A Security Architecture for Personal Networks

Assed Jehangir

A Security Architecture for Personal Networks

Assed Jehangir

A Security Architecture for Personal Networks

Assed Jehangir

CTIT Ph.D.-thesis
Centre for Telematics and Information Technology
University of Twente, P.O. Box 217, NL-7500 AE Enschede

ISBN 978-90-365-2818-4

Email: assed.jehangir@siemens.com

The research presented in this thesis was supported by the Dutch Ministry of Economic Affairs, under the Innovation Oriented Research Program (IOP GenCom, QoS for PN@Home).

A SECURITY ARCHITECTURE FOR PERSONAL NETWORKS

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
Prof. Dr. H. Brinksma,
on account of the decision of the graduation committee,
to be publicly defended
on Thursday, April 9th, 2009 at 16.45

by

Assed Jehangir

born on January 10th, 1978
in Neuilly-sur-Seine (France)

This dissertation has been approved by the promotors,
Prof. Dr. Ir. B.R.H.M. Haverkort and Prof. Dr. Ir. Sonia Heemstra de Groot.

Promotion Committee:

Dr. rer. nat. H.P. Schwefel	Aalborg University
Dr. Ir. A. Pras	University of Twente
Prof. Dr. R.J. Boucherie	University of Twente
Prof. Dr. Ir. A.J. Mouthaan	University of Twente
Prof. Dr. Ir. B.R.H.M. Haverkort	University of Twente
Prof. Dr. Ir. I.G.M.M. Niemegeers	Delft University of Technology
Prof. Dr. Ir. S. Heemstra de Groot	Delft University of Technology
Prof. Dr. S. Etalle	Technical University of Eindhoven
Prof. Dr. Ir. E. Fledderus	Technical University of Eindhoven

Acknowledgment

This thesis describes work that I performed at the Design and Analysis of Communication Systems (DACS) group at the University of Twente, under supervision of Prof. Sonia Heemstra de Groot. Her encouragement, support and guidance were instrumental for its successful outcome and I am much obliged to her.

Many other people were also of important for the completion of this thesis, and I wish to thank them. My committee members for reading and reviewing this thesis. My colleagues at the DACS group for their enthusiasm and support, especially those with whom I shared my office over the last few years: Anne, Jose, Tiago, Lucia and Richa. I would also like to thank my colleagues in the IOP GenCom QoS for PN@Home project, for making the meetings a pleasant experience which I enjoyed very much.

Finally I would like to thank my friends and family for their support over the years. Amongst my friends I would especially like to thank Imran and his wife Samina for their encouragement and support. My greatest thanks go to my parents and brother for their constant encouragement and my wife for her unconditional love and support.

Assed Jehangir, March 2009.

Abstract

The proliferation of personal mobile computing devices such as laptops and mobile phones, as well as wearable computing devices such as belt computers, digital bracelets and bio-medical sensors has created an opportunity to create a wireless network to share information and resources amongst personal devices. One such paradigm which utilizes pervasive and ubiquitous computing to create a network of personal devices, both in the local vicinity and those at remote locations, is called a Personal Network (PN). The aim of a Personal Network is to provide its users with new and improved services.

As Personal Networks edge closer to reality, security becomes an important concern since any vulnerability in the system will limit its practical use. However the mobile and constrained nature of its constituting devices places unique requirements on the design of Personal Networks, such as the need for low power consumption and the ability to self organize in the face of intermittent connectivity. One of our conclusions in this regard was that the new characteristics and possibilities offered by Personal Networks mean that old solutions are often not suitable in their current form. Therefore in this thesis we introduce a novel security architecture especially designed for Personal Networks.

As people with a network background, our aim was not to develop new PN specific cryptographic protocols, but to develop a model for secure *network* architecture. In this regard our focus is more on defining mechanisms for access control, rather than the security properties of specific protocols. For instance, we propose mechanisms for device personalization, key management, resource discovery, authentication and secure network formation/communication. Our proposals are then analyzed analytically based on the main drivers for our design choices, with some parts evaluated using the Ns-2 network simulator. Where possible we have attempted to reuse existing and well established security protocols, knowing that proposing a novel protocol specific to PNs only introduces the possibility of security flaws common to new protocols.

Given the infancy of the PN concept, our *first* contribution is in promoting the development of this concept as related to security. In this regard we have identified the different architectural components which play a part in enabling security and specified their functional roles. This required an understanding of typical user

behavior as well as development of scenarios which highlight the challenges and requirements in connecting heterogeneous personal devices in a self organizing manner. The *second* main contribution is in designing a secure architecture around these entities which meets the rather unique requirements identified earlier. In this regard we have specified mechanisms for the secure formation and communication in Personal Networks as well as communication between different Personal Networks.

Contents

Chapter 1

A new user-centric networking paradigm

1.1 Introduction

It used to be said that as more and more functionality gets packed into one device, users would own fewer and fewer devices. Although devices are becoming more multi-functional, this statement did not turn out to be very accurate because the number of devices owned by a user has not changed significantly. It is still typical for a person to own many of the following devices: mobile phone, PDA, laptop, MP3 player, digital camera, video recorder, audio headset, pager etc. In fact, new types of applications require embedding tiny sensors in home appliances, jewelry, watches, belt buckles and even clothing. There is a growing need for all of these devices to interconnect and share resources with each other in a seamless fashion.

Although some of these devices can connect to infrastructure networks like WiFi Hotspots relatively easily, it is almost impossible for an average user, using existing technologies, to configure a self-organizing network that enables easy and unified access to remote personal content and applications. Furthermore certain devices like biomedical sensors, are very resource constrained and support reduced functionality forcing them to interact with special readers using non-standard technologies. The challenge to connect such a wide variety of devices into a unified, smart, secure and self-configured network only increases when considering the growing mobility needs of today's users. Such is the vision of a Personal Network (PN) [1] [10], which aims at integrating different technologies to develop seamless solutions that are easy to use and enable ubiquitous access to information and communication.

We believe that designing a completely new architecture for Personal Networks not only makes them practically operative, it also allows optimizations that make the overall design more efficient. As PNs edge closer to reality, the supporting security models and mechanisms must also evolve. For instance, given that self-organization

of personal devices into a PN is an important requirement, a suitable concept of trust between personal devices must be developed. Furthermore, physically distributed systems such as PNs rely on the transmission of messages and events that need to be secured before such systems can be widely deployed. The identity of entities involved, the authorization to access available services and the privacy and integrity of transmitted messages must all be established.

In this thesis we introduce a security architecture designed specifically for Personal Networks. We will begin by describing the security requirements of PN along with the main drivers which will influence our proposed design choices for enabling the secure formation and communication between PNs. Of course where possible we will take inspiration from existing security frameworks and mechanisms, however as we will show later, given the heterogeneous and mobile nature of the Personal Network this is not always possible. Our aim is to propose security solutions that enable the secure formation and communication within a Personal Network, and also between different Personal Networks.

1.2 The Personal Network - Concept and architecture

A Personal Network (PN) is a self-organizing, secure and private network of a user's devices notwithstanding their geographic location. It connects the various personal devices of a user seamlessly, at anytime and anyplace, even in the face of mobility. Another way of putting this is that while a Personal Area Network (PAN) connects devices around the vicinity of a person, the PN extends the PAN with devices and services farther away. However a PN is more than just connectivity. The aim of a PN is to create a self-organizing and intuitive support network that supports many different types of networks and devices with the aim of providing its users with new and improved services. As such it has a very strong user-centric view.

As people move they leave behind some of their devices in various living and working domains e.g. home, car and office. Such geographically co-located personal devices organize themselves in the form of secure "subnets" which we call **clusters**. Devices in a cluster communicate using short range radio links. In Chapter 3 we will explain why forming clusters allows the communication, routing and other self organization mechanism between personal devices to be protected. In other words, forming clusters facilitates in securing multi-hop communication and access to the common pool of resources. Clustered devices have one or more short range wireless interfaces like Bluetooth [106], IEEE 802.11 (WLAN) [109], IEEE 802.16 (WiMAX) [61] and ZigBee [108]. They are expected to be mobile and their membership status, as well as their physical location within the PN can change at any time.

The otherwise geographically seperated clusters are interconnected using secure

dynamic tunnels, created between **gateway** devices, resulting in a network of personal devices that are geographically dispersed. This on-demand and transparent extension is physically made via infrastructure-based networks such as an organizations intranet, public networks such as UMTS, WiFi, WiMAX, DSL and even other ad-hoc networks. In this work we focus on securing clusters in which constrained mobile devices communicate over the insecure wireless medium. When communication occurs over more secure channels such as wired Ethernet, security can be enforced using simpler techniques such as firewalls at network entry points.

Figure 1.1 outlines our PN architecture as developed under the QoS for PN@Home project [2]. A special type of cluster known as the **P-PAN** cluster (Section 1.2.2) is the set of personal devices in the vicinity of the user. A personal device is one which has gone through a personalization phase (details in Chapter 3), as a result of which it has the necessary long term trust relationship necessary to join the PN. From the figure we also see that remote personal devices belonging to User 1 are grouped into his car and home clusters. These clusters then inter-connect using secure tunnels to form PN_1. We envision PNs to contain heterogeneous devices, from resource constrained sensors that must last months on small batteries, to powerful devices like laptops and PDAs that are recharged often. The main challenge in securing PNs is in using mechanisms that are suitable for resource constrained devices yet robust enough for secure communication and self organization. Our work focuses on securing constrained devices since any mechanism suitable for them will also function with more powerful devices.

In Figure 1.1 we also see that there are two basic types of devices, **personal** and **foreign**. Personal devices belong to the owner of the PN, while foreign devices do not. In the PN trust model there is a strong focus on the long-term trust that exists between personal devices. As this trust does not extend to foreign devices, only personal devices can be part of a user's PN. Consequently although the boundaries of the PNs do not intersect, PN_2 and PN_3 are able to share select services. In Figure 1.1 User 3 is able to view pictures stored on User 2's camera on his PDA. Similarly, User 2 is able to listen to the songs stored on User 3's iPod on his speakers. In the remainder of this section we will look at the different architectural components in more detail. Note that even though the secure tunnel endpoints terminate at personal clusters, in order to keep the figure readable we have not shown tunnels inside the interconnecting structures.

1.2.1 The Case for the PN

Imagine owning a number of digital devices with varying functionalities and capabilities. Think of the possibilities if all of these devices, irrespective of their location, are able to network together seamlessly and even handover state to each other. Such a network will enable users to share information, play games, control their home remotely and even enjoy everyday tasks more. In this section we will presents two use

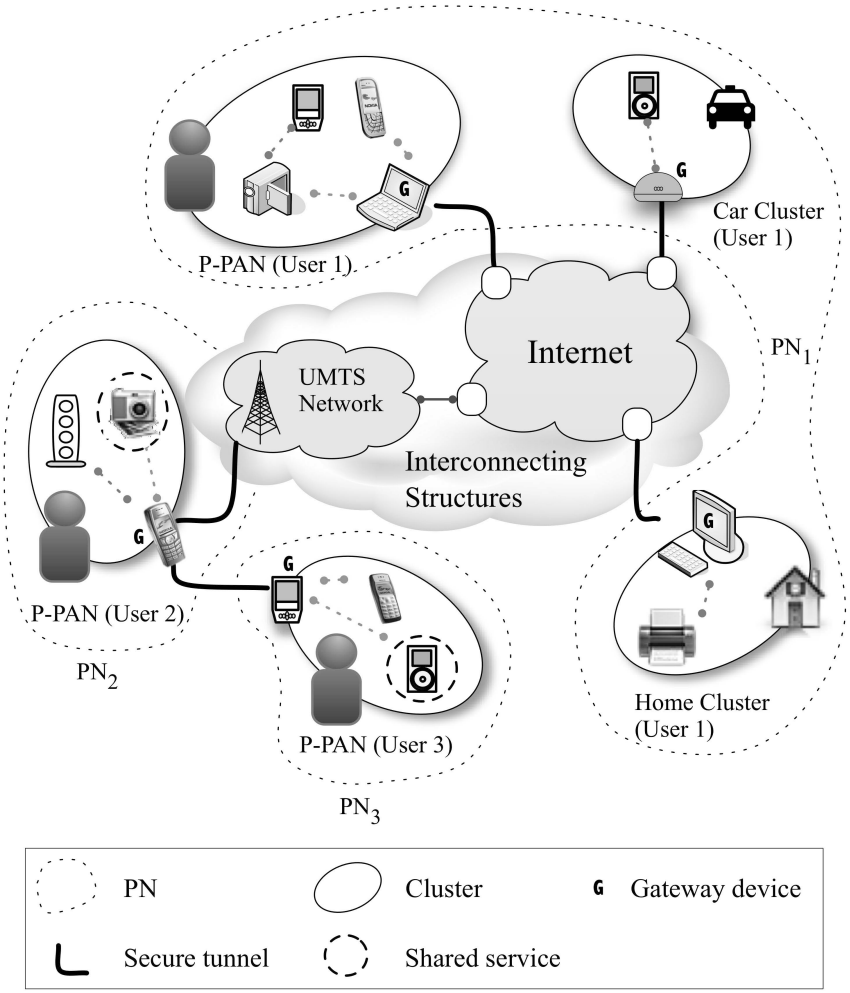

Figure 1.1: Three Personal Networks belonging to three different users. PN_1 is formed of three clusters while PN_3 only comprise one cluster, the P-PAN. PN_2 and PN_3 are sharing a subset of their services.

cases that validate the ubiquitous usage of PNs. Interested readers can find more scenarios are available in [114] and [115].

The traveler

The traveler scenario is based on one of the major benefits of a PN, which is the location-independent seamless access to personal resources. Such resources can be stored in any remote or local location and the user is able to access them easily as long as there is even one type of network access available. Mobility is becoming a common part of everyday life, whether it is traveling to another country or even waiting in traffic on the way to work. This scenario will focus on the case of a tourist (Bob the manager) who is visiting a foreign country. Bob is expected to be on the move a lot, and wants to maximize his touristic activities. Furthermore he also wants to communicate with his family and office while on the move. The ability to do the latter may either be for emergencies only or in some cases may potentially encourage people to take more vacation time without loosing touch with their professional obligations.

In this scenario we will look at three functionalities: Maximizing touristic experience, interacting with family and interacting with the office. These functionalities utilize the PN framework that enables Bob's devices to seamlessly cooperate and to communicate with local as well as distant devices. For instance information points in museums, multimedia devices at home or company servers in the office. In this scenario we assume that Bob is carrying a mobile phone, a PDA, an audio headset and a digital camera. The mobile phone has a UMTS interface as well as a GPS module. The PDA has a WLAN interface and a built-in video camera. All devices also have short range wireless PAN interfaces (IEEE 802.15) like Bluetooth which are used for intra-cluster communication. The different access technologies like UMTS and WLAN are meant to be transparent to the user and are only mentioned for completeness.

Company is always welcome for a single traveler. Rich communication can be achieved using displays, cameras, speakers and microphones which are part of the home cluster. Whether it is sharing the view of the Eiffel tower with the family back home using streaming video or still pictures that are displayed on the family's Hi-Definition television. Through the PN environment, Bob can virtually see and talk to his family with ease. All this is done while on the move using the best available Internet connection. For instance, if Bob is talking to his wife using a VoIP connection on his PDA, as he moves away from a public hotspot his PDA looses Internet connectivity and immediately transfers the ongoing talk over the UMTS connection of the mobile phone. Later if Bob meets interesting people when on the train, he can set up a temporary communication channel and exchange his pictures or other files and maybe even play a game.

Office related duties that can be performed are access to personal and shared files

as well as agendas. These can be especially useful if Bob is contacted by a customer and to whom he needs to provide information or make offers. Even thought this application seems simple, it is quite important. Other variations include Bob making a video conference call with the customer as well as a colleague at the office. This will require different devices in the cluster to collaborate and use the best network access that is available to any one of them. For instance, the Bluetooth headset is used for listening and speaking, the PDA is used for displaying and capturing video, and the UMTS interface in the mobile phone is used as the best available Internet interface.

The PN can also help Bob maximize his touristic experience. At a tourist spot, his PDA can connect to a public hotspot and download information from the tour guide server. Visual items are displayed on the PDA, which forwards audio to the headset. Any pictures taken by the camera are immediately uploaded to the home entertainment center through the PDA, where the rest of the family can also see them. He can even ask their opinion when buying souvenirs and gifts etc. They are also able to follow his travel by reading his GPS signal. Furthermore, Bob is able to locate his favorite fast food chain or the next tourist destination by using online directory services offered by the tourist office or a third party, and get step by step directions. If Bob discovers that a local friend of his is out of town, he can get in touch with him and ask his friend to configure his car so Bob can add it to his personal network for a few days.

These examples show that PNs can be a powerful tool for communication and collaboration. Although there are existing technologies and other still in development which claim to support similar scenarios, they do not combine all these functionalities into a flexiable and integrated solution that is easy to use for normal users. Most offerings are application specific and proprietary and often require technical expertise to set up.

Care for the elderly

In many developed countries it is common for the elderly to live apart from their grown-up children. Given how globalization is talking place, this trend will likely extend to the developing world as well. Even though many elderly people face life alone, they sometimes do not prefer to move to old age homes where they can be looked after. There is potential for the PN to make life for such people easier by supporting them in emergencies and even everyday tasks. We will look at two such functionalities: health care support and smart grocery shopping.

Consider Alice, whose suffers from Epilepsy. Epilepsy is a neurological disorder that affects millions of people around the world. People who suffer from epilepsy are plagued by unprovoked seizures. They find that their quality of life suffers greatly due to the unpredictability of their seizures. Research has shown that these seizures can be predicted by monitoring the various vital signs such as blood pressure, heart

rate etc. of a patient in question. The PN can leverage this knowledge to provide epilepsy patients with a support system that can help them in emergencies.

In this example, Alice's PN consists of one or more medical sensors which enable continuous monitoring of her health. These could be fit into a special bracelet that she wears. This bracelet communicates her vitals to the PDA which is responsible transmitting them and other relevant information to her health monitoring service. The health monitoring service can utilize information such as the Alice's current location and the location of her care-giver to provide customized care. If a seizure is detected, it takes appropriate actions such as directing the primary care givers to her location and contacting her over a special device to reassure her that help is on the way. Depending on the configuration, the PN may also inform Alice's family and her local doctor (GP) about her situation.

As her vital signs are constantly monitored, Alice is able to live independently and is also no longer house bound. Furthermore since Alice's health can be monitored by the doctors remotely, she can lead a more normal life by not having to visit the doctor as frequently. This will also directly reduce operational costs of hospitals and allow doctors to focus more on emergencies. In essence the PN is able to provide Alice with a better quality of life.

Other types of assistance provided by the PN can be more routine in nature. The PN can remind Alice to take their medication if early Epilepsy symptoms are detected. It can also remind Alice about what is needed when she goes to the grocery store by checking sensors at home about the food stock in the fridge. Moreover if Alice still cooks, her PN can remind her of the ingredients required to prepare the day's selected recipe. It will also enrich her quality of life by improving her communication abilities with her family and friends.

1.2.2 From a PAN to a P-PAN – User centric networking

Before the Personal Network concept, there was the concept of a **Personal Area Network** (**PAN**). The PAN is defined at a connectivity level, as a collection of devices within a personal area of the user (typically about 10 meters) that communicate using local interfaces. The PN concept is a user-centric extension of the Personal Area Network (PAN) concept, with a renewed focus on pervasive and ubiquitous computing. This user-centric extension is based on the concept that while the shared resources of a device will typically be accessible to all other personal devices, this will likely not be true for foreign devices. In Figure 1.2 we see how a PAN containing seven devices capable of networking with each other using local interfaces, translates into the PN model.

From the definition of the PAN we know that all the devices in Figure 1.2 are capable of communicating with each other using local interfaces. However the traditional PAN networking model no longer applies since devices in a P-PAN are

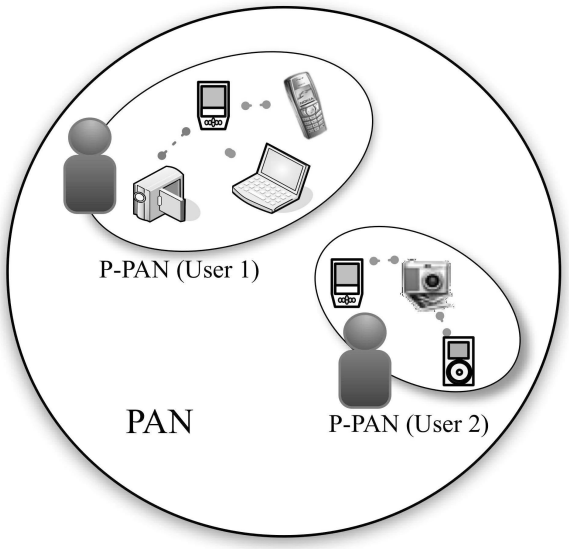

Figure 1.2: A PAN containing two different P-PANs

enhanced with the new concept of group trust based on the fact that they all belong to the same owner. Thus the P-PAN (as well as all other PN clusters) are not defined at the connectivity level.

1.2.3 Cluster organization

Personal devices with direct radio connectivity use their long term trust relationships to organize themselves into secure clusters. As more co-located personal devices are discovered, the cluster expands. Establishing secure connectivity in the form of a cluster allows network level communication between personal devices to take place without using devices that do not have such a long-term trust relationship. We call this **intra-cluster** communication. Any communication involving non-personal devices, including infrastructure based devices is called **inter-cluster** communication if the end points are personal devices. If one of the end points is a foreign device, it is instead called **inter-PN** communication.

We expect intra-cluster communication to be more efficient since inter-cluster communication must rely on inter-connecting structures which can be less reliable (in terms of security, performance, cost, etc) than intra-cluster communication. Note that since the P-PAN operates in the same way as any other cluster, the mechanisms used for self organization, routing etc. are the same.

Clusters are composed of heterogeneous personal devices and are dynamic entities due to the mobile nature of their constituents. Members of a cluster can be switched off (or even run out of battery) as well as move between or out of clusters. A cluster

can split if the user takes a group of its constituents and moves away.

Our cluster formation mechanisms are proactive in the sense that they attempt to make clusters as large as possible. This means that two personal clusters with direct radio connectivity will merge to form one larger cluster. However we do not expect personal clusters to become overly large because a single user will own or can carry around only so many PN capable devices.

Since clustered devices can have different radio technologies, we use network connectivity as the glue to bring these different underlying technologies together. This means that two clustered devices within each other's radio coverage might require multi-hop communication if they do not have a common radio technology. Note that design issues related to the addressing, routing functionality are beyond the scope of this work which only focuses on secure PN formation and service access.

The gateways as the entry and exit points of a secure network (i.e. the cluster) need to control the flow of sensitive information. Consequently they have special functionality such as network address translation, traffic filtering etc. required for this task.

1.2.4 PN organization

As PNs are composed of multiple clusters that are geographically distributed, communication between these clusters takes place using secure tunnels. We have stated that only devices with gateway capabilities are able to connect with the interconnecting structures etc. through which remote clusters can be accessed. Thus the secure tunnels established between clusters which are necessary to form the PN, are only created between the gateway devices of participating clusters. Note that the decision on when such tunnels are created e.g. whether they are on demand or permanent is related to the context aware framework in the PN and is beyond the scope of this work.

Since the clusters are otherwise disjoint they need a mechanism to locate each other before the secure tunnels can be initiated. We call such functionality a **PN agent** (see Figure 1.3). The PN agent functionality is similar to the home agent in Mobile IP, of course suited for PN requirements. All cluster gateways register their new point of attachment with their PN agent. A registration request includes at least the IP of the gateway and the identity of the cluster. When a cluster moves it must deregister with the PN agent otherwise the PN agent will automatically deregister the cluster after a timeout if it fails to receive the keep-alive messages.

Since a PN agent maintains current reachability information about clusters it facilitates their discovery for PN formation. If a foreign devices need to establish communication with the PN, it may be useful to use the PN agent as a point of contact if direct connection is not available. However a PN agent does more than only handling mobility, it also supports other PN tasks such as state synchronization,

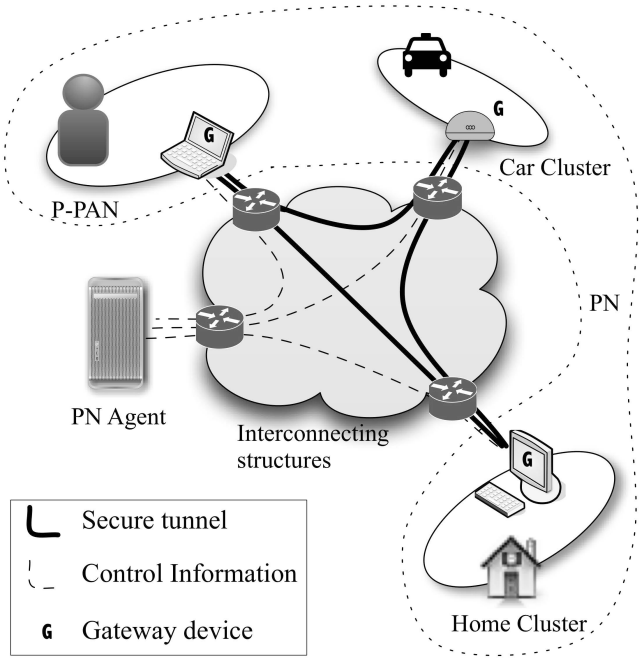

Figure 1.3: A PN agent facilitating PN formation.

service discovery, etc. between distributed clusters. We will look at these roles in more detail later. The PN agent functionality can exist on a suitable PN device e.g. in the home cluster, or it may be offered by a trusted PN service provider. By delegating these tasks to a PN service provider the owner of the PN can reduce his management overhead. In Figure 1.3 we see otherwise disjoint clusters utilizing services offered by a third-party PN agent to connected to form a PN.

Lastly, as the PN agent is a convenient place to maintain information related to cluster location (and their services) it should provide the PN owner with an interface to query information related to PN composition. Although the different functionalities performed by the PN agent can be distributed over different physical entities, for simpler management we envision these functionalities to be centralized on a single entity.

1.2.5 Intra-PN and Inter-PN service access

Securing intra-PN service access utilizes the concept of single PN trust domain, which is based on the long term trust existing between personal devices. Given that devices involved in intra-PN services access all belong to the same user, we do not believe that it is necessary to implement access control based on the specific identity of the personal device but rather its ability to demonstrate its membership of the secure PN group. We argue that personal devices as cyber representatives of a user

can be assumed to be working on his behalf, thus there is no need to incur the additional cost of per device access control.

Inter-PN communication extends the core idea of the PN by allowing users to share their services with each other. As the concept of intra-PN trust does not extend to foreign devices, they cannot be part of a user's PN. However although the boundaries of the PNs do not intersect, they are able to share selected services. For instance, a user may only allow one friend to listen to his songs but allow another friend access to both his songs and his picture album.

We define two types of inter-PN relationships, **pair-wise** and **federations**. Pair-wise relationships are created between two specific PNs, and leverage on the pair-wise trust between the two PNs. The set of shared services depends on the specific identity of the PN and can be modified whenever necessary.

A PN federation is a relationship existing between a group of PNs and leverage on the concept of group trust within the federation. Federations are a useful tool when a group of PNs must cooperate quickly and conveniently for a common purpose. Examples include federations between attendees in a conference, colleagues in an ad-hoc meeting and multiplayer gaming. As with real world relationships, a PN can of course be a member in multiple federations.

For each federation the members PNs make available a set of services. It is important to note that all members are equal in terms of their access to such services. Federations are initialized and managed by one of the members which we call the PN-F manager. Among other tasks, the PN-F manager is responsible for enrolling new members by providing them with credentials to authenticate as part of the federation.

Figure 1.4 illustrates a federation where three users are connected through secure connections between their respective P-PANs. These connections are only created on demand do not need to exist throughout the lifetime of the federation. We can see that even though User 3 is not sharing any services with other federation members, it has the same access to the services offered by User 2 that User 1 has. As with pair-wise PN relationships, once the secure connection between two PNs is established the services being shared by one PN are accessible by all devices belonging to the peer PN.

Inter-PN trust establishment

When devices belonging to different PNs need to communicate or share resources, intra-PN trust mechanisms must be extended to support interaction between multiple PNs. Trust for pair-wise relationships can be created by the manual interaction of the two users, or by using the facilities of a mutually trusted third party. Later we will look at the specific mechanism in more detail.

The main advantage of a federation, as extension to the pair-wise PN trust

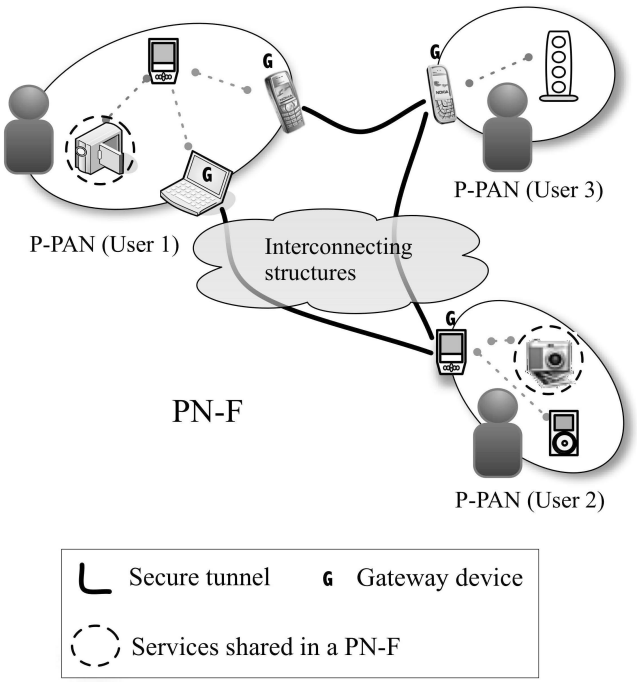

Figure 1.4: A PN-Federation involving three users.

model, is that it does not require each new federation member to share a security association with all existing members in order to access their services. Consequently the amount of manual interaction required to establish the necessary trust relationships is reduced. For instance for a federation with n members, only $(n-1)$ trust relationships must be established compared to $\frac{n(n-1)}{2}$pair-wise relationships. Moreover management tasks like member eviction are the responsibility of one user (the PN-F manager) and do not need to be duplicated by others.

1.3 Security in PNs: Requirements

Given the rather wide scope of Personal Networks it is clear that there are many potential areas of research. In this thesis we have focused on one very specific aspect, that of security in such a distributed and heterogeneous environment. This meant finding answers to questions like: "How can we best protect the privacy and confidentiality of the user's communication?" and "What is the best way of protecting the networking mechanisms from malicious attackers?". For clarification, even though PNs are likely to interact with network providers we have tried to avoid assuming the existence of specialized support systems at the operators.

Our approach to the problem of security in PNs was to start by formulating a set of high level security requirements (listed below). Based on these requirements we

analyzed existing security models and mechanisms proposed for distributed and self-organized systems such as mobile ad hoc networks and sensor networks. In Section 1.4.1 we explain why we concluded that existing solutions were often not suitable in their current form. The next step was to define a high level security architecture that met the requirements listed below and then to make this high level architecture complete by proposing detailed solutions for various components. Although we have evaluated most of our proposals analytically, in Chapter 4 have used network simulations to quantify the overhead of some of our proposals. In future work we would like to incorporate our security models with other components of the PN architecture [2] into a full-fledge prototype in order to validate its usefulness.

1.3.1 Security of personal assets

We envision the Personal Network to function like a support platform, enabling the user secure and seamless access to his resources irrespective of his location. With personal assets at stake, PNs can only succeed if people trust them. Unfortunately wireless networks are more vulnerable to attacks than wired ones due to broadcast nature of the transmission medium. Given that Personal Networks combine characteristics of other wireless and distributed systems, they are expected to share many of the security requirements [97] [21] in existing distributed wireless communication. Thus they also have general security requirements like privacy, trust and availability. These requirements can be subdivided into the following building blocks: confidentiality, integrity, anonymity, authentication, authorization and non-repudiation. Security requirements with respect to the individual PN components are presented in Section 2.3.

However we would like to stress that the new characteristics and requirements of personalized communication also leads to new types of security and trust problems which must be explicitly addressed. For instance, given the user scenarios envisioned for PNs, how do we model trust between the different devices belonging to one person, and also between different persons? One thing to always remember is that if the proposed trust models and their security mechanisms are not comprehensible to average users they may use them incorrectly or become frustrated and simply disable them altogether [54].

1.3.2 Optimized for constrained devices

Devices in a Personal Network will vary in terms of computational power, battery capacity etc. from laptops to a sensors. Therefore when selecting protocols we must consider requirements of devices that are not able to perform public-key operations or frequent radio transmission. Designing a security framework for constrained devices has generally been problematic especially since attackers have none of these

constraints. Personal devices such as cameras, Bluetooth headsets, biomedical sensors, wrist watches, belt computers etc. may be constrained in their:

- Energy: Many personal devices are designed to be mobile. Their small batteries need to last months without recharging or replacement. We consider energy to be the scarcest resource in our system and our security mechanisms must be frugal in their power consumption. This also means that we must minimize the number as well as the size of any transmitted messages.

- Processing Power: Processing abilities are extremely limited, in line with the constraints imposed on their power usage. A typical example of a constrained personal device is that of a Berkeley Mica Mote sensor [24]. It features an 8-bit 4 MHz Atmel ATmega 128L processor with 128 Kbytes program store, and 4 Kbytes SRAM. The processor only supports a minimal instruction set, without support for multiplication or variable-length shifts or rotates.

- Storage space: A limited storage space means that only a limited number of cryptographic keys can be stored in the device. Also, any proposed security framework must be compact in its implementation.

- User Interface: Some devices may only present their users with a few buttons and possibly a LED for indicating their state e.g. sensors, wireless headsets, cameras etc. Our design cannot make any assumptions about minimum user interface requirements.

- Cost: Finally, the cost of a device will likely play a critical role in its success. Any cost overhead of security must be proportional to its benefit.

1.3.3 Symmetric or Asymmetric cryptography

Energy consumption is also a prominent issue in battery operated devices and affects their usability in the real world. Energy is the scarcest resource in our system and our security mechanisms must be economical in its consumption. Table 1.1 [16] summarizes the energy consumption of three common cryptographic algorithms. We can see that AES being a symmetric cipher incurs the same cost for both encryption and decryption. For the other two, the first value represents the cost of decryption and the second the cost of encryption. When implemented in software, RSA [3] [4] and ECC [5] [6] both perform un-satisfactorily in resource constrained devices. For example, key exchange using ECC implemented in software (which actually performs better of the two) when done on an 8-bit, 7.3828-MHz ATmega 128L processor (a MICA2 mote) takes 35 seconds and requires 34k bytes of memory [4]. In future we may have even weaker processors as the number of gadgets people own increase, and the need to increase their battery life becomes more essential.

Table 1.1: Energy consumption of three common cryptographic algorithms

Algorithm	Energy/op (HW)	Energy/op (SW)
AES (128b)	0.045 μJ	17.9 μJ
RSA (1024b)	2.41 / 0.37 μJ	546 / 16 μJ
ECC (163b)	0.66 / 1.1 μJ	134 / 196 μJ

It is well known that using specialized cryptographic hardware instead of the general purpose CPU reduces the overall energy usage, the code size and also allows the product designer to use a lower power CPU since more of it will now be available to applications rather than for communications. However hardware implementations of asymmetric cryptography are more costly than those of symmetric cryptography (due to much higher number of gates [3]) and are not cost effective since it is only used for the single purpose of key establishment. Consequently we have concluded that our basic security mechanisms required for secure PN formation and communication must only rely on lightweight symmetric cryptographic primitives. As a result we must also reject mechanisms based on the Diffie-Hellman protocol [89] given the cost of integer exponentiation. In other words, hybrids of symmetric and asymmetric cryptography, in which asymmetric cryptography is used to exchange the symetric keys are also not possible. Where possible we will use or modify existing well established security models that utilize the inherent assets of PNs, principally that personal devices fall under one administrative domain and that the heterogeneity of device types allows us to delegate tasking requiring larger resources to more able devices.

1.3.4 Other Requirements

High usability

Since the Personal Network is meant to support normal people in their daily activites, one of our main design requirements is to protect the Personal Network while maintain good usability. This means that we would like to protect users from any burdensome manipulation or requirements of remembering long passwords, etc. Very often there is a give and take between increased security and better usability. Given the non-critical nature of most personal communication, we believe that the default configuration should be inclined toward better usability. A more usable system reduces the user involvement and simplified it when it is necessary. Where possible the user should not be presented with a plethora of possible security options, the consequences of most of which he will likely not understand [53]. This includes having to remember multiple passwords or long keys.

Lack of infrastructural support

Given the nature of distributed mobile computing, we cannot assume permanent connectivity to the infrastructure. Consequently our mechanisms for establishing both intra and inter-PN trust should not require the permanent availability of infrastructural support. When available, infrastructure based trusted third parties such as certification authorities may be used, but here must be alternatives to support ad-hoc operation.

The security requirements listed above are meant to guide us in evaluating our final security architecture by verifying them against these requirements. However since most of these requirements exist at a rather high level, the decision of how well we meet them will be somewhat arbitrary. Nevertheless it is important to formulate such requirements as they do play an important part in directing our research. Finally, in terms of the overall PN architecture there are many more non-security related requirements, some of which vary depending on the application or target group etc. We do not attempt to cover these since they are beyond the scope of this work and have already been done rather extensively in [82].

1.4 Related Work

There is a plethora of security solutions proposed in the area of wireless communication. The most relevant are those that have been considered in the area of personal communications, however it is important to point out that there have been very few attempts to achieve a complete solution for all personal communication needs. In this section we will only give a short overview of other attempts in the area of personal communication, with detailed comparisons of security mechanisms (if possible) provided later in relevant chapters.

The two most relevant projects, IST MAGNET and MAGNET Beyond [8] [9] projects are natural evolution of the earlier work done in the IST PACWOMAN [95] [96] and IST SHAMAN [29] [30] projects. Both of these earlier projects started close togther and worked on issues related to PANs. The Power Aware Communications for Wireless Optimised Personal Area Network (PACWOMAN) project aimed at developing enabling technologies for PANs, specifically the design of a low-power, scalable and secure PAN. The main research area was optimisations at the link layer with the aim of providing a physical/MAC layer that enabled low-power operation and scalability. The results were relevant for personal communication given the battery operated nature of most devices. The IST SHAMAN project focused on providing a security architecture for PANs in order to provide secure roaming, access over heterogeneous radio networks and security. They defined corresponding security mechanisms, protocols and procedures based on a public key infrastructure. Another contribution was to define the security features and procedures involving smart cards

and other security modules. The basis for the architecture was a trust model which classified devices into three categories: first party, second party and un-trusted devices. With respect to a given device, other first party devices are those which belong to the same owner. Second party devices are those belonging to a different owner, but with whome the given device shares a security association. The security association is used to enforce access policies defined by the owner of each device. Although the different concepts developed in the SHAMAN project (e.g. its trust model) provided a good foundation, the limited scope (PAN vs. the PN) and the focus on public key infrastructure [28] meant that many of its security mechaisms were not usable. Since the IST MAGNET and MAGNET Beyond approaches are the most related and worthy of detailed comparison, for the sake of easier readability this comparison will be made in the relevant chapters.

The Mobius project [113] from the University of Illinois at Urbana Champaign, groups devices in close vicinity into so called Mobile Grouped Devices (MOPEDs). A MOPED, similar to a PN cluster, is composed of homogenous devices that are connected to a central proxy which is similar to a home agent over the infrastructure. However MOPED type architectures are not suitable for PNs because they require extremely centralized solutions which need guaranteed connectivity to the infrastructure. Moreover MOPEDs do not address the issue of person to person communication since they do not have any suitable mechanisms for such trust creation. In summary, the MOPED vision is not as broad as that of the PN and overlap is rather limited.

The concept of networking personal devices has also not escaped the notice of those with a more commercial interest, however most such products lack the broad vision of the PN concept and are proprietary solutions that are restrictive in their flexibility. For instance, IXI Mobile [77] is a commercial product built around what the company calls the Personal Mobile Gateway (PMG). The PMG is a mobile phone with both 3G and WPAN technologies, that has been extended to facilitate creation of a person's PAN and manage its communications with the outside world. However the vision (in terms of use cases) is not as broad as a PN and all services and external communication must go through the operators network.

Based on the user scenarios and the types of devices we envision in PNs, it is clear that PNs also have similarities with the more general purpose mobile ad-hoc and sensor networks. Since it does not make sense to reinvent the wheel, we also took steps to study security models and mechanisms proposed in this larger context. Unfortunately we found that the differences in the architecture when compared with PNs meant that most security models were not applicable.

1.4.1 Similarities and differences with ad-hoc and sensor networks

PN clusters are basically mobile ad-hoc networks that include a wide variety of devices, from powerful PDAs and mobile phones to embedded devices that are very resource constrained. During our study of security models in typical Mobile Ad-hoc NETworks (MANETS), it soon became clear that existing models key management were not directly applicable to PNs because they assume that participants are homogenous in their capabilities as well as their susceptibility to theft. Furthermore the basic trust model is often different since most scenarios assume that each peer is owned by a different person, while all devices participating in the PN belong to the same person. An example of a typical MANET would be an ad-hoc meeting, P2P network or a gaming session where each individual has a laptop or a PDA. Finally, since we concluded earlier that the basic security mechanisms required for secure PN formation and communication must also support constrained devices, we decided to narrow our focus to look at models and mechanisms designed for wireless sensor networks (WSN). This because it is also not possible to use classic cryptographic algorithms and security protocols in networking architectures designed for WSNs.

Since a successful exchange of keys is the first step towards securing the ensuing communication it is not surprising that most security related work for WSNs has focused on issues related to key management [11] [12] [13]. We found that although many devices in the PN are similarly constrained, key management in PNs presents a different set of requirements due to:

- The difference in the network model (as shown by different use cases),
- The heterogeneity of personal devices (from wrist watches to laptop computers),
- The difference in size and scope. When compared to the typically large and unattended sensor network, lost/compromised devices in a PN are easier to detect and subsequently blacklist.

WSNs have two basic network models, Distributed WSNs (DWSNs) and Hierarchical WSNs (HWSNs). In a DWSN there is no fixed infrastructure, and network topology is not known prior to deployment. Sensor nodes are usually randomly scattered over the target area and once deployed scan their radio coverage area to figure out their neighbors. Most examples of this type of sensor networks are based around the military or rescue operations deploying sensors in inhospitable areas. Given that PNs do not have this constraint, we concluded that such an approach is not optimal since the heterogeneity of personal devices in the PN means that instead of being limited into a totally distributed approach to security, we have an opportunity to offload part of the cost of security to more powerful devices. This will allow us to extend the operating period of the more constrained devices.

In the other network model, called HWSN, there is a hierarchy among the nodes based on their capabilities: base stations, cluster heads and normal sensor nodes. In many cases cluster heads have similar resource constraints as normal sensor nodes and only collect and merge local traffic before sending it to base station. Base stations, on the other hand, are many orders of magnitude more powerful than sensor nodes and cluster heads and are assumed to be trusted and tamper resistant. They manage the network (for instance, they are used as key distribution centers) and have enough transmission power to reach all sensor nodes while sensor nodes depend on the ad hoc communication to reach base station. However unlike sensor networks that assume the existence of a very limited number of static and a-priori trusted base stations, the dynamic nature of PN participants means that such roles will have to be dynamically assigned based on availability (Section 1.5.2). For instance, take the SPINS [11] set of WSN protocols which are designed for HWSNs. The WSN forms around a base station where sensor nodes establish a routing tree, with the base station at the root. Nodes forward messages towards the base station (i.e. sensor readings), receive unicast requests from the base station (that are sent using source routing) and also receive broadcasts sent to all devices. Devices establish a master key with the base station upon introduction to the network, and use that key to derive two new keys for protecting unicast traffic between the base station and themselves. Routing is much simpler since devices are only required to have a path towards the base station. A secure routing tree is created (for traffic to the base station) by having the base station send routing beacons protected using the TESLA [19] broadcast authentication protocol. Additionally since battery replacement is designed to zero out all the keys, there is no further key management.

Nevertheless some of the conclusions drawn from research in securing sensor networks [14], particularly those on the suitability of specific cryptographic algorithms [15] are directly relevant. For example, TinySec [14] is the first implemented link layer security protocol for sensor networks. As with our proposed mechanisms it uses a shared symmetric key which is used by senders to first encrypt the data and then apply a Message Authentication Code (MAC). The receiver uses the MAC to verify that the packet was not modified in transit. The TinySec protocol is not a complete architecture in the sense that it does not specify aspects such as key management and device discovery etc. However, the TinySec protocol has been well analyzed (and designed) in the context of sensor devices, especially related to its energy and communication overhead.

Other things that became clear were that that the constrained energy and communication capabilities of large sensor networks meant that protocols such as TLS [63] and Kerberos [27] (originally developed for wired networks) were deemed impractical. Most popular key management approaches used some variation of predeployed symmetric keys. Amongst common proposals were those that use a global key shared by all nodes [17] [18], those in which every node shared a secret key with a base station [11], and those based on random key sharing [13].

Finally, non-WSN related, were models and protocols (based on symmetric cryptography) designed for securing collaborative multicast communities [19] [20] and other long term communities [21] [23] are also relevant. These formed the building blocks of some of our proposed security framework for PNs and we will look at each of them in more detail later in this work.

1.5 Security in PNs: Approach

Based on the requirements listed earlier, this section provides a general insight into our approach for securing Personal Networks.

1.5.1 A secure bubble around the user

We envision the Personal Network to operate as a secure bubble around the user. This bubble, like a digital secretary, travels with the user and enables him secure and convenient access to his resources. Our approach for enforcing security is to make the perimeter of the network secure, so that it can only be accessed by authorized devices. Also, using gateway devices for controlling the entry and exit points of this otherwise closed network means that we can assert a high degree of control.

The approach of establishing a line of defense between the PN and the rest of the world is done by a separation of trust at the cluster level. In other words, cluster level security distinguishes between cluster members who are trusted and non-members who are un-trusted. This is due to the semi-independent nature of clusters coupled with the fact that inter-cluster connectivity is not always guaranteed (even when possible it is infeasible to maintain it at all times for key synchronizations etc). This trust is then leveraged to secure all intra-cluster traffic at the link-layer using a periodically updated cluster wide group key. When compared to higher layer security, link layer security provides a more compelling and complete solution (we will discuss this in more detail in Chapter 3). Personal devices joining existing clusters must authenticate themselves with valid credentials before they can take part in intra-cluster communication. As explained earlier, secure inter-cluster communication is then ensured using secure tunnels created between the gateway devices of disjoint clusters.

1.5.2 Centralized or distributed – Defending our choices

When designing security for a distributed ad-hoc network like a PN, where communication takes place between mobile devices using wireless communication, one is tempted to design a purely distributed security architecture. After all, with mobile and battery powered devices there is really no way to guarantee communication with any centralized resource. Imagine using Kerberos to authorize service access in a

P-PAN. What would happen if the Kerberos server were unavailable to authorize requests? This could happen for a variety of reasons. For instance, the server could run out of battery, moved away, or have intermittent connectivity resulting from interference of its wireless signals. Unfortunately a distributed approach to security also has its share of pitfalls. With more devices involved in security, there is more synchronization overhead and possibly even more chinks in the armor. Ultimately, we choose to have a somewhat centralized security architecture, both at the PN level and at the cluster level. Although we will only look at detailed security mechanisms later on, we take this opportunity to give our rationale.

PN level security corresponds to securing tasks related to the control of device membership and behavior in the PN. It does not deal with the specific mechanisms used for securing cluster formation and maintenance as well as intra-cluster and inter-cluster communication. We believe that PN level tasks are inherently centralized because they do not require any self organization on the part of the system but rather should be in the complete control of the user who is a central entity. Since it is reasonable to assume that the user will not want to make system wide changes to all devices individually, we have chosen to centralize such tasks a single secure point. As the loss of a non-critical device is often not immediately detected, this allows us to limit the effect of compromised devices that are without this functionality.

Cluster level security relates to the mechanisms used for secure cluster formation, maintenance and communication. Due to the mobile nature of many PN devices, cluster level security needs to have a strong element of self-organization. However given the heterogeneous nature of the devices in the PN, a fully distributed approach is too costly for many constrained devices. Our approach thus centralizes management roles to be performed on-demand by any suitable PN device. For clarification, although the roles are centralized in nature they do not match classic centralized systems like Kerberos where the function performed by the central entity cannot be dynamically handed-over or re-assigned. Specifically:

- A distributed approach is attractive if all participating devices are homogenous in their capabilities as well as their physical safety. However a typical PN is envisioned to contain a wide range of device types. This makes a PN different from typical ad-hoc networks (e.g. at meetings, P2P networks, vehicular networks etc.) composed of more-or-less peer devices. For instance, although a user will own relatively powerful devices like laptops, mobile phones and PDAs he will also have small specialized devices like biomedical sensors, Bluetooth headsets, digital watches and wristbands each with a specialized purpose. Since the cost and physical dimensions of these devices plays an important factor in any purchase, the user is motivated to buy low cost and small form factor devices. Consequently we want to keep the minimum functionality required to be PN *capable* at as limited as possible. We do this by centralizing all management overhead to the more capable devices.

- With many digital devices to keep track of some are bound to get misplaced. We can expect that the physical safety of PN devices will vary, for instance, it is reasonable to assume that a digital pen is more likely to be misplaced or lost than a PDA. Although this does not always mean that resource constrained small form factor devices are more susceptible to loss, this may be true on many occasions. Consequently it may be safer for such devices to not be involved in management tasks because otherwise their loss may have a larger an impact on the system.

- One of the most useful benefits of centralizing security is that it simplifies key management. Devices only need to manage security associations with the central entity and not with all the other devices in the network. Since the overhead of key management can be quite costly, this is important benefit for resource constrained devices. For instance, in a network with n devices where each device has a security association with every other device, the loss of one device will require $(n-1)$ revocations. However if security were centralized then only one security association need to be deleted, that been the revoked device and the central entity.

- Certain tasks are inherently centralized e.g. the periodic updates of the cluster wide group key. Such tasks are initiated by what is known as a leader. Without a centralized entity in place to perform this task, it will have to be done after a cluster wide leader election [7] which will have its own overhead.

1.6 Organization of the thesis

The remainder of this thesis is divided into six chapters. Chapter 2 provides an overview of our security model for PNs. Chapter 3 gives details on security mechanisms and protocols for trust establishment, authentication and key management inside the PN. The network overhead of our proposals for cluster formation is analyzed in Chapter 4 using simulations. Chapter 5 investigates existing VPN technologies for securing inter-cluster communication while Chapter 6 investigates the issues involved in securing inter-PN communication and presents the new mechanisms and protocols for this purpose. Lastly, Chapter 7 presents our earliest research in securing PNs, our proposals for secure intra-cluster routing.

1.7 Summary

Challenges in securing PNs derive mainly from the mobile nature of devices, the wireless nature of the communication and the constrained nature of many personal devices. To summarize our architectural choices:

- We have chosen to centralize security both at the cluster level and at the PN level. This allows us to keep the minimum functionality required to be PN capable at as limited as possible.
- We have chosen to enforce security by making the perimeter of the PN secure, so that it can only be accessed by authorized devices through special personal devices called gateways.
- Given the semi-autonomous nature of clusters and because inter-cluster connectivity cannot be guaranteed, we have chosen to manage security and access control of a cluster locally.
- We have decided that our basic security mechanisms should not require asymmetric cryptography. However asymmetric cryptography can still be used between more powerful PN devices by choice instead of the basic mechanisms.

The remainder of the work considers security in PNs in light of the limitations imposed on our design by the above constraints.

Chapter 2

Overview of secure PN formation and communication

2.1 Introduction

This chapter provides an overview of our security model for PNs, with detailed descriptions of supporting mechanisms presented later in the thesis. We propose our concepts for the secure formation of personal clusters and establishment of the PN to enable secure access to personal services.

Security is bound to be a major concern in the acceptance of PN technologies. However the limited computational and energy resources of many potential PN devices imply that our security solutions must be simple and lightweight so that they do not create a performance bottleneck of their own. Unfortunately the majority of current security mechanisms were designed for more powerful workstations operating with some sort of infrastructural support, making them unsuitable for PNs. Furthermore, as communication bandwidth uses the lion's share of available power [11] any overhead arising from the transmission of extra security bits comes at a significant cost. Consequently, not only must our solution minimize the security overhead in data packets but it must also reduce the overhead resulting from activities such as key management and authentication. With the introduction given in Chapter 1, we can now define in much more precise terms the most frequently used terminology in the rest of the thesis.

2.2 Terminology

- **Device**

  An electronic entity that can communicate with other electronic entities through wired or wireless links.

- **Neighboring devices**

  Two devices within each other's radio transmission range.

- **Personal device**

  With respect to a specific person, a PN capable device belonging to that person. Such a device has gone through personalization phase, as a result of which it has the necessary long term trust relationship necessary to form the PN. Note that although the trust relationship is long term, it can be revoked if necessary.

- **Foreign device**

  With respect to a specific person, any device that is not his personal device. A foreign device may or may not understand any PN specific protocols.

- **Personal cluster**

  A secure ad-hoc network of personal devices that can communicate amongst each other without using any non-personal devices. A single personal node functioning as a Security Agent (defined later), is considered a special case of cluster that only contains itself.

- **P-PAN**

  A personal cluster in the close proximity of the owner.

- **Foreign cluster**

  With respect to a specific person, a personal cluster of another person.

- **Cluster member**

  A physical device which is functioning as a member of a given cluster.

- **Un-clustered device**

  A PN capable device that is unable to form a cluster on its own, and is also not a cluster member.

- **Gateway**

  A device within a cluster which enables other clustered devices to access the outside world. The generic term for referring to all PN devices able to function as gateways is PN gateways. However from the point of view of a clustered device, the subset of PN gateways belonging to the same cluster can be called cluster gateways. A cluster may have zero or more gateways.

- **Personal Network** (PN)

  An overlay network composed of personal clusters connected to each other over interconnecting structures.

- **Owner** (of a PN)

  A person who owns or controls the personal devices that make up the given PN.

- **Foreign user** (of a PN)

  A person who is not the owner of the given PN, but uses some or all of its services.

- **User** (of a PN)

  A generic term for a person using one or more services available in a given PN. The user can be either the owner of the PN, or a foreign user.

- **Security Manager**

  A personal device performing the role of a security manager is responsible for security within the Personal Network. It is able to carry out special security and management related tasks on other personal devices. Every Personal Network has one security manager.

- **PN Agent**

  A device performing the role of a PN Agent maintains reachability information about all personal clusters, thus facilitating cluster discovery. It also supports other PN tasks such as state synchronization, service discovery, etc. between distributed clusters. Every Personal Network has one PN agent.

- **Security Agent**

  A personal device performing the role of a security agent is responsible for security within a cluster (details in Chapter 3). Each healthy cluster has one security agent.

- **Interconnecting structures**

  Infrastructure and ad-hoc based public or private networks such as UMTS and the Internet that are utilized to connect geographically dispersed personal clusters.

2.3 Security Goals

In Chapter 1 we presented an overview of our requirements and the general direction of our approach for solving them. However we did not present any security requirements with respect to the individual PN components. Specifically our security goals with respect to:

- **Personal devices**

  - Prevent unauthorized network access of services (Note that we do not consider issues related to physical access of personal devices by un-authorized users. That would require methods for user-to-device authentication and if necessary methods to enforce different access policies for different users.)
  - Limit accidental or malicious attack capability of personal devices unable function as security agents and/or security managers.

- **Clusters**
  - Secure communication between personal devices that make up a cluster
  - Prevent unauthorized addition of devices to clusters

- **PNs**
  - Select a viable personalization and long term trust creation model
  - Be able to revoke the long term trust
  - Secure PN formation and intra-PN communication

- **Interaction with the outside world**
  - Secure interaction that is restricted to the services defined by the owner of the PN.

2.4 Architectural Framework

2.4.1 The concept of trust amongst personal devices

When a new device is configured to become part of an existing PN, we say that the new device has been **personalized**. This is inherently a user controller procedure using a "master" entity which we call the **security manager**. The introduction of a new device to the PN requires the creation of long term trust between existing personal devices and the new device. We choose to make this trust long term because we expect most personalized devices to be part of a PN for an extended duration. However this long term trust is revoke-able on demand.

Our model creates trust between PN devices by establishing a-priori trust relationships during the personalization phase. We do this a-priori because classic network security mechanism based on online access to trusted third parties are not suitable given the distributed and ad-hoc nature of PNs. The trust existing between personal devices is then leveraged to enable them to self-organize into a secure PN.

In terms of PN connectivity and service access, all personal devices are equally trusted. However management tasks are classified into clearly specified roles, and devices performing such roles play a more important function in the PN's self-organization. Details on the different roles their associated tasks will be provided later in the thesis.

2.4.2 Two categories of devices

Broadly speaking we classify PN devices into two categories. The first category includes multifunctional and relatively powerful devices like laptops, PDAs and mobile phones. The second category contains more resource constrained devices with small form-factors which were designed to perform a specialized function e.g. Bluetooth headsets, digital watches and bio-medical sensors. We design our security architecture to deal with this heterogeneity of personal devices in a constructive way. Specifically, we off-load the additional overhead of management to the first category of devices thus requiring a significantly reduced set of minimal functionality necessary for a device to be PN *capable*. Furthermore our security mechanisms are designed with the awareness that the second category of devices is more prone to loss so their compromise should have a low impact on the security of the system.

Prospective cluster members authenticate with a centralized entity within the cluster that they wish to join. We call this entity the **security agent**. Our model requires each cluster to have one device functioning as a security agent, however there is no restriction to any other role that such a device may serve. In terms of security agent functionality we define two new classifications of devices. Some devices have the capabilities to function as security agents and others do not. When not connected to other devices a **Security Agent Capable** (**SAC**) device will function as a security agent and is thus considered a special form of a cluster which only contains itself. **Security Agent Incapable** (**SAI**) devices, unable to act as security agents, do not constitute a cluster when alone and need to join existing clusters. We expect SAI devices to be less sophisticated and typically not useful by themselves. They are designed to be used in conjunction with other smarter devices, when networked together as a cluster.

2.4.3 Cryptographic material

Although the cluster is established based on bilateral trust between cluster members and the security agent, secure communication within the cluster takes place using a periodically updated group key generated by the security agent. Since this group key is only known to members of the cluster, we call it the **cluster key**. In order to limit accidental or malicious attack capability of cluster members, control traffic generated by the security agent is secured using an efficient and reliable broadcast authentication mechanism based on TESLA [19]. The types of control

traffic protected using this mechanism include cluster key updates and also **cluster advertisements**. Cluster advertisements are periodically generated by the security agent of each cluster and serve to alert internal cluster members to the continued presence of the security agent and the outside world to the presence of the cluster. Later we will see how the cluster advertisements play an important part in the self organization of the cluster and also in enabling of local inter-PN communication.

This means that there are in fact three types of cryptographic material on each clustered device. One long term that is used to authenticate with the security agent in order to join the cluster. The other two being short term, that are used to secure the integrity and confidentiality of intra-cluster data and control traffic respectively in order to prevent unauthorized access by both internal cluster members and also outsiders.

2.4.4 Link layer security for intra-cluster communication

Given that we have chosen to enforce security by making the perimeter of the PN secure, clustered devices use the cluster key to perform link layer verification of all intra-cluster communication. Note that unless stated otherwise, we use the term link layer security to actually mean security at Layer 2.5 (between the network and link layers). In order to have perfect forward secrecy (PFS) for intra-cluster communication we limit *direct* use of the cluster key by using two derived keys for securing intra-cluster communication. This is because cluster key updates are distributed after encrypting them with the existing cluster key. Thus we limit further exposure of the cluster key to cryptanalysis. PFS works on the premise that a key used for the transfer of data cannot be used to derive any keys for future transmission.

All intra-cluster traffic is appended with a MAC (message authentication code) for integrity protection with optionally encrypted for confidentiality. Any subsequent cluster member receiving the traffic can verify that the traffic was generated by a trusted device and not modified in transit by any un-trusted device. For traffic routed over multiple hops, the MAC is checked at each hop[1], thus ensuring that any packet without a valid MAC is immediately dropped.

Message integrity

Clustered devices append a MAC calculated using the key K_{mac} to all intra-cluster traffic that they generate. K_{mac} is generated from the cluster key using a globally known one-way hash function. This allows us to maintain forward secrecy because even if K_{mac} is compromised it cannot be used to decode the next cluster key. Similarly, backward secrecy is also guaranteed since only the security of the current

[1] Since the IP packet is not modified, the MAC can be reused. This implies that the hop-count field of the IP packet is not used in calculating the MAC.

session is compromised. This is illustrated in Figure 2.1 where we see a cluster key of one period being used to encrypt the cluster key for the next period prior to distribution. Details of our cluster key update mechanism, including the distribution of the encrypted cluster keys using reliable and efficient broadcast authentication are available in Section 3.7.

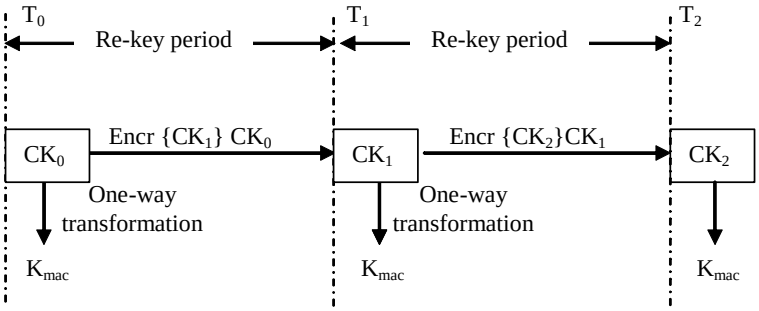

Figure 2.1: K_{mac} generation

It is important to note that as appending the MAC increases the original packet size and thus the communication overhead, it should not be too large. Conventional security protocols use overly conservative security parameters having MAC sizes of 8 or 16 bytes [14]. Although a larger MAC reduces the chance of an adversary blindly guessing the appropriate code, it correspondingly costs more to transmit. For example with a 16 byte MAC an adversary has a 2^{-128} chance of forging the MAC but with a 4 byte MAC this is reduced to 2^{-32}. Reference [14] validates using a 4 byte MAC and explains how it is not detrimental in the context of low bandwidth wireless links.

Message confidentiality

Similarly to K_{mac} the encryption key K_{encr} is also derived from the cluster key using another globally known one-way transformation. However since encryption/decryption consumes power it is difficult to justify confidentiality as a requirement for all traffic. Unfortunately in the absence of some signaling mechanism to enable or disable link layer encryption dynamically (the specification of which is beyond the scope of this document) at this moment all intra-cluster traffic is also encrypted for confidentiality. This means that all communication within the PN, including control traffic belonging to routing and service discovery is automatically secured. In Section 1.3.3 we have explained how hardware implementations of symmetric ciphers can be used to significantly reduce energy usage (and are rather cost effective given their frequent use).

Figure 2.2 illustrates how intra-cluster communication is protected inside a OSI Layer 2 frame. Even though the Layer 2 header cannot be encrypted since the source and destination addresses need to be clear text for transmission and reception, the

MAC is calculated over all the immutable fields thus assuring the integrity of the source and destination addresses.

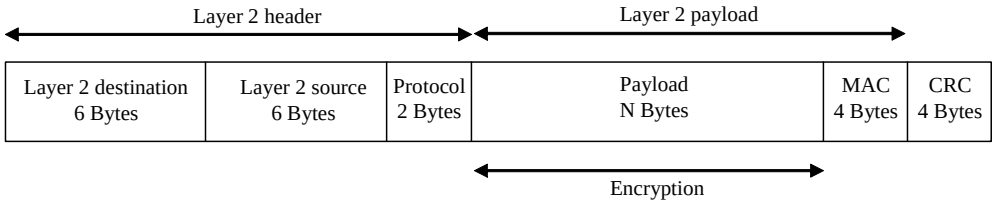

Figure 2.2: A Layer 2 intra-cluster frame

Lastly, we would like to point out that although evaluating the specific cryptographic algorithms for securing communication amongst constrained devices is important, it is *not* a part of our research. This is because a lot of research has already been done in this area in the context of resource constrained sensor networks [15] [25] [26], and results from such studies are directly relevant for us because of similarities in the constrained nature of devices.

The PN Layer

For PN devices, we envision a new **PN Layer** existing between Layer 2 and 3 of the IP stack (see Figure 2.3). This layer enforces the link layer security mechanisms introduced earlier for enabling secure intra-cluster communication.

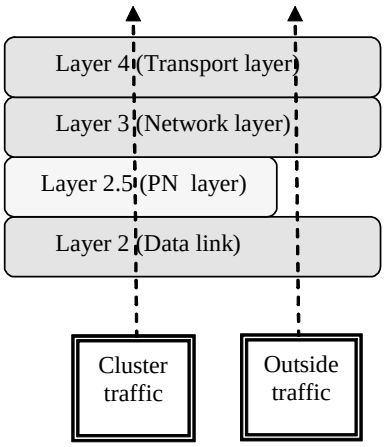

Figure 2.3: IP stack showing the new PN layer

Although it is possible to avoid the use of a new PN layer by mapping the cluster key to specific radio technology keys e.g. a 128 bit link Bluetooth key [106], using a radio technology independent protocol common to all PN devices allows us to have a more consistent level of security (more details in Chapter 7). The PN layer can

also be used to support mobility, routing or optimized radio interface management, however details on such functionality is beyond the scope of this document.

2.4.5 The PN identifier

We imagine the world to be full of wireless devices belonging to a multitude of users. In order to enable interaction between personal devices, we need an efficient mechanism to allow them to distinguish between clusters belonging to their own PN from all the rest. We certainly do not want devices from one user continuously trying to connect to a cluster of another user and failing, thus wasting precious energy in the process. We propose that during initialization of a new PN the security manager generates a **PN identifier** which is later given to all personal devices during personalization. The PN identifier should be calculated randomly over a sufficiently large space so as to reduce the possibility of collisions. Since the PN identifier is included in each cluster advertisement, receiving devices only attempt to authenticate with clusters with a matching PN identifier.

2.4.6 Cluster policy

Each security agent maintains a cluster policy which contains the configuration parameters that all cluster members must implement. This includes parameters such as the periodicity of cluster advertisements and the periodicity of cluster key updates. Using a cluster policy instead of fixed default parameters allows the owner of a PN can configure each of his clusters to behave differently if so required. A copy of the policy is given to each device when it joins the cluster, and cluster members are alerted to modifications by setting a field inside cluster advertisements. Members can then contact the security agent and update their local copy of the cluster policy.

2.5 Concluding remarks

In this chapter we have provided an overview of the terminology, entities and concepts used in our model. We have introduced the new roles of the security manager and the security agent. We have explained how personal devices share a concept of long term trust. Leveraging on this trust, co-located personal devices use cluster advertisements to discover neighboring clusters and authenticate with their security agents. Given that autonomous nature of such clusters, intra-cluster security is enforced by distinguishing between cluster members and non-members based on their knowledge of a shared group key. This key, together with a new PN layer is used to enforce the integrity and confidentiality of intra-cluster communication.

Chapter 3

Securing personal clusters

3.1 Introduction

In this chapter we present our proposals [117][118][121] for the secure formation, communication and self-organization of personal clusters. Given that we expect all personal devices to belong to one trust domain, we begin by looking at the problem of initial key distribution between previously un-associated personal devices. As we stated earlier, classic network security mechanism typically create trust between a group of devices by utilizing online access to a trusted third party. Such mechanisms are not suitable given the distributed and ad-hoc nature of PNs. Trust between PN devices must be created *a-priori* by preloading devices with cryptographic information such that, after deployment, they are able to form a secure PN.

Our scheme must be able to work despite the un-predictability attributable to device mobility and should allow new personal devices to be added to the PN at any time in the future. A very important constraint is that all mechanisms must be implementable on devices with limited hardware and thus must have low computational and communication overhead. Lastly, our proposed mechanisms must fully support the user centric concept of PNs, with the owner in complete control of all his personal devices.

3.2 Overview of related security models

Reference [23] lists the existing security models used to offer security services to members of a long term ad-hoc community such as a Personal Network. However since no single model is sufficient for us on its own, our security framework is based on a combination of three existing models.

3.2.1 Pre-shared common data model

In the pre-shared common data model each member of the community knows a shared secret. We propose that all personal devices that are part of a cluster use a shared secret (called the cluster key) for securing intra-cluster communication. Each personal cluster has a unique and periodically refreshed cluster key which is only used to secure communication between cluster members. The use of a shared key is often rejected in spite of being very energy efficient because compromise of a single member affects the security of the entire community. The shared nature of the key also means that it is difficult to uniquely identify a malicious member. In Section 3.4.1 we will defend our choice of using a shared key for secure intra-cluster communication.

3.2.2 Pre-shared derived data model

In the pre-shared derived data model each community member can be uniquely identified by a seed derived for it by another entity. The use of a pre-shared derived data model using an online-verifiable mechanism such as Kerberos [27] is often rejected in ad-hoc networks because such mechanisms require access to an online entity at the time of communication. The dynamic nature of the PN means that on-demand access to such an entity cannot be guaranteed. On the other hand, the offline-verifiable mechanisms belonging to this model are based on using digital certificates that are signed by the public key of a trusted third party. The personal CA [28] concept developed in the IST SHAMAN project [29] is a good example of how an offline-verifiable mechanism based on digital certificates can be scaled to a personal environment like a PN. Unfortunately we have had to reject any mechanisms based on asymmetric cryptography due to the constrained nature of many PN devices.

Given that PNs have different characteristics when compared with existing multi-user networks, we realize that security solutions designed for the latter are often unsuitable in their *current* form. Therefore we propose using an offline-verifiable mechanism based on symmetric key cryptography, which we call the **Personal TGS** (i.e. a personal ticket granting server). The Personal TGS concept leverages the fact that devices in a PN belong to a relatively small trust domain with long term trust relationships. Such an assumption cannot be made in a typical multi-user environment which is potentially larger and much more dynamic. These differences allow us to propose certain modifications to the Needham Schroeder protocol [31], which is based on symmetric cryptography and uses third party referral in the form of tickets to uniquely authenticate devices. We propose pre-loading devices with tickets (which are valid for an extended duration) during personalization so that they do not have to request them later from the Personal TGS. This enables us to use the Personal TGS in an offline manner, in other words it does not need to be online at the time of communication. More details are available in Section 3.3.3.

3.2.3 Resurrecting duckling model

The resurrecting duckling model [21] ensures the security of a transient association between a mother duck (master) and a duckling (slave), in the absence of an online authentication server. The association is called transient because it does not have to be permanent i.e. it can be broken anytime in the future. This secure association it built when the "mother" device **imprints** the "duckling" device by transferring a secret key using physical contact (in order to ensure the authenticity and confidentiality of the secret key). With reference to ethology, imprinting is the phenomenon which makes a duckling emerging from an egg choose the first seen animated object as its mother.

Although an imprinted duckling is controlled by the mother, it is free to interact with others. A duckling in the imprinted state remains linked to its mother till the latter breaks the relationship between them, at which point the duckling enters the imprintable state again. Hence the term "resurrecting" duckling. Predictably, an imprinted duckling cannot be re-imprinted until it is released by its mother.

When considering how the PN is an extension of the person it is important to represent the trust model associated with PNs using human understandable primitives. We feel that the concept of imprinting suits our requirements for personalization rather well. We have already stated that our proposed mechanisms must fully support the user centric concept of PNs, with the owner in complete control of all his personal devices. We envision that each new PN device goes through a simple owner initiated imprinting process, at which point it becomes inherently linked to the owner who is the only person able to perform administrative/configuration tasks. In our model the role of the mother is performed by the security manager which functions as the cyber representative of the user. If the owner ever wishes to sell a personal device, he only needs to use his security manager to break the link between the device and his PN, at which point the device is ready to be re-imprinted by its next owner.

3.3 Personalization

When a new device is configured to become part of an existing PN, we say that the new device has been personalized. Personalization consists creating a bond between the new device and its owner (through imprinting) and between the new device and other PN devices (through Personal TGS tickets).

3.3.1 Security Manager

The security manager is the cyber-representative of the user within the Personal Network. Each personal device needs to be personalized with the security manager

before it can function as part of the PN. This means that the first step in creating a Personal Network is to initialize its security manager.

As a consequence of imprinting, the security manager maintains a database of secret keys, one for each personal device. This allows the security manager to perform special administrative tasks on personal devices, such as device configuration and revocation. If the security manager is lost or unavailable, such tasks are no longer possible. Therefore it is important that the owner of the PN maintains a backup of its secret keys and in case the security manager breaks down they can be restored to a new device. However if the security manager gets lost or compromised these keys must be used to un-imprint all personal devices, followed by re-imprinting with a new security manager.

Since the security manager also functions as the Personal TGS, new devices are pre-loaded with necessary tickets during personalization. These tickets include the device's ID and capabilities, and the key to be used to secure communication, all protected using the destination device's secret key. The aim is to enable the new device to authenticate with existing devices (details in Section 3.3.4). Automatically transferring tickets after imprinting completes means that it does not constitute an additional burden on the user. The security manager is able to issue tickets because it has a special security association with all PN devices that is created as a consequence of the imprinting process. After personalization the new device is able take part in secure PN formation without requiring any further user intervention.

For purposes of mobility we envision the cryptographic material representing the security manager existing on a password protected smart card carried by the user. Any P-PAN device with a suitable smart card reader will be able to function as the security manager on-demand after insertion of this card.

3.3.2 Creating trust for imprinting

As part of their resurrecting ducking model Stajano and Anderson [21] propose establishing trust between two devices using physical contact to transfer the secret key in plain text. This is later formalized by Balfanz et al. in [32] as a location-limited side channel that ensures the secrecy of communication.

Location limited side channels (LLSC)

Location-limited side channels are separate from the main wireless link, and have the security properties of demonstrative identification and authenticity. They may or may not be able to ensure the secrecy of communication. Demonstrative identification means that the user is able to identify the communicating entity based on physical context e.g. the PDA in front of him. The second property of authenticity ensures that it is not possible for an attacker to transmit data in the location-limited side channel without being detected by the legitimate participants. Both of these

properties are based on the inherent physical limitation of the communication technology being used, for example direct physical contact or directional channels such as infrared. The optional property of ensuring secrecy of communication is a much more restrictive requirement because it means that the location limited side channel must also be resistant to passive attacks.

Imprinting constrained devices

Given our focus on the many resource constrained devices that make up the Personal Network, it is important that our imprinting mechanisms are suitable in this context. Unfortunately all devices, especially the more constrained variety like Bluetooth headsets, do not generally come with suitable location-limited side channels over which the security manager can "push" a symmetric key in plain text. To the extent that such devices may not even come with a suitable user interfaces allowing the PN owner to himself function as the LLSC.

We therefore propose an imprinting mechanism in which the security manager is provided an initialization key which it uses to create trust during imprinting. To support our proposal we give an example of a sensor that comes from the factory pre-configured with an initialization key. The user would have access to this key in the form of a string of text, an RFID or a bar code that comes in the box. During imprinting this initialization key is pushed to the security manager with user assistance, for example, the user takes a picture of the bar code with the built camera of his security manager. From this bar code the security manager can automatically extract the initialization key and the link-layer address of the new device. Now, at any time in the future the security manager can authenticate itself to, and securely imprint the sensor over the insecure wireless channel without requiring the sensor to have a suitable location-limited side channel. Note that we do not use the initialization key of the sensor as the long term key with the security manager, because the PN owner would like such keys to be secret even from the manufacturer of the device.

Since this mechanism does not require the user to position the security manager and the device to be imprinted according to the physical requirements of the location-limited side channel, it is even more useful when imprinting a large amount of sensors. Imagine a scenario where a user buys a house which has been pre-deployed with a number of sensors. He would be given a list of bar codes corresponding to the deployed sensors and only needs to scan the bar codes with his security manager. The remainder of the imprinting process, where the security manager needs to transfer the necessary credentials to the deployed sensors, will be done automatically over the insecure wireless link as the user walks around the house. The protocol specification is available in 3.3.4.

Since the heterogeneity of devices in a PN means that some personal devices may have multiple interfaces available for imprinting, we can generalize our imprinting

model to include more capable devices. If the device to be imprinted has an interface where a secret key can be pushed, then the security manager can choose to use it instead.

Access control

Although we do not implement access control for PN devices based on their unique identity when they access PN services, administrative/configuration tasks can only be performed by specifically authorized PN devices. In [22] Stajano extends the original resurrecting duckling model by proposing the use of security policies to handle more than one-to-one relations. This allows the security manager to control the behavior of a device by uploading different policies to it, for example, specifying one or two other frequently used devices that are allowed to perform limited administrative tasks. This would typically be enforced using Access Control Lists (ALCs), where the list of authorized devices is initially configured during personalization and modified by the security manager whenever necessary.

3.3.3 Creating trust between personal devices

Pre-Authentication

At the end of the imprinting phase, we have a trust model shaped like a star, with the security manager at the center having a security association with all personal devices. However since personal devices also need to communicate with each other, this model as it stands in not enough. In order for two PN devices to interact they require an existing security association (based on symmetric cryptography), the creation of which precedes the actual communication. When one thinks of symmetric cryptography one usually thinks of creating pair-wise keys. Unfortunately pre-distributing pair-wise keys during personalization is not possible because it does not easily allow new devices to be added to a pre-existing network since existing devices will not have the new devices' keys.

Tickets

Our selected approach for enabling mutual authentication between two previously un-associated personal devices is built on third party referral and uses as its basis the Needham-Schroeder protocol [31]. This protocol is based on symmetric cryptography and uses tickets to prove the identity of authenticating devices. In our model, the trusted third party responsible for issuing tickets is the security manager. It is able to issue such tickets because it maintains a database of secret keys (one for each device belonging to the PN) created during the imprinting process.

It is important to point out that our proposal of the Personal TGS, although

a novel twist (i.e. the concept of pre-loading tickets) of the Needham Schroeder protocol, is only useful for scenarios in which communication is taking place between a limited number of devices that have pre-defined long term trust relationships. The trust relationships must be pre-defined because the tickets are pre-loaded in advance of the actual communication. They must also be long term due to the increased overhead of revocation since the pre-loaded tickets require long term validity. Such an approach would not be suitable for large scale networks with many users because such networks require the flexibility to authorize temporary access to services.

Personal devices are pre-loaded with all necessary tickets during the personalization phase and under normal circumstances will never contact the security manager for more tickets. Each ticket is only used once, as the resulting key is stored for future authentications. An unintentional benefit of this scheme is that since personal devices no longer need to request tickets from a third party before beginning the actual key negotiation, it reduces both latency and energy usage.

Unlike a certificate, a given ticket is only valid for authentication with one device. This means that in a network where a device may wish to be able to authenticate with any other device, it will have to be loaded with as many tickets as there are devices in the network. In order to avoid this and keep the storage overhead low, devices are only loaded with tickets needed to form clusters and the tunnels that interconnect different clusters. In other words, resource constrained SAI devices are only loaded with tickets needed to authenticate with SAC devices for forming clusters. Since only a fraction of devices in a typical Personal Network will be SAC, this significantly reduces the storage overhead for constrained devices. This optimization is based on our assumption that SAI devices being less sophisticated (typically without suitable user interfaces) are not useful by themselves but only when networked with other smarter devices in a cluster. Once part of a cluster they can communicate securely with fellow cluster members using the cluster key and members of remote clusters through their cluster gateways.

SAC devices will of course need tickets for all other devices in the PN. This is because SAC devices when functioning as security agents should be able to authenticate all SAI devices joining their cluster and also other SAC devices when two co-located clusters merge into one. However we do not consider this a weakness because SAC devices like PDAs and cellular phones are not resource constrained and have sufficient storage capabilities. Personal devices capable of functioning as gateways will also need tickets corresponding to other PN gateways. This is because inter-cluster communication is secured by creating cryptographic tunnels between the gateways of different clusters.

Personal TGS

For a more inclusive explanation of the Personal TGS we compare it to the Needham Schroeder protocol and it's most well known and featured derivative, Kerberos [27].

We assume that the reader is familiar with both of them. When comparing with the Needham Schroeder protocol, we do not need to use N_A in step 1 and 2 of Appendix A. This is because the keys generated by the Needham Schroeder protocol are session keys that need to be periodically refreshed, whereas in our case they are long term keys used for authentication. The nonce is meant to protect against replay attacks where an attacker replays step 2 with the old session key in order to increase the amount of cryptographic material available to him. Also since the keys generated are long term, Dorothy Denning and Giovanni Sacco's [33] recommendations about time stamps in the Needham Schroeder protocol are not necessary. Lastly, in Appendix A we see that B is authenticating A, but not vice versa. Therefore we use an extra nonce when the ticket is first sent to the authenticator to ensure mutual authentication. When comparing the Personal TGS concept with Kerberos, the main differences are:

- We do not use an online TGS. All necessary tickets are transferred in the personalization phase.
- We use tickets to create long term keys that are used for authentication. Kerberos tickets generate session keys used to protect data.
- Kerberos uses time stamps to create authenticators when sending a TGT (ticket granting ticket) to a TGS. We do not need authenticators since all tickets are transferred during personalization.
- Kerberos also used timestamps for authentication, whereas we must use a challenge response mechanism since our tickets are valid long term. This has an advantage of not requiring PN devices to be time synchronized with each other and the TGS.

The main disadvantage of the offline Personal TGS is in the difficulty of performing timely revocation of personal devices. Details on the revocation process are available in Section 3.11.

3.3.4 Protocol Specification

In the next section we present the protocol specification for personalization, divided into two phases. The first phase is pre-personalization and requires user assistance. The second phase is the actual personalization process during which all relevant information is transferred to the new device. Note that the second phase can take place any time in the future after the first.

Personalization

Device B is new SAI device being introduced into a Personal Network that contains a SAC device A, and a security manager M.

MAC_B	Link layer address of B
K_{init}	Initialization key
ID_B	Device ID of B
N_B	Nonce generated by B
N_M	Nonce generated by M
dec	Decrement operation
$hash$	One-way hash operation
C_B	List of B's capabilities (SAC/SAI/GW)
P_B	PN policy (default configuration parameters)
K_{AB}	Long term key between A and B
K_{AM}	Long term key between A and M
K_{BM}	Long term key between B and M
PN_{ID}	Personal Network identifier

Phase 1 (User assisted over a secure channel)

$1. User \Longrightarrow M : MAC_B, K_{init}$

Phase 2 (Insecure main wireless channel)

$2. M \Longrightarrow B : \{ID_M, N_M\}K_{init}, hash(ID_M)$

$3. B \Longrightarrow M : \{dec(N_M), N_B, C_B\}K_{init}$

$4. M \Longrightarrow B : \{dec(N_B), ID_B, P_B, K_{BM}, PN_{ID}, hash(tickets)\}K_{init},$
$\{ID_B, K_{AB}, C_B\}K_{AM}, \{ID_A, K_{AB}, C_A\}K_{BM}$

The first phase is performed by the user when he loads the security manager with the link layer address and initialization key of the device to be personalized. In the second phase the security manager sends its ID and a challenge to B, protected by the initialization key (step 2). Sending a hash of already encrypted information allows B to verify that M knows its initializing key before B transmits any replies (as ID_M is also sent encrypted with the initialization key). This is because transmitting data is a much more expensive operation and we want to protect B from denial of service attacks. Note that sending the hash in step 2 is not meant for integrity protection, since modifictions of messages in step 2, 3 and part of 4 protected with K_{init} will cause authentication to fail. The integrity of tickets in step 4 is thus protected by including a hash of them inside the part of the message encrypted with K_{init}.

In step 3, B responds to the challenge and sends one of its own for mutual authentication. It also sends a list of capabilities based on which, the security

manager transfers all relevant tickets in step 4. In the example given the security manager is only transferring one ticket, corresponding to A. The ticket also includes the capabilities of A. This ensures for example that an attacker is not able to transfer compromised credentials from an SAI device to his own SAC device in order to hijack a cluster. Besides tickets the security manager also imprints B with its new ID, the shared key, the PN identifier and the default PN configuration parameters. These default configuration parameters can be modified anytime in the future using either:

- A remote configuration service based on imprinted credentials
- Manual configuration by the user on devices with an appropriate user interface

In the next section we continue this example and see how the tickets are actually used for the very first authentication between two personal devices.

Authentication using Personal TGS tickets

The very first authentication between two personal devices utilizes the tickets transferred during personalization. A successful authentication results in the creation of a security association based on the resultant long term key. However the ticket used in the initial authentication can reside on either of the two devices, depending on the order in which they were personalized. Next we look at two examples of authentication, one for each scenario.

Example 1: Device B begins authentication with device A after receiving its cluster advertisement i.e. the ticket necessary for authentication is with device B. In step 2 by only returning its ID, A is indicating to B that it does not have a valid ticket corresponding to B.

$1.B \Longrightarrow A : ID_B$
$2.A \Longrightarrow B : ID_A$
$3.B \Longrightarrow A : \{ID_B, K_{AB}, C_B\}K_{AM}, \{N_B\}K_{AB}$
$4.A \Longrightarrow B : \{dec(N_B), N_A\}K_{AB}$
$5.B \Longrightarrow A : \{dec(N_A)\}K_{AB}$

Example 2: Later a new SAC device C is added to the Personal Network. B is unclustered and begins authentication with C after receiving its cluster advertisement.

$1.B \Longrightarrow C : ID_B$
$2.C \Longrightarrow B : ID_C, \{ID_C, K_{BC}, C_C\}K_{BM}, \{N_C\}K_{BC}$
$3.B \Longrightarrow C : \{dec(N_C), N_B\}K_{BC}$
$4.C \Longrightarrow B : \{dec(N_B)\}K_{BC}$

3.4 The role of the security agent

The security agent represents a centralized entity within each cluster responsible for executing the organizational tasks necessary for the secure formation and secure communication of each cluster. In Chapter 1 we explained why centralizing security thus allows us to keep the minimum functionality required for devices to be PN *capable* at as limited as possible. We chose to have each cluster manage its security locally due to the semi-autonomous nature of PN clusters. The security agent performs management related task, these are advertising the cluster, authenticating new cluster members, storing the cluster policy file, generating and periodically updating the cluster key, facilitating cluster merging and evicting cluster devices. In the next few sections we look at each of these roles in detail.

3.4.1 The cluster key

Fortunately, since devices from a particular PN share the same owner we believe that for purposes of access control it is sufficient for PN devices to demonstrate group membership of a cluster rather than their individual identity. Using a group key (called the cluster key) to verify cluster membership instead of many pair-wise keys significantly improves on system efficiency. This because it reduces the overhead associated with key management including the amount of processing required, the number of messages exchanged, the authentication delay and the storage requirements. In our security model the cluster key is randomly generated by the security agent when it forms a new cluster and is given to new members when they join. It is also periodically refreshed by the security agent and distributed to all existing cluster members. Only devices that know the current cluster key are able to take part in intra-cluster communication.

The use of a group key is often rejected (in spite of being very energy efficient) because compromise of a single member affects the security of the entire group. However, we believe that using a group key for implementing cluster access control is feasible because of:

- The transient nature of the cluster and its cluster key. Consequently the effects of key compromise only last as long as the compromised cluster is intact. Unlike models which use group keys as long-term globally shared keys, for example [17], the cluster key is time-limited and cluster-limited.
- The reduce risk of key compromise from cryptanalysis because of periodic cluster key updates.
- The reduced risk of key compromise from compromised devices because the loss or theft of a personal device is generally easier to detect when compared to large-scale unattended networks like sensor networks.

Since the cluster key is mainly used as a lightweight means of link-layer network access control, the effects of key compromise are limited to (a) access to the set of unsecured personal services and (b) Denial of Service (DoS) attacks against the cluster, for instance, those against the routing protocol. As for the former, more sensitive PN services will typically have additional security at a higher layer of the protocol stack e.g. a password for remote desktop access. Similarly, secure communication with entities outside the PN e.g. credit card transactions, are already protected end-to-end using SSL/TLS [34] so are not directly affected by a compromised cluster key.

Alternatives to a group (cluster) key would be pair-wise or sub-group keys, both of which increase the overhead of key management, storage and even multi-hop communication (because of repeated decryption and encryption for packets traveling multiple hops). Given our constraints such alternatives have to be rejected.

Although we use group authentication for increasing efficiency, we appreciate that cluster key compromise can be problematic since there is no easy way to uniquely identify the source of any attacks. Unfortunately there are no ideal solutions for source authentication in group communication, especially in the case of multiple senders. TESLA [19] provide secure broadcast authentication using symmetric cryptography by delaying the disclosure of authenticable secret keys, however using it in scenarios with multiple senders has an unacceptably large overhead. Normally public key cryptography would be a good alternative but it is too costly to generate and verify digital signatures on every packet. There exist other schemes with the source authenticity allowed by public key cryptography but without the performance penalty [35] [36] however they each have their drawbacks. We believe that our decision to use a group key to secure intra-cluster communication is a good compromise between security and performance.

3.4.2 Cluster advertisements

Clustering, the process by which all PN devices within each other's transmission range connect to form a cluster is an integral part of our vision for a PN. Therefore one aim of all personal devices is to discover others around them that belong to their own PN. In our model this discovery is done by listening for what are called cluster advertisements. Clusters advertise themselves proactively by periodically broadcasting cluster advertisements. PN devices listen for such advertisements in order to discover personal and also foreign clusters within their communication range.

Only clusters are allowed to advertise, while un-clustered devices periodically wake up to listen for such advertisements. As mentioned before, SAI devices are envisioned to operate only as part of clusters and to be less sophisticated. These devices can be left powered up for extended periods either purposely or mistakenly by the owner of the PN. We do not want such otherwise idle devices to waste precious energy continuously advertising their existence to non-existing neighbors. Cluster

members on the other hand, as shown by their interconnected state, are more active in nature. Therefore the onus of generating advertisements is on them.

Cluster advertisements are periodically generated and broadcasted by the security agent. Depending on the cluster policy (Section 2.4.6) and local resource constraints, other cluster members may re-broadcast non-duplicate advertisements on all or selected radio interfaces. Not only does this allow the advertisements to reach devices who do not share a radio interface with the security agent, it also extends the transmission range of the advertisements by enabling them to propagate to devices that are only within the range of peripheral cluster members. Besides allowing outside devices to discover the existence of the cluster, cluster advertisements also update existing cluster members to the continued presence of their security agent. This second aspect plays an important role in the self organization of the cluster and is discussed in detail in Section 3.5.

When an un-clustered device receives a cluster advertisement from a cluster belonging to its own PN it attempts to join that cluster. However this can only happen after a successful authentication with the security agent of the cluster it is trying to join. Similarly when two clusters of the same PN come within each other's transmission range, they also attempt to merge to form one cluster. However in this second case which we call **cluster merging**, the authentication only takes place between the two security agents. Details on cluster merging are available in Section 3.9. Note that the decision to merge or not depends on the user preferences configured at each security agent, for instance, the user may not want his P-PAN cluster to merge with another personal cluster until the two clusters have been co-located for a certain minimum duration.

Figure 3.1 illustrates the format of a cluster advertisement without any extension headers. Extension headers are used to add control information to cluster advertisements and are identified by a distinct next header value. Although extension headers are protected for confidentiality, unless a cluster wishes to remain anonymous the cluster advertisement itself is not encrypted. We have defined three types of extension headers: *Re-Key*, *Merge* and *Hash disclosure*. Later we will look at each of them in more detail.

The *Sequence number* field shown in Figure 3.1 is incremented for each new cluster advertisement. It is used by receiving devices to determine, and drop duplicates. A cluster advertisement may carry zero, one, or more extension headers each identified by the *Next header* field of the preceding header. The *EAP pass-through* field lists the capabilities of the advertising device to function as an EAP pass-through during a subsequent authentication attempt by an outside device (refer to Section 3.8 for more details). The *Flags* field is used by the security agent to indicate a change in status, for instance a modification of the cluster policy. In such a case cluster members can update their local copy of the cluster policy after some short random delay. The *TTL* (time-to-live) field is used to control the maximum number of times a cluster advertisements can be re-broadcasted. The *PN ID* field has

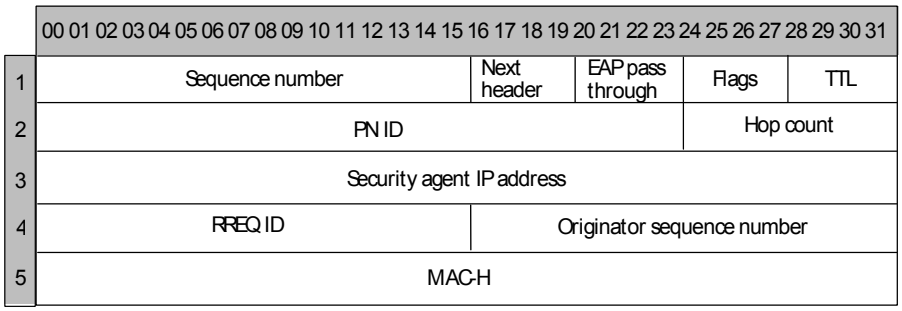

Figure 3.1: A cluster advertisment without extention headers

already been explained in Section 2.4.5.

Since we selected the Ad Hoc On-demand Distance Vector (AODV) routing protocol [39] as the intra-cluster routing protocol in our simulations (see Chapter 4) , the *Hop count*, *RREQ ID* and *Originator sequence number* fields of a cluster advertisement are an optimization used to reduce the path discovery overhead. They have the same functionality as their namesakes in the AODV *RREQ* packets and are used by forwarding cluster members to create temporary routing table entries to store reverse paths towards the security agent. Given the central role played by the security agent, for example during authentication for cluster access this is quite useful because it significantly reduces the overhead of discovering a path to the security agent. Lastly, the *MAC-H* field is used to enable the source authentication of control traffic so that compromise of a cluster key cannot in itself be used to generate verifiable cluster advertisements. Detail on our mechanisms for source authentication of control traffic are available in Section 3.6.

3.4.3 Distributing cluster advertisements

Using a multicast tree for distributing a large number of messages to devices in an ad hoc network is more efficient than blind flooding [37], since it ensure that each device receives the message only once. However because constructing and maintaining such a delivery infrastructure has an overhead and we are only disseminating low volume *control* messages, additional work is required to evaluate its suitability for disseminating cluster advertisements. This is especially true because unlike a typical multicast delivery tree where messages are only meant for tree members, cluster advertisements are also meant for outside devices that may wish to become members. Since we wish clusters to extend with devices that are within the range of even peripheral cluster members, our broadcast routing protocol should be optimized for advertisements by peripheral members. However at this moment we use blind flooding due to its simplicity, robustness and the fact that it gives us a good idea of the worst case overhead for disseminating cluster advertisements.

3.5 Self Organization

The mobile nature of personal devices and clusters has a profound impact on the design of underlying mechanisms which support the creation of the Personal Network. We have explained how devices listen for cluster advertisements in order to discover clusters within their transmission range. However cluster advertisements do more than just advertise the existence of a cluster to non-members, they also let cluster members know that their cluster is **alive**. A cluster that is alive has a functioning security agent and can therefore grow by adding new members. Conversely, a **zombie** cluster is one that has lost its security agent (but has a valid cluster key). Devices belonging to a zombie cluster can still communicate securely with each other until the existing cluster key expires, but the cluster cannot grow because there is no security agent to authenticate new members. Zombie clusters can only be resuscitated by the return and the resulting cluster advertisement generated by the original security agent.

Figure 3.2 illustrates the state transitions of SAI and SAC types of devices. We see that SAI devices start out in the *orphan* state, whereas SAC devices start out in the *security agent* state. Devices in both states are ready to authenticate with other personal devices in order to join or expand the cluster respectively. After authenticating with an existing cluster, SAI devices move from the *orphan* state into the *clustered* state, as members of the expanded cluster (step 1). However when two personal clusters merge, one of the security agents enters the *merging* state (step 6) where it updates its cluster members with the cluster policy and other relevant parameters of the cluster being merged with. Once this process completes it moves into the *clustered* state (step 8), as a member of the two merged clusters. The other security agent will remain as the security agent of the two merged clusters (step 7). Details about the procedure for cluster merging are available in Section 3.9.

In order for clustered devices to discover the departure of their security agent, we define a new parameter called the **redundancy factor** (**RF**). The value of RF corresponds to the number of *sequential* cluster advertisements that must be missed by a clustered device before it believes it has lost connectivity with its security agent. In Chapter 4 we use simulation results to recommend a value for RF. Consequently if a cluster member misses RF consecutive cluster advertisements, it enters a *transition* state (steps 2 and 9). Devices in the *transition* state switch back into the *clustered* state (steps 3 and 10) if they receive a new cluster advertisement from their original security agent. Since devices only enter the *transition* state when they no longer hear cluster advertisements from their security agent they automatically stop advertising their cluster during this blackout period. In this way the system is quite self regulating since no new authentication request are generated by outside devices wishing to join the cluster.

A security agent aware of an impending departure from its cluster (e.g. due to battery depletion) transfers its state to another SAC cluster member. This makes it

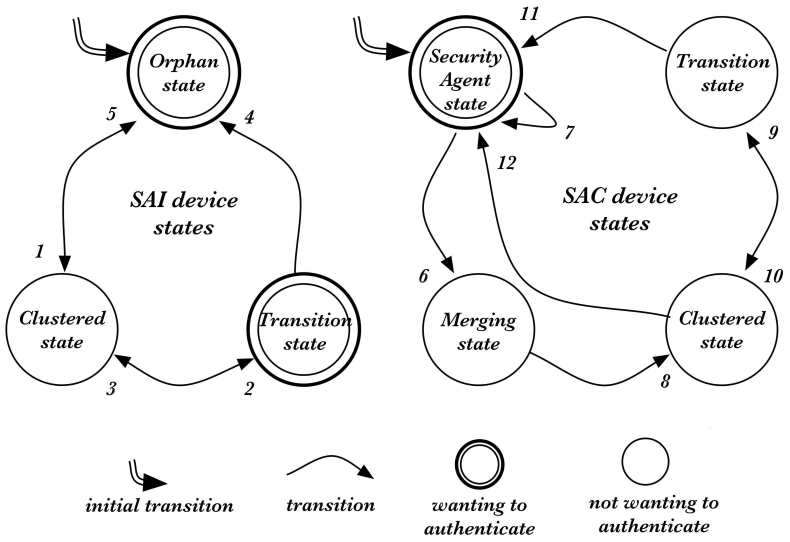

Figure 3.2: State transitions of PN devices

possible to keep the cluster intact even if the original security agent moves away and is more efficient when compared to the amount of traffic which would be generated to re-organize the cluster if the security agent were to depart suddenly. Since cluster formation typically involves the merging of multiple clusters a departing security agent usually has some idea of other SAC cluster members. Even if this is not so, the security agent can broadcast a query message to locate other SAC capable cluster members. Transfer of state is done using secure unicast transmission after locating a suitable candidate.

If the security agent departs its cluster suddenly, then there is no one to update the cluster key and it will eventually expire. Consequently cluster members in the *transition* state leave the zombie cluster and join the first personal cluster whose advertisement they hear (step 3 and 10). If they are not able to join new clusters before the existing cluster key expires, they can no longer communicate amongst each other and the cluster falls apart automatically. When this happens SAI devices enter the *orphan* state (step 4) where they wait indefinitely to join other clusters while SAC devices form their own clusters (step 11).

For SAI devices the duration of time spent in the *transition* state is limited by the time remaining till the existing cluster key expires. If they do not hear any cluster advertisements at all, they remain part of the existing zombie cluster till the current key timeouts and the cluster falls apart (step 4). On the other hand, SAC devices being less dependent can leave the *transition* state sooner and create their own clusters (step 11) after a certain random delay period.

During the simulations carried out in Chapter 4, we discovered that clustered

devices were able to move from the *clustered* state to the *orphan* and *security agent* states without going through the *transition* state (steps 5 and 12). It turns out that such devices were in the *transition* state during the process of updating the cluster key (refer to Section 3.7) so could not update to the new cluster key. Later they received one or more cluster advertisements so moved back to the *clustered* state just before the existing cluster key expired. For this reason, we propose that cluster members moving from the *transition* state into the *clustered* state should synchronize with the security agent after a short random delay. This would be a much cheaper operation than (re) authentication.

3.5.1 Bootstrapping cluster formation

In this section we give an example of how our proposed mechanisms are used to bootstrap the formation of a cluster. When an SAC device powers up it starts functioning as security agent in a cluster that only contains itself. As a security agent it periodically broadcasts cluster advertisements. SAI devices unable to create their own clusters start out in the *orphan* state. Devices in the *orphan* state do not belong to any cluster so do not have a cluster key. Such devices only know their PN identifier, which was configured during the personalization process.

Although SAI devices do not play an active part in discovery, they do broadcast one *Hello* message as soon as they power up. This speeds up cluster formation because a neighboring cluster member hearing a *Hello* message responds with a cached cluster advertisement.

SAI devices attempt to authenticate with the first cluster of their own PN whose advertisement they receive. After a successful authentication SAI devices are able to join that cluster and take part in intra-cluster communication. Similarly, two clusters within each other's transmission range merge to form one cluster. This means that co-located personal devices will systematically coalesce into a single cluster.

3.6 Source authentication of control messages

Even though all intra-cluster traffic is protected using the cluster key we believe that control messages generated by the security agent should have an additional level of security. Having an extra layer of security for control information ensures that compromise of a cluster key cannot in itself be used to generate verifiable but malicious control traffic. For instance, attackers cannot generate verifiable cluster advertisements or hijack a cluster by updating the cluster key. Older cluster advertisements cannot be replayed because they contain sequence numbers and duplicates are discarded by cluster members.

In terms of distribution, such control traffic can either be sent unicast or broad-

casted for more efficient distribution. Acknowledged unicast exchange secured using the long-term keys existing between the security manager and clustered devices (see Section 3.8.3), is possible but has a high communication overhead. It also requires the security agent to maintain an up-to-date list of all cluster members, which further increases overhead. In our model the security agents authenticated each personal device wishing to join the cluster but does not maintain state about it afterwords. Eventually we rejected the unicast approach due to its inefficiency in utilizing the properties of the broadcast medium. Unfortunately the unreliable nature of broadcast transmission also means that we need some mechanism to ensure the reliable distribution of broadcasted control messages.

3.6.1 Reliable broadcasts

The issue with broadcasting control messages is that there is no easy way for the security agent to ensure that all cluster members have received such messages. We can imagine a solution in which members generate negative acknowledgements, for example, if the cluster key that they are using is about to expire and they have not received the new key. However cluster members cannot send negative acknowledgements if the re-key is being done ahead of schedule, for instance as a consequence of two clusters merging.

Our chosen approach for increasing the reliability of broadcasting control messages is to use redundant transmissions. Since control messages are piggy-backed over cluster advertisements, as per our definition of the redundancy factor (RF) any device that is part of the cluster should receive at least one of the RF sequential cluster advertisements. Thus control information is repeated over RF sequential cluster advertisements. This ability to link different system concepts not only simplifies our design but also reduces the number of system parameters that need to be optimized.

3.6.2 Authenticating control message broadcasts

The asymmetry required by broadcast authentication is typically provided using mechanisms based on asymmetric cryptography. However there exist some mechanisms for broadcast authentication based on symmetric cryptography. Carman et al. acknowledge in [18] that although such mechanisms are attractive due to their low energy and processing requirements, they also have certain limitations. For instance, two well known security protocols SNEP [11] and TESLA [19] provide secure broadcast authentication using symmetric cryptography by delaying the disclosure of authenticable secret keys. Emulating asymmetry using delayed key disclosure requires that each device is time synchronized with the sender, performs certain key management functions and has sufficient buffer capacity. If the number of messages being authenticated in each time period is large, the receiver needs to have sufficient

buffer capacity to store them before the corresponding key is released and they can be authenticated.

We employ μTESLA [11] (Timed, Efficient, Streaming, Loss-tolerant, Authentication) protocol for broadcast authentication of control traffic. μTESLA relies on the same basic structure as TESLA, with only a few minor modifications to lower the overhead for resource constrained devices. μTESLA still utilizes the one way hash chains, loose time synchronization, and delayed key disclosure of TESLA however it does not use any digital signatures, and discloses the key only once per time period while restricting the number of potential senders. The fact is that none of these modifications change the basic design, which is the way that the protocol deals with key disclosure.

In μTESLA each sender chooses the final key, K_N, of a hash chain randomly. This key is then run numerous times through a hash function, F, to compute a one-way key chain. For instance $K_{N-1} = F(K_N)$, $K_{N-2} = F(K_{N-1})$, and so on. Each value in the hash chain becomes a key. μTESLA divides time into equal intervals, assigning a different key to each interval. For instance, key K_i would be assigned to the time interval T_i and each key is only valid for the time interval for which it is assigned. All packets generated by the source within a specific time interval use the key assigned to that interval. Messages are then broadcasted with a MAC generated using the secret key, which will be disclosed at a specific time in the future.

Since every key in the key chain can be obtained from K_N by the public formula $K_i = F^{N-i}(K_N)$, the keys are actually disclosed in the reverse of the direction in which they were generated. Therefore while no one can reverse the hash chain in order to find an unpublished key, all devices are able to apply the function F to a received key to verify if it belongs to the key chain. This of course assumes that receiving devices have a way of obtaining an initial, authentic value of the key chain. In our proposal the authentic hash chain value and other cluster parameters are transferred from the security agent to each new cluster member during the authentication performed prior to joining the cluster (see Section 3.8).

Receivers can verify that the sender is the one who sent the message if the key used to create the MAC could not have been released before the receiver obtains the message. If the key may have been published beforehand then the receiver will drop the packet since another device could have used that key to compute a MAC and spoof the sender. Therefore when the receiver receives a message, it confirms that the key has not been disclosed, and then buffers the message since it does not yet have the key to verify the MAC. When the corresponding key is publicly disclosed the receiver authenticates that it belongs to the correct time slot on the key chain and then uses the key to calculate the MAC on the received packet in order to verify its authenticity. It is important to note that although receiving devices are not required to have perfect clock synchronization with the sender, they must assume a maximum synchronization difference and a pessimistic upper bound on network delay.

The entire cluster advertisement including the extension headers is protected by another MAC (known as MAC-H) using μTESLA. In our architecture, the μTESLA time period is equal to the cluster advertisement period however the two do not perfectly overlap since cluster advertisements are sent towards the middle of μTESLA time intervals. This means that even loose clock synchronization will be suitable since we can assume rather pessimistic values for the synchronization difference and network delay. The corresponding key chain value is released in the next cluster advertisement using the *Hash disclosure* extension header illustrated in Figure 3.3. Even if a key chain corresponding to a buffered message is lost, the receiver can verify such messages based on subsequently released key chain values. Piggybacking control information on top of cluster advertisements not only reduces the overhead of disseminating the control information but also allows us to reuse the distribution framework defined for cluster advertisements. Moreover, having a μTESLA period equal to the cluster advertisement period means that we maximize the length of time each hash chain is valid (by making the μTESLA period as large as possible) and also reduces the number of system parameters that need to be optimized. Finally, although the piggy-backed control messages cannot be authenticated till the corresponding key chain value is released in the next cluster advertisement, external attackers cannot launch DoS attacks by generating false cluster advertisements because all intra-cluster traffic is also protected using the cluster key.

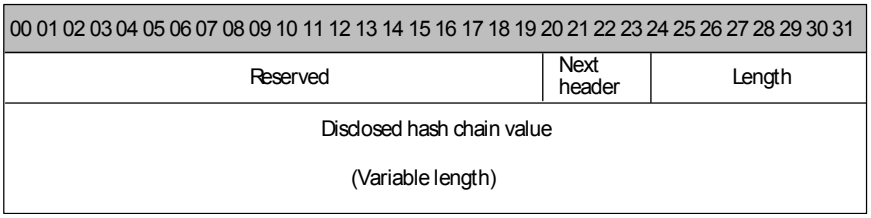

Figure 3.3: The *Hash disclosure* extension header

Suitability for our model

Earlier we stated TESLA's limitations i.e. that it requires each receiving device to have loose time synchronization with the sender, have sufficient buffer capacity and perform key management functions. Here we look at how we propose to overcome these limitations. As for the first limitation, with only one authenticated sender, clustered device only need loose time synchronization with the security agent. This can be done with minimal overhead and complexity by having devices synchronize directly with the security agent after authentication. In this regard, the issue of clock drift must also be kept in mind [38]. However given that personal clusters composed of mobile devices have shorter lifetimes when compared to e.g. static sensor networks, clock drift poses less of a problem. Moreover it may be possible to utilizing the periodic properties of cluster advertisements and use them as time synchronization beacons.

As for the secondly limitation, only securing control messages means that the number of messages being authenticated in each time period is very small and receivers do not need excessive buffer capacities. For the third limitation and also to reduce the energy overhead of cluster wide broadcasts we piggyback control traffic and messages related to TESLA's key management e.g. hash disclosure, on the periodically broadcasted cluster advertisements. Not only does this reduce the overall transmission overhead, it also reduces the number of times devices need to wake up from sleep state.

3.7 Updating the cluster key

Since cluster keys have lifetimes, they must be updated by the security agent before they expire. Updating the cluster key (or re-keying) is also performed after a PN owner expels a revoked cluster member from the cluster or when one cluster has to update its cluster key to that of another cluster during cluster merging. We identify three main issues that need to be resolved before a cluster key can be successfully updated. These are:

1. Given the unreliable nature of wireless transmission, have a mechanism to ensure a reliable distribution of control messages to all cluster members.
2. Ensure that the re-key packets were generated by the security agent.
3. Have a mechanism that allows all cluster members to switch en-masse to the new cluster key while minimizing overhead and packet loss.

The scope of this section is limited to the third issue since we have already described a framework for the first two in Section 3.6. Having an extra layer of security for control information ensures that compromise of a cluster key cannot in itself be used to generate verifiable re-key packets. Figure 3.4 illustrates our proposed re-key mechanism where re-keying starts with the cluster advertisement of sequence number x and the value of the redundancy factor (RF) is i. As per our earlier definition of the redundancy factor, any device that is part of the cluster should receive at least one of RF sequential cluster advertisements (with the *Re-key* extension headers). Once devices have verified that the new key is genuine, they are ready to switch over. However this should be done in a way that no ongoing communication is disrupted.

The Re-key extension header is illustrated in Figure 3.5 and contains the new cluster key (encrypted using the existing cluster key), the validity period of the new cluster key and the **update sequence number**. The *Update sequence number* field of the re-key message holds the sequence number of the cluster advertisement which will signal the switch over to the new cluster key. Devices switch over to the *verified*

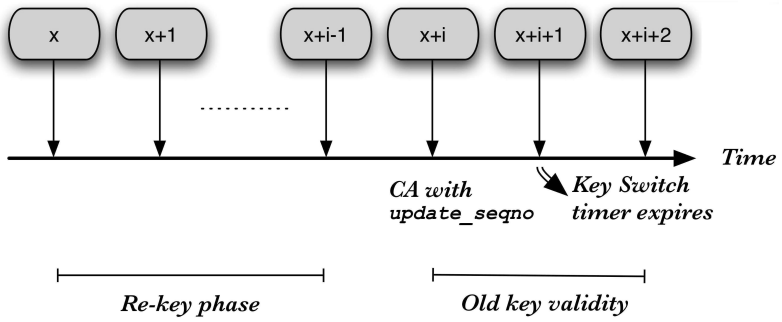

Figure 3.4: The re-key mechanism

cluster key immediately after forwarding this cluster advertisement. This cluster advertisement will enclose the disclosed hash value corresponding to the last repetition of the re-key message. Since the unreliable nature of the broadcast medium means that some devices may not receive this cluster advertisement, we also have two backup mechanisms in place. The first mechanism estimates the time in the future at which the cluster advertisement with the update sequence number should arrive. This is calculated based on the cluster advertisement period and the sequence number of the last verified re-key message. If devices exceed the estimated time plus one cluster advertisement period (to ensure that the message was indeed lost and not delayed), they update their cluster key automatically. This timer (**key switch timer)** is automatically cancelled if the cluster key is updated using other mechanisms. Secondly, if a cluster member waiting for the cluster advertisement with the update sequence number receives a message protected with the new cluster key it updates its cluster key as well. This is because it assumes that the sending device must have received the cluster advertisement with the update sequence number, otherwise it would not have shifted to the new key. However this second mechanism is not enabled when cluster members are re-keying as a result of cluster merging because other devices are already using the new cluster key.

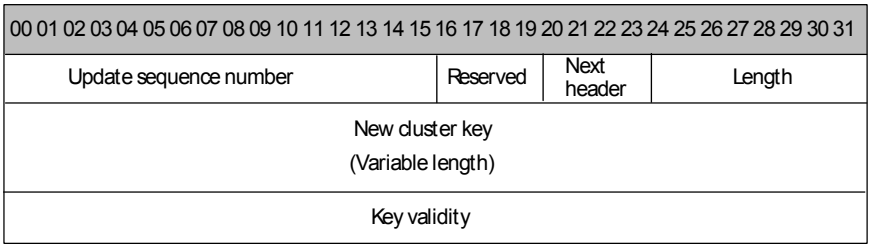

Figure 3.5: The *Re-key* extension header

Finally, since the cluster will not synchronize immediately there will be situations where some members are using the old cluster key and others the new one. Therefore

we define an **old key validity** phase during which cluster members which have updated to the new cluster key also accept packets signed with old key. This phase lasts for two cluster advertisement periods.

3.8 Authentication for cluster access

Earlier we stated that devices wishing to join a cluster must authenticate with its security agent. We also added that members of a cluster only forward authorized traffic with a valid MAC generated using the cluster key. Together, these two statements imply that devices wishing to join a cluster must have a common radio interface with its security agent and be within the transmission range of the security agent. Similarly, for two clusters to be able to merge together the two security agents need to share a common radio interface and be within each other's transmission range.

Such a restriction on the extensibility of the cluster is not very practical. We would like clusters to extend with devices that share a radio interface with any cluster member and also those that are only within the range of peripheral cluster members. To that end our authentication framework for cluster access utilizes the Extensible Authentication Protocol (EAP) [50]. EAP is considered as an authentication framework that can support multiple and even future authentication mechanisms. The EAP protocol is fast becoming an industry standard for network access control. It provides an extensible and transport independent protocol for encapsulating different authentication mechanisms.

In this section we will present details on the integration of the generic authentication mechanism presented in Section 3.3.4 with EAP, to be used as a new EAP authentication method for cluster access. Since this new EAP method uses Personal TGS tickets we call it **EAP-PTGS**. Our approach fits well with the extensible nature of the EAP model which allows the addition of new authentication mechanisms. Specifically this approach allows us to benefit from:

1. Common EAP functionality, especially its integration with the AAA framework [52] which facilitates the use of a back-end authentication server
2. The use of different authentication mechanisms for different types of devices. For instance the use of alternate EAP methods for authenticating foreign devices, as explained in Chapter 6

The remainder of this section is divided into three parts. In the first part we provide an overview of the EAP protocol including details on the format of EAP packets. In the second part we identify and define the different architectural components in our model, as well as the protocols used between them. In the third part we look at the steps necessary for the specification of a new EAP method, and specify the behavior of the different components with respect to how they process EAP-PTGS packets. This includes specifying the EAP-PTGS packet format.

3.8.1 EAP overview

The Extensible Authentication Protocol (EAP) is a client/server protocol that provides an extensible and transport independent mechanism to encapsulate different authentication methods. It is able to select the most suitable authentication method e.g. EAP-TLS [56], EAP-MD5 [50] and EAP-PSK [57] prior to issuing the authentication request by asking for more information from the client. Although it typically runs directly over the link layer, it is also possible to use it over IP.

The EAP authentication framework is based on three entities. With reference to our model, the entity called the **EAP peer** corresponds to the personal device which wishes to authenticate and join the cluster. The second entity known as the **EAP authentication server** is the security agent that verifies the client credentials and is the other end of the EAP communication. The third entity, known as the **EAP authenticator** is the entity which has direct connectivity with the EAP client, and through which the client is authenticating with the server.

The authenticator and the authentication server can be one or two separate entities. If the authenticator is also functioning as the authentication server, all EAP messages are processed locally. Otherwise the authenticator acts as an EAP **pass-through**, and is referred to as a pass-through authenticator and the authentication server is referred to as a **back-end** authentication server. The pass-through authenticator forwards the EAP messages from the client to the back-end server and vice versa, while waiting for an *EAP-Success* or an *EAP-Failure* message [50] from the back-end server which indicates the success or failure of the EAP authentication. An important advantage of EAP is the flexibility it provides to the three-party authentication model. Whenever a new authentication model needs to be supported, rather than requiring intermediate entity (which on forwards messages) to be updated for supporting the new method, only the backend authentication server needs an upgrade.

In such a case, the back-end EAP authentication server is usually part of an Authentication, Authorization and Accounting (AAA) server [52] and communication between the authenticator and the back-end authentication server is done over a suitable AAA protocol like RADIUS [58] or DIAMETER [60]. For more details related to the support for EAP in RADIUS refer to [59] and for Diameter to [62].The authenticator encapsulated the EAP packets received from the client into AAA messages and transports them to the back-end AAA server which then delivers them to the EAP server. Returning EAP packets are again encapsulated in AAA messages and sent to the authenticator, which forwards them to the EAP peer.

The benefit of pass-though authenticators are that they only need to implement limited forwarding functionality. Furthermore, the overall system is more secure because secret credentials do not need to be stored at each network access point. Note that it is possible for the authenticator to implement some authentication methods, while acting as a pass-through authenticator for others.

EAP packets

Figure 3.6 shows the EAP packet format. The *Code* field indicates the type of the EAP packet and can only be one of four values: *Request, Response, Success and Failure*. We have already explained that the *EAP-Success* and *EAP-Failure* packets are used to carry the results on the authentication from the EAP server to the EAP authenticator. The *Identifier* field is used to match *EAP-Requests* with *EAP-Responses*. The *Length* field indicates the byte count of the whole packet. The *Type* and *Type-Data* fields are only present in the *EAP-Request* and *EAP-Response* packets. The *Type* field indicates the EAP method that is encapsulated in the EAP packet, while the *Type-Data* field carries the payload corresponding to that EAP method.

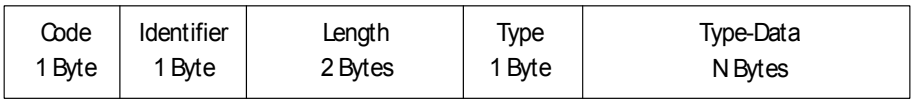

Figure 3.6: EAP packet format

The specification of a new EAP method must assign a new *Type* value greater than four [50] which can be used to identify it. It must also define the layout of the corresponding *Type-Data* field which will carry the payload of the new EAP method. Lastly, it must specify how the new method specific payloads are processed at the three EAP entities i.e. the peer, authenticator and authentication server.

3.8.2 Architectural components

Our aim is to enable un-clustered personal devices to authenticate with the security agent of a co-located personal cluster using pre-loaded Personal TGS tickets, thereby obtaining the cluster key and other configuration parameters necessary to become members of that cluster. Furthermore we would like such devices to be able to authenticate through cluster members within their transmission range and with whom they share a common radio interface. This should not necessarily have to be the security agent of that cluster. Figure 3.7 illustrates the protocol stack of the components in the EAP-PTGS framework and provides an overview of the processing path from the EAP peer (client) to the back-end EAP authentication server (security agent). We can see that the messages belonging to the EAP-PTGS authentication method are transported between the client and the security agent encapsulated in EAP messages which are routed within the cluster using the facility of the underlying AAA framework.

Besides authorized cluster traffic, certain cluster members also accept unauthenticated EAP requests which are forwarded to the security agent for authentication. Predictably, devices that are not part of a cluster do not accept EAP requests. The main components of this framework are the:

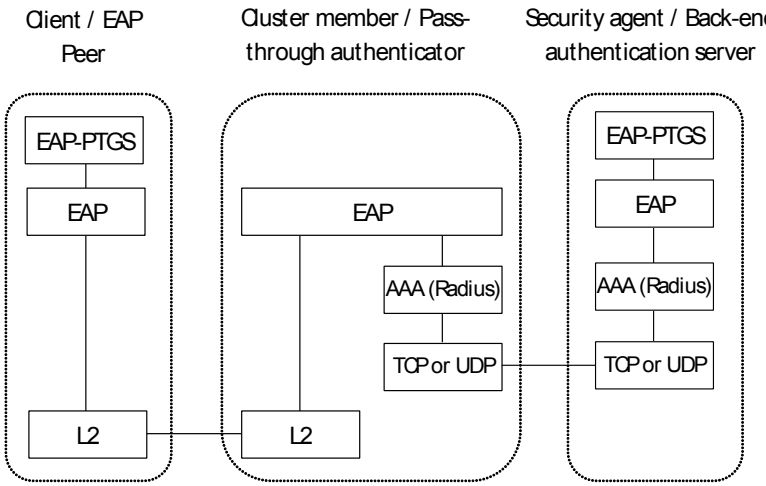

Figure 3.7: Protocol stack of EAP-PTGS framework

Personal TGS

The Personal TGS is the trusted third party responsible for pre-loading all relevant tickets to personal devices during imprinting. The Personal TGS functionality has been introduced in Section 3.3.3 where we also explained how the set of tickets pre-loaded on a given device during imprinting depends on the capabilities of that device.

EAP-PTGS authentication server (PTGS-AS)

The PTGS-AS module at the EAP server is responsible for generating and processing EAP-PTGS messages. It has access to the secret key shared with the Personal TGS as well as all preloaded tickets, using which it is able to authenticate any devices presenting a valid ticket. Furthermore it has access to the cluster key and other parameters that must be securely transferred to the EAP peer after a successful authentication.

At the end of the authentication process the PTGS-AS module returns a result that consists of three things. First, it gives an indication of the success or failure of the authentication. Based on this the EAP server sends out an *EAP-Success* or an *EAP-Failure* message to the pass-through EAP authenticator. Second, if the authentication is successful the PTGS-AS module returns a key (derived from the ticket) which will be stored in a local security database and used for all future authentications. Third, the PTGS-AS module either returns an EAP-PTGS error message used to signal the EAP-PTGS authenticating client (PTGS-AC) module at the EAP peer the failure of authentication, or an EAP-PTGS reply message that contains the necessary parameters for cluster access. Details on the different types

of EAP-PTGS messages are available in Section 3.8.3.

Back-end EAP/AAA authentication server

The role of a back-end EAP/AAA authentication server will be performed by the security agent. This means that all security agent capable devices (SAC) are required to have this functionality. The security agent will need to function as an AAA server because the pass-through EAP authenticator functions as an AAA client in order to route the EAP packets received from the EAP peer to the back-end EAP server. Our mechanisms for self organization ensure that each cluster has exactly one functioning security agent.

Pass-through EAP authenticator

The pass-through EAP authenticator is the cluster member in direct communication with the client and through which the client is authenticating with the security agent. Since all the devices in a cluster connect using short range radio links, the EAP communication between the pass-through EAP authenticator and the EAP peer will run over the link layer.

Although the pass-though nature of the EAP authenticator means that the functionality required at such a device is rather limited, we do not expect all personal devices to be able to function as pass-through EAP authenticators. Cluster members indicate their EAP pass-through capabilities using the *EAP pass-through* field of cluster advertisements which they forward. The cluster advertisement packet format was illustrated in Section 3.4.2 where we also stated that cluster members may forward non-duplicate advertisements on available radio interfaces, depending on the cluster policy and local resource constraints.

Indicating the EAP pass-through capabilities enables devices hearing such advertisements to know beforehand if they can authenticate through the advertising device. In case this functionality exists, the advertising device functions as a pass-through EAP authenticator and encapsulates all EAP messages received from the EAP peer in an AAA protocol and relays them to its back-end EAP server. Any confidential information exchanged between the authentication end-points is not visible to the intermediaries who are just forwarding the EAP messages.

3.8.3 The EAP-PTGS method

In this section we describe details of the EAP-PTGS method which defines how EAP-PTGS messages are encapsulated in EAP packets and how such packets are processed by the different components in our authentication framework. As stated earlier the first step in specifying a new EAP method is to define the layout of the *Type-Data* field (see Figure 3.6) which will carry the payload of the new method.

The EAP-PTGS payload

As illustrated in Figure 3.8, the layout of *Type-Data* field for the EAP-PTGS payload is composed of a *Msg-Type* field, depending on which there may be a sequence of one or more *Type-Length-Data* attributes.

Msg-Type (1 Byte)		
Type (1 Byte)	Length (2 Bytes)	Data (N Bytes)
Type (1 Byte)	Length (2 Bytes)	Data (N Bytes)
Type (1 Byte)	Length (2 Bytes)	Data (N Bytes)

Figure 3.8: Layout of the *Type-Data* field for the EAP-PTGS payload

The attribute *Date* field is variable in length and is a placeholder that contains a certain data, the size of which is specified by the value in the attribute *Length* field. The different EAP-PTGS *Msg-Types* and as well the included attribute *Types* are described below.

- MSG-Type

  A value that indicates the type of EAP-PTGS message being encapsulated in the EAP packet. The different values that *Msg-Type* can take are shown in Table 3.1. The *PTGS_Start* and the *PTGS_Start_Tkt* messages are generated by the security agent and indicate the initiation of the EAP-PTGS authentication method corresponding to the two examples described in Section 3.3.4. The former only includes the identity of the security agent, while the latter also includes the Personal TGS ticket held by the security agent. The *PTGS_Finish* and *PTGS_Error* messages indicate the successful or failed termination of the EAP-PTGS method respectively. The *PTGS_Request* messages are generated by the client after the reception of the start messages, to which the security agent responds with a *PTGS_Reply* message.

- Attribute: PN-Identity

  This attribute hold the complete PN identity of the sender, including the senders own ID and his PN ID.

- Attribute: PTGS-Data

  This attribute contains the ANS.1 encoding [67] of the authentication data, defined by the *Msg-Type* field and based on the expectation of the EAP-PTGS state machine.

- Attribute: Cluster-Data

Table 3.1: EAP-PTGS message types

Name	Value
PTGS_Start	0
PTGS_Start_Tkt	10
PTGS_Finish	20
PTGS_Error	30
PTGS_Request	40
PTGS_Reply	50

This attribute is only present in an EAP-PTGS *Msg-Type* of *PTGS_Reply* and contains the ANS.1 encoding of the organizational parameters necessary for cluster membership. Such information can be transferred in multiple *PTGS-Reply* packets if necessary.

Next we specify the behavior of the different architectural components with respect to how they process EAP-PTGS packets.

Behavior of the client

The authentication process is initiated by the client after the reception of a cluster advertisement transmitted by an existing cluster member. The client then indicates to the cluster member (e.g. using the *EAPOL-Start* message in IEEE 802.11i [68]) its interest in joining the cluster, based on which the cluster member transmits the *EAP-Request* packet of Type 1 (Identity). For the purpose of explanation we will assume that this cluster member is functioning as an intermediary, forwarding the EAP communication to the security agent.

At this point the EAP entity at the client does not know the identity of the security agent and replies with an *EAP-Response* packet of Type 1 (Identity). Based on the identity of the client, the security agent checks its security database to see if it has an existing long term key with the client. If not, it invokes the PTGS-AS module which initiates authentication based on preloaded Personal TGS tickets. The first EAP-PTGS message sent by the PTGS-AS is the *PTGS_Start* or the *PTGS_Start_Tkt* message, with the former including just the identity of the security agent, while the later also including a ticket.

After the reception of the start message, the client responds with a series of *PTGS-Request* messages to which the security agent responds with *PTGS-Replies*. The requests and their associated replies are transported by the authentication framework between the security agent and the client as explained in the previous section. At the completion of the EAP-PTGS authentication the PTGS-AS sends out a *PTGS-Finish* message, while any errors discovered in the middle generate

PTGS-Error messages. For instance, if the PTGS-AC module at the client does not have a ticket corresponding to the identity of the security agent as stated in a *PTGS-Start* message, and instead expected a *PTGS-Start-Tkt* message.

Behavior of the intermediary cluster member

As specified in [50] the functionality of the pass-through EAP authenticator is just to forward EAP packets to both sides of the communication end-points i.e. when a pass-through authenticator receives an *EAP-Request* from the back-end EAP server it forwards the message to the EAP peer. When it receives an *EAP-Response* from the EAP peer it forwards it to the EAP server. This functionality is independent of the EAP method in use.

However in the event of a successful authentication, the EAP-PTGS method does add one modification to the typical behavior of an EAP authenticator. Typically a successful EAP authentication results in the generation of what is known a Master Section Key [50]. This symmetric key is transported from the back-end EAP server to the pass-through EAP authenticator using AAA transport and used to secure the network access of the client. The actual transport and use of the MSK is beyond the scope of the EAP specification and depends on the security architecture in question e.g. IEEE 802.11i. However in our case the client is not restricted to accessing the cluster through the authenticator involved in the authentication process. In other words after a successful authentication the client does not maintain a long term association with the authenticator and is able to send traffic through any other cluster member within its transmission range. Thus after reception of the *EAP-Success* packet signaling the completion of the authentication and the successful transfer of related configuration parameters, the EAP authenticator breaks the connection with the EAP peer and deletes any state information related to it.

Due to the simple functionality required for EAP pass-through and the limited duration of the association with the client (till authentication completes), we expect many PN devices to be able to function as pass-through EAP authenticator for cluster access. In this regard, we define three categories of pass-through EAP functionality. The first category of devices does not have any pass-through EAP functionality and cannot take part in facilitating any authentication. The second category of devices is only able to do EAP pass-through for cluster access i.e. they do not have the functionality or resources to act as a dedicated access points for foreign devices. The third category of devices are able to operate as PN gateways and are able to function as pass-through EAP authenticators for cluster access as well as for foreign devices. Details on the secure service access of foreign devices is available in Chapter 6.

Behavior of the security agent

The EAP server evokes the PTGS-AS module to process EAP packets in three cases:

1. When it receives an *EAP-Response* of Type 1 (Identity) corresponding to a personal device for which it does not have a long term key in its security database. If the key exists in the security database then another suitable existing EAP mechanism such as EAP-PSK [57] can be used.
2. The client specifically requests the EAP-PTGS method by sending an EAP Legacy NAK.
3. The EAP server receives an *EAP-Response* message of type EAP-PTGS.

When an EAP server initiates the PTGS-AS module after the reception of an EAP identity packet or an EAP Legacy NAK packet the EAP server generates an EAP packet built as follows:

- EAP *Type* is set to EAP-PTGS
- EAP-PTGS *Msg-Type* is set to *PTGS_Start*, or *PTGS_Start_Tkt*
- The *PN-Identity* attribute holds the complete ID of the security agent
- For *PTGS_Start_Tkt* packets, the *PTGS-Data* attribute holds the relevant ticket

This EAP packet is then encapsulated in an AAA message and sent to the intermediate cluster member who forwards it to the client. Further behavior of the PTGS-AS and PTGS-AC modules depend on the type of EAP-PTGS message (as indicated by the *Msg-Type* field) and the expectations of the EAP-PTGS state machine (based on the generic authentication mechanism presented in Section 3.3.4).

3.9 Cluster Merging

Cluster merging is the process whereby two clusters of the same PN, within each other's transmission range, merge to form one cluster. Merging two co-located personal clusters allows devices from one cluster to access resources in the other cluster directly, instead of going through their respective gateways. However before two clusters can merge into one, they need to agree on common cluster parameters such as the cluster policy, cluster key and security agent. Furthermore the decision to merge or not depends on the user preferences configured at each security agent, for instance, the user may not want his P-PAN cluster to merge with another personal cluster until the two clusters have been co-located for a certain minimum duration.

The authentication prior to cluster merging only takes place between the two security agents. If the authentication is successful, one of the security agents will step down after updating its cluster members with the parameters of the cluster being merged with. This update information is distributed to all cluster members by utilizing the *Merge* extension header as illustrated in Figure 3.9. Since cluster merging also requires an update of the cluster key, the *Merge* extension header is always used with the *Re-key* extension header and its associated synchronization mechanisms.

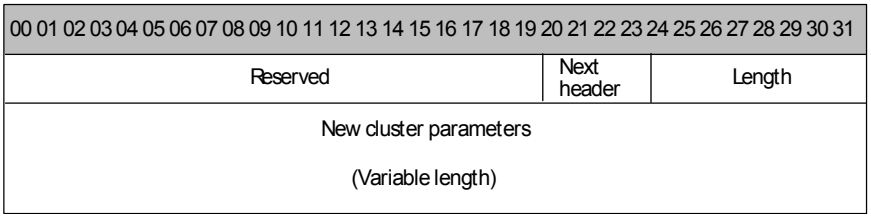

Figure 3.9: The *Merge* extension header

The decision on which of the two security agents steps down is based on their **Security Agent Weight** (**SAW**). The value of the SAW summarizes their capabilities and is used to identify the more capable device to remain as the security agent. The different parameters used to calculate the SAW e.g. remaining battery time, mobility, number of cluster member, number of radio interfaces etc. do not all have the same impact and are weighed differently. We assume the following expression for the computation for a device X's SAW:

$$SAW_X = \sum_{i \epsilon I} c_i P_i$$

Where c_i is the (constant) weighing factor of the I^{th} system parameter of interest P_i. Note that the parameters of interest used to calculate the SAW also include the number of devices in the cluster. This is because the energy cost associated with updating the necessary information during cluster merging depends on the size of the cluster.

3.9.1 Authentication framework extensions for cluster merging

At this point two clusters can only merge if either of their security agents is able to directly receive cluster advertisements from the other cluster. This security agent would function as a client and authenticate through the intermediary cluster member with the other back-end security agent. However it may be useful if the two clusters are able to merge when their periphery overlaps i.e. when a cluster member from one

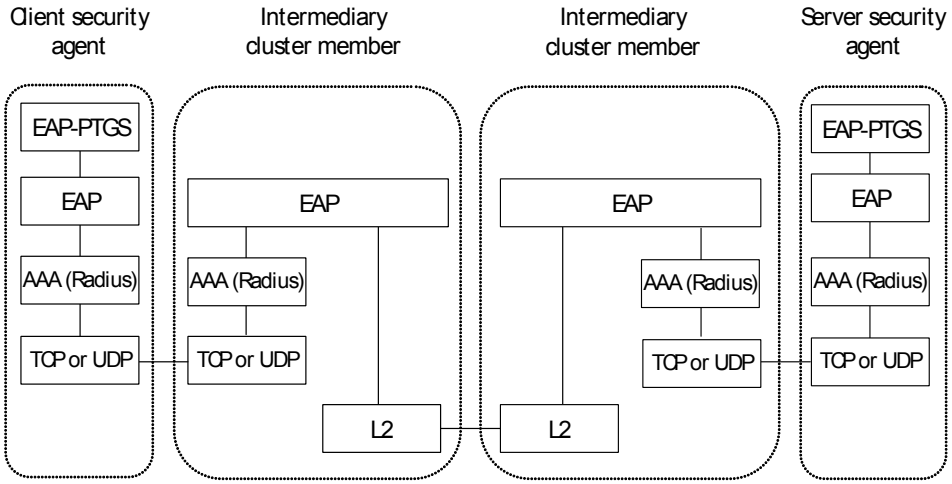

Figure 3.10: Protocol stack of modified authentication framework

cluster is within the transmission range of a cluster member from another cluster. Based on this we propose an extension to our original authentication framework (see Figure 3.7) as illustrated in Figure 3.10. Note that the functionality related to the EAP-PTGS method, specifically the roles of the PTGS-AS and PTGS-AC modules as well as the EAP-PTGS packet format remain the same.

Modifications are only required at the intermediate cluster members which must forward non-duplicate advertisements of other personal clusters to their own security agent. To reduce overhead intermediate cluster members can utilize some back-off algorithm to throttle the rate at which cluster advertisements are forwarded. Additionally, it will also be necessary to implement a new mechanism to signal the intermediary cluster member in the client cluster to initiate the authentication. The intermediary cluster member in the server cluster will respond with an EAP identity request, which (including all further EAP communication) is forwarded to their respective security agents. The two security agents then authenticate each other, enabling the two clusters to merge when their periphery overlaps.

3.10 Processing cluster advertisements

Cluster advertisements play a central role in the self-organization and formation of clusters. Figure 3.11 illustrates the steps a cluster member goes through on the reception of a cluster advertisement. We have already looked at individual aspects separately; here we attempt to show how they fit together.

For cluster advertisements belonging to its own cluster, the EAP pass through decision is used to check if the host device is EAP capable. If not, it re-transmits the cluster advertisement marked *1*. If yes, it re-transmits it marked *2* or *3* depending

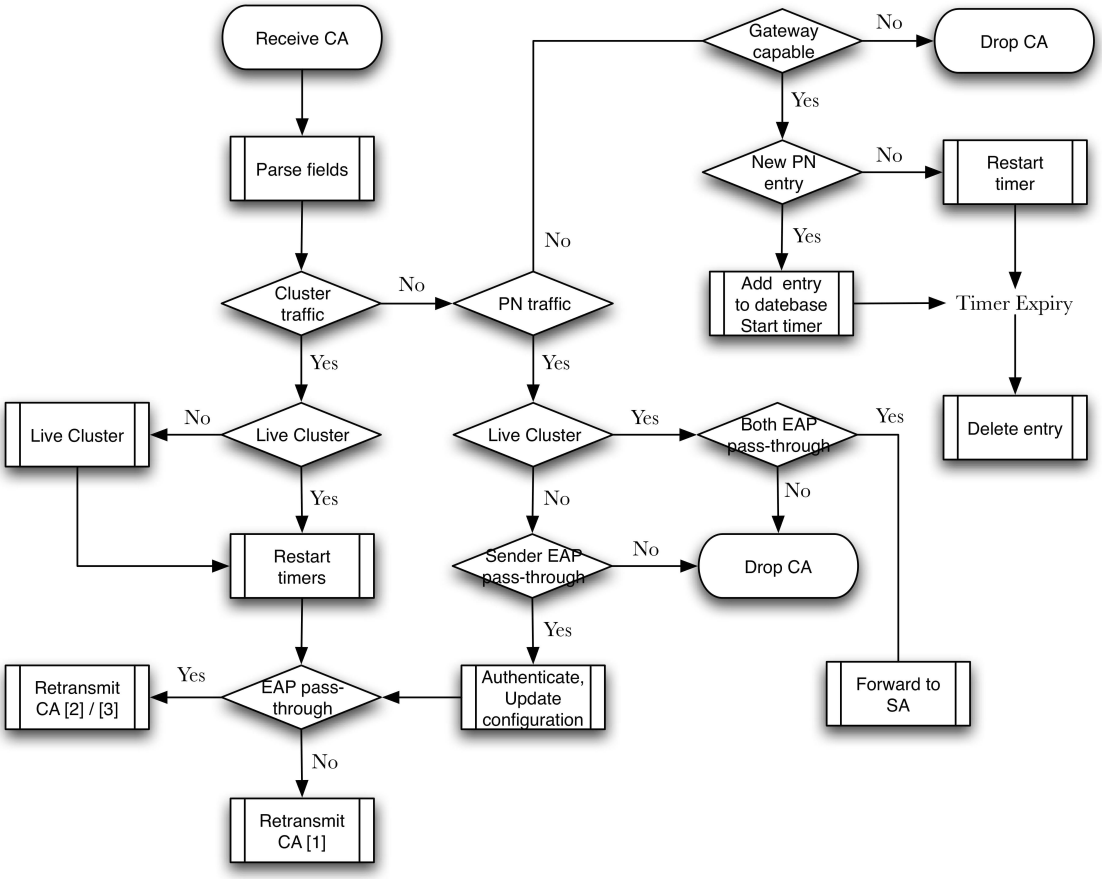

Figure 3.11: Processing cluster advertisements

on whether it is only able to authenticate personal devices or also foreign devices. For cluster advertisements belonging to other personal clusters, the host checks if both the sender and it are EAP pass through. If yes, the cluster advertisement is forwarded to the security agent for enabling cluster merging. If it is a part of a zombie cluster and the advertising device is EAP pass through, then it instead attempts to join the newly discovered personal cluster. Cluster advertisements belonging to foreign clusters are dropped unless the host device is gateway capable. If yes, it caches the information inside the foreign cluster advertisement in a local database, which is exported to other cluster members as a service. Stale information is periodically deleted.

3.11 Evicting personal devices

Evicting long-term members is a rare occurrence which for example needs to be done if a device gets lost or compromised. In such a case, the device's long-term security association with the PN will be severed, followed by its eviction from the PN. A personal device that has not been compromised but is otherwise removed from the PN, for example if it is sold, does not need to be revoked. It will just have its locally stored PN credentials purged during de-personalization.

Revoking the membership of a personal device in the PN is in fact a two step process. The first step is to identity the device which needs to be revoked and the second step is to initiate the actual process of revocation. In an ideal case we would like compromised devices to be automatically detected and blacklisted by the system. Using a true intrusion detection system (IDS) to detect misbehavior would be overkill for the average home user since they are difficult to install and time intensive to keep properly maintained. Distributed mechanisms to detect misbehavior in wireless ad-hoc systems such as those based on assigning a reputation to each device [69] [93] [102] are also not feasible given the constrained nature of battery operated mobile devices. Such reputation systems, used to establish trust and encourage trustworthy behavior, suffer from inherent problems in the way the reputation value is defined and calculated, the detection of disreputable behavior, and the coordinated distribution of reputation information. Clearly the cost of security should not be more than the benefit derived from it. Ultimately, lacking a suitably cost effective alternative our architecture makes it the responsibility of the user to identify devices that need to be revoked.

Once identified, personal devices are revoked by adding them to a local revocation list. Given our model where the security manager is the only entity to be fully trusted, the revocation notification has to be generated by the security manager. The simplest method would be to have the security manager contact each personal device and update the local revocation list there. However given the disconnected nature of the PN and the fact that the security manager is not permanently online,

we propose an alternate mechanism which we feel is applicable to the majority of cases. Specifically the user can chose this alternate mechanism if the device to be revoked is SAI. Given that the majority of personal devices are expected to be small form factor SAI devices, it will be appropriate in most cases.

In Chapter 1, we stated that our *basic* security mechanisms required for secure PN formation and communication must only rely on lightweight symmetric cryptographic primitives. This was done to reduce the minimum requirement for devices to be PN capable, so that even small devices like bio-medical sensors could potentially be part of a PN. This means that for more powerful devices like security agents, it is possible to relax this constraint since it will not be a limiting factor.

When a personal device needs to be revoked, it should be added to the revocation list maintained by the security manager. The alternate approach would have the security manager sign this list with its public key and upload it to the PN agent[1] (Section 1.3). Security agents periodically check the state of the revocation list at the PN agent and synchronize any changes. The reason this step is enough to revoke SAI devices is because SAI devices must always access the PN through a security agent. Details on the mechanisms for synchronisation and the limited usage of public keys within the scope of PNs is provided in Chapter 6.

Once a compromise or lost device is revoked, it is recommended to re-form the PN. This will ensure that when the PN re-forms the evicted device will not have any valid credentials to authenticate itself. It is safest to re-form the PN, because once part of a cluster devices receive cluster key updates using group authentication so it is not possible to definitively identify the cluster to which a personal device belongs. However if the user knows that the device in question was a member in a given cluster then it is sufficient to re-form that cluster only.

Since personal devices are expected to have long term associations with the PN, for efficiency reasons our proposed cluster re-key mechanism does not offer a way to selectively deny re-keying to evicted devices. This is why we must re-form the cluster in order to deny access to a recently evicted device. However if cluster re-formation is not an option, it is possible to modifying our re-key mechanism to use GKMPAN [20] instead. GKMPAN is based on probabilistic key pre-deployment and also uses TESLA for secure broadcasting. However it allows us to evict selected personal devices during re-keying. However, given that we only expect personal devices to be evicted rarely, we do not propose it as the default mechanism. This is due to increased overhead from storing the extra set of keys, increased transmission overhead and the fact that it requires a multicast delivery tree to distribute the new group key. GKMPAN was designed for multicast groups where such trees already exist.

[1] This implies that all SAC devices must be imprinted with the security mangers public key during personalization.

3.12 Threat analysis

In general, attackers are classified into two main classes, passive and active. Passive attackers only eavesdrop on the communication thus they are a threat against the confidentiality and the anonymity of the communication. Although most applications that require confidentiality already encrypt their traffic end-to-end, confidentiality for PN traffic can also be assured via optional link level encryption using K_{encr}. A compromised K_{encr} only compromises the confidentiality of data till the next cluster key update. On the other hand if the actual cluster key gets compromised, usually as a result of device compromise, then the confidentiality of all future cluster traffic is compromised. This will require revoking the compromised device and reforming the cluster.

Anonymity can only be assured if devices belonging to the user do not expose their identity or something that can be linked to their identity to un-trusted parties. Providing anonymity for the users of PNs is quite challenging and affects performance and possibly even usability. Consequently our proposed mechanisms do not provide anonymity as default. However we have identified at least two modifications that would be necessary in our design if we were required to support anonymity.

- Using encrypted cluster advertisements: Broadcasting of clear text PN ID in cluster advertisement is no longer possible. The user can stay anonymous and undiscoverable (i.e. such clusters cannot increase in size) by temporarily encrypting cluster advertisements. This also has the automatic effect of stopping authentication attempts in which devices transmit their IDs in clear text .

- Using dynamic MAC addresses: Since authentication is based on device IDs, devices may change their MAC addresses if necessary. However this creates overhead in the system as it requires periodic updates of the ARP cache and checking for potential MAC address collision before each update.

Active attackers not only eavesdrop on the communication but also modify and inject packets into the network. An active attack against personalization is not possible because the attacker does not have access to the initialization key (assuming *Phase 1* in Section 3.3.4 was performed securely). Similarly, since all link layer communication is protected with a MAC created using K_{mac}, attackers cannot become active unless they have compromised this key. For this reason K_{mac} is regularly updated and the system has forward security since a compromised K_{mac} does not compromise the next K_{mac}. Nevertheless if K_{mac} is compromised while it is still being used the cluster becomes open to attacks, for example, those against the ad hoc routing algorithm [71] and resource consumption attacks on TESLA [19].

Without compromised keys the attacker can only launch DoS attacks. We disregard physical layer attacks based on jamming because mechanisms such as spread spectrum have already been extensively studied to resist such attacks. DoS attacks

against computational resources can be performed by fake authentication attempts as well as injecting messages into the cluster that will fail the integrity check. Our aim is to reduce harm from such DoS attacks by limiting their energy imprint. As transmission uses the lion's share of energy, we want to reduce wasteful transmissions triggered by attacks. Therefore during personalization (Section 3.3.4) we also transmit a hash of ID_M in clear text. This allows device B to confirm that the device M knows its initializing key before it transmits any replies (as ID_M is also sent encrypted with the initialization key). Replay attacks are not possible because both imprinting and authentication between two given devices using tickets is only performed once. Lastly, since devices have no way of verifying cluster advertisements of personal clusters to which they do not belong, attackers can easily generate such fictitious cluster advertisements. However as devices always perform mutual authentication before joining a cluster, this is at best a DoS attack. Devices can protect themselves against such false advertisements by limiting the rate at which they attempt to re-authenticate after recent failed attempts.

3.13 Related work

The concept of Personal Networks is fairly new. The only publically accessible PN specific architectures we know of are those developed in the IST MAGNET [8] [80] and the IST MAGNET Beyond projects [8]. As such they are the only related work with which we can make a direct comparison. First we look at how our model for secure clustering compares with the model developed in the MAGNET project.

There are fundamental differences in the mechanisms used for the creation of trust between personal devices, the formation of clusters and secure intra-cluster communication. We have proposed establishing trust through the use of Personal TGS tickets, pre-loaded on new devices during the personalization phase. On the other hand, the MAGNET approach to creating trust between personal devices is to create the very first trust relationships between a new PN member and a subset of existing PN members through user assistance. It was expected that the new member would be able to create the remaining trust relationships using the concept of "transitivity of trust". The expectation was that if two personal devices A and B had a trust relationship, and B also had a trust relationship with C, then A could leverage its trust relationship with B to create one with C. However this model had some inherent flaws, mainly related to scalability (it was not feasible for more than 20 devices) and the potential for abuse by compromised devices. As a result of these flaws it was abandoned and replaced in the MAGNET Beyond project by a model based on PKI [66].

When comparing the mechanisms for cluster formation, our mechanisms are based on selecting the most suitable device to perform the role of a security agent. The MAGNET approach is totally distributed, where a device is only able to join

a cluster provided it has a trust relationship with a cluster member within its communication range. New cluster members use the transitivity of trust to establish separate trust relationships with all neighboring cluster members, something that results in increased communication overhead especially during device mobility. This is necessary because in the absence of a shared group key, a MAGNET cluster is composed of co-located devices connected using secure pair-wise links.

When comparing the mechanisms for secure communication, the MAGNET approach of using pair-wise keys to secure both unicast and broadcast communication results in increased processing overhead of packets travelling multiple hops (because of repeated decryption and re-encryption). In comparison our approach of using a cluster key reduces the overhead of key management and multi-hop communication substantially and the centralized nature of our security architecture can be viewed as an asset as it provides the PN owner with a higher degree of control. Conversely the main advantage of the MAGNET approach is that there is no single point of failure. Consequently cluster merges and splits do not require any re-organization given that there is no centralized entity in the cluster.

Even though the MAGNET project stated that some potential PN constituents were too resource constrained to be able to do asymmetric cryptography, the MAGNET Beyond project no longer envisions devices like sensors to be part of a Personal Network [81]. In our opinion this is more to do with the difficulty associated with using purely symmetric cryptographic primitives than a change in the paradigm. Consequently its mechanisms for the creation of trust are based on the concept of a personal PKI [28] and are no longer directly relevant to the work presented in this chapter. However since the MAGNET Beyond project generally focuses on inter-PN communication, relevant work in that regard is presented in Chapter 6.

3.14 Concluding remarks

In this chapter we have presented our model and mechanisms for securing personal clusters. We have presented a trust model based on imprinting and Personal TGS tickets. In this context we have explained the role played by the security manager, which functions as the cyber representative of the user. We have also presented details on the functionality of the security agent, as related to its management role within the cluster. This includes details on the new EAP-PTGS mechanism used for authenticating cluster members and our mechanisms for the source authentication of control traffic. Finally, we have justified out use of a periodically updating the cluster key and explained the role played by cluster advertisements in the self-organization of the cluster. Consistent with our requirements all required mechanisms are based on light symmetric cryptography and are applicable to a wide variety of devices. In the next chapter we will look at the transmission cost of our proposed mechanisms by simulations in NS-2 [83].

Chapter 4

Simulations

In this chapter we evaluate the mechanisms proposed in Chapter 3 for the formation and self organization of personal clusters [119]. In order to study the behavior of these mechanisms we developed a supportive simulation environment in NS-2 [83]. The objective was to (a) understand the effects of certain parameters on system performance and (b) get quantifiable values for the overhead of proposed mechanisms. For (a) we carried out simulations to study the effect of the value assigned to the redundancy factor (Section 3.5) on reliable message delivery. For (b) we looked at the cost of mechanisms for: cluster formation with one or more SAC devices, control traffic for cluster: maintenance, key updates and merging. The results show that our mechanisms have low delay and transmission overhead and are feasible for the heterogeneous devices we envision in PNs.

4.1 Details on authentication

The mechanisms used by an unclustered personal device to join a personal cluster have been explained in Section 3.8. As stated, the authentication process is initiated when an unclustered device receives a cluster advertisement from a personal cluster. This cluster advertisement is then cached for the duration of the authentication process. If the authentication fails the cached cluster advertisement is discarded, otherwise it is re-broadcasted. Doing so reduces the total clustering delay since all clusterable devices can be incorporated into the cluster after the transmission of only one cluster adversiment, instead of incorporating the first hop device after the first cluster advertisement and the second hop devices after the second cluster advertisment etc. As a result the time necessary for cluster formation is much lower than previously presented [117]. Note that in terms of EAP pass-through capabilities, we assume that all personal devices in our simulations are able to function as EAP pass-through for authenticating other personal devices.

An exception to the above procedure is that unclustered devices do not innitiate

authentication based on cluster advertisements containing extension headers. This is because such cluster advertisements are an indication that the cluster key is being updated. They will wait for the re-key phase to complete before authenticating with the security agent otherwise they may receive the old cluster key. Similarily, security agents entering the re-key phase cancel ongoing authentications and reject new authentication attempts till the re-key phase completes.

In our simulation model security agents are able to authenticate multiple SAI devices at the same time. However they reject authentication attempts from other security agents if they have an existing authentication (with a security agent) in progress[1]. This is because a successful authentication with another security agent will result in cluster merging. When this happens, all ongoing authentications at the yielding security agent are cancelled with a suitable error message[2]. The only exception to this is that the security agent with the lower weight will not yield if the cluster key of the other cluster is valid for less than the time required for rekeying. It will then wait for the new cluster key to become active, and then re-authenticate. Predictably, security agents in the merging state reject any authentication attempts from other clusters and also SAI devices.

Given that authentication for cluster access is based on symmetric key cryptography, the processing delay simulated at authentication endpoints was set at a nominal 10ms. Furthermore, we required every authentication attempt to complete within a certain timeout (*auth-timeout* in Table 4.1) or it was rejected. Lastly, each authentication message (transmitted over UDP) was required to get a reply within *rtn-timeout* (Table 4.1) or it was retransmitted.

4.2 Details on self-organization

The mechanisms used for self organization have been explained in Chapter 3 and include the periodic transmission of cluster advertisements and updating of cluster keys. In that regard each PN capable device in our simulations goes through the state transitions illustrated in Figure 3.2 and implements the rekey mechanism illustrated in Figure 3.4. In this section we will only provide the simulation specific details on these mechanisms. Table 4.1 shows values of some simulation parameters.

The *max-SAC-delay* timer refers to the maximum time that an SAC device waits in the *transition* state before it decides to form its own cluster. The actual amount of time waited is inversely proportional to its SAW. This means that when a functional cluster breaks apart, the first SAC device to form the new cluster will be the most capable SAC device and other devices can then join this cluster without needing to

[1] SAI devices also only attempt to authenticate with one security agent at a time.

[2] Ongoing authentications are also cancelled with a suitable error message if a cluster enters the rekey phase.

Table 4.1: Default values of certain simulation parameters

Parameter	Value
cluster-advertisment-period	5s
cluster-key-validity	600s
auth-timeout	15s
rtn-timeout	3s
max-SAC-delay	15s
redundancy-factor	4
cluster advertisment size	40 bytes
re-key ext hdr size	28 bytes
merge ext hdr size	28 bytes
hash ext hdr size	20 bytes

go through the process of cluster merging which has a much higher overhead.

4.2.1 Forwarding authentication requests and cluster advertisments

Cluster members forward all authenticated traffic from fellow cluster members, and drop all un-authenticated traffic from non-members except authentication requests and cluster advertisements. Such authentication requests from external devices are forwarded to the security agent for verification. Since these authentication requests cannot be authenticated before being forwarded, cluster members will need to limit the amount of authentication requests that they forward in any given period in order to guard against DoS attacks.

Similarly, cluster advertisements belonging to other clusters of the same PN i.e. with the same PN ID, are also forwarded to the security agent of the cluster. The security agent will then attempt to authenticate with its advertising counter part in order to merge the two clusters. As cluster advertisments from one clusters cannot be authenticated by forwarding devices of another cluster, it is conceivable that an attacker can generate false cluster advertisements with the same PN ID in order to launch a DoS attack. Therefore cluster members should control the rate at which they forward unauthenticated cluster advertiements, in our simulations cluster members only forward one external cluster advertisement per local cluster advertisement period. Such configuration information is usually set in the cluster policy.

4.2.2 Reducing Path Discovery Overhead

When a security agent transmits a cluster advertisement its AODV module sets the *Hop count*, *RREQ ID* and *Originator sequence number* fields shown in Figure 3.1. These fields have the same functionality as their namesakes in an AODV RREQ packet [39] and are used by forwarding devices to create temporary routing table entries to store reverse paths towards the security agent.

When a device receives a cluster advertisement from its security agent that has not been processed before, it updates the *Hop count* and searches for a reverse route to the security agent in its routing table. Depending on the result, the route is either created or updated using standard AODV behavior e.g. the lifetime of the reverse route entry is set using default parameters. The cluster advertisement is then processed by the device and re-broadcasted. Consequently, when an ensuing authentication requests needs to be forwarded to the security agent, the complete path to the security agent already exists and does not need to be discovered. This reduces both overhead and delay. The forward path to the authenticator is created when its first authentication packet travels to the security agent. The authenticator adds the following three fields to the first authentication packet that it sends to the security agent; *Hop count*, *Destination sequence number* and *Path lifetime*. As intermediate devices forward the returning authentication packets to the security agent, they use the information existing in the packet to create/update the forward route to the authenticator. The processing of the fields is the same as that of an AODV RREP packet [39]. We can think of cluster advertisements like AODV RREQ packets that are disseminated in the entire network, and the resulting authentication packets as the AODV RREP packets that are unicasted back to the security agent.

For this optimization, we only use a subset of AODV features that allow intermediate devices to create the freshest/shortest route to the security agent. Since the cluster advertisements are not meant for any specific device they do not use all the fields of a typical AODV RREQ packet, such as the *Destination address* and *Destination sequence number* etc. Also, further features of the AODV implementation such as the expanding ring and exponential back off are not required since we need network-wise dissemination of cluster advertisements and because security agents do not wait for any reply to their advertisement. Lastly, since these optimizations are derived from the AODV implementation in NS-2 they require using AODV as the ad-hoc routing algorithm for the simulations.

4.3 Simulations

Each simulation contains both standard NS-2 and PN capable NS-2 devices. Given the shared property of the wireless medium, the standard NS-2 devices are there to create background traffic that can compete with PN related traffic for contention of

the transmission medium[3]. This allows a more real world scenario to be simulated. An NS-2 simulation script is used to create both types of devices and initialize their parameters such as initial position and mobility model. The PN capable devices, which are initialized as SAC or SAI are pre-configured with parameters that would typically be configured in real life using imprinting (Section 3.3.4). Since we are not actually using cryptographic keys during authentication, the only value that needs to be configured for all devices is their PN ID and the parameters shown in Table 4.1. Lastly, since SAC devices start out as clusters they are also initialized with a random cluster key and a static weight (SAW).

PN capable devices do not function in the same way as normal NS-2 devices in that they are only willing to communicate with a subset of devices in the simulation i.e. those that belong to their own PN. However, as expected this communication is only possible after they are able to form a cluster.

4.3.1 Setup

We simulate a CSMA/CD (802.11b) wireless ad-hoc network with the wireless mobility extensions of NS-2.29. The network size is 50m x 50m and the transmission range of each device, both PN capable and others is 10m. Our routing protocol is AODV, and the wireless propagation model used is *TwoRayGround*.

In order to cover a wider range of possible scenarios each simulation is carried out with 5, 20 and 50 PN capable devices. For each of these scenarios we generate 30 connected graphs with random device positions. The simulation for each of the 30 graphs is performed 3 times using a random seed, for a total of 90 simulation runs per scenario. The overhead of our proposed mechanisms is judged by the average time and total transmissions at Layer 2 required to organize the cluster. This gives a quantitative idea of the cost, in terms of delay and energy consumption. Note that measuring the overhead in terms of bits and bytes that need to be transmitted allows us to give a transmission technology independent value for power consumed (using measured values of the transmit power of each technology when transmitting a data packet). The values calculated for energy consumption are interesting when compared to the available energy at each device, for example a device running on a 1.5 volt AA battery has a total of 15kJ [112] available.

4.3.2 Background traffic

Each simulation has 16 normal NS-2 devices placed 10m apart, equally spaced in a grid. The sender for each stream of the background traffic is chosen randomly

[3]We decided not to use PN capable devices to create background traffic because they can only communicate with each other after the cluster has already formed, so the effect of background traffic on cluster formation is limited.

from the top row and receiver from the bottom, thus ensuring that the background traffic traverses the network. Our background traffic is audio and video streaming, something which we believe is typical for PNs. As the amount of background traffic has a significant effect on the optimal value of the redundancy factor, we simulate three types of background traffic: low, medium and high. Low background traffic corresponds to 128kbps MP3 audio streaming. We assume that each payload contains 3 MP3 packets of 417 bytes each, for a total of 1251 bytes. We use a CBR (Constant Bit Rate) traffic generator with a packet interval of 80ms. Medium traffic corresponds to low traffic plus an MPEG2 video stream of 256 kbps. The payload size is 1024 bytes and the CBR traffic generator has a packet interval of 32ms. High traffic corresponds to low traffic plus an MPEG2 video stream of 512 kbps. Besides packet loss due to collisions we can also have packet loss due to interferencefrom external sources, therefore devices in the system are assigned a packet error rate of 2% for all incoming traffic.

4.3.3 Choosing the redundancy factor (RF)

In Table 4.1 we showed values of some parameters used in the simulations. The parameter which has the largest effect on performance is the value of the redundancy factor. As explained earlier we use redundant transmissions to ensure a reliable distribution of broadcasted cluster advertisements and re-key messages. For simulations with one re-keying phase, Figure 4.1 shows the average number of devices that are unable to re-key successfully due to lost re-key messages. The setup of this simulation run has been described in Section 4.3.1.

As expected, with low background traffic there is less contention for the transmission medium and a lower redundancy factor is sufficient to ensure reliable delivery. However, there are opposing effects as the number of devices in the system increase. On the one hand there is more contention in the system but on the other a higher device density means more redundancy. It is interesting to see that with light background traffic a cluster with 20 devices actually has slightly more re-key failures that that with 50 devices. This can be explained by the fact that with 20 devices spread over the same area as 50, there are fewer duplicate paths between a device and its security agent. So if one device fails to re-key it affects other devices that can only be reached through it. To conclude, since we would like re-key failures to be probabilistically low while also keeping the overhead as low as possible, we chose the value of the redundancy factor as 4 for all further simulations. Furthermore, since the amount of background traffic has a smaller effect on our remaining simulations, we also only use medium background traffic for the remainder of the simulations (for the chosen redundancy factor).

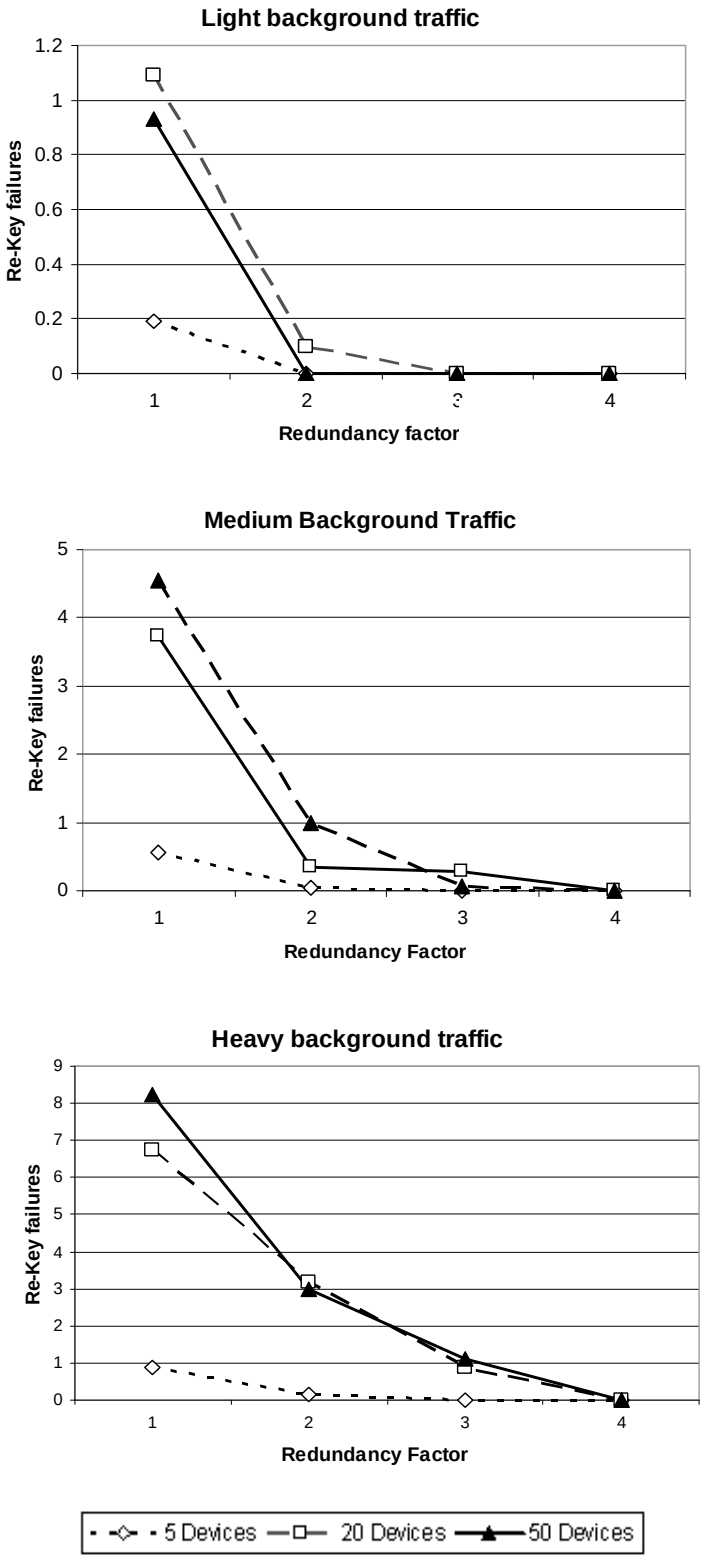

Figure 4.1: Re-key failures with light, medium and heavy background traffic.

Table 4.2: Cluster formation overhead for 5 devices

	1 SAC, 4 SAI devices	**5 SAC devices**
Duration (s)	2.1 ± 0.5	51.4 ± 3.9
Overhead (KB)	42.9 ± 2.3	50.2 ± 2.7

Table 4.3: Cluster formation overhead for 20 devices

	1 SAC, 19 SAI devices	**5 SAC devices, 15 SAI devices**
Duration (s)	6.5 ± 0.8	81.8 ± 5.0
Overhead (KB)	450.1 ± 21.7	367.4 ± 20.2

4.3.4 Cluster formation overhead

We simulate cluster formation with one and with potentially five SAC devices. Table 4.2 shows the average cluster formation overhead with 95% confidence interval for a cluster with 5 devices. The overhead field measures the average sum of Layer 2 transmission by all PN devices to create the cluster. When there is only one SAC device in the system, the cluster is formed rather quickly. After the security agent advertises itself, all devices within the first hop range authenticate (in parallel). As soon as they have authenticated, they retransmit the cached cluster advertisment allowing devices which are two hops away from the security agent to authenticate, and so on.

With 5 SAC devices, each device starts out being a security agent. Therefore each authentication can potentially result in cluster merging depending on if the security agent that is stepping down after merging has any clustered devices. If the security agent that is stepping down has not authenticated any devices to be part of its cluster, then it skips the cluster re-key process and the merge is instantaneous. The slight increase in the overhead between the two scenarios is the result of rejected authentication attempts, which need to be repeated. Some authentication attempts are initially rejected because a security agent can only authenticate with one security agent at a time. Table 4.3 shows the average cluster formation overhead with 95% confidence interval for a cluster with 20 devices.

As expected, with 1 SAC device the time taken for 19 devices to authenticate is more than that for 4 devices (Table 4.2). This is because with a larger number of devices, there is a larger probability of having a device more hops away. Generally, devices within the first hop will authenticate first (then retransmit the cached cluster advertisment), next the devices within the second hop and so on. Similarly with the increase in the number of authentications (and other control traffic such as RTS/CTS packets) and the larger number of hops they need to traverse to get to the security agent, the total number of bytes transmitted increases non-linearly.

Table 4.4: Cluster formation overhead for 50 devices

	1 SAC, 49 SAI devices	**5 SAC devices, 45 SAI devices**
Duration (s)	11.1 ± 1.0	82.1 ± 4.0
Overhead (KB)	1440 ± 83	1099 ± 64

From Table 4.3 we can see that although the time taken for cluster formation with multiple SAC devices is larger due to the delay resulting from cluster merging, the transmission overhead is actually lower. This is because although the total number of authentications in the system remains the same, the number of hops between the authenticating device and the security agent it is authenticating with is actually reduced. Devices authenticate with the first security agents whose advertisement they receive, often this is the closest one in terms of the number of hops. As clusters grow, they overlap and merge after the two security agents authenticate each other. In this way instead of multiple devices authenticating with the security agents over several hops, there is only one aggregated authentication. This effect is more pronounced here when compared with simulations of 5 devices because of the increased number of hops in the system. Table 4.4 shows the average cluster formation overhead with 95% confidence interval for a cluster with 50 devices.

With simulations of 20 and 50 devices, the time needed to complete cluster formation with multiple SAC devices is larger than the simulation of 5 devices. This is because in the later simulation as there are no SAI devices, each cluster merge does not require a cluster key update by the security agent that is stepping down. Furthermore, with 50 devices in the system the average number of hops between two random devices is greater, therefore the benefit of aggregated authentication due to cluster merging is more pronounced (24% in Table 4.4 compared to 18% in Table 4.3).

4.3.5 Cluster maintainance overhead

We proposed using periodic cluster advertisements to announce the presence of the security agent. Our simulations confirm that the security agent transmits one cluster advertisment (without any extension headers) of 103 bytes at Layer 2, per cluster advertisement period of 5 seconds. This cluster advertisment is then re-broadcasted once by each cluster member.

4.3.6 Re-keying overhead

During the re-key phase security agents transmit four cluster advertisments with the re-key extension header (131 bytes). This comes out to be 524 bytes per security agent and each clustered device; since they retransmit every CA they receive. With

Table 4.5: Re-key overhead

	5 devices	**20 devices**	**50 devices**
Overhead (KB)	2.4 ± 0.04	9.2 ± 0.13	24.6 ± 0.31

Table 4.6: Cluster merging overhead

	5 devices	**20 devices**	**50 devices**
Duration (s)	25.9 ± 0.3	26.4 ± 0.3	27.2 ± 0.5
Overhead (KB)	26.8 ± 1.9	92.0 ± 3.8	200.6 ± 36.5

a cluster advertisment period of 5 seconds, re-keying usually takes 25 seconds to complete. Table 4.5 summarizes the transmission overhead of updating the cluster key with 95% confidence interval.

As expected the growth is linear, since each device merely re-broadcasts the cluster adverisments it receives. The results are not exact factors of 524 due to lost cluster advertisments resulting from interference and collisions.

4.3.7 Cluster merging overhead

This simulation is carried out by creating two identical but seperate clusters, one of which then moves within the transmission range of the other. Three sets of results were obtained, for cluster merging between two clusters with 5, 20 and 50 devices each. The results are shown is Table 4.6.

Once the two clusters come within transmission range the time taken for them to complete merging is dependant on the re-key period of 25s. However, as the number of devices in the cluster increases, the number of hops between the two security agents (as well as the probability of packet loss over this longer path) increases. That is why as the size of the cluster increases we see an increase in the time needed for cluster merging. Furthermore, the overhead of cluster merging increases linearly as the number of devices in the cluster increases.

4.4 Conclusions

In this chapter we have studied the effects of different values of the redundancy factor as well as the the overhead of our mechanisms for cluster formation. Using mechanisms based on fast symmetric cryptography means that they are applicable to a wide range of devices. Since most of the (energy) overhead of these mechanisms arises from the transmission of control traffic rather than computation costs we quantify the total transmissions at Layer 2 that need to be exchanged.

Chapter 5

Secure inter-cluster connectivity

In Chapter 3 we have explained how personal devices initially organize themselves in the form of clusters. Forming clusters facilitates in securing multi-hop communication and access to the common pool of resources. The otherwise isolated clusters are interconnected using secure dynamic tunnels, created between gateway devices, resulting in a network of personal devices that are geographically dispersed.

In this chapter we investigate means of securing communication between geographically distributed Personal Network clusters. Using Virtual Private Network (VPN) technology enables the creation of secure tunnels between gateways of different clusters, making it possible to transfer all types of intra-cluster traffic securely, over the insecure public network. We investigate the suitability of existing VPN technologies to secure inter-cluster communication. Since the Internet Protocol (IP) is becoming the ubiquitous networking protocol, of particular interest will be IPSec [84] which is already widely deployed in different networks and terminals. We will consider how IPSec can be adapted to meet our requirements, as well as leverage on the existing key management environment of PNs. We are concerned with application of IPSec to PN networking in general, and in particular to networking small devices with limited computing power. Our contribution is in identifying aspects of existing solutions that render them unsuitable in their current form for constrained personal devices. We propose a modified framework [120] based on IPSec and KINK [86] that better satisfies the requirements of personal devices by reducing the cost of security and uses existing Personal TGS tickets for key management.

5.1 Introduction

When designing mechanisms to secure communication between PN clusters it is a logical step to investigate the use of Virtual Private Network (VPN) technologies. VPNs refer to creating semi-permanent cryptographic tunnels over public networks between two private machines or networks, to pass arbitrary traffic. The benefit of

using a cluster-to-cluster tunnel and not application level security such as an SSL [34] gateway means that we ensure the exact same connectivity and privacy as on a typical local private network. VPNs have been designed and used successfully for many years to connect remote offices of companies over the Internet. We too would like to use public networks (like the Internet) to coalesce geographically distributed subnets (clusters) into a private network (Personal Network). A VPN will allow us to create an encrypted tunnel between two clusters enabling the secure exchange of a wide range of traffic regardless of application or protocol.

In spite of the similarities, a Personal Network has some important differences with a typical geographically distributed corporate network. Firstly, people are mobile thus the cluster gateways should be able to handle mobility. Secondly, and arguably more significantly, there is considerable resource disparity between a mobile personal gateway and a static corporate style gateway. Whereas the former would resemble a small battery operated mobile phone the latter would be more like a typical desktop computer. Consequently, we need to investigate the different VPN technologies available today and evaluates their usability in securing communication between mobile resource constrained personal devices.

In Chapter 1 we concluded that the hetrogenity of devices in a Personal Network meant that our security mechanisms would only rely on lightweight symmetric cryptography. However later we made an exception for the more powerful PN devices which are able to perform the role of security agents. Although it is possible to make this assumption for gateway devices also, using purely symmetric cryptographic primitives to create inter-cluster tunnels allow us to support a wider variety of devices as cluster gateways. In this chapter we look at our options in this regard.

5.2 Existing Approaches

The most well known and standardized means of creating secure tunnels over the Internet is the Internet Protocol Security (IPSec) [84] standard. IPSec is the IETF standard VPN technology defined for the TCP/IP suite and is used to create the majority of commercial VPNs found today. However IPSec is overwhelmingly used with a Public Key Infrastructure (PKI) [66] requiring us to evaluate its suitability for Personal Networks.

In the last few years there has been a challenge to the widespread implementation enjoyed by IPSec in the form of Secure Socket Layer (SSL) based VPNs. SSL VPNs use the highly mature and prevalent SSL/TLS libraries to handle the tunnel creation and cryptographic elements necessary to create a VPN. They are the only serious competitor to IPSec VPNs in terms of functionality, scalability and flexibility. Next we investigate the suitability of the two approaches given our constraints.

5.2.1 SSL VPNs

One of the most well known is the free and open source OpenVPN [85]. In terms of functionality it has the same site-to-site connection functionality found in IPSec VPNs. The OpenVPN designers claim that because it operates in user-space (unlike IPSec which operates in kernel space) it increases security and system stability. Furthermore because of reduced configuration options and the fact that it runs in user space, it is easier to install and configure.

Traditionally SSL based VPNs used TCP to tunnel packets; however IP-over-TCP-over-TCP can create several latency problems when packet loss occurs. This is because with TCP encapsulated within TCP, we now have two flow control layers that are working against each other in an attempt to get packets resent. Therefore many SSL based VPNs found today (such as OpenVPN) use UDP to tunnel packets instead of TCP.

An often touted advantage of SSL based VPNs is that they are better at traversing Network Address Translators (NATs) because IPSec with Authentication Header (AH) [87] hashes the source address as part of its authentication process and therefore is unable to function behind NATs. However since we are only considering IPSec with Encapsulating Security Payload (ESP) [88] we do not consider this a drawback as the source address is no longer part of the packet integrity check. On the other hand, a downside to SSL VPNs is in its packet drop performance. IPSec will inspect and drop packets at lower level in the protocol stack than SSL which will process it more before rejecting it. However in all fairness, this will only be an issue with DoS attacks in high capacity usage scenarios. We do not envision this to be a problem if used in Personal Networks.

Ultimately, we reject an OpenVPN based solution because it does not provide an alternative to using asymmetric cryptography. The SSL/TLS handshake protocol allows a client and a server to negotiate a cyphersuite, authenticate each other and obtain a shared master key usually using public key algorithms. In practice OpenVPN requires a PKI (either RSA challenge/response or DSA digital signature) to be used in any meaningful way. Manually pre-distributing static keys is not only un-scalable but also unsafe as it does not provide Perfect Forward Secrecy (PFS) by automatically updating the keys at regular intervals.

5.2.2 IPSec VPNs

The benefits of using IPSec include that as a standard it enjoys widespread implementation and must be included in all IPv6 implementations. Furthermore it provides a lot of flexibility and configuration options for different situations, notably in key management. Recall that we rejected OpenVPN because its key management did not provide an alternative to using asymmetric cryptography. Since the processing cost of IPsec is overwhelmingly composed of processing required for

cryptography [3] we now look at some of these aspects in more detail.

Once the initial keys are exchanged, the protection from IPSec and SSL VPNs is similar as it depends on the specific ciphers selected. Data passing through the virtual tunnel is kept secure using symmetric cryptography, where both sides of the tunnel share common keys. Symmetric encryption and message digests use fast block ciphers so the security overhead once the tunnel is created, is low. The primary challenge therefore is in key distribution which typically relies on asymmetric cryptography to get the common keys to both sides of the tunnel.

Key management in IPSec is handled by the Internet Key Exchange (IKE) protocol [92] which requires two phases. Phase 1 called the IKE Security Association (SA) phase, establishes an authenticated secure channel between the two communicating peers. Phase 2 uses the secure channel established in phase 1 to negotiate key material and security parameters for the IPSec SA. The following are the three authentication methods available to use with IKE phase 1:

- Public Key

  The exchange is protected by encrypting parameters such as ID and nonce with the sender's private key

- Digital Signature

  The exchange is protected by signing a hash value over ID, nonce etc. using the sender's private key

- Pre-shared key (PSK)

  The exchange is protected by encrypting parameters such as ID and nonce using a symmetric key derived by some out-of-band mechanism.

The first two are clearly based on asymmetric cryptography and must be rejected based on our constraints. Unfortunately upon closer examination of the IKE protocol it becomes clear that because PSK uses the Diffie-Hellman protocol [89] in negotiating the IKE SA, it therefore also relies on asymmetric cryptography. Furthermore when using PSK with mobile devices, if their IP address changes the corresponding shared key cannot be located during standard authentication. This necessitates the use of what is called "aggressive mode" in phase 1, where the ID of the initiator is sent to the responder in clear text. Not only is aggressive mode susceptible to (clogging) DoS attacks but it also means that IDs are not kept confidential during the exchange. In the next section we look at a little known alternative to IKE phase 1, known as KINK [86], which is purely based on symmetric cryptography.

5.3 Kerberized Internet Negotiation of Keys

The IETF Kerberized Internet Negotiation of Keys (KINK) working group has created a protocol to facilitate centralized key management for IPSec security based on the Kerberos [27] architecture, as an alternative to IKE phase 1. Kerberos provides an efficient means of mutual authentication and replay protection using a trusted third-party model based purely on symmetric cryptography. The performance goals of the KINK protocol are having a low computational cost, low latency, and a small footprint. It limits computational cost by avoiding the use of assymetric cryptography and reduces latency by only using two messages to establish the necessary security associations. Its limited complexity versus IKE phase 1's necessity of using public key cryptography means that it is more compatible with the constrained devices typically found in PNs.

Since KINK uses Kerberos as the underlying authentication mechanism, a KINK initiator needs to get a Kerberos ticket for each responder before the actual key negotiations begin. This would be a typical Kerberos exchange (depending on local policy considerations) and is not part of the KINK specification. The specification therefore only defines the steps to follow once the ticket is obtained. KINK uses a simple and lightweight *Command/Reply* messages exchange for each of its *Create*, *Delete* and *Status* message flows.

Once the two peers share a Kerberos session key using KINK, the session key is used to create the shared state necessary to run normal IKE phase 2 negotiations. As a result everything defined for IKE phase 2 (new group mode, quick mode, etc.) is supported by KINK. The specification, now a proposed standard, is publicly available and has already been implemented in racoon2 [90], an open source implementation of IPSec key management protocols. Figure 5.1 illustrates the system architecture based on the KINK protocol.

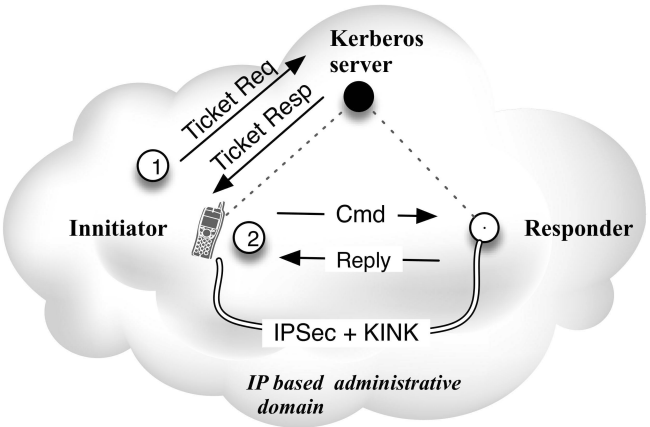

Figure 5.1: KINK architecture

Since KINK is designed for communicating entities within a centrally managed domain we see that the initiator and responder belong to the same domain as the Kerberos server. The server is able to issue tickets because it maintains a database of secret keys, one for each device belonging to the domain. This relationship is illustrated by the dotted lines in Figure 5.1. In step 1, the initiator requests a ticket from the Kerberos server corresponding to the responder. Once the ticket is obtained, the initiator uses the *Command/Reply* messages exchange (step 2) to derive the necessary IKE phase 2 parameters used to create the IPSec SAs. With the SAs in place, data passing through the tunnel is kept secure using use fast block ciphers so the security overhead is low.

Unfortunately the KINK architecture as shown in Figure 5.1 does not fit in with our requirement of supporting mobile personal devices. The dynamic nature of the Personal Network means that online access to a central authority for authentication cannot be guaranteed. Therefore we propose simple modifications to the KINK architecture that will allow us to use it in an offline manner. Specifically, we leverage on the Personal TGS concept introduced in Chapter 3, which allows us to pre-load gateway capable devices with tickets during personalization which can be later used in place of Kerberos tickets. Since devices no longer need to request tickets from the central entity before beginning the actual key negotiations with their peers (step 1), this also reduces tunnel setup latency and energy usage.

As of yet we have not proposed any modification to the KINK message exchange, but only to how tickets are made available to it. However pre-loading tickets can require a slightly modified KINK message exchange because the relevant ticket is now no longer always available with the KINK initiator. This is because new gateway devices are only pre-loaded with tickets corresponding to gateway devices that are already part of the PN. Consequently the ticket necessary for authentication is always with the device added later to the PN. Depending on the order that the two gateway devices were personalized, either the initiator or the responder can be holding the ticket. In the former case, KINK message exchange would proceed as usual. However, the latter case would require an extra *Request* message to be exchanged whereby the initiator would request the responder to begin authentication. Figure 5.2 illustrated the modified KINK architecture where the initiator does not hold the necessary ticket. We see that the two authenticating peers do not need access to the offline personal TGS in order to create an IPSec SA.

5.3.1 Other related work

Gupta et al. [91] have presented an alternate end-to-end security architecture for low power devices (Sizzle), which uses TLS based on ECC. Even though we reject the use of asymmetric cryptography like ECC for all devices in the PN, and Sizzle cannot use our existing key management which is based on PTGS tickets, it is nevertheless an interesting and standards-based related work. Sizzle allows embedding a secure web server in low power devices for monitoring and control purposes. A control station

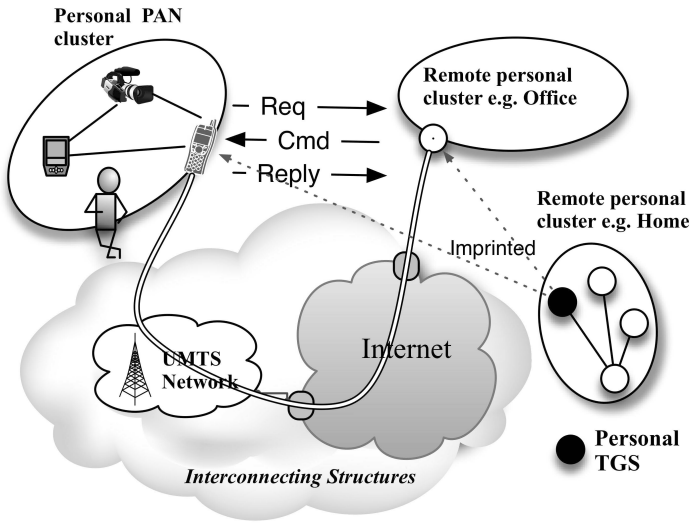

Figure 5.2: Modified KINK for Personal TGS tickets

located remotely is able to access the web service through a gateway that serves as a bridge to sensors which can only communicate via low-speed and short range wireless links. The gateway does not perform any actual cryptographic functions but only forwards messages between the two communicating end-points. Furthermore in the current architecture it does not authenticate itself to the control station or the sensors.

5.4 Concluding remarks

In this chapter we proposed a framework for secure inter-cluster communication in Personal Networks based on IPSec and KINK. The limited complexity of our proposal versus IKE's necessity of using public key cryptography means that it is more compatible with the resource constrained devices typically found in Personal Networks. Consequently we will be able to support a wide variety of personal devices as cluster gateways. We propose pre-deploy tickets into gateway devices during personalization, making KINK more suitable for mobile ad-hoc environments and reducing its latency and energy usage. The main disadvantage of pre-deploying tickets is in the increased overhead of device revocation, but given the nature of Personal Networks where devices have long term trust relationships we feel that this is acceptable side-effect.

Chapter 6

Secure inter-PN service access

6.1 Introduction

In earlier chapters we have presented our mechanisms for secure intra-PN communication. When considering secure service access for personal devices we take a rather straightforward approach. Given that all devices involved in intra-PN service access belong to the same user, we feel that it is not necessary to implement access control based on the specific identity of the personal device but rather its ability to demonstrate its membership of the PN group. Consequently secure service access for personal devices leverages on the security of the underlying intra-PN communication network, as introduced earlier. We argue that personal devices (as cyber representatives of their owners) are expected to be working on their behalf thus it is not efficient to incur the additional cost of mechanisms which enable per device access control. This means that in our model there is no additional cost of securing intra-PN service access.

Although the main focus of a Personal Network is to provide services to its owner, PNs are not expected to work in complete isolation from the rest of the world. This means that personal devices which interact with foreign entities must implement certain access control mechanisms. Specifically:

- Personal devices which allow physical access to foreign users should be able to limit the PN access available to such users. This means that such devices must implement methods for user-to-device authentication as well as methods to enforce different access policies for different users.
- Personal devices which allow network access to foreign entities, must implement methods for authentication and authorization of such users.

The scope of this chapter is limited to the second problem. PN devices must be able to interact with foreign devices operating as members of a foreign PN as well as standalone foreign devices that may not even be PN aware.

There are many scenarios in which the owner of a PN would like to provide others with access to some of his services, or to use their services. The entities involved in such relationships may include family members, colleagues, third-party service providers and even strangers. For an example of a PN interacting with a service provider, consider a patient suffering from epilepsy which requires constant monitoring of vital signs like blood pressure and heart rate. The PN belonging to such a patient can be registered with a healthcare provider which is able to access live readings of relevant vital signs. If a seizure is detected the healthcare provider takes appropriate action such as directing the primary care givers to the patients current position. Such a use case can also be extended with appropriate access control requirements. For instance, although the patient's vital signs are also available to his GP, the GP may not be allowed access to the patients current location since he is not involved in coordinating the response to the seizure.

It is important that our proposals for enabling secure service access by foreign entities should not increase the functionality and complexity required at ordinary PN devices disproportionately. In Chapter 1 we stated that we would like to reduce the minimum capability criterion for personal devices in order to accommodate all potential applications of the PN concept. A PN can be much more useful if some PN devices are small enough to be embedded in jewelry, watches and even clothing. This means that we would like to (a) limit the minimum functionality required for PN capability and (b) reduce as much as possible the complexity of this functionality. Consequently in this chapter we propose a framework for enabling secure PN interaction with foreign entities that suits the heterogeneous and distributed model of the PN. We want mechanisms that are simple and intuitive for the user and which can work without support from the infrastructure.

6.2 Securing Communication: Link layer or Network layer?

When considering foreign access of PN services, we can categorize foreign entities into two distinct groups. Foreign entities that have direct (OSI Layer 2) connectivity with the PN and foreign entities that need to access the PN through a routed IP network (OSI Layer 3). Our terminology for the former is **local service access**, and for the latter **remote service access**.

Network layer security like IPSec [84] is perfectly suitable for remote service access because it provides end-to-end security for IP layer and upper layer protocols such as TCP and UDP across a routed IP network. The fact that it is unable to protect non-IP traffic and data at lower layers is not an issue because all relevant communication takes place at or above the network layer.

In general, any security mechanism designed for remote service access can also

operate in the context of local service access as long as network layer connectivity can be successfully established. However because wireless networking presents significant security challenges, we have decided (where possible) to address the security of local service access at the link layer. Our main reason for not using network layer security to protect local service access is because it leaves the communication open to attacks that work outside the network layer. For instance, man-in-the-middle attacks using forged Address Resolution Protocol (ARP) replies designed to fool the client to send data to a malicious peer. Furthermore when compared to link layer security, denial-of-service attacks against network layer security are more costly because they must be processed higher up in the protocol stack.

Next we give an overview of the main architectural components involved in secure local and remote service access. Given that we are dealing with a potentially large set of foreign entities with dynamic relationships, our chosen model should minimize the overhead of any changes in these relationships. With a security model that is centralized, the PN owner can manage everything from a central controller. This reduces the complexity at ordinary PN devices because they no longer need to create, manage, or even understand foreign credentials and access policies.

6.2.1 An overview of local service access

In Figure 6.1 we see the secure interaction of two co-located PNs belonging to Alice and Bob. Alice would like to view pictures stored on Bob's camera using her PDA. We can see the two PNs are connected using a secure link layer connection between their respective gateways.

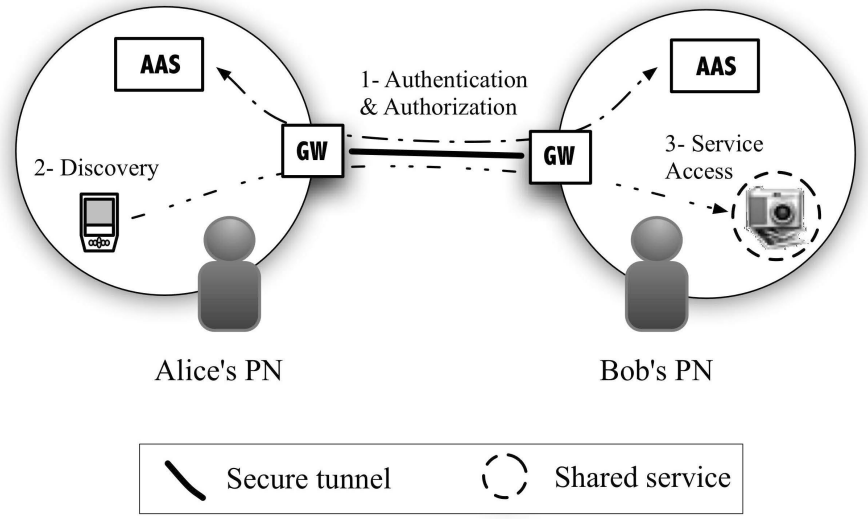

Figure 6.1: Generic architecture for local inter-PN service access

In Section 1.5.1 we explained how our approach for enforcing security in the PN

was based around making the perimeter of the PN secure, so that it could only be accessed by authorized entities. Therefore each gateway guards access to its PN and depends on the answer from a backend authentication server to allow or deny access to foreign entities. In Section 6.3 we will give an overview of the many benefits of this approach, which are related to both security and performance. The backend server, which we call the **Authentication and Authorization Server** (**AAS**) has the functionality necessary to perform authentication and authorization of foreign entities.

If the foreign entity is a PN, then authorized services can be accessed by all members of the foreign PN after a suitable discovery phase. The end-to-end service access is secured by leveraging on the security of intra-PN communication till the respective gateways, and the secure point-to-point connection between the two gateways. The ability to separate the gateway functionality from the AAS makes it possible to use the most suitable gateway device for service access. The PN owner only needs to manage one set of credentials for the foreign entity, instead of managing a separate set on each of his gateways. Also, the ability to separate the AAS functionality from the client and server means that they do not incur the overhead of creating and managing the related credentials and access policies. Later in this chapter we will look at the role performed by the AAS and in more detail. We start by restricting the AAS functionality to one device within the PN. Later we will analyze the effect of AAS unavailability as a consequence of intermittent or lack of connectivity between geographically distributed clusters, and relax this restriction by propose how multiple AASs can be accommodated in our proposed architecture.

Although we have shown the gateway, client and AAS functionality separately, these roles can be performed by one device (typically the case with PN unaware legacy devices). Such a scenario is illustrated in Figure 6.2. It is interesting to point out that in terms of the mechanisms used for authentication and service access, it makes no difference if the authenticating peer is a PN or a standalone device.

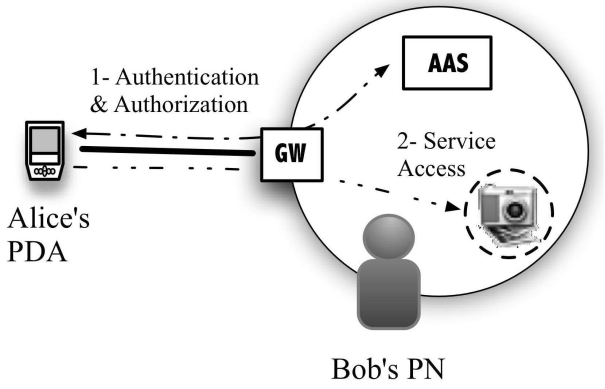

Figure 6.2: Local service access by one foreign device

6.2.2 An overview of remote service access

The two previous examples of local service access only dealt with situations where there was an OSI Layer 2 connection between the two involved gateways. However interaction can also take place between remote foreign entities as per the epilepsy patient scenario mentioned earlier. This is illustrated in Figure 6.3. Note that as in Figure 6.2 the gateway, client and AAS functionalities could also be located on one remote entity.

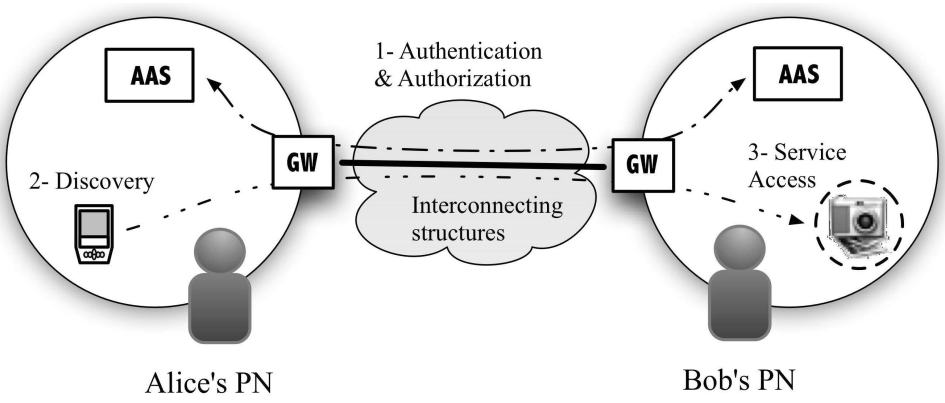

Figure 6.3: Generic architecture for remote inter-PN service access

As in Figure 6.1 the secure connection (now at OSI Layer 3) is between the two pictured gateways. However it is important to point out that there is no change in the credentials used for authentication as well as the mechanisms used by the client in Alice's PN to access the services in Bob's PN. Shielding the client and server from such knowledge means that we achieve part of our goal, which is to limit the complexity at ordinary PN devices.

In the next section we examine a standardized framework for secure service access, known as the Authentication, Authorization and Accounting (AAA) framework. The AAA framework is of interest to us because it also enforces security by making the perimeter of the network secure so that its services can only be accessed by verified users. However since the AAA framework is designed for networks with fixed infrastructure, porting it to ad hoc networks like the PN is not a simple task.

6.3 AAA in distributed ad-hoc personal environments

Authentication, authorization and accounting (AAA) is a term for a generic architectural framework for controlling access to computer resources. The AAA framework allows us to configure the three given functions (which are essential for effective network management and security) in an intelligent and consistent manner. Since our

work focuses solely on security issues we will only investigate the first two functions, specifically authenticating users and handling their authorization requests.

6.3.1 Overview

Figure 6.4 illustrates how the AAA framework is used to control who is allowed access to the network and what services they are allowed to use once they have access.

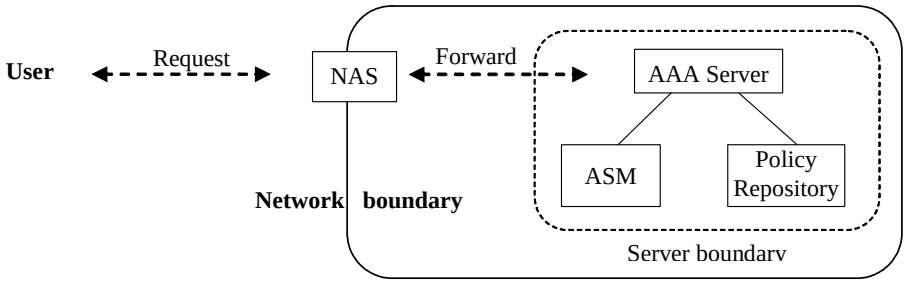

Figure 6.4: Secure network access under the AAA framework

When a user wants access to a network, he sends a request to the Network Access Server (NAS) which tunnels the request to the backend AAA server to check if the user's credentials are valid. The NAS acts as a gateway to guard access to the network and its resources and depends on the answer from the AAA server to allow, or deny access to the user. As a generic framework, it does not specify the exact authentication and authorization protocols used between the client and the NAS. The requirements AAA protocols (e.g. RADIUS [58]) must meet in order to support service authorization between the NAS and the AAA server are listed in [55].

The generic AAA architecture is centralized and consists of three basic components: the AAA server, the policy repository and the Application Specific Modules (ASMs). The AAA server has rules to inspect authentication requests and come to a decision. The policy repository contains the list of available services about which authorization decisions can be made and the policy rules to make them. Since the applications requiring AAA services may have their own unique needs in terms of configuration and initialization, the generic AAA server has an application interface to a set of ASMs. Authorization is performed by the AAA server based on the content of the policy repository and the decision by the ASMs (with their application specific knowledge). The ASMs may also use their application specific protocols to communicate with application specific service components.

The advantage of the centralized AAA approach is the ease of management for the network administrator who only needs to make changes (for both policies and user credentials) at the AAA server for them to have a system wide effect. Any changes at the AAA server e.g. to access rights are immediately reflected since every

authorization request is evaluated at the AAA server. Furthermore, from a security perspective it is also more secure to have user credentials and access policies stored in a single, secure location. When used within the fixed infrastructure, the main disadvantage of the centralized AAA approach is having a single point of failure. However this can be easily mitigated by having one or more backup servers.

6.3.2 Porting generic AAA to PNs

Given that the PN is a single administrative domain, under the generic AAA architecture we would have one AAA server for the entire PN. Foreign entities would be allowed network access to the PN after performing a successful authentication with the backend server, through any gateway (of any cluster). Once with network access, they would need to make authorization requests for each PN service they wished to access.

In the AAA framework [52] authorization can be performed using any of the three available models (1) *Agent* (2) *Pull* and (3) *Push*. Each of these leads to a different sequence for the order in which the operations are performed. In the Agent model the AAA server forwards requests between the foreign entity and the service equipment. The foreign entity sends a service request to the AAA server which, after a successful authorization, forwards it (possibly with other configuration information) to the service equipment. The service equipment is the entity that provides the actual service. The service can be something that the user considers as a service or even something that provides some functionality within the network. The latter type is usually transparent to the user. After the service equipment is prepared (e.g. sets up state and so on) to provide the service, it acknowledges to the AAA server that it is ready. The AAA server replies to the foreign entity that the service is set up. In the Pull model the foreign entity sends the request directly to the service equipment. The service equipment forwards the request to the AAA server which evaluates the request and returns an appropriate response. Service is then granted or denied based on the response. In the Push model, the foreign entity gets a token from the AAA server. Anytime that it requests a service from the service equipment, it presents the token from the AAA server as a way of verifying that it has been authorized to have access to the service.

It is noteworthy that each of these three models requires the AAA server to communicate, for each service access, its authorization response to the service equipment. Unfortunately the ad-hoc nature of the PN means that this is not very suitable. Personal devices (including the AAS) can join or depart the network at any time, while others although remaining within the PN may have intermittent reachability. As of now little research has been done in the context of AAA for ad-hoc networks so there is no standardized approach for solving such problems.

When considering the generic AAA architecture for the PN, we identify four main problems:

1. A single AAS for the PN domain is not always reachable from remote PN clusters even if it is located in the home cluster where it can always be online. We believe that the most logical way to solve this problem is to have a separate AAS in each PN cluster, which is in fact a semi-autonomous entity within the PN. However because all clusters belong to the single PN domain their rules and policies should be identical. For instance the credentials and access policy for a particular foreign entity must be the same at each AAS. Consequently we will need mechanisms to synchronize the different AASs when their state changes. State changes would be caused by the user modifying policies etc. at his/her local AAS. Note that this is different from the well known AAA setup for multi-domain environments where complex authorizations are realized through a network of interconnected AAA servers communicating via a standard protocol. In such a case AAA servers forward requests from out-of-domain users to other trusted AAA servers for evaluation. In our case all the AASs belong to one domain as they are all able to authenticate same credentials and apply identical policies.

2. Even with one AAS per cluster the unpredictability of device availability in ad-hoc networks means that a centralized server is not always reachable. This is not so much a problem for authentication because under the AAA framework authentication for network access only needs to be performed *once* for the duration of the connection, while authorization needs to be performed for *each* service access. If each authorization request needs to be authorized by the AAS, foreign entities will not be able to access services while the AAS is unavailable. Furthermore, not requiring the AAS to be involved in each authorization request will reduce the network overhead as well as the load on the AAS, and hence the cost of authorization in terms of energy. Consequently we propose to pre-approve services that are available to the foreign entity at the moment of authentication. Given that the number of services is not expected to be high (in hundreds) we donot envision any scalability issues.

3. Under the generic AAA architecture foreign entities are allowed network layer access to devices within the protected network after an initial authentication. Given the different set of requirements for PNs when compared to commercial multi-user networks, we are able to enhance security by limiting the access of foreign entities to the gateway devices, and export any authorized services through them.

4. The generic AAA architecture and its supporting protocols are designed for interaction between a standalone supplicant device (representing the user) and a network. Under the PN paradigm the user is no longer represented by an individual device but by a network of personal devices in the form of a PN. Thus, inter-PN service access represents the interaction between two networks. This requires modification of the AAA supplicant side behavior.

These proposed modifications are explained in more detail in the remaining sections. We begin by explaining at the roles performed by the gateway devices and the AAS.

6.4 Architectural components

6.4.1 PN Gateways

As a secure network the PN must enforce security safeguards at all network entry point i.e. foreign entities should not be passed information about (or allowed to take part in) internal mechanisms such as those for routing and service discovery. PN device able to perform such a function are known as gateway devices. A PN gateway guards access to the Personal Network and its resources and depends on the answer from the AAS to allow, or deny access to foreign entities. Furthermore devices within a PN can only access the outside world through PN gateways.

Even though a PN (or its clusters) may include multiple gateways we do not expect most single purpose/low cost devices to have such capabilities. What is important is that these single purpose/low cost devices do not require the gateway functionality to be implemented *locally* in-order to interact with the outside world. The benefit of having personal devices organized into a PN is that their owner can access all the different functionalities distributed across them without requiring such functionality to be duplicated in each device. The ability to offload the gateway functionality to a subset of PN devices means that we reduce the minimum functionality needed for a device to be PN capable. This is importance for designers of low cost/low form factor SAI devices. Needless to say, the disadvantage of this approach is that without at least one gateway device in the cluster it cannot interact with the outside world. However since forming a cluster requires at least one multi-functional device (e.g. SAC devices) we can expect such a device to be gateway capable.

Gateway devices also play a role in discovering neighboring foreign clusters. In Chapter 3 we explained how depending on the cluster policy cluster gateways may automatically forward cluster advertisements received from neighboring personal clusters to their security agent. However cluster gateways also receive cluster advertisements of non-personal clusters. Although such cluster advertisements are not automatically forwarded, the information included within these advertisements is stored for a limited time and made available as a "foreign-PN" discovery service. Therefore in Figure 6.1 the first step for Alice before establishing a connection with Bob's PN would be to use her AAS to discover all neighboring PNs by utilizing the discovery service offered by her gateway devices. She may then select one or more gateways (for redundancy) to establish a connection with Bob's PN. The AAS will then initiate the process of authentication through each of these gateways.

6.4.2 AAS

We have stated that the functionality for authentication and authorization of foreign entities is found in a personal device which we call the Authentication and Authorization Server (AAS). The AAS functionality is optional in that it is only required for foreign service provisioning and access. For example in Figure 6.1 Alice will need to have AAS functionality in her PN if she wants to access any services in Bob's PN or if she wants Bob to be able to access services in her PN.

In Chapter 1 we decided that our basic security mechanisms for secure PN formation and communication must be based on purely symmetric cryptography, so that even small devices like bio-medical sensors could potentially be part of a PN. However we will relax this constraint for the AAS because:

- For a PN to use or offer services to a foreign entity, only few of potentially many personal devices needs to be AAS capable. Given the heterogeneous nature of devices in a PN, requiring the AAS to be capable of asymmetric cryptography should not be a limiting factor.

- Unlike the limited set of personal devices, the set of foreign entities interacting with a PN can be much larger and more dynamic. Mechanisms based on symmetric cryptography are not always practical; in many scenarios asymmetric cryptography in the form of widely used PKI is more suitable.

Next we take a detailed look at how two AASs involved in inter-PN service access can be configured with appropriate credentials required for mutual-authentication.

6.5 Pre-Authentication – Creating trust

The two basic types of authentication credentials supported by the PN are (a) pre-shared pair-wise keys and (b) a private key where the corresponding public key is either stored at the authenticating AAS or validated at the AAS using a digital certificate. The pre-existing trust relationship between the two authenticating AASs is used to carry out an authentication and authorization sequence that establishes a dynamic relationship between the service end points.

The exact type of credential used or created for a given situation depends on the real life needs of the users involved in the authentication. For instance, we expect public key signatures to be the standard option for authentication between two users who know each other. This is because a public key is verifiable cyber identity of a person that can be easily distributed, a lot like a business card. However there are situations where for privacy reasons people may not wish to reveal their identity, for instance during brief interaction (e.g. sharing music) between two people seated

together on the train. In such a case it would be best to use pair-wise keys with short term validity.

Authentication both for local and remote service access takes place using the Extensible Authentication Protocol (EAP). The EAP authentication framework is ideal for use with backend authentication servers like the AAS. It can be used with a number of authentication methods such as EAP-PSK (Pre Shared Keys) and EAP-TLS (Transport Layer Security), the exact method used depends on the type of credentials being used for authentication. Another benefit of using EAP authentication is that we can easily handle potential scenarios where the two basic types of credentials, specifically public key signatures and shared secrets, are not flexible enough to meet the requirements of the deployment scenario.

Clearly the owner of the PN is the only one to configure access control for his services. However the establishment of trust between the PN and the foreign entity can be leveraged on the services of a trusted third party (TTP) or it can be manually configured by the two parties. These two options are illustrated in Figures 6.5 and 6.6 respectively.

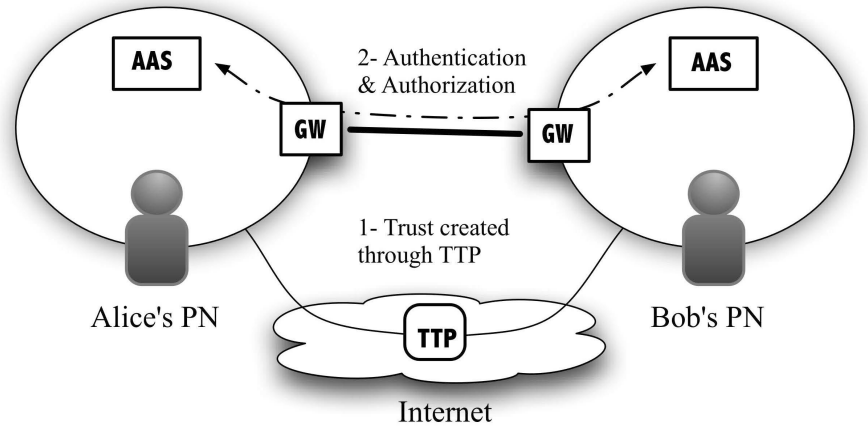

Figure 6.5: Using a TTP to create trust

The TTP concept, usually in the form of a public PKI [66], is well understood and involves the use of a mutually trusted entity to facilitate interaction between two parties that do not have a direct trust relationship. However given the financial cost of obtaining such certificates, the requirement for infrastructural connectivity and the overhead associated with following chains of trust for certificate validation, we do not expect this approach to be the dominant option given the rather informal nature of the PN.

Thus we expect the second approach, which creates direct trust relationships based on user facilitated secure exchange of credentials, to be the dominant option for securing PN interaction. Our main requirements are that the proposals to bootstrap this trust must be user-friendly and intuitive. In Section 6.5.1 and 6.5.2 we look at

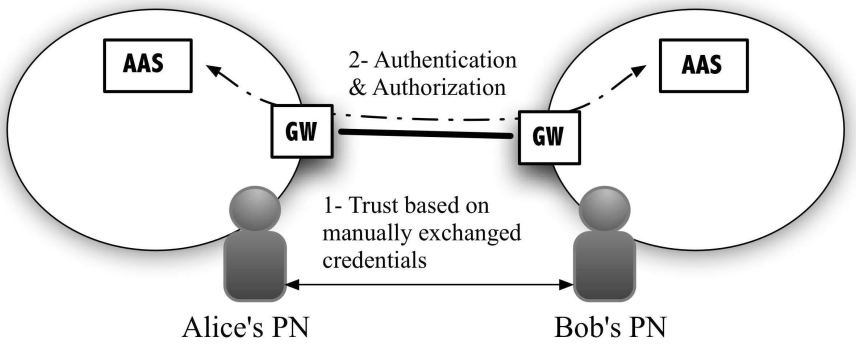

Figure 6.6: User assisted creation of trust

user assisted mechanisms to exchange pair-wise and public keys respectively.

6.5.1 Exchanging pair-wise keys

We recommend using pair-wise keys between two *co-located* users when the relationship is *anonymous* in nature and meant to be *short term*. If the relationship is not anonymous then it is better to use public keys (Section 6.5.2) because then the users need to maintain fewer credentials. The secure exchange of pair-wise keys between two co-located devices (in our case the AASs) has been well researched. We have already looked at mechanisms based on location limited side channels (LLSC) in the context of device personalization (Section 3.3.2). Typically it involves the user in setting his device in a certain state where it is (a) able to receive the key or (b) the information to derive the key. There is of course the more trivial solution where prior to the actual communication the two users enter the key by hand into the two authenticating devices. However it is easier, less error prone and more secure to do this with minimal user interaction.

Bluetooth is one of few technologies which has specified a key management mechanism, with which, the users of two Bluetooth devices can establish a shared secret link key. This mechanism, also called pairing, uses a short user generated code (normally 4 digits) that is entered to a pair of Bluetooth devices. The cryptographic algorithms are all symmetric key algorithms, which means that the resulting link key is at most as strong as the initial user generated code. Since this code (which must be kept secret) is handled by the user it is normally accepted that an adequate level of security is thus difficult to achieve.

Given the varying properties of different location limited side channels it is tempting to propose multiple mechanisms for exchanging pair-wise keys, each suitable for a specific type of communication medium. For instance with a location limited side channel that ensures the secrecy of key transfer, it is possible to transfer the key in plain text. For instance Stajano and Anderson [21] propose a solution that uses

physical contact i.e. using electrical contacts, to transfer the pair-wise key in plain text. This ensures that there is no ambiguity about which two entities are involved in the relationship. Other location limited side channels like infrared do not have this property of secrecy, so a totally different mechanism must be used.

The user as the LLSC

We believe that in order for our proposals to be user-friendly it is important to select *one* mechanism that is suitable in a general case. Since all devices may not come with the same hardware supported location-limited side channel (or perhaps any such channel), our proposed solution will employ the user itself for this role. Since people represent a notoriously low-data rate (and error prone!) channel any workable solution should minimize the amount of information they are required to exchange. Thus while some limited information necessary to commit to keys can be exchanged manually by the users, the actual key exchange must be done over the main wireless link.

The most well known mechanism for exchanging pair-wise key between two users over an insecure communications channel is the Diffie Hellman protocol [89]. However the Diffie-Hellman protocol by itself does not provide authentication of the communicating parties and is thus vulnerable to a man-in-the-middle attack. Since there is no way for Alice to guarantee that she is communicating with Bob (and vice versa) an attacker can establish two separate Diffie-Hellman key exchanges, one with Alice and the other with Bob. This will allow him to masquerade as Alice to Bob (and vice versa) by decrypting and then re-encrypt the messages passed between them. This type of attack can be prevented by having some way of validating that the key derived by both Alice and Bob is the same. This can be done by utilizing the user as a location limited side channel.

The IST MAGNET [8] and IST SHAMAN [29] projects have both proposed establishing pair-wise keys using mechanisms based on authenticated Diffie-Hellman protocol. Both utilize the users as the location limited side channels to retrieve information displayed at one device and enter it into another device. Importantly, even though the pair-wise keys generated are of sufficient length the users do not need to enter long codes. We have short listed both the PAN formation protocol (PFP) proposed in [8] and the MANA III mechanism proposed in [29] as two possible means to exchange pair-wise keys between two co-located users. The main difference in the MANA III and the PFP mechanism is that the latter uses the short code *during* the Diffie Hellman exchange, while the former *after* the exchange. However both mechanisms appear much the same to the user because both require the user to enter a short code into his AAS and then wait for either an accept or reject message. This means that an AAS must have an input interface for entering short hexadecimal string as well as an output interface to display this string and the result.

With the PFP mechanism Alice acts as a location limited side channel and en-

ters a 5 hexadecimal digit code displayed on Bob's AAS into her own AAS. Bob's AAS will then use this code to append an HMAC to the exchanged Diffie Hellman messages, thus ensuring mutually authentication. With the MANA III mechanism, the authentication actually takes place as a separate step performed *after* the completion of the Diffie Hellman protocol. One of the users selects a short code (at least 4 hexadecimal digits) which is then entered in both devices. This code together with a random number and a known Message Authentication Code (MAC) function is then used to verify that the pair-wise key is the same in both devices. As a slight modification, we believe it would be better for one of the devices to generate this short code, because users tend to repeat codes or use simple combinations.

6.5.2 Exchanging public keys

Using public key signatures for authentication requires the two parties to mutually authenticate using their private keys. In the next two sections we see how the corresponding public keys can either be configured manually at the AAS or exchanged dynamically using verifiable digital certificates like with SSL. Note that SSH-type protocols do not solve the problem of exchanging public-keys because the transferred public key is unauthenticated and in principle requires the user to use a suitable side channel (like telephoning the helpdesk) to verify it. However this is rarely done in practice. This public key is then cached over subsequent connections and a warning is only raised if the new public key received prior to connection establishment does not match with the cached one.

Of course, as with pair-wise keys the AAS must be configured by the owner of the PN to allow the appropriate access rights to users that can prove possession of the corresponding private key.

Manually exchange

Similar to the exchange of pair-wise keys, it is convenient for two co-located users to exchange their public keys using themselves as location-limited side channels. As before, the two users do not need to exchange their complete public keys across the side channel but only need to commit to those keys by exchanging some limited information. Ideally we would like our chosen method be similar to that proposed for exchanging pair-wise keys so that users can use a familiar mechanism for both cases.

The MANA III mechanism explained earlier is quite suitable in this case because it only operates after the key material has been exchanged. We propose a simple modification to the original, whereby the users initiate the MANA III mechanism after exchanging their public keys over an insecure wireless channel. This is illustrated in Figure 6.7. As before, since the users configure the public keys manually and are not using a public key infrastructure, automatic revocation checks based on

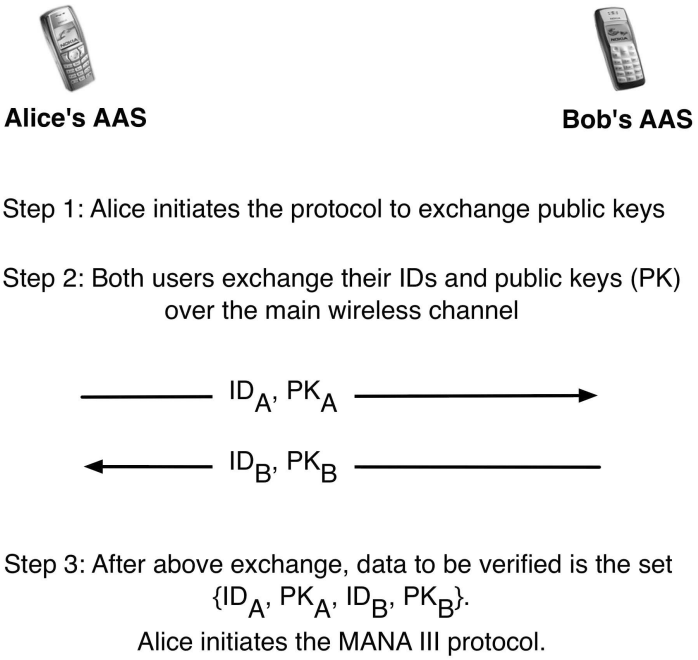

Figure 6.7: User assisted exchange of public keys

online access to the revocation lists issued by relevant certification authorities do not take place. Thus revocation of manually exchanged public keys also needs to be done manually by the users.

The manual exchange of public keys, as described above is most useful where online access to a PKI is not available. However interaction between two entities, which are no co-located usually takes place over supporting infrastructure so it is reasonable to assume access to a PKI. Nevertheless we do consider the case where one or both entities may not have, or wish to use their digital certificate. If the two users are not co-located it becomes awkward to exchange public keys in a secure manner without the use of verifiable digital certificates. However if the two parties have access to a secure means of exchanging data the public key can, in principle, be exchanged over that channel. For instance although not completely secure, it is common for small organizations to distribute their self-signed certificates through official websites. Similarly exchanging public keys through email or online messaging, although not fully secure even if coordinated between the two parties, may be acceptable to a large number of people due to ease of use.

Dynamic exchange using certificates

Public keys can also be exchanged dynamically (using verifiable digital certificate) as a standard part of an authentication protocol e.g. EAP-TLS. Since digital certifi-

cates based on the existing PKI are costly, both in financial terms and the overhead of following the chain of trust, it may be better to have a separate PN specific PKI. For the purpose of this work we assume that there is a single PN certification authority (PN-CA), so there are no chains of trust to follow.

Services of PN-CA could be provided as part of a subscription with a PN service provider. In such a case we expect most users to register themselves with a PN service provider, usually in the beginning of the PN life cycle. Revocation checks will of course require online access to the revocation lists issued by this CA. However users may enable admission even in case revocation checks cannot be performed due to unavailability of online access.

An important benefit of using certificates is that they make it possible to provide services to a group of entities without requiring the user to configure access for each entity individually. This is useful if, for instance, the owner of the PN is a member of a club and wishes to allow all other members a certain service. The owner of the PN can then configure access for the entire group based on the certificates issued by the CA representing the club. Most likely, the owner of the PN will have to manually install the self-signed certificates of the club CA.

6.6 Authentication

6.6.1 Authentication for local access

Instead of proposing a novel protocol specific to PN authentication (and introducing the possibility of security flaws common to new protocols) authentication for local access is based on the IEEE 802.1X authentication framework and the Extensible Authentication Protocol (EAP). This allows us to take advantage of all of the existing work in protocol design and security analysis.

IEEE 802.1X overview

IEEE 802.1X [101] is a standard for port based network access control of IEEE 802 LANs. It is the most suitable candidate for authentication in our model because it leverages backend authentication servers and EAP. Furthermore, it works at the OSI Layer 2 to allow or deny access to the network through what in IEEE 802.1X terminology is called the **authenticator** (gateway). The authenticator is the intermediary that facilitates authentication between the **supplicant** (foreign entity) and the **authentication server** (AAS). Messages belonging to the actual authentication method are transported between the supplicant and the authentication server encapsulated in EAP messages. The authenticator and the authentication server can be one or two separate entities. If the authenticator is also functioning as the authentication server, all EAP messages are processed locally. Otherwise the au-

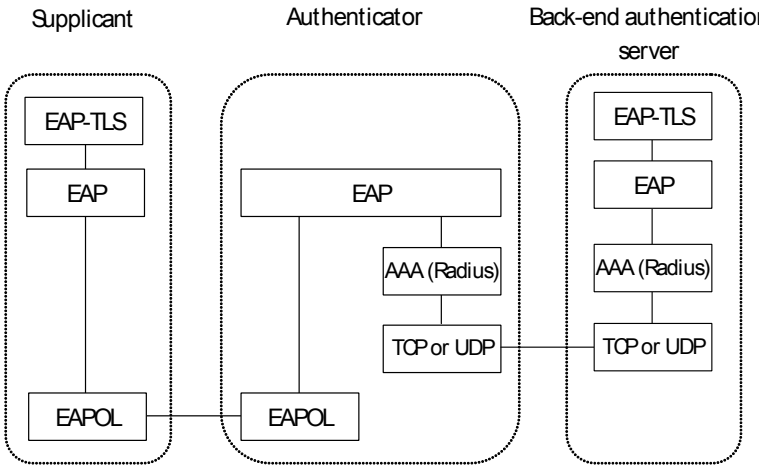

Figure 6.8: Protocol stack of EAP-TLS authentication when used with IEEE 802.1X

thenticator acts as an EAP pass-through and the authentication server is referred to as a back-end authentication server. In the latter case, the back-end authentication server is generally part of an AAA server and communication between the authenticator and the back-end authentication server is done over a suitable AAA protocol. For clarification, we would like to point out that although the lines amongst the three AAA functions have been blurred with regard to IEEE 802.1X, in reality it only provided the authentication piece.

Depending on the result of the authentication between the supplicant and the authentication server, the authenticator may allow or deny access to the supplicant. Specifically, the authenticator waits for the *EAP-Success* or *EAP-Failure* messages from the server indicating the success or failure respectively of the EAP authentication. Along with its decision to allow access to the supplicant, the authentication server also exports associated parameters, such as the Master Session Key (MSK) for use post-authentication to protect link layer communication between the supplicant and the authenticator. Figure 6.8 illustrates the protocol stack of the EAP-TLS [56] authentication when used with IEEE 802.1X.

The keying material is transported from the backend authentication server to the authenticator and used to establish a unicast security association between the supplicant and the authenticator. For instance, IEEE 802.11i [68] uses the MSK to derive Transient Session Keys (TSKs), which are used by the TKIP and AES cipher suites to protect communication between the supplicant and the authenticator. Since the TSKs are derived from the MSK using the 4-way handshake that includes a nonce exchange, the MSK can be reused to generate new TSKs periodically without requiring an EAP re-authentication. This fact is important in our design because the AAS may not be available for entire duration that the foreign entity is connected to the PN. However the maximum lifetime of TSKs cannot be more than of the MSK,

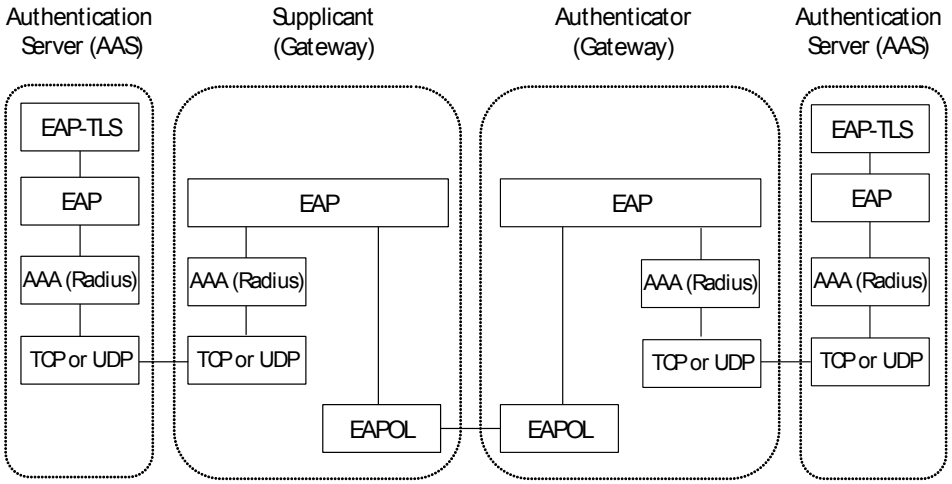

Figure 6.9: Modified protocol stack to support inter-PN authentication

meaning that for long duration connections it is necessary for the foreign entity to re-authenticate with the AAS. Although the life time of the exported keys can be managed as a system parameter, a default lifetime of 8 hours is recommended [51].

IEEE 802.1X for PNs

Avoiding PN-enhanced protocols means that foreign entities that are PN-unaware will also be able to access PN services. However existing mechanisms for secure network access are designed with the assumption that the user is represented by just one client device; a paradigm that no longer applies when working with PNs. The IEEE 802.1X framework as implemented today is adequate for providing mutual authentication in the scenario where PN services are being accessed by one foreign entity, as illustrated in Figure 6.2.

Fortunately inter-PN authentication (Figure 6.1) only requires minor modifications to the IEEE 802.1X supplicant functionality and AAA protocol signaling. First, the AAA protocol used for communication between the AAS and the supplicant must be able to signal the supplicant to initiate the IEEE 802.1X authentication. Second, the EAP module at the supplicant must be able to act as an EAP pass through. This allows authentication between the two AAS, and not between the supplicant and the AAS. The supplicant will be dependent on the AAA protocol to know the outcome of the conversation (from its AAS), since it does not look at the encapsulated EAP packets. Along with the outcome, the AAS will also have to transport the MSK exported by the EAP method to the ciphersuite module at the supplicant. Figure 6.9 illustrates the modified protocol stack used for inter-PN authentication.

The sequence of messages corresponding to the above protocol stack is illustrated

in Figure 6.10. Once IEEE 802.1X authentication is initiated by the supplicant, EAP messages will be forwarded at both ends of the EAPOL [68] exchange, making possible the authentication between the two backend AASs. Both the supplicant and the authenticator will wait for the *EAP-Success* or *EAP-Failure* messages from their respective AASs. The dotted lines show the EAP messages that are relayed back and forth by the gateways. The only new functionality are the *Connect* and *Disconnect* messages, which tell the personal gateway to connect or disconnect from the specified foreign gateway (F-GW). Information related to which foreign gateway connects to which foreign PN is retrieved using the "foreign-PN" discovery service, which is offered by all personal gateways (Section 6.4.1).

This means that it is possible to make the PN look like a conventional WLAN network, such that legacy foreign devices can use standard mechanisms for establishing connectivity. With PN to PN authentication, what is important is that the functionality required for the transport and wrapping of EAP messages and AAA parameters between the gateways and the backend AASs already exists in the form of AAA protocols such as RADIUS and DIAMETER. Thus we do not propose fundamental modifications to the IEEE 802.1X framework, or new EAP authentication methods.

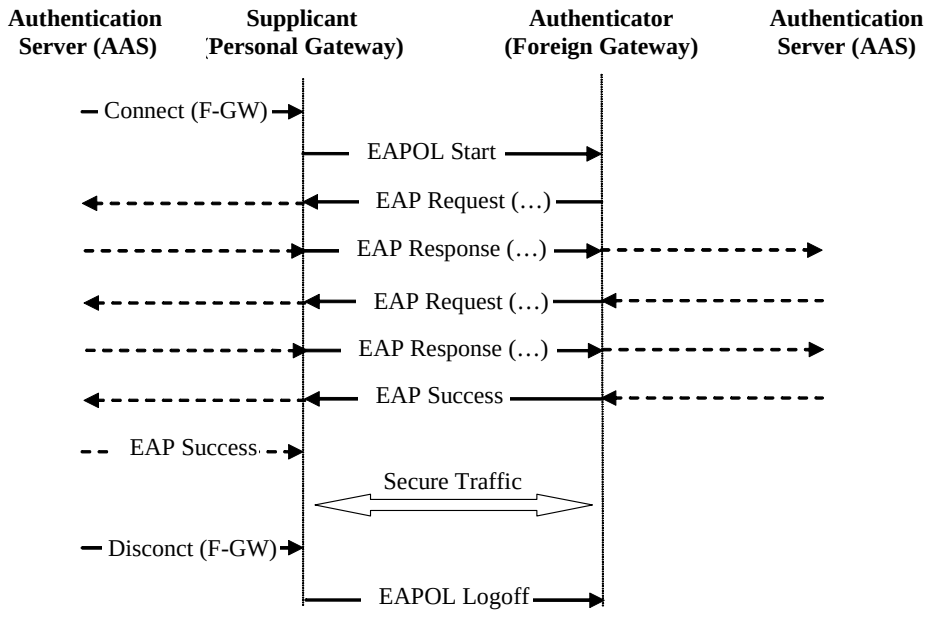

Figure 6.10: Message sequence for inter-PN authentication

6.6.2 Authentication for remote access

The mechanisms proposed in Section 6.6.1 cannot be used for remote access because the IEEE 802.1X framework works at Layer 2 of the OSI model, whereas the two gateways in Figure 6.3 only have Layer 3 connectivity. Consequently, the protocol stack for remote authentication will look identical to Figure 6.9 except that EAPOL will be replaced with a suitable OSI Layer 3 protocol. This allows us to reuse the same credentials and EAP authentication method, only the protocol used to establish the secure tunnel between the two gateways (based on EAP derived keying material) will be different. To put it simply, what we need is a protocol for creating a secure tunnel between the two gateway devices which supports EAP authentication. In Chapter 5 we stated that IPSec and TLS were the only two standards based approaches for creating secure tunnels. Between these two, only IPSec supports EAP authentication. The component of IPSec used for performing authentication and establishing security associations is called IKEv2 (Internet Key Exchange version 2) [94] and has two basic forms. The first one consists of four messages, and supports scenarios where the communication parties authenticate with each other using public key certificates or shared secret keys. The second form makes use of the EAP protocol for client authentication, and potentially accommodates any EAP type. This suits us because in Chapter 5 we already selected IPSec for creating tunnels between remote clusters, so this functionality already exists in PN gateways.

IKEv2

Using EAP authentication not only makes it possible to separate the tunnel endpoint (gateways) from the authentication endpoint (backend AASs), it also makes our architecture flexible enough to meet the requirements of unforeseen deployment scenarios requiring alternate authentication methods. It is much easier to update the AAS to support alternate authentication methods than requiring IPSec updates in the gateways. Figure 6.11 shows details of the IKEv2 message exchange when using EAP authentication. The dotted lines show the EAP messages relayed back and forth by the responder. The initiator and responder represent the secure tunnel endpoints.

Table 6.1 described the notation used in Figure 6.11. The SA payload states the cryptographic algorithms supported by the sender and responder. The KE payload includes their public Diffie-Hellman value. Payloads that are optional appear in square brackets, while the notation SK { ... } indicates that these payloads are encrypted and integrity protected using keys derived from initial the Diffie-Hellman exchange (messages 1 and 2).

The messages exchanged between the initiator and the responder to establish an IKE security association (IKE_SA) are categorized into the IKE_SA_INIT and IKE_AUTH exchanges. The IKE_SA_INIT exchange negotiates the security pa-

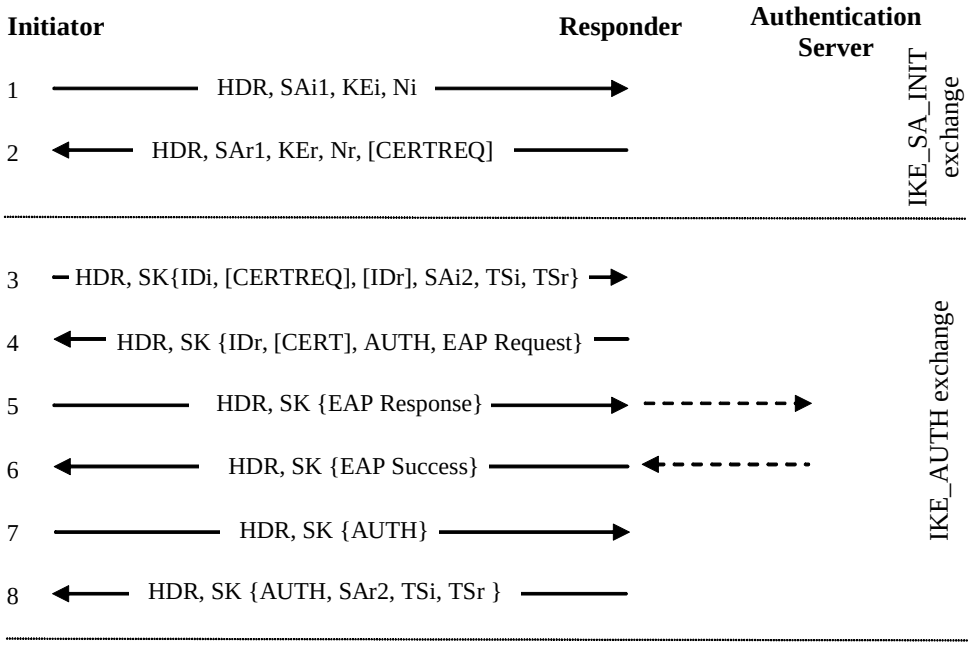

Figure 6.11: IKEv2 message exchange when using EAP authentication

Table 6.1: Notation used in IKEv2 message exchange

Notation	Payload	Notation	Payload
$AUTH$	Authentication	IDi	Identification - Initiator
$CERT$	Certificate	IDr	Identification - Responder
$CERTREQ$	Certificate Request	Ni, Nr	Nonce
SA	Security Association	TSi, TSr	Traffic Selector
HDR	IKE Header	KE	Key Exchange

rameters for the IKE_SA, sends nonces and Diffie-Hellman values. The IKE_SA is the result of this Diffie-Hellman exchange. Once established the IKE_SA is used to establish the security associations for Encapsulating Security Payload (ESP) [88] and/or Authentication Header (AH) [87]. The IKE_AUTH exchange authenticates the Diffie-Hellman exchange by transmitting identities knowledge of the corresponding secrets. Authentication is performed using the *AUTH* payload which is actually a MAC created using keying material generated by a successful EAP authentication and a block of data which includes the nonces received earlier.

Unfortunately in order to support legacy EAP methods that only provide unilateral authentication e.g one-time passwords or token cards, the IKEv2 standard specifies that EAP authentication must be used together with public key signature based responder authentication. This is clear from Figure 6.11 where the responder is shown to be authenticating using certificates. However since newer EAP methods do provide mutual authentication and shared key agreement there are proposals [103] to support responder authentication based on EAP (see Figure 6.12).

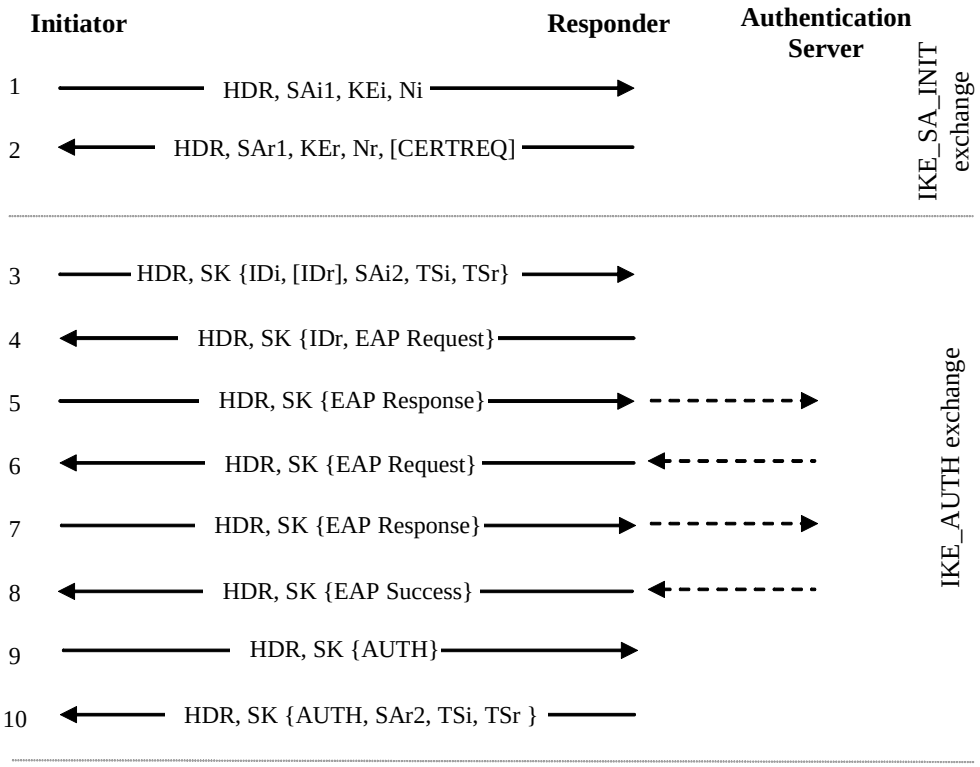

Figure 6.12: Modified IKEv2 messages supporting mutual EAP authentication

After a successful EAP authentication the generated keying material is sent from the back-end authentication server to the responder using the AAA infrastructure. It is then used to authenticate the IKE_SA based on the *AUTH* payload. It is

important to note that the IKE_SA is not authenticated by a successful completion of the EAP authentication itself, but a valid exchange of the $AUTH$ payload based on the EAP-generated keys.

Modifying IKE message exchange for PNs

Although the modifications proposed in Figure 6.12 suit our requirements for only using EAP as the authentication protocol, reuse of the IKE_SA_INIT means that the Diffie-Hellman exchange is still taking place. Unlike the actual EAP authentication, this exchange takes place at the tunnel endpoints which are the PN gateways. In Chapter 5 we explained why it was beneficial to use IPSec with KINK in order to avoid asymmetric operations at PN gateways. Since EAP methods already provide mutual authentication and key agreement, we believe it is possible to omit the Diffie-Hellman exchange without adverse affects and instead use the EAP generated keys for generating the session keys for IPSec SAs. Figure 6.13 shows details of our proposed modifications. Note that SK { ... } now indicates payloads that are encrypted and integrity protected using keys derived from the EAP exchange.

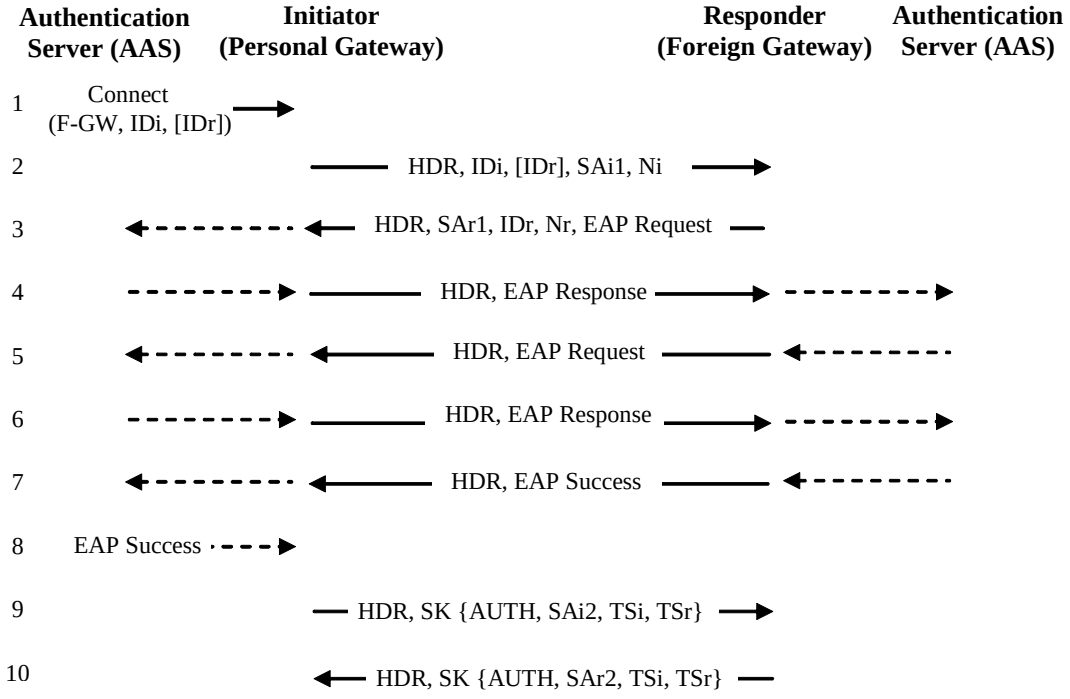

Figure 6.13: Modified IKEv2 message exchange to support inter-PN EAP authentication

When compared with the Figure 6.12 there are two main changes:

1. IDs are sent in clear text

2. The EAP exchange is no longer protected using the Diffie-Hellman key

For the first, the IKEv2 message exchange in Figure 6.12 provides identity confidentiality against passive attackers by encrypting them using the Diffie-Hellman key. However it cannot provide any protection against active attackers pretending to be the initiator or the responder since authentication only takes place in message 9 and 10, while identities are revealed in messages 3 and 4. Thus the main disadvantage of our modified proposal is that we do not provide identity confidentiality against passive attackers. However we believe this is not a serious disadvantage because e.g. when EAP-TLS is used in the IEEE 802.1X framework the responder always sends its certificate in clear text and consequently exposes its identity to eavesdropping. For the second, although the EAP payloads in Figure 6.13 are not protected using the Diffie-Hellman key it is not a disadvantage because in the IEEE 802.1X standard the EAP payloads are neither encrypted nor integrity protected either (at the link layer), so EAP methods are designed to take that into account. In summary, we find it acceptable that the security offered by authenticating for remote access is the same as that for local access (based on IEEE 802.1X and EAP). Further modifications to IKE are limited to the *order* in which the IKE control information is transferred. An important point to note is that because the security of the system remains based on the EAP mechanism actually used it is not necessary to carry out a new security protocol analysis.

6.7 Authorization

A PN device that offers services to foreign entities has to make access control decisions about which of its services are available and under what conditions. Typically, the serving device can do this by using one of the following methods:

1. It is configured with appropriate credentials and policy statements to make access control decisions locally

2. It obtains online verification for each service request from a policy server or the PN owner

3. It requires being presented with access tokens that prove it is OK for the foreign entity to access the requested service

4. It only allows access through a trusted agent, which is then required to implement access control per one of the above three methods

We feel that the first method is not feasible. Not only will it require the user to configure access policies and foreign credentials on each accessible device, it also

requires such device to have a user interface suitable for this purpose. Given the number and heterogeneity of devices in the PN, this is not a reasonable requirement.

Furthermore, we feel that the second method is also not feasible. In Section 6.3.2 problem statement 2, we pointed out that the unpredictability of device availability in ad-hoc networks means that a centralized server is not always reachable for authorizing *each* service request. Additionally, requiring the PN owner to authorize each service request is not only inconvenient but also not possible when considering service access in remote clusters.

For these reasons we proposed pre-approving services at the time of authentication using user configured access policies at the AAS. This would allow us to enjoy all the benefits of a centralized single sign-on system, vis-à-vis the ease of management and low overhead. The third and fourth methods present interesting possibilities in this regard, which are investigated in the next two sections.

6.7.1 Access Tokens

A token is defined as something a claimant possesses and controls which can be used to verify the claimant's identity. Access tokens e.g. SAML tokens [104] also contain conditions that apply for invoking services, thereby giving the holder the capability to access the target service. In other words, an access token contains the credentials necessary for service access, the list of services that are allowed access and the permissions under which access is allowed.

If our authorization scheme is based on the use of access tokens, we envision each AAS to generate access tokens for its peer immediately after a successful authentication. Since each token is protected with a keyed hash (HMAC) generated using a symmetric key known to the serving device and the issuing AAS, there will be one access token generated per serving device. This is because use of a static shared symmetric key is not secure, and tokens cannot be protected using asymmetric cryptography. Furthermore, since the list of authorized services may change each token is only valid for a limited duration.

In terms of proof-of-possession (PoP) of the token we have two options. The token can include a symmetric proof key or no proof key (i.e. bearer token). We have already stated that the AAS will sign the token to indicate its veracity. In addition, if used with a symmetric proof key the token will be cryptographically bound to the holder of the proof key. Thus the presenter of the token (subject) must prove knowledge of this key before service access is granted. An attacker intercepting a proof key token will find it worthless without the proof key. On the other hand a bearer token is not cryptographically bound to the subject. In other words, whoever presents it (i.e. the bearer) is considered to be the subject, or at least acting on behalf of the subject. Consequently bearer tokens must be protected in transit using lower layer security.

Both approaches have their advantages. The main advantage of the proof key token is that it allows the service endpoints to use an extra layer of security on top of the default PN security. The main disadvantage of course is that additional end-to-security has its overhead, and given that intra-PN service access uses the default PN security mechanisms, it is difficult to justify.

Figure 6.14 illustrates our representation of a proof-key token. The white fields are not part of the actual token and include information meant for the subject; specifically the ID of the serving device and the proof key. The token (grey part) also includes a copy of the proof key encrypted with the same secret key used to create the HMAC. As stated earlier, these token are transferred between the AAS and its authenticating peer over a secure channel immediately after authentication.

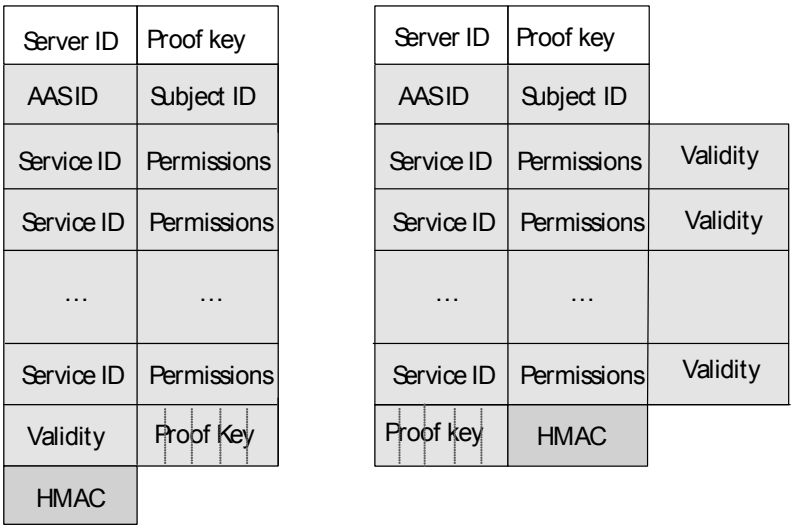

Figure 6.14: First proof-key token format has one validity value for all services. Second allows each service to have a different validity value. A comparable bearer token just omits the two proof key fields.

If the peer represents a single foreign device, then the mechanisms used for service access are straightforward. Upon service request the foreign device will present the serving device with the token and if necessary authenticate with the proof key. The situation becomes more complex if the authorized peer represents a PN. Since authorized services should in principle be accessible by every device in the peer PN, there must be a mechanism to allow other PN devices secure access to the tokens. As both types of tokens require a secure channel for distribution (bearer tokens must be protected in transit, and proof key tokens need to be distributed with their proof key) the distribution mechanism will be the same for both. Since creating a security association between the AAS and each client device will be costly, we expect this distribution to use the underlying PN security.

The tokens itself should be stored at gateway devices to ensure their availability

as long as the underlying PN to PN connectivity is available. The services contained in the tokens should be made available to the default service discovery method, whatever that may be. Once other PN devices identify the foreign service they wish to access they must retrieve the token, and present it during the service request.

Unfortunately because an issued token cannot be easily revoked, modifying selected access permissions once the token is issued is not possible. The only option would be to revoke the foreign entity entirely (i.e. deny it PN connectivity) till the tokens expire. Besides that, the main disadvantage of using tokens is that they require maintaining a security association between the AAS and each serving device. Furthermore, they do not meet our requirement as laid out in Section 6.3.2 problem statement 3. Next we look at an approach based on using a trusted agent.

6.7.2 Trusted Agent

If authorization is performed based on the trusted agent method, then the trusted agent is required to act as service proxy and implement access control on behalf of the serving device. The serving device is configured to allow the trusted agent access to all proxy-able services based on a security association existing between the two.

In the introduction to this chapter we explained how in our model intra-PN service access was secured by leveraging on the security of the underlying intra-PN communication network, and that all personal devices had access to the complete set of non-administrative PN services. Consequently when implementing the trusted agent method in the PN, we do not require a new security association between the serving device and the trusted agent or modification to the default access control.

Location

We have chosen to position the trusted agent functionality at the gateway devices. This implies that the trusted agent functionality will be availability as long as the underlying connection between the PN and the foreign entity is available. Additionally, by exporting authorized services through gateways we can remove the need for foreign entities to have direct access to non-gateway PN. Thus by presenting a more controlled and limited interface to the outside world we make the overall architecture more secure. This meets our requirement as laid out in Section 6.3.2 problem statement 3. Access control between the trusted agent and the foreign entity can be implemented using different techniques. We present our proposal in the next section.

Access Control

When considering the method used by the gateway to authorize foreign service requests, we can leverage on the fact that as the endpoint of the secure tunnel

connecting the foreign entity to the PN, the gateway device is acutely aware of the foreign entity's identity. Consequently if the AAS provides the gateway with a list of services that the foreign entity is allowed to access, the gateway can enforces this without requiring additional authentication. Based on its authorization decision, the gateway will either allow or deny service requests coming from the foreign entity. Once authorization it allowed, it will proxy the communication by maintaining some state information per service session.

At the client side, the two AASs exchange a list of authorized services immediately post-authentication and integrate them into their local service discovery mechanism. Such services are listed as being located at the target gateway devices. In terms of the behavior of the service end-points, the mechanisms they use for inter-PN service access remain the same as intra-PN service access. This simplifies the functionality at the serving device which does not need to implement access control itself and instead treats the service request from the gateway identical to other intra-PN service requests. End-to-end security is provided in two steps, traffic between the service endpoints and their gateways is secured using intra-PN security, and between the two gateways using the secure tunnel.

The main disadvantage of the trusted agent approach is that it increases the overhead at the gateway devices which must evaluate and proxy service requests. However based on the advantages listed above, we have selected this as our chosen method for access control.

6.8 Multiple AASs

Up till now we have assumed a single and always reachable AAS which stores the credentials and access policies enabling secure interaction with foreign entities. Given that the trust creation mechanisms described in Section 6.5 require user interaction, it is tempting to have this AAS located in the P-PAN. However this has several drawbacks:

- As stated in Section 6.3.2 problem statement 1, the dynamic nature of the PN means that a single centralized AAS may not be reachable from remote clusters. Thus, until connectivity is reestablished foreign entities will not be able to access services located in these clusters.

- Storing the PN's PKI key set on a mobile device functioning as an AAS in the P-PAN means that if this device were to be misplaced or stolen, all existing trust relationships with foreign entities will have to be revoked and then reestablished with a new set of credentials.

- If we restrict the AAS functionality to one device in the P-PAN we risk having one point of failure, while having one or more backup AASs requires the

duplication of the PN's private key over multiple mobile devices, which is not safe.

On the other hand, if we decide to avoid these problems by placing the AAS at a more secure location in the infrastructure (e.g. a home AAS), not only will we still have issues associated with reachability and increased delay but we will also have to implement some mechanisms to proxy the creation of initial trust. We believe that the best way to solve the reachability problem is to support a separate AAS in each mobile cluster, which is in fact a semi-autonomous entity within the PN. However, for security reasons the PN's PKI key set will only be stored at a single secure location. Details are presented in Section 6.8.2.

Since all clusters belong to the single PN domain, credentials and access policy for a particular foreign entity must be the same at each AAS. When using PKI this is not an issue for foreign credentials because they will be represented by the foreign entity's *public* key. However duplicating the PN's own PKI key set over multiple mobile devices is not safe; therefore we feel it is necessary to enable unique identification of each AAS. This means that revocation of a lost AAS will not require the PN owner to sever existing trust relationships with foreign entities, but only to indicate the PN's updated revocation status to such entities.

6.8.1 Synchronization

With multiple AASs representing the same administrative domain, there needs to be a mechanism to synchronize state changes. State changes are caused by the PN owner adding new foreign credentials and/or modifying access policies at his local AAS. From an increased security and reliability perspective, we propose using an infrastructure supported AAS as the central point of synchronization. Such an AAS, henceforth referred to as the master AAS, would typically be a part of the PN agent functionality (Figure 1.3). The other mobile AASs will not synchronize directly with each other but only with the master AAS. Thus not only will the master AAS perform the rather important role of a secure backup, but by maintaining a log of updates performed by each AAS allowing erroneous entries by any AAS to be easily identified and rolled back. All this ensures that the loss of an AAS does not affect the PN anymore than the loss of any other personal device.

For clarification, at a minimum we propose to have one infrastructure supported master AAS and one mobile AAS in the P-PAN. What is important is that our proposed architecture supports the use of a local AAS in each cluster without compromising security. However is not necessary for all remote clusters to have their own local AAS. Clusters that have reliable network connectivity can be configured to use services of the master AAS when the owner is not around.

Depending on the mechanisms used for mobility management, any updates at the master AAS can be synchronized across the PN using a reactive or proactive

approach. A reactive approach will require the master AAS to push the updates to all other AASs, while a proactive approach will require other AASs to poll the master AAS at periodic intervals. Regardless of the chosen approach, with different disconnected fragments of the PN operating semi-autonomously it is impossible to *guarantee* synchronization at all times. As things stand, until newly added foreign credentials and access policies have been synchronized, foreign entities are not able to connect through another cluster. Similarly, a credential valid in one cluster may have been recently revoked in another. Therefore it is therefore up to the user to configure how up-to-date an AAS capable device must be before it can function as the AAS of a mobile cluster. This is a basically a choice between increased security and better usability; given the non-critical nature of the PN we believe that the default configuration should be inclined toward usability.

Earlier we identified the two types of credentials typically used to authenticate users. The first, pair-wise keys were to be used for short-term anonymous communication between co-located users. Consequently pair-wise keys and their corresponding access policies are not synchronized between AASs i.e. access is restricted through the AAS which was involved in creating these credentials. Given that the communication is short-term, this does not present an issue. Synchronization issues related to the second, PKI based credentials, are discussed in the next section.

6.8.2 Creation and verification of AAS credentials

We have explained why it is not safe to duplicate the PN's private key across multiple mobile devices capable of operating as AASs. However with a dynamic group of AASs representing the owner of the PN, it is not possible to simply derive for each AAS a new PKI key set. This is because each time a new AAS capable device is added to the PN, the owner will have to update all existing foreign trust relationships to include the credentials of the new AAS. Therefore we propose using digital certificates (or alternatively ID based cryptography) during authentication so that we can dynamically identify each new AAS as part of the PN's trusted group of AASs.

Digital certificates

The personal CA concept introduced in [28] originally proposed to provide certificates within a PAN, is quite useful in this context. The main benefit of using a personal CA instead of delegating this functionality to a conventional CA outside the personal environment is that it allows the owner of the PN full control over his own local environment. Furthermore the financial overhead and scaling deficiencies associated with conventional PKI model do not apply because a small domain like a PN will only have one CA so there will not be any long chains of certificates to validate.

In our scenario the device operating as the personal CA is the security manager which initializes each personal device during the imprinting phase. When initializing a device capable of operating as an AAS, it also imprints such a device with the PN's public key and a digital certificate (sighed with the corresponding private key) certifying its membership in the PN. Considering that the security manager is the controlling entity of all PN devices, it is ideally suited to operate as the personal CA and the most secure location to store the PN's PKI key set.

Consequently when using PKI based credentials for inter-PN trust creation, the credentials exchanged between two AAS are their PN's ID and public key. The actual authentication for service access then takes place using digital certificates which certify the capacity of each AAS to represent its PN.

Identity based cryptography

An interesting alternative to digital certificates is identity based cryptography (IDC). An identity-based cryptosystem also requires a trusted third party which is called the Key Generation Center (KGC). In our scenario, we expect this role to be performed by the security manager. The KGC creates the system parameters comprising the master key (which is only known to the KGC) and a public extraction function f, which is publically known to all the participants in the system. The KGC is responsible for generating and distributing the private keys of participating devices based on its master key and participating device ID. For personal devices with AAS functionality, their private key as well as the public key extraction function is configured during the imprinting step. The corresponding public keys can be automatically generated by any other device with the knowledge of the public key extraction function f. Therefore during trust creation the two users who want to share services actually exchange their public key extraction functions.

In our context, the main advantage of IDC over PKI is that it avoids the need to exchange digital certificates during authentication. In general, the main disadvantage of IDC (known as key escrow) is that the KGC can generate any private key so the system cannot offer true non-repudiation. However this is not a problem in a small autonomous system like a PN where all devices belong to one user, so non-repudiation is not an issue.

6.8.3 Revocation

In this section we will look at issues related to the revocation of credentials created in Section 6.8.2. However before we do so, it is important to clarify how the other types of credentials in the system are cancelled. Pair-wise keys and user integrated PKI key sets are, strictly speaking, not revoked but cancelled. This is done by deleting the credentials and corresponding access policies, and for PKI credentials, initiating PN wide synchronization. Digital certificates using conventional PKI are

revoked using standard revocation checks. This does not have an impact on intra-PN revocation which only deals with revoking select foreign AASs in the context of inter-PN service access.

In Section 6.8.2 we explained how each AAS has its own unique set of credentials that allow us to revoke individual AASs while maintaining inter–PN trust relationships. Given that AAS credentials are issued by its own security manager, it is the responsibility of the security manger to update all affected foreign PNs with the list of revoked personal AASs.

Once the PN owner selects an AAS capable device for revocation, the security manager will not only revoke this device within the PN (Section 3.11) but also send a signed notification to each foreign entity with which it has a trust relationship. Recall that inter-PN trust creation involved the exchange of public keys belonging to the two security mangers. If connectivity with the foreign entity can be established, this notification can be transferred using a suitable (revocation) service offered by the foreign AAS. Once the foreign AAS verifies the signed notification, all revoked devices will be added to the local revocation list which is then synchronized throughout the PN. On the other hand, if connectivity with the foreign entity is not available, the signed notification can also be sent via email. Note that personal AAS capable devices that are sold or otherwise removed from the PN will have their credentials purged beforehand so do not need revocation at foreign entities. Furthermore, revocation of the security manager will require the PN owner to manually contact all foreign entities and request deletion of its credentials.

For clarification, since compromised device cannot be kept on the revocation list indefinitely, the credentials that are being revoked must have an expiry date. Note that unlike digital certificates which have a field for validity, private keys issued by KGC using identity based cryptography can only be given a validity period by linking the identity with a time period. For instance, a device with an identity MY_PDA in a PN with an identity MY_PN can be given a private key valid for the year 2007 by generating the private key for the identity MY_PDA@MY_PN:2007.

6.8.4 Co-Locating AAS and security agent functionality

Till now we have kept the functionality designed for inter-PN service access (AAS) separate from that required for intra-PN service access (security agent). However when considering multiple AAS, one in each cluster, there is significant benefit in having both functionalities co-located on one device. Specifically,

- It is easier for the user to use one device to manage security for both local and foreign access. It also simplifies the number of roles in the system by merging two functionalities into one.
- It ensures that each cluster contains one device capable of functioning as an AAS

- Gateway devices do not need to filter EAP traffic from local and foreign authentication to two different devices i.e. they do not need to make a forwarding decision based on the EAP mechanism being used.
- The intra-PN revocation list (Section 3.11) used by security agents can be synchronized alongside the AAS specific synchronization
- But most importantly, co-locating these functionalities means that there is no need to have a separate AAS selection process e.g. [110] [111] (including mechanism to detect and recover from AAS mobility etc). This is because such mechanisms already exist for maintaining one functioning security agent per cluster.

Regardless of whether the two functionalities are actually co-located, we do expect the security agent (as the local management entity) to be involved in the selection of the cluster's AAS.

6.9 PN federations

We have already pointed out why it is desirable to extend PN communication beyond the boundaries of the individual PNs. Based on this, we proposed mechanisms to establish trust and share services between a *pair* of PNs. However there also exist scenarios where it is desirable for a group of co-located people to interact quickly as a group without having to go through the hassle of creating a lot of pair-wise trust relationships. It is possible (even likely) that each individual member does not know everybody else, but is willing to interact for some common purpose. Such a group of PNs operating with a sense of shared trust is known as a PN federation (PN-F). Examples include federations between attendees in a conference, colleagues in an ad-hoc meeting and multiplayer gaming. As with real world relationships, a PN can of course be a member in multiple federations.

For each federation the members make available a set of shared services. It is important to note that all federation members are equal in terms of their access to such services. In this context, our requirement are that the interaction between the federation members is secure, self-organized and restricted to the services defined by the members.

Even though the different member of the federation may not know each other a-priori, for security purposes there must be some mechanism to control the membership of the federation. Therefore federations are defined, initialized and managed by one of the federation members which we call the **PN-F manager**. The PN-F manager plays an essential role in the operation of the federation while other members only contribute their services. Therefore it is important to note that the PN-F manager will have to be mutually agreed by all parties in the federation. Specifically,

this role is performed by the AAS in the P-PAN of the PN-F manager. We will look at each of the roles in turn, with a focus on issues related to security.

6.9.1 Definition

A federation is defined by the PN-F manager when it specifies the following:

- Federation ID
- Federation name
- Federation goal
- Federation rules

Each federation has an identifier which is a unique value that is randomly generated. This non-descriptive value is represented in a more user friendly manner by a user generated federation name. The federation goal and rules are also user specified texts that state the goals of the federation and any rules that participants must obey. The goals and rules of the federation may or may not be available to prospective members before they join, however this information must always be available to existing members via a service provided by the PN-F manager. It is possible that the goals and rules of a federation may change over time.

6.9.2 Initialization

Federations can only be initialized by the PN-F manager after they have been defined. Initialization takes place when the PN-F manager begins to advertize the federation using advertisements. These can be broadcasted in an ad-hoc manner by the PN-F manager or distributed using the infrastructure by publishing it on websites or sending via emails etc. Advertisements include the federation ID, federation name, PN-F manager's PN ID and point of contact of the PN-F manager. However in order to limit their size, advertisements may not include the federation goals and rules.

If the PN-F manager advertizes using radio broadcasts the advertisements should be broadcasted by all gateway devices that are willing to act as interface points for the federation. In this case the point of contact will be the link layer address of the advertising gateway. On the other hand, for infrastructure supported advertisements this will be the globally reachable IP addresses of participating gateways. Prospective federation members use the information contained in the advertisements to authenticate with the PN-F manager (based on a-priori trust), which issues them a set of credentials that they can use to prove membership of the federation. If authentication is a result of radio broadcasting, then the authentication mechanisms

used are those proposed for local service access (Section 6.6.1). For infrastructure supported advertisements that include the IP address of the gateways, authentication is performed using mechanisms proposed for remote service access (Section 6.6.2).

For security reasons federation credentials are only stored on the participant's P-PAN AAS and backed up at the master AAS during synchronization. This means that interaction related to PN-Fs always takes place through the participants P-PAN. If the P-PAN's AAS is lost or compromised, all PN-F memberships need to be revoked. However unlike pair-wise relationships, we envision most federations to be ad-hoc and short term so this should not be an issue.

Once a new member joins the federation, it must make available to the PN-F manager the list of its services available for sharing. In return the new member can query the list of other members, their offered services and their points of contact. Details on how mobile points of contact are reachable are beyond the scope of this document. Generally speaking we can use static or dynamic IP addresses. The use of static IP addresses will require infrastructure support in the form of PN agents or Mobile IP home agents. On the other hand, dynamic IP addresses would require periodic updates with the PN manager which can be a cause of some overhead.

6.9.3 Management

The mechanisms we use to facilitate group trust in a federation are based on a PKI where the trust domain is limited to the federation. We take our inspiration from the personal CA concept explained earlier. The PN-F manager's AAS acts as a certification authority and generates a PKI key set for each federation that it manages. Based on this key set the AAS issues certificates to all members of the federation. Specifically, we expect it to issue a digital certificate that includes the PN ID and the public value of a PKI key set generated by each participating member. For increased security federation members can employ unique key sets per federation; however it is debatable whether this will increase the overall security in the system.

Federation members use their certificates to certify their membership of the federation and gain access to federation services. The mechanisms for inter-PN authentication and authorization are the same as those explained in Section 6.5 and Section 6.6. Since members may have their certificates revoked before they expire (e.g. malacious PNs), the PN manager will maintain a certificate revocation list that is be periodically checked by other members. We expect the certificate revocation list to be a standard service provided by all PN-F managers.

The main advantage of federation as extension to the pair-wise PN trust model is that it does not require each member to share an existing security association with all other members in order to access their services. Thus federations are very suitable

for quick group interaction because their construct reduces the amount of manual interaction required to establish and maintain the necessary trust relationships. For instance for a federation with n members, only $n-1$ trust relationships must be established compared to $\frac{n(n-1)}{2}$ pair-wise relationships. Moreover management tasks like member eviction are the responsibility of one user (the PN-F manager) and do not need to be duplicated by others.

6.9.4 Service Access

The set of services shared by different members of a federation depend on the stated goals of the federation and the amount of trust members place in the federation. Specifically, the amount of trust members place in the PN-F manager plays a direct part in their willingness to share services.

Once two federation members mutually authenticate and establish a secure tunnel between their gateways, access control will be performed on the same trusted agent model that is used for pair-wise relationships. Figure 1.4 illustrates a federation where three users are connected through secure connections between their respective P-PANs. These connections are only created on demand do not need to exist throughout the lifetime of the federation. Once the secure connection between a pair of PNs is established the services being offered by one PN will be accessible to all devices of the peer PN.

6.10 Related work

In this chapter we have proposed mechanisms for allowing a PN to interact with the outside world. Where appropriate we have given an overview of related work, for example, an overview of the generic AAA architecture in Section 6.3 and of security protocols like IPSec and IEEE 802.1X in Section 6.6. In this section we look at related work for our proposals in their entirety. To the best of our knowledge there is only one other work in this new field of inter-PN communication, that being carried out in the MAGNET Beyond project.

6.10.1 MAGNET Beyond

The MAGNET Beyond project started in 2006, and like MAGNET, is a large project with 30 partners from 15 countries. Where MAGNET focused on PN formation and secure intra-PN communication, MAGNET Beyond focuses more on inter-PN communication and realization of pilot services. However before we get into comparisons, it is important to point out that at this point the MAGNET Beyond architecture is not finalized and our knowledge is restricted to publically available project deliverables available on the project website [8]. Of particular interest to us is work

package 4 tasks 1 and 2 of which only the initial ideas are available, with the detailed final solutions due in Q3 2008.

In the MAGNET Beyond architecture all inter-PN interaction takes place with the formation of PN Federations. Although the concept of Federations is roughly the same, we differ in the model used to realize them. For MAGNET Beyond participation in Federations is realized through the functionality of a Federation Manager (FM), which is a role implemented in the gateway of the home cluster. The FM contains authentication and access control policies for each federation members and uses a policy engine to authenticate such requests. This implies that even local cluster access will require (non-optimal) communication through the home gateway. There is some mention of coping FM functionality to a local gateway when connectivity with the home cluster is not available however there are no details available on this topic.

When we compare this with our model, since we separate the gateway functionality from the authentication and access control functionality, local clusters can be access directly through their own gateways (even when using a remote AAS). Furthermore, as we explained in Section 6.8.1 the copying of credentials is not safe, hence we propose mechanisms for synchronization of access policies ad public credentials and choose to derive for each AAS a unique set of credentials which certify its capacity to represent its PN. Furthermore, our decision of separating the AAS functionality from the gateway device allows us to use the most convenient gateway for inter-PN communication and possibly even multiplex the connection over multiple gateways, and not only use the gateway which implements the FM functionality.

As yet MAGNET Beyond does not provide details on the mechanisms used for authentication and secure tunnel creation, so comparison on that topic is not possible. Our proposed mechanisms based on IEEE 802.1X and IPSec, use well established security protocols and are also suitable for legacy devices.

For authorization during inter-PN service access, MAGNET Beyond has three different architectures under review. However details are lacking and until the final solution is available it is difficult to know how the three models will fit in with the FM concept. It is possible that because MAGNET Beyond has less restrictive requirements in terms of PN device capability i.e. it assumes that all PN devices can carry out asymmetric elliptic curve cryptography, the final model may be difficult to compare.

As stated earlier, MAGNET Beyond defines all inter-PN interaction in terms of Federations. However we believe that the concept of PN Federation is more suitable for quick and convenient group communication, usually amongst co-located users and not for long term pair-wise relationships. This mainly because of the unequal role played by one Federation member (the PN-F manager) and also due the additional overhead associated with broadcasting of advertisements, the definition of Federation goals, specification of rules and policies for participation etc. Also since our model for pair-wise PN communication does not require an external party

to issue credentials, each PN is able to derive for its AASs a unique credential during personalization. As updating each existing AAS capable device with a new Federation credential is not convenient, the credential certifying membership of a Federation can only be stored in a central location. In our case it is the AAS in the P-PAN, and in MAGNET Beyond it is the FM at the home gateway.

6.11 Concluding remarks

In this chapter we have proposed mechanisms that enable secure intra-PN and inter-PN service access. By leveraging on the security of the underlying intra-PN communication network, we ensure that there is no additional cost of securing intra-PN service access. For inter-PN service access we have proposed protocols and mechanisms for local and remote authentication and service authorization. Foreign service access only takes place through gateway devices that depend on backend servers for authentication and authorization. End-to-end service access is secured by leveraging on the security of intra-PN communication till the respective gateways, and the secure point-to-point connection between the two gateways. This allows us to reduce the overhead of security for service end points, which can potentially be resource constrained. Pre-approving services at the time of authorization and using the gateways as trusted agents, means that the mechanisms used for service access between the service end-points are the same for both intra and inter-PN service access.

In order to benefit from the existing work we have made an effort to use well known standardized mechanisms, abet with modifications necessary to support our model. These include mechanisms necessary to create trust between two users. Our proposals are designed to be simple and intuitive to configure (for the user), and lightweight to enforce (for personal devices). They are lightweight in both the computational requirements and the device code complexity. This translates in to a lower financial cost of devices, a smaller form factor, longer battery time, and thus a better user experience.

Chapter 7

Secure routing for PNs

Our approach for securing communication inside wireless ad-hoc clusters, as presented in Section 2.4.4, can be described as Link layer security that requires a new PN layer situated between the Network and Link layers. There were of course other options, specifically (a) utilizing the Link layer cryptographic functionality pre-existing in different radio technologies like Bluetooth and WLANs (b) using Network layer security like IPSec or (c) requiring applications to manage end-to-end security. We will first point out the advantages and disadvantages of each approach and then present the results of our very first research into secure routing within the P-PANs, which was not based on Link layer security.

Radio technologies like Bluetooth and WLANs have the functionality to encrypt all traffic at the Link layer, including user data and control messages like those used for routing. This is quite useful because it implies that ad-hoc routing protocols like AODV (Ad hoc On-demand Distance Vector routing protocol) [39] and DSR (Dynamic Source Routing protocol) [49] which work above the Link layer can be used without any special security functionality. However given the different types of radio technologies existing in PNs e.g. Bluetooth, IEEE 802.11, ZigBee [108] etc. this implies the use of different cryptographic algorithms and their associated keys. In other words, the level of security will vary significantly depending on the radio technology being used, with the system only being as secure as the weakest technology. Another issue with this approach is the need for encryption and decryption for packets travelling over multiple hops, to the extent that such packets will need to be appended with a new MAC. Lastly, with different radio technologies in the P-PAN we would have to distribute and manage multiple keys, one for each radio technology.

Our second option, that of using Network layer security like IPSec [84] which is widely deployed, has the advantage of providing a more consistent level of security because it is independent of lower layer technologies. However IPSec is primarily used for enabling *end-to-end* authentication and security between two network entities that already have routing between them. Control traffic such as ad-hoc routing

algorithms are not protected and need to be secured using additional secure routing functionality. The last option, that of application level security, has the same disadvantage of not supporting hop-by-hop security and moreover places the user at the mercy of the application developer.

Although we eventually decided to enforce security based on a new PN layer which has all the benefits of Link layer security without some of the disadvantages identified above, our initial approach was to try to solve each security problem independently, using whichever of the three options was best suited. The first topic of our focus was to secure the ad-hoc routing algorithms used within the P-PAN, and this chapter presents our results in that respect. Although our later decision to use the PN layer means that it is no longer relevant for P-PANs, it is however still relevant in the context of secure routing when PN Federations are created over multihop ad-hoc links.

Although alot of adhoc routing protocols have been proposed over the last few years [72] [74] [75] [79], our work focused on the two protocols that were under consideration by the IETF for standardization, namely AODV and DSR. AODV is a reactive routing protocol which only finds the route on-demand by flooding the network with route request messages while DSR is a proactive routing protocol which maintains an up-to-date list of destinations and their routes by periodically distributing routing tables throughout the network. Our analysis of the pros and cons of each approach led us to conclude that reactive routing protocols were more energy efficent given the periods of in-activity expected in PNs. Consequently we chose to work with AODV, with the requirements identified earlier such as that with regards to the exclusive use of lightweight symmetric cryptography.

We will begin by detailing the security threats against most adhoc routing protocols (which only focus on providing routing services without explicitly considering security), with specific examples using AODV. To achieve availability in the P-PAN, the routing protocol used must be robust against both dynamically changing topology *and* malicious attacks. Unfortunately the only two existing secure routing protocols for AODV [40] [98] require digital signatures to be verified on a per-hop basis and are too resource intensive for the constrained devices envisioned in PNs. In light of the identified threats and our requirements we then propose a new secure routing protocol based on AODV, which we call SL-AODV (Secure Lightweight AODV) [116]. SL-AODV requires some amount of prior security coordination and relies on efficient cryptographic primitives to avoid most of the identified attacks. It prevents external attackers and even compromised nodes from tampering with routes consisting of uncompromised devices, and is safe against multiple un-coordinated active attackers. Finally, it has minimal performance costs for the increased security in terms of processing and networking overhead.

7.1 Secure Routing

There are convincing reasons for enabling multi-hop communication in the P-PAN. For example, in order to ensure network connectivity amongst different wireless technologies, devices with more than one type of wireless interface will need to route packets between the different radio domains. Moreover, depending on the range of the wireless signals multiple hops may be needed for communication. In fact, it may be more energy efficient to transmit packets over multiple small hops, even if end to end communication is possible using one large hop. The absence of centrally administered secure routers in wireless adhoc networks means that secure topology discovery cannot be taken for granted. The cooperative nature of ad-hoc routing algorithms allows malicious devices to easily cripple the network by inserting false route information, replaying old messages, modifying messages of other devices, etc.

7.1.1 Overview of AODV

With the Ad-hoc On-Demand Distance Vector (AODV) [39] protocol when a source wants to send data packets to a destination but cannot find a route in its routing table, it broadcasts a Route Request (RREQ) message to its neighbors. The neighbors then check their routing table to see if they have a fresh enough route to the requested destination, if so they generate a Route Reply (RREP) message, otherwise they rebroadcast the RREQ message. A fresh enough route is one whose associated sequence number is equal to or greater than that contained in the RREQ message. This process continues until the RREQ message reaches the destination, or an intermediate device that has a fresh enough route.

Every device maintains its own sequence number and its RREQ ID. Together the RREQ ID and the source's IP address uniquely identify a particular RREQ message. Intermediate devices only accept the first copy of a RREQ message, and drops all duplicates recieved later. The sequence numbers are used to guarantee that all routes are loop-free and contain the most recent routing information. It is the destination sequence number (included in a a RREQ message) which is used by intermediate devices to decide if the route they have for the destination is fresh enough.

One a fresh enough route is found, the destination or intermediate device updates the reverse route to the source by sending a RREP message to the neighbor from which it receives the RREQ message. As the RREP message travels towards the source, intermediate devices update their forward route to the destination to the neighbor from which they received the RREP message. They also update the destination sequence number in their routing table to the maximum of the one already in the routing table and the one in the RREP message. Finally, AODV may also use HELLO messages to update the status of the local connectivity between two neighbors.

The maintenance of an established route is done with Route Error (RERR) messages. If an intermediate device detects a link break in an active route it sends out a RERR message to its upstream neighbors that are using it as the next hop in the broken route. When a device receives such a RERR message, it forwards the RERR message to its upstream neighbors as well. When this message arrives at the source, it initiates a new route discovery.

7.1.2 Exploits

Most adhoc routing protocols are are efficient in terms of network performance, but they allow attackers to easily advertise false route information, redirect routes, and launch DoS attacks. Most attacks against an adhoc routing algorithm are typically carried out for the following reasons:

- Route Disruption: The aim of the attacker is to disrupt existing communication or prevent new routes from being set up.

- Route Insertion: The aim of the attacker is to insert itself between the communication endpoints, either to snoop on the data being exchanged or to drop certain critical data packets in order to distrupt the applications that are exchanging the data. The former would be an attack against the privacy of communication while the later would be a denial of service attack which would be very difficult to detect.

- Device isolation: The aim of the attacker is to isolate a specific device in the network by preventing the establishment of valid routes towards it.

- Resource consumption: The aim of the attacker is to waste the resources in the network such that the performance degrades significantly. This attack can be carried out against the communication bandwidth, battery or even storage space. For example generating large quantitiesof false RREQ messages which have to be broadcasted in the entire network.

The attacks presented below are described in terms of the AODV protocols. Since the focus of our work will be on attacks that result from the specification of the ad hoc routing protocol, we disregard for example physical layer attacks based on jamming because mechanisms such as spread spectrum have already been extensively studied to resist such attacks. Also we purposefully do not attempt to defend against attacks based on non-cooperation, for instance during route discovery, since the attacker could also otherwise drop data packets sent to those destinations.

Modification of route sequence numbers

AODV maintains routes by assigning monotonically increasing sequence numbers to routes toward specific destinations. Any device can divert traffic through itself by advertising a route to a destination with a destination sequence number greater than the authentic value. For example in the adhoc network illustrated in Figure 7.1 device S has broadcasted a RREQ message for destination D with a destination sequence number X.

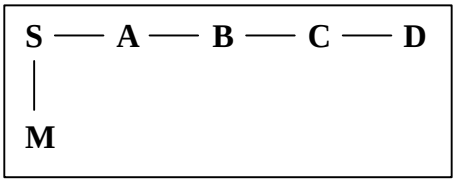

Figure 7.1: A sample adhoc network.

Upon recieving this RREQ message, even though the malicious device M does not have a route to D (and also cannot stop the discovery of the valid route to D through device A), it replies with a false RREP message that includes a destination sequence number of X+1. Even though S will receive the valid RREP message from A also it will have already received the false RREP message from M. Since the false RREP message included a higher destination sequence number, S will drop the valid RREP message thinking that the valid route is stale.

Modification of hop counts

For a given destination sequence number, AODV uses the hop count as the metric to select the shortest path. Malicious devices can increase the chances of being included in the newly created route by resetting the hop count of the RREQ message they forward to zero. This would be an example of route insertion. Similarily by setting the hop count to a very large value, they can make sure that they are not included in the route. This can be an example of route disruption or non-cooperation (possibily to save local resources).

Modification of source addresses

In the scenario illustrated in Figure 7.1, assume that a valid path already exists between devices S and D. A device M entering the network can spoof its IP address to that of D and generate a RREP message to S containing a hop count that is less than that advertised by A e.g. zero. S therefore changes its route to the destination.

Generation of false error messages

AODV impliments route maintainance to recover quickly from broken paths usually caused by device mobilty or sudden unavailability. If the destination device or an intermediate device along an active path moves, the device immediately upsteam of the link generates a RERR message indicating that the destination is now unreachable. Since the RERR messages in AODV are not verified it is easy for attackers to generate false message and break otherwise valid routes. For example, in Figure 7.1 assume that a valid route exists between S and D. Device M can easily break this route by sending a RERR message to S pretending to be A.

Generation of unnecessary route request messages

Since RREQ messages are broadcasted in the network, an attacker wishing to launch DoS attacks can generate false or unnecessary RREQ messages as an example of a resource consumption attack. The only way to guard against such an attack is to authenticate the source of the RREQ message and to impliment a mechanism which limits the rate at which RREQ messages from any one device are forwarded.

Wormhole/Tunneling attack

A wormhole attack is launced by a pair of *colluding* attackers and the aim is route insertion. The two attackers, separated by several hops, build a direct link in the form of a tunnel and communicate with each other through the tunnel. This tunnel can be established in different ways, for instance through an out-of-band channel, packet encapsulation, or high-powered transmission. The route through the tunnel is attractive to legitmate devices because it appears to provide a route with less number of hops. The wormhole attack is particularily difficult to identify because the attackers do not modify the contents of the routing messages. A routing message recieved at one end of the wormhole comes out the other end unmodified. Existing algorithms to detect wormhole require special hardware [76] or tight time synchronization [78] which makes them rather unpractical.

7.1.3 Related Work

Since SL-AODV is based on the basic operation of the AODV protocol, it is also a reactive or an on-demand routing protocol. Other well known reactive secure routing protocols include S-AODV [40], ARAN [98], ARIADNE [71] and SRP [100].

The S-AODV (Secure AODV) scheme is also based on AODV and assumes that each device has certified public keys of all other network devices, so that they can validate all in-transit routing messages. The idea behind S-AODV is to use signatures to authenticate most fields of the route request and route reply messages and

to use hash chains to authenticate the hop count. The originator of a routing message includes in it the last element of a new hash chain and appends to it a digital signature. In addition, it includes an element of the hash chain, called the hop count authenticator, based on the actual hop count in the routing message header. With the exception of the hop count field and hop count authenticator, the remaining fields are immutable and therefore can be authenticated by verifying the signature. As the message traverses the network, intermediate devices cryptographically validate the signature, thus validating the integrity of the message and also validate the number of hops to the sender by using the properties of the hash chain. However intermediate devices with a fresh enough route can still not reply to RREQs as they cannot sign on behalf of the destination. S-AODV also uses signatures to protect the route error messages used in route maintenance. Unfortunately, the widespread use of public key cryptography in S-AODV imposes a high processing overhead on intermediate devices and is therefore does not meet our requirements.

Authenticated Routing for Ad-hoc Networks (ARAN) [98], like S-AODV, is based on AODV and uses asymmetric cryptography in the form of certificates to protect the routing protocol. Before joining an ad-hoc network, a device must authenticate with a trusted certificate authority (CA) whose public key is known to all valid participants. A sucessful authentication results in the issuance isof a digital certificate which associates the IP address of the device with a public key. The certificate is used to sign the route request message which includes a nonce and timestamp which together ensure freshness when used in a network with a limited clock skew. Unfortunately not only can a CA not be guaranteed in an ad-hoc environment, but the use of asymmetric cryptography imposes a high processing overhead. Since ARAN uses asymmetric cryptography for route message verification it is particularly vulnerable to DoS attacks based on flooding the network with fake control messages. As long as a device cannot verify signatures at line speed, the attacker can force it to discard some of the control messages it receives. The main differences between ARAN and SAODV is that ARAN uses an *extra* signature to authenticate the previous hop while SAODV uses a hash chain to authenticate the hop count. The route request message size is also larger than normal AODV because it includes digital certificates of the originator and the last hop forwarder. Lastly, both S-AODV and ARAN are vulnerable to wormhole attacks.

ARIADNE [71] is a security extension of the DSR protocol and uses only efficient symmetric cryptography. The routing protocol messages are protected by message authentication codes (MACs) and are authenticated at each hop using TESLA [19]. During route discovery, each hop authenticates new information in the route request and the destination device buffers the route reply until intermediate devices can release the corresponding TELSA keys. Once the TESLA security condition is verified by the destination, it includes a MAC in the route reply to certify that the security condition was met. ARIADNE requires an existing mechanism to distribute (throughout the network) one authentic public TESLA key for each device and a

secret key between all pairs of communicating devices. It also requires loose time synchronization between all the devices and the network links to be bidirectional. When compared with SL-AODV the main disadvantage of ARIADNE is the requirement for time synchronization and the additional delay in route discovery due to the use of TESLA.

The design of the Secure Routing Protocol (SRP) [100] is also based on the basic operation of the DSR protocol. The strength of SRP lies in the fact that the correctness of discovered routes can be verified from the route geometry itself. SRP is designed as an extension header that is attached to RREQ and RREP messages and does not attempt to secure the RERR messages which are delegated to the Secure Message Transmission protocol [73]. In SRP the destination validates the incoming route request using a shared key, and constructs a route reply that is protected using a MAC. The route reply is then returned to the source over the reverse path. False or corrupted routing information is discarded by the end devices using an existing end-to-end security association. SRP uses a sequence number in the RREQ to ensure freshness, but this sequence number can only be checked at the target. Similarily modification of the RREQ is only detected at the source. SRP, like ARIADNE, requires network links to be bi-directional and requires existing shared secret keys between pairs of communicating devices. However, even though it does not have the other requirements of ARIADNE, it is not an ideal candidate because routing messages are not authenticated in intermediate devices. As a result the protocol is very vulnerable to denial-of-service (DoS) attacks using fake source addresses, such as repeatedly flooding the ad-hoc network with route requests. Also, because the MAC cannot be verified by intermediate devices, some DSR optimizations cannot be applied.

7.2 Secure Lightweight AODV (SL-AODV)

7.2.1 Requirements

Most attacks against the routing protocol are caused by malicious injection of incorrect routing information, or modification of the correct messages from other devices. To prevent these attacks, it is necessary for each intermediate device to verify the origin and integrity of routing messages that it receives. Therefore the security requirements for SL-AODV include the ability to limit DoS attacks by performing hop-by-hop authentication of all routing messages, ensuring routing message integrity and resisting replay attacks.

Furthermore it must also have a low communication and computation overhead since an inefficient authentication mechanism could be exploited by malicious devices by flooding the network with invalid messages and overwhelming devices with the cost of verifying them. Additionally, in order to support devices that have limited

processing power the security mechanisms of SL-AODV must be based on efficient symmetric cryptography such as one-way hash chains [41] and must not require computationally expensive asymmetric cryptography. Thus we use one-way hash chains to do both the message authentication as well as the hop count verification. As with ARIADNE, the use of a one-way hash chain means that we assume an existing mechanism for devices to distribute the commitment of their hash chain to all other devices. Note that in order to reduce overhead our protocol does not aim to provide confidentiality or privacy for the sender of the routing message.

7.2.2 One-Way hash chains

One-way hash chains, originally proposed by Lamport [41] for one-time password authentication (which Haller later refined to the S/KEY standard [42]), are an important cryptographic primitive in many security applications. These include for example digital cash [44], constructing one-time signatures [45] and (most relevant to us) for authenticating routing updates [46][47][105]. Since one-way hash chains are very efficient to verify they have become rather popular for designing security protocols for resource constrained mobile devices such as sensors [11]. This is because the low-powered processors found inside sensors can compute a one-way hash function in milliseconds instead of up to a minute for generating or verifying digital signatures [43]. The increase in popularity also means that a variety of improvements to hash-chains to make storage, access and verification more efficient have been proposed [48][70][65].

A one-way hash chain is built on a one-way hash function which maps an input of any length to a fixed-length output string. It is a one-way hash function because it should be simple to compute, yet computationally infeasible to invert. A number of such functions have been proposed, including MD5 [107] and SHA-1 [99]. Figure 7.2 shows a one-way hash chain construction that is built on the one-way hash function F. To generate a chain of length n each device randomly picks the first element of it's chain (c_0) and repeatedly applying F to compute the list of values $c_1 \ldots c_n$, where $c_i = F\ (c_{i-1})$. It is important to understand that the elements of a hash chain are revealed in the opposite order of their generation i.e. c_n, $c_{n-1} \ldots c_0$. As stated earlier, the function F should be simple to compute yet must be computationally infeasible to invert. c_n is called the commitment of the hash chain because it can be used to verify any element of the chain, for example to verify that element c_i is indeed the element with index i of the hash chain, we check $F_{n-i}\ (c_i) = c_n$. More generally if $i < j$ and we know that c_j is the j-th element of the chain, we can say that c_j commits to c_i if $F_{j-i}\ (c_i) = c_j$.

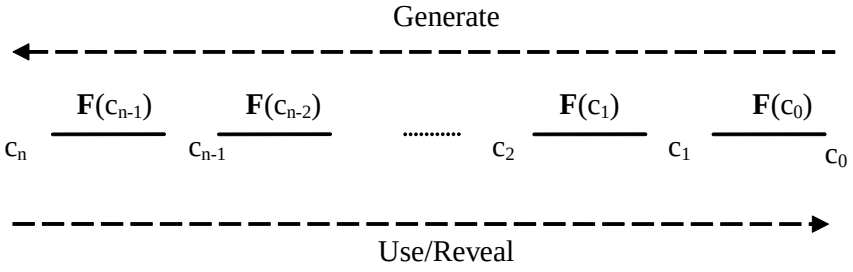

Figure 7.2: Generating and using a one-way hash chain.

7.2.3 Overview

SL-AODV requires devices to forward only verified routing protocol messages. This ensures that unauthenticated attackers cannot take part in the routing process and even compromised devices cannot tamper with routes consisting of uncompromised devices. The algorithm requires each device to maintain a hash chain, elements of which are used over time by the routing protocol to secure routing messages. Each device uses a specific element from its hash chain in each route request (RREQ) and route reply (RREP) message it generates. Based on the position of this hash element in the hash chain, it will provide verification of the sender, the included sequence number and the (lower bound value of the) hop count in each routing protocol message. As mentioned before, these hash values are used in the order of decreasing subscript, meaning that given an existing authenticated element of a one-way hash chain, it is possible to verify elements later in the sequence of use (i.e. those with a lower subscript). The hash chain element is contained in the *Hash* field of the modified AODV RREQ and the RREP messages, as shown in Figures 7.4 and 7.5.

SL-AODV requires an upper bound to be assumed on the diameter of the ad-hoc network; we use $m - 1$ to denote this bound. Thus, all hop count values in routing messages must be less than m. A value of m that is too conservative will mean that destination more than m hops away from a source will be unreachable. However an overly large value of m will cause the hash chain values to be used up faster, and increase the overhead of maintaining the hash chain.

If a device's hash chain is the sequence of values $c_0, c_1, c_2, c_3 \ldots c_n$ where n should be divisible by m, then for a routing message of sequence number i, let $k = \frac{n}{m} - i$. An element from the group of elements c_{km}, c_{km+1}, $c_{km+2} \ldots c_{km+m-1}$ from the hash chain will be used to authenticate the sequence number and hop count for that routing message. For example, when a device sends a RREQ message with sequence number i, it sets the hop count field to 0 and the hash value field to the first element (c_{km}) in the *group* of its own hash chain elements corresponding to that sequence

number. Before this message is retransmitted, the intermediate device will increase the hop count by one and re-hash the value included in the message (i.e. the first hop device will hash c_{km} to c_{km+1}, the second c_{km+1} to c_{km+2} etc.). Figure 7.3 gives an example of the hash chain element used in a routing message corresponding to its sequence number (i) and hop count (j).

	j=0	1	2	3	4
i=1	h_{15}	h_{16}	h_{17}	h_{18}	h_{19}
2	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}
3	h_5	h_6	h_7	h_8	h_9
4	h_0	h_1	h_2	h_3	h_4

Figure 7.3: Hash chain value usage with a network diameter of 5 and a chain length of 20.

Devices receiving a routing messages can easily authenticate the hash value, given any earlier authentic hash chain element. If the entry validates the combination of the source address, sequence number and hop count value, the device processes it in the routing algorithm otherwise it drops the message. Note that devices will only accept routing messages from a given source with a sequence number that is higher than what they already have on record for that source. Due to the one-way nature of the hash chain this prevents malicious devices from advertising a route to other destinations claiming a greater sequence number than that destination's own current sequence number. Likewise, no device can advertise a route better than that for which it has received an advertisement, since the hop count in an existing route cannot be decreased. Lastly, routing messages received at a later time using the same sequence number are automatically dropped, so malicious devices cannot carry out replay attacks.

Each time a device generates a routing update, it will use a one larger sequence number and use the first element (e.g. h_{10} for $i = 2$) of the group of elements corresponding to that sequence number. Receiving devices will verify that they have not seen a message from the sender with that sequence number or higher, and then verify that the hash value corresponds to the included sequence number and the hop count.

To safeguard against replay attacks, devices wishing to authenticate a routing message need to know the latest sequence number used by the sender of that message. Therefore, unlike the AODV expanding ring approach [39] for disseminating RREQs, SL-AODV requires the RREQs to be disseminated through the ad-hoc network. This

ensures that all the devices are aware of the latest publicly available hash value of the sender (corresponding to the latest sequence number used). Assuming that network diameter is not large, this should not result in perceptible performance degradation. For the same reason, the RREPs from the destination also need to be disseminated through the entire network, because they too contain the latest publicly available hash value of the destination. This is why we stated earlier that is only suited to small ad-hoc networks like P-PANs. Note that if these RREPs are not properly disseminated, an attacker overhearing a RREP can replay that message in another part of the network.

Since routing messages are broadcasted through the ad-hoc network, they have a potential for abuse in a resource consumption attack. Devices should, in principle, limit the rate at which they forward routing messages for a given device. Note that as route error (RERR) messages are only transmitted one hop, they are authenticated using security associations existing between neighboring devices, and do not need additional protection. A simple check to verify that routes listed in RERR messages do indeed go through the sender of the RERR message, will provide sufficiently robust security against malicious RERR messages.

7.2.4 Message Format

One of the security requirements for SL-AODV is that the source of the RREQ can only trust RREPs that are sent by the destination itself. In standard AODV intermediate devices with fresh routes to the destination can also generate RREPs on behalf of the destination. Since intermediate devices are no longer trusted to reply to RREQs on behalf of the destination, there is no need to have the 'D' (Destination only flag) or the 'G' (Gratuitous RREP flag) in the AODV routing messages. Details on the AODV message format can be found in [39].

In standard AODV the *Destination Sequence Number* field in a RREQ message has two uses. First, it lets intermediate devices understand how up-to-date the sender requires the RREP to be. Second, it helps the destination choose a new sequence number when it reboots. After rebooting, a device does not remember its last sequence number and trusts anybody that sends to it a RREQ with the correct number. However, such functionality is not acceptable since a malicious device can put a much bigger destination sequence number than the real one. This allows a very easy attack that consists in setting the *Destination Sequence Number* field to its maximum value. The next time the device increments this sequence number, its counter will overflow and this will cause unexpected results. Moreover, the first functionality is also not necessary since intermediate devices are no longer allowed to reply to route request messages and the destination is expected to reply with the most up to date route. Therefore, this field and its associated 'U' flag do not exist in our modified RREQ message. Finally, as there is no support for multicasting the 'J' and 'R' flags are also not needed.

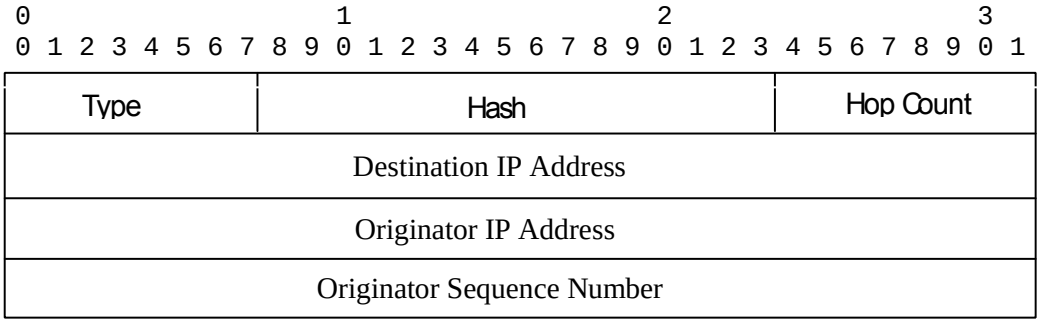

Figure 7.4: SL-AODV RREQ packet format.

As a result of the expanding ring approach in AODV, senders often needed to send multiple RREQs before receiving a RREP back. To ensure that sequence numbers are not used too fast, each new broadcast increments the RREQ ID value. However, since SL-AODV does not use the expanding ring approach, senders always increment their sequence number before retransmitting RREQs. Intermediate devices can now use the sequence number and source address in each RREQ message instead of RREQ ID, to drop duplicates. Figure 7.4 and 7.5 show the new SL-AODV RREQ and RREP message formats respectively. The fields in the new RREQ message are:

- *Type*: 1
- *Hop Count*: The number of hops from the *Originator IP Address* to the device handling the request.
- *Destination IP Address*: The IP address of the destination for which a route is desired.
- *Originator IP Address*: The IP address of the device which originated the RREQ.
- *Originator Sequence Number*: The current sequence number to be used in the routing table entry, pointing towards the originator of the route request.
- *Hash*: Hash value from the sender's hash chain corresponding to the sequence number and hop count of the message.
- *Type*: 2
- *Destination IP Address*: The IP address of the destination for which a route is supplied.

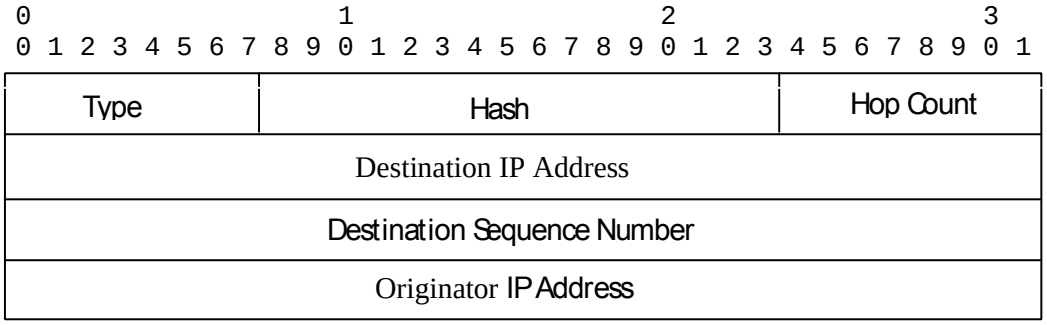

Figure 7.5: SL-AODV RREP packet format

- *Destination Sequence Number*: The destination sequence number associated to the route.
- *Originator IP Address*: The IP address of the node which originated the RREQ for which the route is supplied.
- *Hash*: Hash value from the senders hash chain corresponding to the sequence number and hop count of the message.

7.2.5 Analysis

An obvious disadvantage of fully disseminating routing messages in the network, when compared with AODV's expanding ring approach is the extra number of transmissions necessary. The exact overhead of course depends on the size of the network and the level of activity. However, this extra dissemination does have some benefits in that it allows other devices in the network to discover a path to the two communicating devices. It is conceivable that these active devices may be involved in future communication, and in many cases new path discovery will not be necessary. Since AODV is designed for potentially hundreds of devices, the reverse route made during route discovery is deleted if it is not chosen during the route reply. This is done in order to reduce the number of entries in local routing tables. P-PANs on the other hand, are made of much smaller number of devices so caching these routes for a longer period could be advantageous. Therefore if both the routes in the RREQ and the RREQ are cached, all the devices in the P-PAN will have routes to these communicating devices. Over time, all the active devices will find themselves in the routing tables of other devices. Note that if any if these routes is broken, then the standard AODV route error messages will be returned and a new route discovery can be started.

Malicious devices can still attempt to reduce the amount of routing information available to other devices, by not advertising certain routes or by destroying routing

messages that pass through them. This shows the unwillingness of the malicious device to forward messages for such destinations and we purposefully do not attempt to defend against this "attack", since the attacker could also otherwise drop data packets sent to those destinations. Moreover, our protocol does not aim to prevent attackers from injecting data messages into the network since such a resource consumption attack is costly for the attacker and thus not so interesting. Since attacks against the routing infrastructure are much more effective, attackers may attempt to modify a routing message by changing its destination, source address or hop count. For example, an attacker advertising a zero hop count for all destinations may want all devices around it to route packets for all the destinations through itself. SL-AODV provides defense against such types of attacks by authenticating (at each intermediate device) the sender of each routing message as well as the number of hops to the sender.

In another attack, malicious routing messages claiming a large sequence number can attempt to force the receiving device to perform a large number if hash operation in order to authenticate the routing message. In order to guard against such attacks the receiving device should limit the number of hashes it is willing to perform for each authentication, discarding messages that do not meet that criteria.

Another well known attack known as the wormhole attack is carried out by a pair of colluding attackers in the network, linked via a tunnel. In such a case, every routing message received by an attacker A, is sent over the tunnel to attacker B, to then be forwarded normally by B. Similarly, B tunnels every routing message it receives to A. This means that the hop count field in these routing messages will not be changed, so most routes between devices towards the two ends of the network will pass through A and B thus creating a virtual vertex cut of the devices in the network. This type of attack is very difficult to detect as no false messages are being created or maliciously modified, and cannot be protected against by SL-AODV.

Lastly, compromised devices can also be used to launch attacks. Since SL-AODV requires devices to authenticate the sender of each routing message compromised devices cannot spoof their source address, and are restricted to attempting DoS attacks. SL-AODV protects against such DoS attacks by limiting the number of routing messages from any one source, so neighbors of malicious devices may temporarily or permanently block any further messages from that address. As the source is not able to authenticate itself as any other address, it cannot perform any more attacks.

7.3 Conclusions

This chapter has presented the design and evaluation of SL-AODV, a new ad-hoc routing protocol that requires some amount of prior security coordination and relies only on efficient symmetric cryptography to secure routing in resource constrained

ad-hoc networks. It operates on demand; the design being based on the basic operation of AODV. SL-AODV prevents attackers or compromised devices from tampering with routes consisting of uncompromised devices, and also protects against a variety of DoS attacks. However due to the way we employ hash chains and network wide broadcast of routing messages, SL-AODV is only suitable for small networks with limited number of hops. It enables devices to only forward authenticated routing messages and protects against multiple un-coordinated active attackers, in spite of compromised devices in the network.

Chapter 8

Concluding Remarks

The world wide success of mobile telephony can in large part be contributed to the changing requirements of our mobile and busy lifestyle. However as we get busier and move around even more, there is need for a smarter kind of connectivity. Just being able to have voice conversations with family or exchange emails with business partners is not enough. What remains is that devices and environments need to become more responsive to the needs of the user. Such is the vision of the Personal Network, and in the first chapter we attempted to give the reader an overview of its concept, architecture and security requirements.

With computing devices being integrated into everyday objects, pervasive computing is happening. However we believe that Personal Networks will only succeed if they meet (amongst other things) the security requirements of their users. This has been the area of focus for our work; the importance of which is highlighted by the fact that security must be an integral part of system design and not an add-on later in the design process. One of our conclusions in this regard is that the new characteristics and possibilities offered by Personal Networks means that old solutions are often not suitable in their current form. This could be something simple like a matter of scale, for example given the number of personal devices direct user interaction with passwords will no longer be a suitable authentication mechanism, or it may be something more fundamental like modifying security and trust models to incorporate the concept of "personal".

When discussing security, an important aspect lies in formulating a security model for Personal Networks that is both powerful enough to meet the needs of the users, yet at the same time comprehensible to them. Personal Networks are meant to be used by people from all walks of life and not only the technically inclined. Security models and mechanisms that are too complicated will inevitably fail in their purpose. Consequently one of our tasks was to formulate a simple way for Personal Networks to model trust amongst personal devices. We believe that the three fundamental security properties of confidentiality (preventing unauthorized reads), integrity (preventing unauthorized writes) and availability (ensuring autho-

rized users are not denied service) all rest firmly on a distinction between personal (authorized) and foreign (unauthorized) devices. As personal devices belong to the same user, we feel that it is not necessary to implement access control to PN services based on the specific identity of the personal device but rather its ability to demonstrate membership of the PN. Consequently secure intra-PN service access leverages on the security of the underlying communication network, which is based on link layer group security within the cluster and VPN-like tunnels between clusters.

With many of the devices envisioned to be part of a PN being portable, it was necessary to handle heterogeneous hardware constraints and device mobility. Many existing standards, such as IEEE 802.1X and IPSec were developed to apply cryptographic concepts and algorithms to fixed networking problems and are not applicable to the ad-hoc and constrained environment of the PN in their current form. Of course cryptographic algorithms and protocols such as IPSec are not enough alone to secure a system. Given the heterogeneous nature of the devices in the PN, we concluded that a fully distributed approach was too costly for many constrained devices. Our approach thus centralizes management roles to be performed on-demand by suitable PN devices. For intra-PN security this meant defining the role of the Security Manager and Security Agent. The Security Manager was to be the cyber representative of the PN owner and enable the creation of trust in the PN by the use of preloaded Personal TGS tickets. The PN was to be established based on these tickets and the EAP-PTGS authentication mechanism we defined. For inter-PN security we ported the generic AAA model to PNs and defined the role of the Authentication and Authorization Server (AAS).

Given how personal devices are integrated into the PN, loss of a device not only has implications for the information stored on the device itself, but on services and data stored within the rest of the PN. One of our more important assumptions in this regard was that it is too costly to require personal devices to be tamper proof. Consequently we also looked at mechanisms for revoking trust. Our other contributions include quantifying the network overhead of some of our proposals and introducing a new lightweight secure ad-hoc routing protocol called SL-AODV.

The purpose of this thesis, as a first work in proposing a security architecture for this new concept of personal communication, has been met. However any complete work on security in Personal Networks must certainly go beyond a single PhD thesis. Some important issues remain, many of which relate to evaluating how far our model meets our high level requirements. These include checking if the model can meet the security requirements of the users, support their necessary social interactions and be as usable as we hope. However these questions can only be solved by building a prototype and testing it with real users. In this regard one focus of the study should be in re-evaluating the question: what needs protection, and why, and for whose benefit? It is possible that some of the implicit assumptions we have made do not stand up to real world usage and require a rethink of policy.

Another issue to consider is the overhead of managing security. As a self-

configurable and "smart" entity PNs should be able to minimize the need for manual management. Ideally we would like the security management functionality (e.g., regular backups of the Security Manager) to be integrated with the other non-security related PN management functionality (e.g., software and firmware updates). Finally, we understand that standardization work is required to guarantee the success of any proposed or modified security protocol. If the modified security protocols proposed in this work do well in meeting the needs of the prototype group, then efforts must be made towards their standardization.

Appendix A

Protocol specification Needham Schroeder Symmetric Key

Alice (A) authenticates herself to Bob (B) using a mutually trusted server M.

N_A, N_B	Nonce
K_{AM}, K_{BM}, K_{AB}	Session key
dec	Decrement operation

1. $A \Longrightarrow M : A, B, N_A$
2. $M \Longrightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}K_{BM}\}K_{AM}$
3. $A \Longrightarrow B : \{K_{AB}, A\}K_{BM}$
4. $B \Longrightarrow A : \{N_B\}K_{AB}$
5. $A \Longrightarrow B : \{dec(N_B)\}K_{AB}$

Terminology

AAA	Authentication, Authorization and Accounting
AAS	Authentication and Authorization Server
AES	Advanced Encryption Standard
AODV	Ad Hoc On-demand Distance Vector
ARP	Address Resolution Protocol
CA	Certification Authority
DoS	Denial of Service
DSR	Dynamic Source Routing
DH	Diffie Hellman protocol
DWSN	Distributed WSN
EAP	Extensible Authentication Protocol
ECC	Elliptic Curve Cryptography
GW	Gateway
HMAC	Hashed Message Authentication Code
HW	Hardware
HWSN	Hierarchical WSN
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronic Engineers
IKE	Internet Key Exchange
IP	Internet Protocol
IPSec	IP Security
KDC	Key Distribution Center
KINK	Kerberized Internet Negotiation of Keys
LLSC	Location Limited Side Channel
MAC	Message Authentication Code
MAGNET	My Adaptive Global Net
MANA	Manual Authentication
MANET	Mobile Ad Hoc Network
NAS	Network Access Server
Moped	Mobile Grouped device
NAT	Network Access Translation
Nonce	Number used once
NS	Network Simulator 2

PAN	Personal Area Network
P2P	Peer 2 Peer
P-PAN	Personal PAN
PAN	Personal Area Network
PACWOMAN	Power Aware Communications for Wireless Optimised Personal Area Networks
PDA	Personal Digital Assistant
PFS	Perfect Forward Security
PKI	Public Key Infrastructure
PN	Personal Network
PN-F	Personal Network Federation
PSK	Pre Shared Keys
PTGS	Personal TGS
PTGS-AS	Personal TGS Authentication Server
PTGS-AC	Personal TGS Authentication Client
QoS	Quality of Service
RERR	Route Error
RREP	Route Reply
RREQ	Route Request
RF	Redundancy Factor
SA	Security Agent
SAC	Security Agent Capable
SAI	Security Agent Incapable
SAW	Security Agent Weight
SHAMAN	Security for Heterogeneous Access in Mobile Applications and Networks
SL-AODV	Secure Lightweight AODV
SSL	Secure Socket Layer
SW	Software
TCP	Transmission Control Protocol
TGS	Ticket Granting Server
TGT	Ticket Granting Ticket
TLS	Transport Layer Security
TTL	Time To Live
UMTS	Universal Mobile Telecommunications System
VoIP	Voice over IP
VPN	Virtual Private Network
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network

Publications

During the course of this project, a number of public presentations have been made which are based on the work presented in this thesis. They are listed here for reference.

- A. Jehangir and S. M. Heemstra de Groot, "A secure and lightweight adhoc algorithm for Personal Networks", International Wireless Summit (IWS 2005), Aalborg, Denmark, September 2005, pp. 1998-2002, Causal Productions, ISBN 87-90834-79-8.
- A. Jehangir and S. M. Heemstra de Groot, "A Security Architecture for Personal Networks", First International Workshop on Personalized Networks (PerNets 2006), San Jose, California, USA, July 2006, IEEE, ISBN 0-7803-9791-6.
- T.J.M. Coenen, et. al., "Architectural aspects of QoS aware Personal Networks", First International Workshop on Personalized Networks (PerNets 2006), San Jose, California, USA, July 2006, IEEE, ISBN 1-4244-0498-3. .
- A. Jehangir and S. M. Heemstra de Groot, "Evaluating secure cluster formation in Personal Networks", Wireless communication and networking conference (WCNC 2007), Hong Kong, China, March 2007, pp. 3134-3140, IEEE, ISBN 1-4244-0659-5.
- A. Jehangir and S. M. Heemstra de Groot, "Securing inter-cluster communication in Personal Networks", Second International Workshop on Personalized Networks (PerNets 2007), Philadelphia, USA, August 2007, IEEE, ISBN 1-4244-1024-8.
- A. Jehangir and S. M. Heemstra de Groot, "Securing Personal Network Clusters", Third International Conference on Security and Privacy in Communication Networks (SecureComm 2007), Nice, France, Sep. 2007, pp. 320-329, IEEE, ISBN 1-4244-0975-6.

Bibliography

[1] I. G. Niemegeers and S. M. Heemstra de Groot, "Research issues in ad-hoc distributed personal networking", Wireless Personal Communications, vol. 26, no. 2-3, pp. 149–167, August 2003.

[2] QoS for Personal Networks @ Home, http://qos4pn.irctr.tudelft.nl/

[3] N. Okabe, S. Sakane, et al, "Security Architecture for Control Networks using IPsec and KINK", International Symposium on Applications and the Internet (SAINT), Trento, Italy, Jan 2005, pp. 414-420, IEEE, ISBN 0-7695-2262-9.

[4] D. J. Malan, M. Welsh and M. D. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," Sensor and Ad Hoc Communications and Networks (SECON), Santa Clara, CA, USA, Oct 2004, pp. 71- 80, IEEE, ISBN 0-7803-8796-1.

[5] N. R. Potlapally, S. Ravi, A. Raghunathan and N. K. Jha, "Analyzing the Energy Consumption of Security Protocols", International Symposium of Low Power Electronics and Design (ILSPED), Seoul, Korea, August 2003, pp. 30-35, IEEE, ISBN 1-58113-682-X.

[6] H. Yan and Z. J. Shi, "Studying Software Implementations of Elliptic Curve Cryptography", 3^{rd} International Conference on Information Technology: New Generations (ITNG), Las Vegas, Nevada, USA, April 2006, pp. 78-83, IEEE, ISBN 0-7695-2497-4.

[7] S. Vasudevan, J. Kurose and D. Towsley, "Design and analysis of a leader election algorithm for mobile ad-hoc networks", 12^{th} IEEE International Conference on Network Protocols (ICNP), Berlin, Germany, Oct 2004, pp. 350 - 360 , IEEE, ISBN 0-7695-2161-4.

[8] IST MAGNET, http://www.ist-magnet.org/

[9] IST-507102 MAGNET/WP2.1/RWTH/D2.1.2/PU/001/24.10.2005, Overall Secure PN Architecture, October 31 2005.

[10] Martin Jacobsson, "Personal Networks - An Architecture for Self-Organized Personal Wireless Communications", PhD Thesis ISBN 978-90-9023196-9, Technical University Delft, The Netherlands, June 17, 2008.

[11] A. Perrig, R. Szewczyk, V. Wen, D. Culler and J.D. Tygar. "SPINS: Security protocols for sensor networks", Wireless Networks, vol.8, no.5, pp. 521-534, Kluwer Academic Publishers, 2002.

[12] S. Zhu, S. Setia and S. Jajodia, "LEAP: Efficient security mechanisms for large-scale distributed sensor networks," 10^{th} ACM Conference on Computer and Communications Security (CCS), Washington, D.C, USA, Oct 2003, pp. 62 - 72, ACM, ISBN 1-58113-738-9.

[13] S.A. Camtepe and B. Yener, "Key Distribution Mechanisms for Wireless Sensor Networks: a Survey", Technical Report TR-05-07, Rensselaer Polytechnic Institute, March 2005.

[14] C. Karlof, N. Sastry and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks", The 2^{nd} ACM Conference on Embedded Networked Sensor Systems (SenSys), Baltimore, MD, USA, Nov 2004, pp. 162-175, ACM, ISBN 1-58113-879-2.

[15] Y. W. Law, J. Doumen and P. Hartel, "Survey and Benchmark of Block Ciphers for Wireless Sensor Networks", ACM Transaction on Sensor Networks (TOSN), vol. 2, no. 1, pp. 65-93, Feburary 2006.

[16] Jan-Erik Ekberg, "Key establishment in constrained devices", Research Seminar on Network Security, Helsinki, Finland, May 2006.

[17] S. Basagni, K. Herrin, D. Bruschi and E. Rosti, "Secure pebblenets", 2^{nd} ACM Interational Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Long Beach, California, USA, Oct 2001, pp. 156-163, ACM, ISBN 1-58113-428-2.

[18] D. W. Carman, P. S. Kruus and B. J. Matt, "Constraints and approaches for distributed sensor network security", Technical report, NAI Labs, Security Research Division, Glenwood, Maryland, USA, Sep 2000.

[19] A. Perrig, R. Canetti, J.D. Tygar and D. Song, "The TESLA broadcast authentication protocol", RSA CryptoBytes, 5:2, Summer/Fall, pp. 2-13, 2002.

[20] S. Zhu, S. Setia, S. Xu and S. Jajodia, "Gkmpan: An efficient group rekeying scheme for secure multicast in ad-hoc networks", 1^{st} Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous), Boston, MA, USA, Aug 2004, pp. 42-51, IEEE, ISBN: 0-7695-2208-4.

[21] F. Stajano and R. Anderson, "Resurrecting Duckling: Security Issues for Ubiquitous Computing" in Computer, vol. 35, no. 4, pp. 22–26, April 2002.

[22] F. Stajano, "The resurrecting duckling—what next?", Lecture Notes in Computer Science, vol 2133, pages 204–21, Springer-Verlag, 2001.

[23] N. Prigent, C. Bidan, J.-P. Andreaux and O. Heen, "Secure long term communities in ad hoc networks," 1^{st} ACM workshop on Security of Ad Hoc and Sensor Networks (SASN), Fairfax, VA, USA, Aug 2003, pp. 115-124 , ACM, ISBN 1-58113-783-4.

[24] Berkeley MICA motes, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA.pdf

[25] J. Großschädl, S. Tillich, C. Rechberger, M. Hofmann and M. Medwed, "Energy Evaluation of Software Implementations of Block Ciphers under Memory Constraints", The 10^{th}Conference on Design, Automation and Test in Europe (DATE), Nice, France, April 2007, pp. 1110-1115, EDA Consortium, ISBN 978-3-9810801-2-4.

[26] A. Vitaletti and G. Palombizio, "Rijndael for sensor networks: is speed the main issue?", Electronic Notes in Theoretical Computer Science (ENTCS), vol.171, no. 1, pp. 71-81, April 2007

[27] C. Neuman, T. Yu, S. Hartman and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[28] C. Gehrmann, K. Nyberg and C.J. Mitchell, "The personal CA - PKI for a Personal Area Network", 11^{th}IST Mobile & Wireless Communications Summit, Thessaloniki, Greece, June 2002.

[29] IST SHAHMAN project, http://www.isrc.rhul.ac.uk/shaman/

[30] IST-2000-25350 SHAMAN, D13 - Final technical report - results, specifications and conclusions, November 30, 2002.

[31] R. Needham and M. Schroeder, "Using encryption for authentication in large networks of computers", Communications of the ACM, 21(12):993–999, Dec 1978.

[32] D. Balfanz, D. K. Smetters, P. Stewart and H. C. Wong, "Talking to strangers: Authentication in ad-hoc wireless networks", Network and Distributed Systems Security Symposium (NDSS), Diego, California, USA, Feb 2002, pp. 23–35, Internet Soc, ISBN 1-891562-14-2

[33] D. E. Denning and G. M. Sacco, "Timestamps in key distribution protocols", Communications of the ACM, vol. 24, no. 8, Aug 1981.

[34] A. Freier, P.Karlton and P.Kocker, "The SSL Protocol, Version 3.0", Internet Draft, Nov 1996.

[35] P. Rohatgi, "A compact and fast hybrid signature scheme for multicast packet authentication", 6^{th} ACM Conference on Computer and Communications Security (CCS), Singapore, November 1999, pp. 93-100, ACM, ISBN 1-58113-148-8.

[36] A. Perrig, R. Canetti, D. Song, J. D. Tygar, "Efficient and Secure Source Authentication for Multicast", Network and Distributed System Security Symposium (NDSS), San Diego, CA, Feb. 2001, pp. 33-46, Internet Soc, ISBN 1-891562-11-8.

[37] H. Lim and C. Kim, "Multicast tree construction and flooding in wireless ad hoc networks", 3^{rd} ACM Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM), Boston, Massachusetts, USA, Aug 2000, pp. 61-68, ACM, ISBN 1-58113-304-9.

[38] D. J. Scott, "Relying on Time Synchronization for Security in Ad Hoc Networks", 43^{rd} ACM Annual Southeast regional conference, New York, NY, USA, March 2005, pp. 87-91, ACM, ISBN 1-59593-059-0.

[39] C. Perkins, E. M. Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, July 2003.

[40] M. Guerrero, "Secure ad-hoc on-demand distance vector (SAODV) routing", draft-guerrero-manet-saodv-01.txt, August 2004.

[41] L. Lamport, "Password authentication with insecure communication", Communications of the ACM, 24(11):770–772, November 1981.

[42] N. Haller, "The S/KEY one-time password system", RFC 1760, February 1995.

[43] M. Brown, D. Cheung, D. Hankerson, J. Hernandez, M. Kirkup, and A. Menezes, "PGP in constrained wireless devices", The 9^{th}USENIX Security Symposium - Volume 9, August 2000, pp. 19-19, USENIX Association.

[44] R. Hauser, M. Steiner, M. Waidner, "Micro-payments based on iKP", Research Report 2791 (# 89269), IBM Research, 1996.

[45] P. Rohatgi, "A compact and fast hybrid signature scheme for multicast packet", The 6^{th} ACM Conference on Computer and Communications Security, Singapore, November 1999, pp. 93-100, ACM, ISBN 1-58113-148-8.

[46] R. Hauser, A. Przygienda, and G. Tsudik, "Reducing the cost of security in link state routing", The Symposium on Network and Distributed Systems Security (NDSS '97), San Diego, California, February 1997, pp. 93-100, Internet Soc, ISBN 0-8186-7767-8.

[47] K. Zhang, "Efficient protocols for signing routing messages", The Symposium on Network and Distributed Systems Security (NDSS '98), San Diego, California, March 1998, Internet Soc, ISBN 1-891562-01-0.

[48] D. Coppersmith and M. Jakobsson, "Almost optimal hash sequence traversal", The Fourth Conference on Financial Cryptography (FC '02), Lecture Notes in Computer Science, vol. 2357, pp. 102-119, 2003.

[49] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", In Mobile Computing, Tomasz Imielinski and Hank Korth (Eds.), chp. 5, pp. 153-181, Kluwer Academic Publishers, 1996.

[50] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[51] B. Aboba, D. Simon, P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", Internet-Draft, Nov 2007.

[52] J. Vollbrecht, et al, "AAA Authorization Framework", RFC 2904, August 2000.

[53] Paul Dourish, Rebecca E. Grinter, Jessica Delgado de la Flor and Melissa Joseph, "Security in the Wild: User Strategies for Managing Security as an Everyday Practical Problem", Personal and Ubiquitous Computing, vol. 8, no. 6, pp. 391–401, Springer, November 2004.

[54] Alma Whitten, J. D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0", In the 8^{th} USENIX Security Symposium - Volume 8, Washington DC, USA, August 1999, pp. 14-18, USENIX Association.

[55] J. Vollbrecht, et al, "AAA Authorization Requirements", RFC 2906, August 2000.

[56] B. Aboba and D. Simon, "PPP EAP TLS authentication protocol", RFC 2716, Oct 1999.

[57] F. Bersani and H. Tschofenig, "The EAP-PSK Protocol: a Pre-Shared Key EAP Method", RFC 4764, Jan 2007.

[58] C. Rigney, S.Willens, A. Rubens and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 3575, June 2003.

[59] B. Aboba and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, Sept 2003.

[60] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko, "Diameter Base Protocol", RFC 3588, Sept 2003.

[61] IEEE 802.16, Part 16: Air Interface for Fixed Broadband Wireless Access Systems, IEEE Std 802.16, 2004 edition, October 2004.

[62] P. Eronen, T. Hiller, and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, Aug 2005.

[63] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, Apr. 2006.

[64] C. Adams, S. Farrell, T. Kause, and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, Sept 2005.

[65] D. Liu and P. Ning, "Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks", The Network and Distributed System Security Symposium (NDSS '03), February 2003, Internet Soc, ISBN 1-891562-16-9.

[66] J. Jonsson and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, Feb 2003.

[67] International Telecommunication Union. Information technology — ASN.1 encoding rules — specification of basic encoding rules (BER), canonical encoding rules (CER), and distinguished encoding rules (DER). ITU-T Recommendation X.690, July 2002.

[68] IEEE P802.11i/D10.0. Medium Access Control (MAC) security enhancements, amendment 6 to IEEE Standard for local and metropolitan area networks part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications, April 2004.

[69] Po-Wah Yau and Chris J. Mitchell, "Reputation methods for routing security for mobile ad hoc networks", Joint IST Workshop on Mobile Future and Symposium on Trends in Communications (SympoTIC), Bratislava, Slovakia, October 2003, pp. 130- 137, IEEE, ISBN 0-7803-7993-4.

[70] M. Jakobsson, "Fractal hash sequence representation and traversal", The 2002 IEEE International Symposium on Information Theory (ISIT '02), July 2002, pp. 437-445, IEEE, ISBN 0-7803-7501-7.

[71] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks", Wireless Networks, vol. 11, no. 1-2. pp. 21-38, January 2005.

[72] S. Murthy and J.J. Garcia-Lunca-Aceves, "An efficient routing protocol for wireless networks", ACM Mobile Networks and Applications Journal, pages 183–197, October 1996.

[73] P. Papadimitratos and Z.J. Haas, "Secure Message Transmission in Mobile Ad Hoc Networks," Elsevier Ad Hoc Networks J., Elsevier, vol. 1, no. 1, pp. 193–209, 2003.

[74] V. Park and M. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks", The 16^{th} Annual Conference on Computer Communications (INFOCOM), Kobe, Japan, April 1997, pp. 1405-1413 vol.3, IEEE, ISBN 0-8186-7780-5.

[75] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers", ACM SIGCOMM Computer Communications Review, vol 24, no. 4, pp. 234–244, October 1994.

[76] L. Hu and D. Evans, "Using directional antennas to prevent wormhole attacks", 11^{th}Annual Network and Distributed System Security Symposium (NDSS), San Diego, California, USA, Feb 2004, Internet Soc, ISBN 1-891562-18-5.

[77] IXI Mobile, http://www.ixi.com/.

[78] Y. C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless networks", The Conference on Computer Communications (INFOCOM), San Francisco, USA, Mar 2003, pp. 1976- 1986 vol.3, IEEE, ISBN 0-7803-7752-4.

[79] P. Papadimitratos and Z.J. Haas, "Secure Link State Routing for Mobile Ad Hoc Networks", Symposium on Applications and the Internet Workshops, 2003, pp. 379-383, IEEE, ISBN 0-7695-1873-7.

[80] IST-507102 MAGNET/WP4.3/D4.3.2/v1.0, "Final Architecture of the Network-Level Security Architecture Specification", March 2005.

[81] IST-027396 MAGNET Beyond/WP4/D4.2.1/v1.0, "First solutions for implementation of Key Management and Crypto techniques", Dec 2006.

[82] IST-507102 MAGNET/WP1.1/D1.1.1c/v1.0, "Final User Requirements for the PN Service Architecture", December 31 2005.

[83] Network Simulator 2 (NS-2), http://www.isi.edu/nsnam/ns

[84] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, Nov 1998.

[85] OpenVPN, http://openvpn.net/

[86] S. Sakane, K. Kamada, M. Thomas and and J. Vilhuber, "Kerberized Internet Negotiation of Keys (KINK)", RFC 4430, March 2006.

[87] S. Kent, and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[88] S. Kent, and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC2406, November 1998.

[89] W. Diffie and M. Hellman, "New Directions in Cryptography", In IEEE Transactions on Information Theory, vol. IT-22(6), pp. 644-654, Nov 1976.

[90] The Racoon2 project, http://www.racoon2.wide.ad.jp/w/

[91] V.Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, H. Eberle, S. C. Shantz, "Sizzle: A Standards-based end-to-end Security Architecture for the Embedded Internet", The 3^{rd} IEEE International Conference on Pervasive Computing and Communications, March 2005, pp. 247-256, IEEE, ISBN 0-7695-2299-8.

[92] D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, Nov 1998.

[93] S. Buchegger and J. Le Boudec, "Performance Analysis of the CONFIDANT Protocol: Cooperation of Nodes and Fairness In Dynamic Ad-hoc NeTworks", The 3^{rd} International ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC), Lausanne, CH, June 2002, pp. 226-236, ACM, ISBN 1-58113-501-7.

[94] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[95] IST PACWOMAN, Power Aware Communications for Wireless Optimised Personal Area Networks, http://www.imec.be/pacwoman/Welcome.shtml.

[96] IST-2001-34157 PACWOMAN, D2.1 - System Requirements and Analysis, October 8 2002.

[97] L. Zhou, and Z. Haas, "Securing ad hoc networks", IEEE Network Magazine, vol. 13, no. 6, pp. 24-30, 1999.

[98] E.M.Belding-Royer, "A secure routing protocol for ad-hoc networks", 10^{th}IEEE International Conference on Network Protocols (ICNP), November 2002, pp. 78-87, IEEE, ISBN 0-7695-1856-7.

[99] National Institute of Standards and Technology (NIST), "Secure hash standard", May 1993. Federal Information Processing Standards (FIPS) Publication 180-1.

[100] Z. J. Haas, P.Papadinitratos. "Secure routing for mobile ad-hoc networks", SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), Jan 2002.

[101] IEEE Standard for Port-Based Network Access Control, IEEE Std 802.1X-2001, Oct 2001.

[102] S. Marti, T. J Giuli, K. Lai and M. Baker, "Mitigating routing misbehavior in mobile ad-hoc networks", 6^{th} International Conference on Mobile Computing and Networking (MobiCom), Boston, MA, USA, August 2000, pp. 255-265, ACM, ISBN 1-58113-197-6.

[103] P. Eronen and H. Tschofenig, "Extension for EAP Authentication in IKEv2", Internet-Draft, June 2006.

[104] Web Services Security: SAML Token Profile 1.0, OASIS Standard, 2004.

[105] Y. Hu, D. Johnson, A. Perrig, "SEAD: secure efficient distance vector routing for mobile wireless ad-hoc networks", The 4^{th} IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), 2002, pp. 3-12, IEEE, ISBN 0-7695-1647-5.

[106] Bluetooth SIG, "Specification of the Bluetooth System – Version 1.1 B", http://www.bluetooth.com/, 2001.

[107] R. L. Rivest, "The MD5 message-digest algorithm", RFC 1321, Internet Engineering Task Force, April 1992.

[108] ZigBee Alliance, http://www.zigbee.org/.

[109] IEEE P802.11 - The Working Group for WLAN Standards, http://www.ieee802.org/11/.

[110] J.Villadangos, A. Cordoba, F. Farina, and M. Prieto, "Efficient leader election in complete networks", 13^{th}Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP), February 2005, pp. 136-143, IEEE, ISBN 0-7695-2280-7.

[111] S. Vasudevan, J. Kurose, and D. Towsley, "Design and analysis of a leader election algorithm for mobile ad hoc networks", 12^{th}IEEE International Conference on Networks Protocols (ICNP), October 2004, pp. 350-360, IEEE, ISBN 0-7695-2161-4.

[112] http://hypertextbook.com/facts/2001/KhalidaNisimova.shtml

[113] Robin Kravets, Casey Carter, and Luiz Magalhaes, "A Cooperative Approach to User Mobility", ACM Computer Communications Review, vol. 31, pp. 57-69, October 2001.

[114] IST-507102 MAGNET/WP1.3/D1.3.1.b/DTU/R/PU/001/1.0, "UserCentric Scenarios for PNs of a valid architecture", December 2005

[115] R. Prasad and K. Skouby, "Personal network (PN) applications," Wireless Personal Communications, vol. 33, no. 3-4, pp. 227–242, 2005.

[116] A. Jehangir and S. M. Heemstra de Groot, "A secure and lightweight adhoc algorithm for Personal Networks", International Wireless Summit (IWS 2005), Aalborg, Denmark, September 2005, pp. 1998-2002, Causal Productions, ISBN 87-90834-79-8.

[117] A. Jehangir and S. M. Heemstra de Groot, "A Security Architecture for Personal Networks", First International Workshop on Personalized Networks (PerNets 2006), San Jose, California, USA, July 2006, IEEE, ISBN 0-7803-9791-6.

[118] T.J.M. Coenen, et. al., "Architectural aspects of QoS aware Personal Networks", First International Workshop on Personalized Networks (PerNets 2006), San Jose, California, USA, July 2006, IEEE, ISBN 1-4244-0498-3. .

[119] A. Jehangir and S. M. Heemstra de Groot, "Evaluating secure cluster formation in Personal Networks", Wireless communication and networking conference (WCNC 2007), Hong Kong, China, March 2007, pp. 3134-3140, IEEE, ISBN 1-4244-0659-5

[120] A. Jehangir and S. M. Heemstra de Groot, "Securing inter-cluster communication in Personal Networks", Second International Workshop on Personalized Networks (PerNets 2007), Philadelphia, USA, August 2007, IEEE, ISBN 1-4244-1024-8.

[121] A. Jehangir and S. M. Heemstra de Groot, "Securing Personal Network Clusters", Third International Conference on Security and Privacy in Communication Networks (SecureComm 2007), Nice, France, September 2007, pp. 320-329, IEEE, ISBN 1-4244-0975-6.