KARI KOSTIAINEN

INTUITIVE SECURITY INITIATION USING LOCATION-LIMITED CHANNELS

Master's Thesis
14th April 2004

Supervisor: Professor Teemupekka Virtanen
Advisor: N. Asokan, Ph.D.

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering
Telecommunications Software and Multimedia Laboratory

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Kari Kostiainen
Title:	Intuitive Security Initiation Using Location-Limited Channels
Date:	14th April 2004
Pages:	78
Department:	Department of Computer Science and Engineering
Professorship:	T-110
Supervisor:	Professor Teemupekka Virtanen
Instructor:	N. Asokan, Ph.D.

Ad hoc communication is becoming increasingly popular. Mobile phones, PDAs and laptops can be used to create ad hoc networks for many different purposes. For example a waiting passenger could set up an ad hoc network for wirelessly printing a personal document in an airport printing lounge, or alternatively a householder could use set up an ad hoc network for controlling his home appliances with his mobile phone. The need for security in ad hoc communication is also obvious. The traveler does not want to share the personal document with the whole printing lounge and the householder does not want to let outsiders control his home appliances.

Ad hoc communication differs from traditional communication in many ways and the traditional security methods are not always applicable in the different ad hoc environments. Some new security methods have been developed for ad hoc environments, however, these methods have proved to be either cumbersome for the user or insecure. For example Bluetooth pairing requires the use of PIN codes. Long codes make the security initiation process difficult and short ones do not provide sufficient security.

Recent academic research has introduced the idea of using so called location-limited channels, such as infrared or short range radio connections, for proximity based user friendly security initiation. Location-limited channel is a separate channel from the main communication link that can be used to exchange initial security information, such as keys and addresses, between devices that are physically close to each other. After the location-limited channel security initiation, the secure connection can be set up for the main communication link.

The purpose of this thesis is to design a location-limited channel based user friendly security initiation system. We identify different possibilities for using location-limited channels in security initiation, design a set of location-limited channel based protocols and describe a software architecture for the security initiation system. We also describe the implemented prototype of the security initiation system and evaluate the design and implementation.

Keywords:	location-limited channel, security, usability, ad hoc networks
Language:	English

TEKNILLINEN KORKEAKOULU

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Kari Kostiainen
Työn nimi:	Käyttäjäystävällinen turvallisuuden alustaminen etäisyysrajoitteisilla kanavilla
Päivämäärä:	14. huhtikuuta 2004
Sivuja:	78
Osasto:	Tietotekniikan osasto
Professuuri:	T-110
Työn valvoja:	Professori Teemupekka Virtanen
Työn ohjaaja:	N. Asokan, Ph.D.

Ad hoc -verkoissa tapahtuva kommunikointi on koko ajan yleistymässä. Kannettavat tietokoneet, puhelimet ja PDA-laitteet voivat muodostaa ad hoc -verkkoja moniin eri tarkoituksiin. Esimerkiksi odottava matkustaja voi käyttää ad hoc -verkkoa tulostaakseen langattomasti dokumentin lentokentän odotusaulassa, tai vaihtoehtoisesti kotikäyttäjä voi hallita kodinkoneitaan matkapuhelimella ad hoc -verkon yli. Myös turvallisuutta tarvitaan ad hoc -verkoissa. Odottava matkustaja ei halua jakaa henkilökohtaista dokumenttia kaikkien odotusaulassa olevien ihmisten kanssa ja kotikäyttäjä ei halua antaa ulkopuolisten ohjata hänen kodinkoneitaan.

Ad hoc -verkot ovat monin tavoin erilaisia verrattuna perinteisiin tietokoneverkkoihin ja tämän takia perinteiset turvallisuuden luomiseen käytettävät menetelmät eivät aina sovellu ad hoc -verkoissa käytettäviksi. Joitain uusia menetelmiä on kehitetty varta vasten ad hoc -verkoille, mutta on osoittautunut, että nämä menetelmät eivät ole joko käyttäjäystävällisiä tai tarpeeksi turvallisia. Esimerkiksi Bluetooth-paritus vaatii PIN-koodien käyttöä. Pitkät koodit ovat hankalia käyttäjälle ja lyhyet koodit eivät takaa kunnollista turvallisuutta.

Viimeaikaisen tutkimustyön tuloksena on esitty, että niin kutsuttuja etäisyysrajoittteisia kanavia, kuten infrapunaa tai lyhyen kantaman radioaaltoja, voitaisiin käyttää laitteiden fyysiseen läheisyyteen perustuvaan käyttäjäystävälliseen turvallisuuden alustamiseen. Etäisyysrajoitteinen kanava on erillinen kanava pääkommunikointikanavasta ja sitä voidaan käyttää alustavan turvallisuusinformaation, kuten avainten ja osoitteiden, vaihtamiseen fyysisesti lähellä olevien laitteiden kesken. Tämän turvallisuuden alustamisen jälkeen turvallinen yhteys voidaan muodostaa pääkommunikointikanavalle.

Tämän diplomityön tarkoituksena on suunnitella etäisyysrajoitteisiin kanaviin perustuva, käyttäjäystävällinen turvallisuudenalustamisjärjestelmä. Tässä työssä me tunnistamme erilaisia tapoja etäisyysrajoitteisten kanavien käyttämiseen, suunnittelemme joukon etäisyysrajoitteisten kanavien käyttöön perustuvia protokollia ja suunnittelemme turvallisuudenalustamisjärjestelmän arkkitehtuurin. Tämä lisäksi kuvaamme toteutetun järjestelmän prototyypin ja evaluoimme sekä arkkitehtuurin että toteutuksen.

Avainsanat:	etäisyysrajoitteinen kanava, turvallisuus, käytettävyys, ad hoc -verkot
Kieli:	englanti

Preface

This thesis was done as a part of Magic Wand project at the Communication Systems Laboratory of Nokia Research Center. Many of the ideas presented in this thesis are results of long discussions with other Magic Wand project team members and I would like to express my gratitude to the following people.

Sampo Sovio spent countless hours with me thinking about the different aspects of this work. Philip Ginzboorg was always there when needed. Seamus Moloney helped me with my Symbian related problems and corrected my English. Kalle Kuismanen was fun to work with when setting up the demonstrations. Jan-Erik Ekberg always found the time to help us. And Pekka Laitinen provided good advice throughout the project.

I would also like to thank Asokan, my advisor on this thesis, who always found time to help me and guide me. I am grateful for all the valuable comments and guidance throughout the whole process. Also Pasi Eronen helped me tremendously during the writing by reading my drafts, giving comments and helping me to structure this thesis. I would also like to thank Professor Teemupekka Virtanen, my supervisor, who helped me to finalize this thesis.

Finally I would like to thank all my colleagues at the Communication Systems Laboratory for the enjoyable working environment, and special thanks to Elina for all your support.

Helsinki
14th April 2004

Kari Kostiainen

Contents

Abbreviations

AH Authentication Header protocol is part of IPsec and it provides authentication and integrity checking of received IP packets.

API Application Programming Interface provides certain services to application programmers through a set of functions.

CA Certification Authority is a party that signs the certificates of other parties.

CRL Certificate Revocation List is used to announce about certificates that should no longer be trusted.

DLL Dynamic Link Library denotes to program code that can be linked dynamically at runtime.

ESP Encapsulated Security Payload a protocol that is part of IPsec and it provides confidentiality, authentication and integrity checking.

IETF Internet Engineering Taskforce is an open international community for network designers, operators, vendors and researchers.

IKE Internet Key Exchange is a protocol for automatic negotiation of IPsec Security Associations.

IP Internet Protocol is the basic network layer protocol used in Internet.

IPsec Internet Protocol security architecture is a general and flexible framework that provides various security services on IP level.

KDC Key Distribution Center is a central server responsible of the distribution of cryptographic keys.

LLC Location-limited channel is a communication link that can be used for communication only within proximity.

MAC Message authentication Code is used to validate the authenticity of data.

MANA Manual Authentication is protocol for manually authenticating key exchange protocols.

NFC Near Field Communication is a short range radio communication technology.

OCSP Online Certificate Status Protocol is protocol for checking the status of a certificate online.

PDA Personal Digital Assistant is a small hand-held computer.

PGP Pretty Good Privacy is a public key encryption application and key distribution scheme.

PIN Personal Identification Number is a short code that is used to identify the user of some application or device.

RFID Radio-Frequency Identification is a radio system for short range communication.

RSA Rivest Shamir Adleman is a popular public key algorithm that is used for encryption and digital signatures.

SA Security Association is the set of security parameters used in one secure connection.

SDK Software Development Kit is a set of programming tools for creating applications.

SADB Security Association Database stores the different Security Associations in IPsec implementation.

SPD Security Policy Database contains the rules that IPsec implementation uses for IP packet handling.

SPI Security Parameter Index identifies the Security Associations that are used for different connections in IPsec.

SSH Secure Shell is an application for creating secure shell connections over TCP.

SSL Secure Socket Layer is widely used protocol for creating secure connection over TCP.

TCP Transmission Control Protocol provides reliable and connection oriented communication over IP.

TLS Transport Layer Security is the successor of SSL protocol.

USB Universal Serial Bus is a specification for connecting different devices with the same connector.

WEP Wireless Encryption Protocol is a security extension for wireless local area networks.

WLAN Wireless Local Area Network.

Chapter 1

Introduction

1.1 Motivation

Ad hoc communication is becoming more and more popular. Today mobile phones, PDAs and laptops are equipped with wireless communication interfaces that use technologies such as Bluetooth and WLAN. These devices can be used to set up local, temporary and proximity based *ad hoc networks* that do not require any existing fixed network infrastructure.

People can use these ad hoc networks for many different purposes. For example a waiting passenger could use an ad hoc network for wirelessly printing a personal document in an airport lounge. A home user could set up an ad hoc network between his computer and mobile phone for file transferring. Or alternatively, a group of businessmen could form an ad hoc network between their laptops for conferencing when they have a meeting outside the office. The possibilities for ad hoc communication are nearly endless. And we are likely to see many yet undiscovered usage scenarios for ad hoc environments as the technology gets more mature.

The need for security in ad hoc communication is also obvious. The waiting traveler does not want to share the personal document with the whole airport lounge. The home user wants to keep the communication between his computer and mobile phone private from his neighbor. And the businessmen require security because they may be transferring company confidential data in a public environment.

1.2 Problem statement

Ad hoc communication is fundamentally different from traditional computer communication. Traditional computer networks are based on fixed and static connections with centralized architectures. Ad hoc networks are dynamic in nature and possibly isolated from all fixed network infrastructures and centralized resources.

The purpose of this thesis is to address the problem of secure connection initiation between peers in ad hoc environments. The traditional security methods were developed for traditional fixed networks and in the new ad hoc environments the traditional methods are not always applicable.

Some new security methods have been developed for secure connection initiation in ad hoc environments. However, these methods have proved to be neither very usable or secure. For example Bluetooth pairing requires the use of PIN codes. The use of long PIN codes makes the pairing procedure cumbersome and the use of short PIN codes does not provide sufficient security. What is needed is a user friendly method for initiating secure connections between peers in ad hoc environments.

Recent academic research has introduced the idea of using so called *location-limited channels* (such as infrared or short range radio connections) for proximity based pre-authentication and secure connection initiation in ad hoc environments. Stajano and Anderson introduced the Resurrecting Duckling model [59] that used physical connections for transferring shared secrets. Balfanz et al. [16] extended the model to include wireless location-limited channels with the use of public key cryptography.

The main goal of this thesis work is to design a security initiation system that is based on location-limited channels. This system should initiate security associations between peers in various ad hoc environments using location-limited channels and set up complete secure connections with the initiated security associations. A prototype of the security initiation system is also implemented and both the design and the implementation are evaluated in this thesis.

1.3 Evaluation criteria

The design and the implementation will be evaluated according to the following criteria. These same criteria will be also used to compare this work with other existing solutions.

- *Security:* What are the security properties of the system? How do the security properties of the system vary in different usage scenarios and environments? How do the user interactions affect the security of the created connections?
- *Usability:* What kind of user interactions does the use of the system require? Is the security initiation process intuitive and user friendly. How many user interactions are needed to set up a secure connection?
- *Generality:* Is the system architecture applicable to a wide range of different usage scenarios and environments? Can the system be used with a variety of existing security protocols and applications. Can the system be extended easily without major modifications to existing design?
- *Simplicity:* Is the system design simple, easy to understand and implement?

1.4 The scope of this thesis

This thesis concentrates on intuitive security initiation in ad hoc environments. The goal is to design a system that can be used to initiate and set up secure connections. The primary purpose is not to design a new security protocol, but to initiate the use of existing security protocols, such as IPsec [39] or TLS [24].

The goal of this thesis is to design a general security initiation system that can be used in various ad hoc environments with different location-limited channels. However, the purpose is not to implement the whole designed system, but only a prototype of the system with selected supported features.

This thesis also addresses the usability aspects of secure connection initiation. The aim is to design a system that makes the task of initiating and setting up secure connections more user friendly than with the existing solutions. However, this thesis is not supposed to be a full-fledged usability study.

1.5 Organization of this thesis

The rest of this thesis is organized as follows. Chapter 2 presents necessary background information for this thesis. Chapter 3 defines the requirements for the location-limited channel based security initiation system. Chapter 4 analyses different location-limited channels, defines the location-limited channel protocols and presents security initiation system design. Chapter 5 describes the implemented prototype. In Chapter 6 the design and implementation are evaluated and compared to other existing solutions. Finally, Chapter 7 contains a summary and the conclusions of this work.

Chapter 2

Background

This chapter describes the background information for this thesis. The rest of this chapter is organized as follows. Section 2.1 discusses traditional computer networks and Section 2.2 focuses on the security issues that exist in traditional computer networks. Section 2.3 describes ad hoc networks and Section 2.4 focuses on security issues of ad hoc networks. Section 2.5 discusses proximity based security initiation. Section 2.6 addresses issues related to group communication and Section 2.7 focuses on usability. Finally, Section 2.8 presents related work.

2.1 Traditional networks

In this thesis we use the term *traditional network* to denote to networks that are based on static devices connected with stable wired connections. The most popular network type today is TCP/IP network and the biggest representative of that network type is the global Internet.

The different networks can be characterized in many ways. One criteria for characterizing is whether the devices connected to the network are static or mobile. The traditional networks are typically all static in nature. There might some devices that change their point of attachment to the network, but the main physical core of the network is not moving.

The different kind of networks can be also characterized by the infrastructure support of the network. Two types of infrastructure are relevant for this thesis. The first type is the server infrastructure of on-line servers which provide various services such as name service, directory service, and trusted third party service. The second type is the organizational and administrative support such as registration of users, administration of the network, issuing of certificates and cross-certification agreements between different user domains [14]. Typically both of these support infrastructures are available in traditional networks.

2.2 Security in traditional networks

There are many definitions for the term *security*. One approach is to define security in terms of different security services [60]. From the point of view of this thesis, the most relevant security services are *confidentiality*, *integrity* and *authentication*. Confidentiality means

that only authorized parties can access some piece of information. Integrity ensures that the information is kept unchanged during the transportation or storage of the information. And authentication guarantees that the origin of the message is correctly identified.

This thesis addresses the problem of initiating and setting up secure connections. With the term *secure connection* we denote a connection that is confidentiality and integrity protected and authenticated. This means, that the communicating peers can be sure that they are talking to each other and the communication cannot be altered or eavesdropped by illegitimate parties.

Modern cryptographic methods make the confidentiality and integrity protection and authentication of a connection easy and reliable when the communicating parties have established *security associations* in a trusted manner. In this thesis the term security association denotes a data structure that contains the needed cryptographic keys for securing the connection and identity information about the other peer, such the network addresses or host name. The actual hard part is the distribution and management of the needed cryptographic keys and identity information in a large network environment [55].

2.2.1 Key distribution and management

To solve the problem of key distribution and management, several techniques have been proposed. In this section we will look at some of the methods that are typically used for distributing and managing the keys in traditional networks.

Key distribution center

One widely adopted method for the key distribution and management problem is to use a central server called *key distribution center* (KDC) [60]. All the participants of this scheme share some secret, such as cryptographic key or password, with the key distribution center. When for example a participant A wants to establish a secure connection with participant B, he sends a request to the KDC. This request contains the identities of A and B. The KDC generates a response to the request that contains a *session key* for protecting the connection between A and B. The response is typically encrypted using the secrets that A and B share with the KDC and only A and B can use this session key. The session key is typically valid only for the duration of the particular connection and for the next connection a fresh session key has to be acquired.

Probably the most used key distribution center implementation is Kerberos [43]. In Kerberos the network users, clients, authenticate themselves to servers using central authentication servers. The actual Kerberos protocol is relatively complex and the details of that protocol are out of scope of this thesis. However, the main ideology behind Kerberos is as explained above: All the clients share secret passwords and all the servers share secret keys with the authentication servers. The clients send requests to the authentication servers and the authentication servers provide responses that can be used to acquire the needed session key. The kerberos protocol and typically all the key distribution protocols are based on conventional symmetric cryptography.

The key distribution center methods do not solve the initial problem of setting up shared secrets between the key distribution center and other participants, and typically the initial shared secrets are set manually to all the devices. For this reason key distribution center

methods are typically used in somewhat closed environments such as university or company networks.

Public key certificates

Certificate is a collection of data that has been signed with a private key of a certification authority (CA) and thus the certification authority is validating the contents of the certificate [60]. The most common type of a digital certificate is an identity certificate which is used to bind a public key to its owner. An example of this kind of certificate is X.509 [35]. An identity certificate contains a public key and information that is needed to identify the owner of that public key. This certificate is signed with the private key of certification authority and given to the owner of the key. All the participants of this key management scheme are expected to know the public key of the trusted certification authority.

When two participants want to establish a connection between each other, they just exchange their certificates. The certificates contain the public keys of the communicating participants and the participants trust these keys because the keys have been signed with private key of the trusted certification authority.

In addition, the certificates can also contain timestamps. The certification authority sets a duration for the validity of the certificate. After the certificate has expired, the participants are not supposed to trust the information provided by the certificate. In some circumstances, users should stop relying on the certificate even prior to the end of its validity. Compromise of the private key of the certificate holder is an example of this kind of scenario [35]. The most common method for *certificate revocation* is the use of certificate revocation list (CRL) [35]. CRL is a time-stamped list of revoked certificates that is signed by the CA. The CRL is published in a server that all the participants are expected to have access to.

Instead of, or in addition to, the periodic publishing of CRLs, the validity of certificates can also be checked using an online protocol. The Online Certificate Status Protocol (OCSP) [48] enables the participants to determine the status of a certificate by sending a status request to an online server. If the certificate is valid, the server provides a response.

Announcement of public keys

Another scheme for key management is the announcement of public keys [60]. In the PGP model [27], users publish their public keys for example by sending their public keys to all the network users they wish to communicate with, or by placing their public key in some public place, such as the user's web page. The user can also obtain the public keys of other users from similar sources.

To make sure that the obtained public key is authentic the users engage in some out-of-band communication, such as telephone call, face-to-face conversation or exchange of business cards, with the party they believe to be the owner of the key in order to obtain the *fingerprint*, or digest, of the key [29]. Using this fingerprint they can judge the authenticity of the key obtained from an insecure channel. The fingerprint is trusted because it was obtained from a secure channel.

In the PGP model, the public keys can also be used to create so called web of trust [66]. The main idea in the web of trust model is to trust only those public keys that you have au-

thenticated in person (for example by verifying that the public key matches the fingerprint received from a secure channel) or alternatively you can trust public keys that have been signed by the key you have authenticated in person.

For example, let us assume that you have obtained the public key of a person called Bob. You have not received the Bob's key in person, and therefore you cannot be absolutely sure that the key actually belongs to Bob. However, you notice that Bob's public key has been signed with the private key of Alice and you have previously authenticated Alice's public key in person. Knowing that Alice is a trustworthy person and would not sign a key unless she was absolutely certain of its owner you can trust that the key really belongs to Bob.

The PGP web of trust model can be used to create chains of trust with unlimited length. Compared to the traditional model of certificates and CAs the benefits of this model is that it does not require any centralized servers or authorities. The task of signing the public keys is shifted from the CAs to all the participants of the web of trust model.

2.2.2 Securing the initiated connection

After the communicating parties have established security associations, a secure connection can be created using some security protocol. The two most common protocols used in the traditional networks are Internet Protocol security architecture (IPsec) [39] and Transport Layer Security (TLS) [24].

IPsec

The purpose of IPsec is not to be a single security method, but a general security framework. IPsec defines two protocols: the Authentication Header (AH) [37] and the Encapsulating Security Payload (ESP) [38]. The purpose of Authentication Header is to provide authentication and integrity of received IP packets. This is done by protecting the IP header with a message authentication code (MAC). The Encapsulating Security Payload also provides confidentiality. This is done by encrypting the contents of the IP packets. In theory IPsec can be used with AH only, ESP only or both of the protocols at the same time. However in practice, IPsec is typically used with ESP only since ESP provides also authentication and integrity protection.

AH and ESP can be used with different kinds of security parameters. The security parameters of a connection define the used algorithms, the keys etc. The set of security parameters used in one connection is called Security Association (SA). The Security Associations are unidirectional, so there can be two separate Security Associations for a two-way connection. The SA used in one connection is identified with an identifier called Security Parameter Index (SPI).

Security Associations must be negotiated before the peers can start communicating securely. IPsec allows both manually and automatically configured Security Associations. The manual SAs can be inserted into the IPsec implementation using for example PFKEY API [47]. The protocol most commonly used for automatic SA negotiation is the Internet Key Exchange protocol (IKE) [32].

In order to provide its functionality, an IPsec implementation must have access to a Security Association Database (SADB) and to a Security Policy Database (SPD). The SADB provides the IPsec engine with the necessary Security Association parameters, such as key,

lifetime, sequence number etc. The SPD provides information about which traffic should receive protection and how the protection should be enforced. The SPD is organized as a set of rules having a condition part and an action part indicating whether to apply IPsec processing, drop or permit.

TLS

Transport Layer Security protocol (TLS) [24] provides confidentiality, integrity and authentication. TLS is the IETF version of Secure Sockets Layer (SSL) [33] protocol. TLS protocol is server-client oriented and typically used between web browser and server, but the protocol can also be used in peer-to-peer connections. TLS consists on four different protocols: TLS handshake, TLS change cipher spec, TLS alert and TLS record protocols.

The most critical part of TLS is the handshake, where the client and the server authenticate each other, and exchange keys. After that, protecting the connection is easy. In TLS peers typically authenticate each other by verifying that certificate of the other peer is valid and the other peer knows the corresponding private key. Verification means that peers check that the name of the identity certificate matches the name of the intended peer, the certificate is not expired, and it is signed by a trusted CA.

2.3 Ad hoc networks

In general, *ad hoc networks* are temporary networks build on the spot for some particular purpose [49]. The connections in ad hoc networks are typically wireless and the ad hoc networks are usually very dynamic in nature. New network nodes may join the network and existing network nodes may leave the network all the time.

There are different types of ad hoc networks. Some ad hoc networks are connected to fixed traditional networks such as the Internet trough one or more network nodes. In some cases the whole purpose of the existence of the ad hoc network might be to provide access to fixed networks for all the ad hoc network nodes. Or alternatively, the ad hoc network might be totally isolated from all traditional networks.

Earlier we discussed the two support infrastructures that characterize a network: the online servers and the administrative support. Ad hoc networks typically have neither of these. As the ad hoc networks are often isolated from all traditional networks there are no connections to any online servers. Also the administrative support is typically missing. In many cases the lack of administrative support is the most crucial difference between traditional networks and ad hoc networks. Traditional networks are set up, maintained and monitored by professional administrators. However, ad hoc networks are typically set up by the users and the networks are not administered by anyone.

This thesis concentrates on ad hoc networks that are not connected to any fixed networks or centralized resources and lack the administrative support. In the rest of this thesis the term *ad hoc network* will be used to denote to that kind of isolated and autonomous networks that are set up by the users for temporary usage.

2.3.1 Ad hoc usage model

The usage model in traditional networks is typically based on domain names and network addresses, and the network usage is client-server oriented. Most of the popular applications in traditional networks (such as web browsing and email) are based on the fact that there exists a set of server devices and the clients that wish to use the services contact the servers using the domain names or network addresses of the servers.

The usage model of ad hoc networks is more peer-to-peer oriented, since there are no online servers. In ad hoc environments the identification of the devices is often based on the fact that the other device is "here", or that we want to communicate with "that" device, and the actual name or the address of the device is not relevant for identifying the device. Balfanz et al. call this *demonstrative identification* [16], and in the Resurrecting Duckling model of Stajano and Andersson, this kind of authentication is called *anonymous authentication* [59].

2.3.2 Ubiquitous computing

Ubiquitous computing refers to a scenario in which computing is always present, and particularly in which devices that do not look like computers have computing capabilities [58]. In addition to the computing capability the ubiquitous devices are often capable of network communication.

The networks formed by ubiquitous devices can be often categorized as ad hoc networks, since the networks are built on the spot for some particular purpose and the networks do not have any support infrastructure. However, it should be noted that not all networks formed by ubiquitous devices match our definition of the term ad hoc network. For example a refrigerator that is connected to the Internet and makes automatic food orders to the local supplier is an ubiquitous device, but the network is not ad hoc.

2.3.3 Ad hoc network technologies

The two most popular technologies for creating wireless ad hoc networks are Bluetooth and WLAN. However, it should be noted that also wired connections could be used to create ad hoc networks. The most common technologies for creating ad hoc networks are shortly described in the following sections.

Bluetooth

Bluetooth [5] is a short-range wireless radio technology, which adopts a master-slave architecture to form restricted types of ad hoc networks called piconets. A Bluetooth piconet can consist of eight devices, of which one is the master and the rest of the devices are slaves. Each device may take part in three piconets at most, but a device may be master in one piconet at most. Several piconets may also form so called scatternets. [23]

Bluetooth piconets and scatternets are set up by users and the networks are typically not connected to any fixed resources and the networks lack support infrastructure. Therefore, Bluetooth networks match our definition of ad hoc networks.

Wireless LAN

Wireless Local Area Network (WLAN) is a term used to refer to local networks with wireless radio connections. The IEEE 802.11 standard family [7] specifies many different WLAN protocols of which 802.11b is currently the most used. The WLAN standards specify two modes of operation: *infrastructure* approach and *ad hoc* networking approach. [20]

In the infrastructure approach all the devices of the WLAN are connected to a central access point. This access point is typically connected to fixed networks thus providing infrastructure support for all the devices of the WLAN. In the ad hoc networking approach the devices of the WLAN are not connected any central access point, but only to each other. This kind of WLANs are typically set up by users and isolated from all the support infrastructure and correspond to our definition of ad hoc network.

Wired ad hoc networks

An ad hoc network does not necessarily have to be a wireless network. For example if a group of friends meet in a cafe and connect their laptops to a portable hub with ordinary network cables, this network is an ad hoc network even though it is not wireless. Correspondingly a digital camera and a printer form an ad hoc network when they are connected with an USB cable.

2.4 Security in ad hoc networks

In this section we will concentrate on the security issues in ad hoc networks. At first we will discuss the applicability of traditional key management methods for ad hoc networks, and after that we will look at ad hoc network specific methods.

2.4.1 Traditional methods

As noted in the previous section, ad hoc networks are in many ways different from traditional networks. From the point of view of this thesis, the main difference is the lack of support infrastructure in ad hoc environments and the different usage models between traditional and ad hoc networks. The traditional security methods were designed for traditional network environments and these methods are not always applicable in ad hoc environments.

Key distribution center

The use of central key distribution centers is not possible in ad hoc networks since we cannot assume that they have a connection to the central key distribution server.

Public key certificates

In traditional network environments, certificates are most often used for client-server communication. The server has acquired a certificate that has been issued by a well known CA and all the clients trust the certificates of that CA. The use of certificates is practical in traditional networks since most applications used in traditional networks are client-server oriented.

In ad hoc environments the majority of the communication is peer-to-peer oriented, and therefore the use of certificates is not practical as a general method for setting up security. If we would like all the devices of an ad hoc network to be capable of setting up secure connections with each other, than all the devices should have acquired certificates from a mutually trusted CA. This assumptions is not very realistic.

A more realistic assumption would be that only some devices, for example the devices owned by the same company, would have certificates issued by a mutually trusted CA. These devices would be capable of setting up secure connections between each other, but secure connections to other devices would not be possible.

There is also another aspect that makes the use of certificates problematic in ad hoc environments. The certificates used in traditional networks are typically identity certificates. This is often practical, because the identity of the another party is usually known. For example, if a web user connects some website and receives a certificate of that website, he can verify that the public key really belongs to that website, because the web user knows the name of the website he is connecting to.

However, in ad hoc environments we often use demonstrative identification, and we do not know the name or address of the other party beforehand. For this reason, we cannot use an identity certificate to verify that the public key actually belongs to that party. One way to solve this problem would be to attach physical labels that tell the name of the device to all the devices, but then we would have to assume that no one tampered those labels [16].

Yet another reason why the use of certificates is not praticical in ad hoc environments is the lack of online servers. If we want the CA to be able to revoke certificates we need to have access to an online server that maintains a list of revoked certificates. This is not possible in ad hoc environments. [58]

Announcement of public keys

Also the announcement of public keys has some problems in ad hoc environments. This method is based on the fact that you know the parties that you wish to communicate with beforehand and verify their public keys using some out-of-band communication. This is not the case in ad hoc environments, where you often wish to communicate with unknown devices that just happen to be near you.

In general, most existing methods for creating secure connections require that there exists some *prior security context*, such as certificates acquired from a commonly trusted CA or out-of-band verified fingerprints of public keys. In ad hoc environments you are often talking to complete strangers and we cannot assume that there is any prior security context. What is really needed, is a method for *bootstrapping* secure connections between peers with no established security associations.

2.4.2 Bluetooth security

Bluetooth is a communication protocol with built-in security features. The basic idea in the Bluetooth security concept is that the trust between the devices is created at a *pairing* procedure [50]. A pairing is performed between two Bluetooth units. The purpose of the pairing procedure is to generate a common shared secret between the two units. From the perspective of the user Bluetooth pairing goes roughly as follows.

First, the user selects from his device that he wants to perform pairing. His Bluetooth device searches all the devices in proximity and shows the user a list of all possible devices. At this point, the user can select the correct device from the list and the user is asked to enter a personal identification number (PIN) into both of the devices. The purpose is that the user would select a long and random PIN and enter it into both of the devices. After that, both of the paired Bluetooth devices generate the shared secret using the entered PIN.

Some devices (such as Bluetooth headsets) do not contain keypads or any other input mechanisms, and therefore these devices must use fixed PIN codes. If one of the paired devices uses a fixed PIN, this PIN must be entered into the other device. Pairing is not possible between two fixed PIN devices.

The Bluetooth pairing process requires that the user types in a PIN code on one or preferably both of the devices. The strength of the shared secret is based on the strength of the selected PIN code, and thus the whole security of the Bluetooth connection depends on the selected PIN code.

A proper PIN code should be approximately 64 bits long random bit string [50]. On many Bluetooth devices it is possible to type the PIN code only in terms of numerals. A random PIN code of 64 bits would require 20 digits long random number. Selecting and typing such PIN codes into devices is really difficult for the user, and it is natural that the user tries to avoid this task by selecting either too short or systematic PIN code.

To achieve better Bluetooth security, the process of selecting long and random PIN codes should not be left to the user. A better alternative would be to automate the generation of strong PIN codes, or even better, automate the whole pairing process.

2.4.3 Wireless LAN security

The basic WLAN communication protocols do not include any security features and consequently the WLAN networks are very insecure. To fix this problem, many security extensions, such as WEP [10], have been developed. However, most of these security extensions have proved to be insecure [22] and practically all of these security are based on authentication servers or other fixed resources, and these methods are not applicable in ad hoc WLANs.

IEEE 802.11i [12] is the newest WLAN extension that tries to increase the security of WLAN networks and it is expected to be available in early 2004. The 802.11i specification is not yet available, but according to the current draft [12] this extension provides security also for ad hoc WLANs with so called pre-shared key mode, in which users manually enter password which are converted into shared secrets. This solution is very similar to Bluetooth pairing and it suffers from the same security and usability weaknesses.

2.4.4 Security protocols in ad hoc environment

External security protocols are often used to secure ad hoc communication, if the ad hoc networking technology itself does not provide built-in security, or the security support is not sufficient. The security protocol does not have to be ad hoc network specific, as long as its operation is not based on online servers or other infrastructure support.

For example IPsec can be used in IP based ad hoc networks with manually negotiated SAs, however, domain name based IKE negotiations [32] are not possible, if the ad hoc network does not support name service. Likewise TLS secured connections are possible in ad hoc environments, but name certificates cannot be used to authenticate the connections, if name service is not available.

2.5 Proximity based security initiation

As seen in the previous section, the traditional security methods are not always applicable in ad hoc environments. Also some ad hoc specific security methods have been developed, however, none of these methods have proved to be both secure and usable at the same time.

Recent academic research has introduced the idea that the information about the physical location of the device could be used for initiating security in ad hoc environments. This section presents the Resurrecting Duckling model [59] and the work of Balfanz et al. [16] that both tackle the problem of bootstrapping secure connections in ad hoc environments by taking advantage of the fact that the other device is physically in proximity.

2.5.1 Resurrecting Duckling model

Frank Stajano and Ross Anderson introduced their Resurrecting Duckling model in [59]. This model originates from the fact that ubiquitous computing and ad hoc networking are becoming part of our every day life. If for example a householder owns a universal remote control that lets him control various other devices in her home (such as television, heating, lights and even locks), then he will need to ensure that the new device he buys from the shop will obey only his commands and not the commands of his neighbor. He will want to be assured that a burglar cannot take over his household appliances with just by sending command signals with a similar remote control bought from the same shop.

As well as being *secure*, the association between the remote control and the rest of the household appliances must be *transient*. When a householder resells or gives away his television, the appliance will have to obey another controller. Or alternatively, when the remote control breaks down, he must be able to rebind all the household appliances to a new remote control.

In the Resurrecting Duckling model these *secure transient associations* can be established by a process called *imprinting*. When a new household appliance is bought, the device cannot yet be controlled by anyone. Now the remote control can send a secret key to the new device and thus imprint the new device to its command. After the imprinting, the new device can be controlled only by the remote control, that is the party who has the secret key. When the imprinted device is no longer needed the secure transient association can be ended. The device returns to the initial state and it is willing to accept a new secret key.

In the imprinting phase, the secret key must be sent to the new device securely. Traditional security methods are not applicable for previously discussed reasons. Thus, Stajano and Anderson suggest that the imprinting is handled using a physical connection. The secret key can be transmitted as plaintext without any ambiguity about two parties involved in the binding, since the two parties are physically connected. In fact, the Resurrecting Duckling model uses *anonymous authentication*. The name or the address of the imprinted device is not relevant. The master device just wants to imprint the device that he physically identifies.

The original Resurrecting Duckling model deals with master-slave relationships. However, master-slave relationships do not cover all possible usage scenarios in ad hoc environments. Sometimes it might be useful that for example the television and VCR could communicate together, but there is no sense in television being the master of the VCR or vice versa. Sometimes what is really needed is peer-to-peer interaction.

Stajano extended the Resurrecting Duckling model to include also peer-to-peer communication in [57, 58]. The main idea of this extension is that the master could upload *security policies* to the imprinted devices. These policies could for example state that this television can also be controlled by this particular VCR.

2.5.2 Public keys in pre-authentication

The Resurrecting Duckling model has some limitations. First, it is mainly applicable in master-slave situations. The peer-to-peer extension of this model really does not solve the problem of creating secure connection between two parties that have not previously met. This leaves out the fairly common situation in ad hoc environments where two strangers want to communicate.

Secondly, in the Resurrecting Duckling model the imprinting is done by sending a secret key to the imprinted device. The use of secret keys sets serious limitations on the imprinting channel. An adversary must not be able to eavesdrop the the communication on the imprinting channel. And finally, the terminology used by Stajano and Anderson is a bit confusing.

Balfanz et al. note these limitations and they try to correct them in their work [16]. As the Resurrecting Duckling model focuses on situation where a master device imprints the slave, Balfanz et al. generalize this idea for wider range of situations. Instead of talking about imprinting, Balfanz et al. use the more clear term *pre-authentication*. In addition, Balzanz et al. propose the use of public keys in pre-authentication which makes it possible to use other pre-authentication channels than physical contact.

In the model proposed by Balfanz et al. the pre-authentication is carried out between two peers by exchanging pre-authentication material using a so called *location-limited channel*. This location-limited channel is a separate channel from the main communication link with certain security properties.

First, the location-limited channel must support *demonstrative identification*, that is the identification is based on physical context (for example “that” printer in front of me). Channels with directionality, such as infrared, or alternatively channels with very short communication range, such as radio-frequency identification systems, fulfill this criteria. Secondly, the location-limited channel must support *authenticity*. It must be impossible for an adversary to transmit in this channel, or at least transmit without being detected by

legitimate participants.

The third property that can be set on this channel is *secrecy*. In the Resurrecting Duckling model the imprinting was done by sending a secret key. If the pre-authentication material contains secret key, then the location-limited channel must support secrecy. The use of public keys removes this requirement and forces the adversary to mount an active attack.

Balfanz et al. propose several different pre-authentication protocols for both two-party communication and group communication. The main idea in all of these protocols is the same. In the pre-authentication phase, the communicating parties exchange initial security information using the location-limited channel. After the pre-authentication, the parties start some existing security protocol negotiation such as TLS [24] or IKE [32] and authenticate this negotiation using the security information exchanged over the location limited channel.

In the most basic protocol proposed by Balfanz et al. parties exchange commitment to their public keys over the location-limited channel. The information that is actually exchanged can be the public keys themselves, their certificates or hashes of the public keys. In addition, the pre-authentication exchange may contain some address information about the devices.

2.5.3 Pre-authentication compared to traditional methods

Pre-authentication using an out-of-band channel is not a new invention. In the public key announcement scheme the public keys are authenticated by verifying that the public key matches the fingerprint that was received from a secure channel such as telephone.

This pre-authentication method is very similar to the one proposed by Balfanz et al. The biggest advantage in the model of Balfanz et al. is the fact that we can use channels that do not require any user interactions. Verifying a fingerprint in telephone is not user friendly and it is tempting for the user to skip the verification task. Since the location-limited channel based pre-authentication can be automated and integrated into the connection initiation process, this solution is more user friendly.

The location-limited channel based pre-authentication can also be compared to some other traditional security protocols. SSH [65] is a protocol for secure remote login and other secure network services over an insecure network. In this protocol SSH clients connect SSH servers and establish a secure connection.

The server authentication in SSH is typically based on so called *leap of faith* [13] (although other authentication methods, such as certificates, can also be used). SSH connection establishment starts when the client connects to the server. The server provides its public key to the client and the client implementation shows a fingerprint of this public key to the user. If the user knows the public key of the server, he can verify that the fingerprint really matches that key. In most cases the public key of the server is not known to the user, and he just blindly trusts that the server really is the machine who it claims to be. After the first SSH connection the server public key is stored in the client side and the following connections can be authenticated automatically and securely provided that the first connection establishment happened with the intended server.

In the model of Balfanz et al. the key material received from the location-limited channel is always trusted to come from the device that we believe to be communicating with. The use

of wireless location-limited channels, such as infrared, makes spoofing attacks difficult, but not impossible. There exists always some possibility that the received location-limited message came from some other party than that we believed to be communicating with. Thus, trusting the location-limited channel communication can be compared to the leap of faith in SSH protocol, although the difficulty in circumventing location-limited channel communication is much higher.

2.6 Group communication

So far we have concentrated on scenarios, where secure connections are initiated between two peers. However, this thesis also addresses the problem of creating secure groups. This section explains the basic terminology and concepts related to the creation of secure groups.

A *secure group* is a collection of peers, or *members*, who may be senders, receivers or both receivers and senders to other members of the group [17]. The groups are most often dynamic and the membership may vary over time. A *group key management protocol* helps to ensure that only members of a secure group gain access to group data (by gaining access to group keys) and can authenticate group data. The goal of the group key management protocol is to provide legitimate group members with the up-to-date cryptographic state they need for their secrecy and authenticity requirements.

The group key management protocols can be divided into two separate categories [15]. Centrally managed *key distribution* schemes use a central group controller or key server that is responsible for distributing the needed keys to all the group members. In *key agreement* schemes there is no central group controller or key server, but all the group members participate in the generation of group key material in a distributed manner.

2.6.1 Key distribution

Centrally managed groups are easier to control, especially when the size of the group is relatively large. Thus, most of the group key management protocols used on the Internet (for example for secure Multicasting [31]) are based on a central group controller or key server. [17]

In centrally managed schemes joining group members and the group controller mutually authenticate each other. If the joining member is authorized, then the group controller confidentially provides him with the needed key material to securely communicate with the other group members. In some group management schemes the group controller also provides the joining member additional information that can be used to refresh group keys. Refreshing of group keys may result from for example group membership changes or key expiration.

2.6.2 Key agreement

Although centralized key management might initially appear simple and effective, it is very unsuitable for groups that exist in the dynamic ad hoc environments [40]. First, centralization concentrates all the key generation in a single point, hence centralizing all the trust

to one peer of the network. Secondly, a centralized group controller becomes both a single point of failure and an attractive attack target. The availability of the group controller cannot be guaranteed and, therefore, each group member must be prepared to become the group controller. The selection of the new group controller raises a policy issue and, furthermore, each new group controller must establish a pairwise secure connection with every group member in order to distribute the new keys.

To overcome these limitations of centralized key distribution schemes several distributed key agreement schemes have been developed (for example [40, 36]). In key agreement schemes all the group members participate in group key generation. Most of the key agreement schemes are based on the same basic ideology. Each group member contributes its own share to the group key. This share is usually private to each group member and when sent to other members this share is often manipulated in some way. One way to manipulate the share is to use modular exponentiation in prime order groups which is analogous to the Diffie-Hellman protocol [25].

All the shares are organized in some data structure (such as *key tree* [40]) and all the group members are capable of computing the group key using the key data structure that contains the private key share of the member and manipulated key shares of other group members. As new members join the group, new shares are factored into the key data structure but old members' shares remain unchanged.

Key agreement schemes seem like a promising solution for creating secure groups in ad hoc environments. However, most of these key agreement protocols employ some sort of modified Diffie-Hellman key exchange among group members. Just like in two-party Diffie-Hellman, we know that we establish a shared secret with *someone*, but we do not know necessarily who that someone is. These protocols assume that all the members of the group participate in some public key infrastructure, or have previously exchanged public keys in a trusted manner. [16]

2.6.3 Group creation using location-limited channel

Location-limited channels could be used to pre-authenticate and initiate group key management protocols. A centralized group key management protocol could be easily authenticated using normal two-party location-limited channel pre-authentication protocols. One participant of the group is designated to become the group controller. After that, the group controller authenticates all the group members using a normal two-party location-limited pre-authentication protocol.

The above described method is simple, but not very user friendly. The group controller has to pre-authenticate point-to-point links with all the group members separately which might be time consuming since location-limited channel pre-authentication usually involves some effort from the user (such as moving the device close to the other peer). To overcome this defect Balfanz et al. [16] present the idea that audio could be used as location-limited channel for pre-authenticating group key management protocols, since audio channel has a natural broadcast ability.

When using an audio pre-authentication channel, a centralized group key management protocol could be pre-authenticated as follows. One member is designated to become the group controller. The group controller broadcasts key material (such as hash of his public key) to all the group members over the audio location-limited channel. The other group

members respond by sending their key material to the group controller using the same channel.

The audio channel could also be used to easily pre-authenticate distributed key agreement protocols. Each member of the group broadcast key material to all the other members of the group using the audio location-limited channel. After all the members have broadcasted their key material, the normal distributed group key management protocol can start.

2.7 Usability and security

The purpose of this thesis is also to address the usability aspects of security initiation process. In this section we will look at the basic concepts that are related to usability and security.

There seems to be relatively little computer security research that have seriously emphasized user interaction issues. Perhaps, because of this, usability is often ignored or misunderstood when designing security software. [64]

Security problems are often caused by software errors such as buffer overruns or race conditions, or by bad security protocol design. This has focused a great deal of attention on assuring the correctness of software design and implementations. According to Ka-Ping Yee [64], however, the correct use of software is just as important as the correctness of the software itself. For example, there is nothing incorrect about a program that deletes files. But when such a program happens to delete files against our wishes, we consider it a security violation. In a different situation, the inability to command the program to delete files could also be a serious security problem.

Many software designers assume that improving security necessarily degrades usability, and vice versa. The decision of whether to favor one or the other is typically seen as a compromise. However, security and usability do not have to be at odds with each other. In fact, as Ka-Ping Yee states in [64], the opposite makes more sense. A system that is more secure is more controllable, more reliable, and hence more usable. A more usable system reduces confusion and is thus more likely to be secure.

2.7.1 Design principles

As already stated, the area of usability and security is not thoroughly researched and there have not been any guidelines or established conventions that could have been followed when making usability and security design decisions. In his recent paper Ka-Ping Yee [64] presents a concise list of ten design principles for designing secure systems. The most relevant principles for this thesis are presented in a list below:

- *Path of Least Resistance.* The most natural way to do any task should also be the most secure way.
- *Revocability.* The interface should allow the user to easily revoke authorities that the user has granted, wherever revocation is possible.
- *Expected Ability.* The interface must not give the user the impression that it is possible to do something that cannot actually be done.

- *Identifiability.* The interface should enforce that distinct objects and distinct actions have unspoofably identifiable and distinguishable representations.
- *Clarity.* The effect of any security-relevant action must be clearly apparent to the user before the action is taken.

These design principles will be taken into consideration when defining the usability requirements for the location-limited channel based security initiation system and when designing the user interactions of the system.

2.7.2 Normal software vs. security software

When talking about usability and security, we should always keep in mind the distinction between *secure normal software* and *usable security software*.

In this thesis the term *normal software* refers to software that primarily provides some other than security services to the user. For example a web browser could be considered as normal software. Even though security is important, the usability of the normal software is usually the primary design principle when designing normal software. The term *security software* refers to software that primarily provides security services to the user. For example firewall software could be considered as security software. The usability of security software is of course important, but the main purpose of the software naturally is to provide security.

Even though usability and security affect one another and support each other, the designer of the software should always understand the nature of the application at hand. Designing usable security software is often a different process from designing secure normal software.

2.8 Related work

This thesis addresses the problem of bootstrapping trust and initiating security in ad hoc environments. The Resurrecting Duckling model and the work of Balfanz et al. are the most relevant work done so far in this field and form the basis for this thesis. In addition to these models there are also several other approaches for bootstrapping trust and initiating security in ad hoc environments. This section gives a brief overview of other related work.

2.8.1 Near Field Communication

Near field communication (NFC) refers to very short-range radio frequency communication protocol developed by Ecma International [1]. The NFC protocol uses unlicensed radio frequencies and the communication is restricted to distances of 0 - 20 centimeters. [11]

A recently published white paper [11] presented the use of NFC protocol for local communication and initiation of other communication protocols. In this paper the NFC protocol was used to transfer files between devices. In addition, the paper presented the idea that Bluetooth connections could be set up by transferring the PIN code from one device to another using the NFC protocol.

2.8.2 Spontaneous networks

Laura Marie Feeney et al. have also studied the intuitive set up of secured infrastructureless and wireless networks which they call spontaneous networks [26]. In their solution one device is selected to become the session initiator for example by asking “Alice, can you please get the network started?” This device generates a session key and sends it in plaintext format to all other devices using some short range wireless channel. Their implementation uses infrared links separately established for each device. The received session key is used to secure the wireless network traffic.

2.8.3 Manual authentication

Manual authentication refers to authentication methods that require some user interaction [28]. Any unauthenticated key agreement protocol, such as Diffie-Hellman key exchange, can be verified by calculating hash values from the results of the key agreement and comparing these values. If the values are the same, the verification is successful. The hash values must be collision resistant, and therefore the length of 160 bits is typically recommended. Comparing two 160-bit (or 40 hexadecimal digit) hash values is too cumbersome and the user is tempted to skip this task completely.

Christian Gehrmann, Chris Mitchell and Kaisa Nyberg describe a more user friendly approach to manual authentication in [28]. Instead of computing long hash values, they propose the use of short message authentication codes which are keyed using short randomly generated keys. The security is based on the fact that for each verification a new key is generated randomly. The use of their protocols reduces the length of the value that must be compared to approximately 5 hexadecimal digits. (The exact number of digits depends on the probability of forgery.) This is a major improvement and makes the verification procedure considerably more user friendly.

2.8.4 Password authenticated key exchange

Password authentication is another approach for initiating security. Suppose that Alice and Bob want to establish a secure connection and they share a secret password. Alice generates a random key, encrypts this with the password, and sends it to Bob. Bob decrypts the key with the shared password and they can start communicating using the random key. This method is appealing, but vulnerable to brute-force dictionary attacks, especially when the used password is short. [18]

Steven Bellovin and Michael Merrit present a more secure password authentication protocol in [18]. In the most basic form of their method the shared password is used to encrypt a fresh public key. The encrypted public key is then sent to the other party who can decrypt it. After that, the other party generates a random key and sends it encrypted with the public key and the password. This protocol is considerably more resistant against brute-force dictionary attacks.

After the original work of Bellovin and Merrit, their password based authentication scheme has been developed into various different protocols, such as SRP [63], many of which are stronger than the original or add new desirable properties.

2.8.5 Ultrasound security initiation

An alternative solution for initiating secure connections in ad hoc environments is the use of ultrasound for authenticating devices. In [42] Tim Kindberg and Kan Zhang present an approach where the devices are equipped with ultrasound transmitters and two or more receivers.

A device that wants to become contacted acts as a beacon by sending a certain ultrasound signal periodically. When another device wants to communicate with a beacon, it starts a message exchange with the beacon and determines the direction of the beacon using the time difference of the received ultrasound signals. After this, the direction of the other device can be shown to the user. The user must verify that the device in this direction really is the one he wants to communicate with. If this is the case, the device has been successfully authenticated and a secure connection can be established with the key material that was exchanged with the beacon device.

2.8.6 Context authentication using constrained channels

Another paper [41] by Tim Kindberg and Kan Zhang presents a model for *context authentication*. This term refers to authentication of principal's status in a certain context, usually the physical location of the device.

Conventional authentication protocols establish the identity of principal based upon the premise that only that principal possesses some secret. However, in some circumstances in addition to the identity of the principal we are interested in the characteristics of the principal's context, such as their location. The principal's context can be characterized by a set of *contextual predicates*. A contextual predicate might be for example: 'principal must be in location X', or 'temperature in the principal's context must be over T'.

Kindberg and Zhang also introduce the term *constrained channel*. A constrained channel can be either *send-constrained* or *receive-constrained*. When a message is received from a send-constrained channel, we know that the sender principal must satisfy the context predicate set for this channel. And when a message is sent to receive-constrained channel, we know that the principal who receives the message must satisfy the context predicate set for this channel.

These definitions capture some properties established by traditional security protocols. For example, consider two participants connected by a TLS connection, who send and receive messages that are encrypted and decrypted by the connection. This channel is both send-constrained and receive-constrained on the context predicate 'possesses secret key K' for some K negotiated by the TLS connection.

2.8.7 Entity recognition

Jean-Marc Seigneur et al. have also studied issues related to authentication and ad hoc environments in [56]. Seigneur et al. state that the traditional authentication process is based on the use of names and it requires an *enrollment* phase. The enrollment typically involves an administrator or human intervention, such as verifying ones identity or issuing a certificate, and therefore this process is not suitable for ad hoc environments.

Instead, Seigneur et al. present an alternative process that is based on *entity recognition*.

The entity is recognized through its name, location, digital signature or any other means. The outcome of the entity recognition process should be data that can be used to recognize the same entity in future with sufficient level of confidence. Entity names do not have to be used and the entity can remain anonymous.

2.8.8 Round-trip times

Yet another approach is to use round-trip times in authentication, as all communication technologies are bounded by the speed of light. This approach has been introduced by Stefan Brands and David Chaum in [19].

Recently the same approach has been applied to ubiquitous computing by Laurent Bussard and Yves Roudier [21]. They presents a design, in which one-bit long challenges and responses are exchanged using simple dedicated hardware. If the round trip time is less than one nanosecond, than the other device cannot be farther than 15 centimeters. The implementation of Bussard and Roudier is based physical contact with an electric ring.

2.8.9 Shaking

A quite different approach is taken by Lars Erik Holmquist et al. in [34]. In their approach, a connection between two portable devices is formed by holding them together and shaking them. The devices have accelerometers and the common measurements allow them to identify each other.

Chapter 3

Requirements

The purpose of this chapter is to define requirements for the location-limited channel based security initiation system. The rest of this chapter is organized as follows. Section 3.1 defines the high level goals for the security initiation system. Section 3.2 describes a few example security initiation scenarios. Section 3.3 identifies the lower level security requirements, Section 3.4 concentrates on the usability requirements and Section 3.5 identifies the lower level generality requirements.

3.1 High level goals

The main goal of this thesis is to design a general and user friendly location-limited channel based security initiation system. The following list presents the high level goals of the system:

- **Goal 1:** The security initiation system should bootstrap security associations in ad hoc environments.
- **Goal 2:** The security initiation system should set up secure connections with the bootstrapped security associations.
- **Goal 3:** The security initiation system should be user friendly.
- **Goal 4:** The security initiation system should be a general framework that is not restricted to any particular environment or scenario.

3.2 Security initiation examples

This section presents a couple of example security initiation scenarios. These examples will be used to identify the lower level requirements for the system.

Example 1: A traveler wants to use the airport waiting lounge printer to wirelessly print a document from his PDA. The document is personal and the traveler does not want to share it with the whole waiting lounge.

The PDA of the traveler and the waiting lounge printer are both equipped with wireless network interfaces that can be used for local communication. In addition, an RFID tag is attached to the printer and the PDA has an RFID reader. There is no prior security context between these two devices, that is the PDA and the printer do not share any keys or possess certificates issued by mutually trusted CAs.

The traveler forms the secure connection between the two devices by touching the printer with his PDA. During the touch, the RFID reader of the PDA reads the contents of the RFID tag. After that, the secure connection is set up using the the information received from the RFID tag of the printer. The traveler can now print his personal document securely.

Example 2: A home user wants to set up a secure connection between his computer and camera phone, so that he can securely upload pictures from his camera phone to his computer. The connection must be secured, so that the wireless picture traffic takes place between just the camera phone and the computer, and no outsider devices can receive the pictures sent by the camera phone.

Both of the devices are equipped with wireless network interfaces, and in addition, the computer and the camera phone both have infrared ports. Like in the previous example, there is no prior security context between these devices.

The home user brings his camera phone close to his computer. The two devices set up an infrared link and exchange information using that link. After that, secure connection is set up between these two devices, and the home user can upload the pictures from his phone to his computer securely.

Example 3: A group of businessmen forms an ad hoc network for conferencing when they meet outside the office. The businessmen want to share company confidential material and thus the communication must be secured.

Each of the businessmen has a laptop with a wireless network card and an infrared port. Also one of the businessmen has a mobile phone that has an infrared interface. No prior security context exists between the laptops of the businessmen.

The businessman with the mobile phone moves his phone close to all the laptops one at a time. The mobile phone and the laptops establish infrared links and exchange information using that link. After that, the secure group connection is set up between the users.

3.3 Security requirements

The first two high level goals defined that the security initiation system should bootstrap security and set up secure connections in ad hoc environments. In this section, we will identify the lower level security requirements for the system.

3.3.1 Security objectives

In the first example, the most important security aspect is the fact that the traveler knows that he is establishing a secure connection with the correct printer and with nobody else. From the point of view of the printer, it is not relevant who the other party of the connection is. In the second example, mutual authentication is required. The home user wants the phone to establish the connection with his computer and that the computer establishes the

connection with his phone.

Overall, the following three security objectives are identified for situations in which a secure connection is established between two devices. The device that starts the connection initiation is called *initiator* and the other device is called *listener*.

- *Listener authentication:* The initiator device knows that a secure connection is established with the correct listener device.
- *Initiator authentication:* The listener device knows that a secure connection is established with the correct initiator device.
- *Mutual authentication:* The initiator device knows that a secure connection is established with the correct listener device and vice versa.

In the third example, the businessman with the mobile phone must know that the secure group connection is established with the correct other devices. Overall, the two following objectives are identified for group security initiation scenarios.

- *Distribution authentication:* Only one of the group members, the group controller, knows that the secure group connection is established between the correct other devices.
- *Agreement authentication:* All the group members know, that the secure group connection is established between the correct other devices.

3.3.2 Low level security requirements

The previous section presented the different security objectives for establishing secure connections. In this section we will identify the lower level security requirements.

Requirement 1: A set of pre-authentication protocols is needed for different location-limited channels.

There are many different kinds of location-limited channels. Some location-limited channels are one-way, like the reading of the RFID tag of the airport printer, while others are two-way, like the infrared link between the phone and computer of the home user. Some location-limited channels have high bandwidth, while others are capable of sending only very little information.

What is needed is a set of pre-authentication protocols that can be used for initiating security with various different location-limited channels.

Requirement 2: The location-limited channel protocols should initiate security associations that fulfill the requested security objective if that is possible.

Different scenarios have different security objectives. In some cases mutual authentication is required, while in other cases unilateral authentication is sufficient. The pre-authentication protocols should initiate the cryptographic keys that fulfill the security objective set for this scenario if that is possible with the location-limited channel in use.

The following list presents the lower level security requirements for the location-limited channel protocols depending on the security objective.

- Listener authentication: The initiator device must be able to recognize the public key of the listener device.
- Initiator authentication: The listener device must be able to recognize the public key of the initiator device.
- Mutual authentication: The two devices, initiator and listener, must be able to recognize the public keys of each other.

The location-limited channel protocols use public key cryptography for setting up the secure connection whenever possible. However, not all devices support public key cryptography. If public key cryptography is not supported, shared secrets must be used, and the lower level security requirements are as follows.

- Listener authentication: The initiator device must know that it has established a shared secret with the correct listener device.
- Initiator authentication: The listener device must know that it has established a shared secret with the correct initiator device.
- Mutual authentication: The two devices, initiator and listener, must know that they have established a shared secret between each other.

The following list presents the lower level security requirements for the group security objectives.

- Distribution authentication: The group controller must be able to recognize the public keys of all the other members and all the other members must be able to recognize the public key of the group controller.
- Agreement authentication: All the group members must be able to recognize the public keys of each others.

If the devices do not support public key cryptography, typically only key distribution is used.

- Distribution authentication: The group controller must know that it has established a shared secret with all the other group members and all the other group members must know that they have established a shared secret with the group controller.

Requirement 3: The pre-authentication protocols should exchange all the other information that is needed for establishing complete security associations.

The fact that the peers have exchanged keys is not sufficient for establishing complete security associations and setting up secure connections. The peers must also know the identities and network addresses of each other, agree on the used security protocol and other connection parameters. The pre-authentication protocols should take care of establishing all this information.

As the high level goals stated, the traffic in the main communication channel should be protected with the bootstrapped security association. Because different applications require

different security mechanisms, the selected protection approach depends on the application that uses the communication channel. Three different application types are identified in the following list:

- *Unsecured applications:* Some applications communicate without any security features. The messages are sent in plaintext format and the received messages are not authenticated. For example FTP is such an application. In this thesis, these applications are called unsecured applications.
- *Secured applications:* Other applications explicitly use some security protocol or security services for securing the network traffic. For example web browsers use TLS to make secure connections to web servers. In this thesis, these applications are called secured applications.
- *System-aware applications:* Some applications could be also be designed to use the services of the security initiation system directly. In this thesis, these applications are called system-aware applications.

Requirement 4: To provide security for unsecured applications, the security initiation system should configure a network level security protocol.

To provide security for unsecured applications, the communication must be protected at the network level. For example to secure the use of TCP/IP based FTP application, the system should configure for example an IP level security protocol, such as IPsec.

Requirement 5: To protect the communication of secured applications, the security initiation system should provide the needed security association for the particular security protocol that the application uses.

To protect the communication of secured applications does not require the configuration of any security protocols. However, the security applications cannot use the security protocols, if the required security association is not available. For example an authenticated TLS connection cannot be established, if the browser does not have the required security association to authenticate the server.

Requirement 6: To provide security for system-aware applications, the security initiation system should provide a set of security functions.

The system-aware applications should be able to secure their traffic using the security initiation system alone. Thus the system should provide a set of security functions that can be used to protect the communication without any external security protocols.

Requirement 7: The user of the security initiation system should be able to define the duration of the created secure connection.

Some connections are natural one-time connections. For example the connection for printing the document at the airport waiting lounge is probably needed only once. However, other connections are used many times. For example the secure connection between the phone and the computer of the home user is intended to be long lasting. Therefore, the user should be able to define a duration for the created secure connection.

Requirement 8: The security initiation system should store the initiated security association.

The security initiation system should be able to store the long lasting connections. Therefore, some kind of storage is needed.

Requirement 9: The user of the security initiation system should be able to give a name to each initiated secure connection.

Every time a new connection is initiated, the user should be able to give a name to that connection. When, for example, the home user initiates a secure connection to his home computer with his camera phone, the home user should be able to name this connection with a nickname he chooses. The initiated security association will be saved and this association can be retrieved from the data storage using the name given for this connection.

Requirement 10: The level of security of the initiated connection should not depend on the security awareness of the user.

In this thesis the term *security awareness* denotes to the comprehension of all kind of security related terminology and details. If for example some security software asks from the user whether he wants to accept a certain certificate or not, this software requires security awareness from the user, since the user has to understand what is a certificate in order make the correct decision.

In most of the ad hoc specific security initiation methods, the security of the initiated connection is dependent on the security awareness of the user. For example in Bluetooth pairing, the security of the connection depends on the length and randomness of the selected PIN code. Most of the users do not understand these concepts, and therefore the initiated connections are often insecure. In this system, the security of the initiated connection should not depend on the security awareness of the user.

3.4 Usability requirements

The third high level goal stated that the security initiation process should be intuitive and the security initiation system should be easy to use. The section presents the lower level usability requirements.

Requirement 11: The security initiation process should be integrated to the normal use of applications.

Typically, the user is not interested about security - he just wants to use some service or application. For example the traveler in the airport waiting lounge is not primarily concerned about security; what he really wants to do, is to print that document. Therefore, in this scenario the security initiation should be integrated to the printing service.

In ideal case, using the applications with the security initiation system would be almost as easy as using the applications without any security. The end user would not have to understand anything about security, and the security initiation system would create the secure connections almost automatically. The following scenario describes the ideal usage of the security initiation system.

If the user wants to use some communication application, he starts the application normally. Before the application starts communication over an insecure network, the application connects the security initiation system and queries for existing security association for this kind of connection. If one exists, the security initiation system configures the security protocols and the application can start communicating over secured channel.

If there is no existing security association for this kind of connection, the security initiation system user interface is brought up. The user interface of the security initiation system guides the user through the security initiation process and when the initiation is complete the security initiation system configures the security protocols for the application. After that, the application can start communicating securely.

The above described scenario is based on the assumption that all the applications are *system-aware*. This assumption limits the usage of the system to those applications that have been designed with the security initiation system in mind. The purpose of this thesis is to design a system that allows security initiation for all kind of applications. Therefore, the security initiation system should be a sort of master application that has to started by the user every time he wants to initiate a new connection.

Requirement 12: The number of needed user interactions should be small.

The system should make the security initiation as intuitive as possible by reducing the required user interactions. If the initiation of a secure connection takes more than for example five key actions from the user, the connection initiation is not very user friendly.

Requirement 13: The user interactions should be reversible.

The user should be able to change his mind after a made choice without any remarkable difficulties. In addition to this, the user interface should always provide means for the user to abort or restart the security initiation.

Requirement 14: The initiation process should be understandable.

The normal user usually does not understand what security associations, keys, security objectives or security protocols mean. Therefore, the initiation system should hide all those details of security initiation process and all the interfaces of the system should explain the process in language understandable by the user.

3.5 Generality requirements

The fourth high level goal stated that the security initiation system should be a general framework that is not restricted to any particular environment or scenario. This section presents the lower level generality requirements.

Requirement 15: The software architecture should be based on separate and interchangeable software components that can be added to the system later on.

The security initiation system should be able to initiate security in various ad hoc environment, with help of different kind of location-limited channels, using many different security protocols for different kind of applications. Implementing such a system that provides support for all possible environments and scenarios is impossible.

Therefore, the security initiation system design should be based on interchangeable components that can be implemented separately for different environments and scenarios, and added to the security initiation system when needed.

Chapter 4

Design

This chapter presents the security initiation system design. The rest of this chapter is organized as follows. Section 4.1 presents the design assumptions. Section 4.2 analyses the properties of different location-limited channels. Section 4.3 presents different peer-to-peer security initiation models and Section 4.4 defines the actual peer-to-peer pre-authentication protocols. Section 4.5 presents different group models and Section 4.6 defines the group pre-authentication protocols. Section 4.7 gives an overview of the security initiation system and Section 4.8 describes the security initiation system software architecture.

4.1 Design assumptions

The devices used in the security initiation system are expected to be portable devices with wireless network interfaces. The battery lifetime of the devices is expected to be limited. The displays and keypads of the devices are expected to be sufficient for construction of simple user interfaces with somewhat restricted user input possibilities. The devices are expected to be capable of performing either symmetric or asymmetric cryptographic operations.

The underlying network environment is expected to be an ad hoc environment which is constantly available without any required user interactions such as turning on the network interface when communication is wanted. The ad hoc devices are expected to know their own network addresses.

4.2 Location-limited channels

Before the actual design of the security initiation system, a deeper analysis on the location-limited channels is presented. This section identifies the general properties of location-limited channels and analyses the different location-limited channel technologies.

As Balfanz et al. defined, a *location-limited channel* is a separate channel from the main communication link with certain security properties. The most important of these properties are demonstrative identification, authentication and secrecy. A location-limited channel can provide support for some or all of these security properties.

In this thesis we will provide a more detailed description of the term location-limited channel. We define that the location-limited channel is a separate channel from the main communication link formed between two *location-limited components*. A location-limited component is the actual physical component, such as infrared port, that takes care of sending and receiving the messages in the location-limited channel.

Most location-limited components are *online* components. An online component is physically connected to the device itself and the online component can be accessed at any time. The sent messages can be generated on the device and the online component can pass the received messages to the device. An infrared port is an example of an online location-limited component.

Some location-limited components are *offline* components. An offline component is physically disconnected from the device and the device cannot communicate with the component during the location-limited channel interaction. For example RFID tags are usually offline components. The messages that will be sent must be generated on the location-limited component and the messages that will be received cannot be passed to the device itself.

Typically, most of the location-limited components can both send and receive messages. However, some location-limited components are capable of only sending or receiving messages. For example an audio speaker is an example of *send-only* location-limited component and a microphone is an example of *receive-only* location-limited component. The directionality of the location-limited channel depends on the types of the location-limited channel components at the end points of the channel. The location-limited channel can be either *one-way* or *two-way*, depending on the capabilities of the components.

The messages that will be sent using an online location-limited component can be generated on the device. However, the messages sent using an offline component must be generated on the component itself. The following list identifies the different types of offline location-limited components that can only send messages.

- *Static* component sends the same message every time.
- *Random* component generates and sends a different message every time. Either the whole message can be random or the component can include some randomness into the message.
- *Timestamp* component includes some time information into the generated and sent messages. This requires a real-time clock from the offline component.

If the offline component is capable of both sending and receiving messages, it is a challenge-response component.

- *Challenge-response* component first receives a message and then generates the response according to the received message. Some challenge-response components are capable of performing complex task, such as calculating public key encryption operations, while others are capable of only very simple challenge-response behavior.

As Balfanz et al. defined, location-limited channels support authentication. Some location-limited channels, however, provide only partial support for authentication. Tim Kindberg

and Kan Zhang introduced the term *constrained channel* in [41]. A constrained channel can be either send-constrained, receive-constrained or both. In a similar fashion, we can define that the location-limited channel is:

- *Sender location-limited*, if the receiver of the message can be sure that the message was sent by someone that is in proximity at the moment. The channel supports unilateral authentication of the sender.
- *Receiver location-limited*, if the sender of a message can be sure that if someone has received his message that someone was in proximity when the message was sent. The channel supports unilateral authentication of the receiver.
- *Sender and receiver location-limited*, if both of the above mentioned requirements hold. The channel supports mutual authentication.
- *Not location-limited*, if the sender cannot make any claims about the receiver of his message and vice versa.

The location-limited channel can be also characterized by the way the actual communication signal behaves. The first type is *broadcast* channels. The communication signal spreads in every direction. For example Bluetooth channels are typically broadcast channels. The second type is *directed* channels. The communications signal advances into a certain direction. For example infrared is a directed channel. The third type is *point-to-point* channels. Typically only channels with physical connection are truly point-to-point.

Also the *data transfer capability* of location-limited channels varies. Some channels are capable of transferring megabits per second while the overall transfer capability of other channels might be restricted to hundreds of bytes.

4.2.1 Transport alternatives for location-limited channel

There are many communication channels that could be used to implement a location-limited channel. Some of the possible alternatives are described in this section.

Physical connection

Physical connection is the most obvious location-limited channel type. For example Universal Serial Bus (USB) [6] could be used to implement a physical point-to-point location-limited channel.

The location-limited channel could simply be a cable from the USB port of one device to the USB port of the other device. A more handy and elegant solution would be to use a small USB token. A one-way location-limited channel could be implemented by touching the first device with the token and than the other device. The token would carry the needed data from the first device to the second. A two-way location-limited channel would require an additional third touch.

Implementing the location-limited channel with USB token or cable is a secure solution, since the USB location-limited channel supports secrecy and mutual authentication, that is the channel is both sender and receiver location-limited. However, the usability of the

USB solution is rather poor, since the user is required to connect and disconnect cables or tokens.

Range reduced Bluetooth

A standard Bluetooth channel is not suitable as location-limited channel since the communication range is measured in tens of meters. However, if the communication range of the Bluetooth channel could be restricted, also Bluetooth could be used to implement location-limited channels.

There are two ways to implement range reduced Bluetooth channel [30]. The first possibility is to reduce the output power of the Bluetooth radio. This solution ensures that only devices in proximity can receive the signal. The other possibility is to put additional noise into the transmitted signal. This solution guarantees that only devices in proximity can decode the signal. Implementing a range reduced Bluetooth location-limited channel would require that the system should be able to either control the transmission power of the Bluetooth radio or add additional noise to the signal. However, the current Bluetooth specification [5] does not provide such an API.

Range reduced Bluetooth provides a method for implementing a broadcast location-limited channel. By using range reduced Bluetooth the sender knows that the receiver must be in proximity. However, the receiver cannot be sure that the message was sent in proximity, since a high power signal could be transmitted from long distance. Thus the range reduced Bluetooth channel is only receiver location-limited. Also because the channel is a broadcast channel, it does not support demonstrative identification.

RFID

Radio-Frequency Identification (RFID) systems are composed of two key elements [9]. The RFID tag, or *transponder*, contains the data, and the RFID reader, or *transceiver*, interfaces with tags to read or write the tag data. The RFID tags are used to communicate to readers via radio frequencies. The tag memory may be only readable, or alternatively both readable and writable.

The tags may be either *actively* or *passively* powered. Passive tags inductively receive power through an RF signal from the reader, while active tags can boost reply signals with an on-board power. Active tags may carry an on-board clock and perform calculations in the absence of a reader. Passive tags may only operate in the presence of a reader.

The cheapest RFID tags are only readable and contain static data. The more expensive tags can do some simple computations and generate the message for each situation separately. And the most advanced tags are capable of even performing public key cryptographic challenge-response operations [4].

RFID tags and readers can be used to implement different kind of location-limited channels. Typically, all the tags are offline components. Read-only tags form one-way location-limited channels, and readable and writable tags allow two-way communication. The cheapest tags are static components, while the most expensive ones are challenge-response components.

The eavesdropping range of the RFID channel depends on the frequency used for the com-

munication. The channel from the reader to the tag could be theoretically monitored from hundreds of meters and the channel from the tag to the reader from tens of meters. However, in practice these ranges are much smaller. Typically the eavesdropping ranges are much shorter and any active communication with the tags must occur within a short distance, perhaps two meters. [53]

RFID is a promising candidate as a location-limited channel, since the communication distances are quite limited. Like practically all the wireless channels, also RFID is only receiver location-limited. The usage of RFID channel is very intuitive because touching the tag with the reader is simple. The downside of the RFID location-limited channel is the fact that most RFID channels are only one-way and the amount of data that can be stored on the tags is very restricted.

Infrared link

Infrared links use light radiation for communication. [3] The wavelength of the of the used light is a little longer than visible red light. Infrared links are traditionally used in remote controls and similar devices. Infrared Data Association (IrDA) [3] has specified a series of specifications relevant to a local data link. The continuous operating range between IrDA devices is specified to be at least one meter. Mobile devices, such as phones, typically use a low power version of IrDA, which limits the connectivity ranges to a few meters.

Infrared provides a good location-limited channel alternative. Infrared support is built in to many mobile devices and the infrared link is directed which makes the use of infrared location-limited channel more intuitive. Also the infrared link is only receiver location-limited. The sender of a message can be sure, that the receiver of the message is in proximity. However, the link is not sender location-limited, since a high power infrared signal could be beamed from long distances.

Laser pointer

The transmitted data could be also modulated into a signal that is sent by a laser pointer [51]. Laser pointer channel has similar communication characteristics as in infrared, but it allows considerably larger range. Because of this larger range, the security properties of laser location-limited channel are not as good as the security properties of infra-red. On the other hand, the laser pointer location-limited channel is truly point-to-point, whereas an infrared channel is only directed.

Optical channels

A small amount of information can also be encoded into a *bar code*, which is a pattern of black bars and white spaces. The information can be read from the bar code using a bar code reader, which is a combination of a bar code scanner and its decoder. Bar codes could be used to implement offline one-way location-limited channels. The bar code channel could not provide secrecy since the bar code would have to be readable by anyone. The bar code channel would be only sender location-limited, if reading of the bar codes from long distance could not be prevented.

Also a technique called *Optical Character Recognition* (OCR) could be used to implement

location-limited channels. OCR refers to the translation of optically scanned bitmaps of printed or written text characters into character codes, such as ASCII [45]. For example the current camera phones could be used to implement OCR location-limited channels. The properties of OCR location-limited channel would be very similar to the properties of bar code location-limited channel.

Audio channel

Audio channel is typically not used for data transmissions, because the data rates are relatively low and the sounds tend to be rather annoying [46]. However, there are also a few reasons why sound channel could be a promising bearer for a location-limited channel [16]. First, the audio communication naturally has limited range. Second, mounting an active attack on the audio channel without the awareness of legitimate participants is difficult. And third, the audio channel could be a cost efficient way of implementing a location-limited channel, since the required hardware already exists in many mobile devices, such as mobile phones.

The audio channel naturally does not provide secrecy. The channel itself is only receiver location-limited, since loud sounds could be transmitted from distance. However, this would be typically noticed by the user. In addition to audible sound, also ultrasound could be used as location-limited channel bearer. The characteristics of ultrasound are very similar to audible sound excluding the annoying sound.

Human interaction

A small amount of information could be also transmitted using human interaction. Human interaction is not a suitable channel for transmitting all the necessary data, but it could be used to authenticate or verify an exchange that took place using another channel. Sections 2.8.3 and 2.8.4 explained some developed security methods based on human interactions.

4.3 Models for peer-to-peer initiation

Location-limited channel can be used in many ways when initiating security between two peers. This section describes the different communication model alternatives that can be used in the security initiation system design.

The most straight forward use case is that the two peers, who want to establish a secure connection, communicate directly using the location-limited channel. The location-limited channel is formed between two online components and the channel can be either one-way or two-way. The peers exchange security initiation information using the location-limited channel, and after that, the secure connection can be set up for the main communication channel. This scenario is illustrated in the part P1 of Figure 4.1.

Some location-limited channels have very restricted data transfer capacity and it might be impossible to exchange all the needed security initiation information using the location-limited channel itself. Therefore, in some scenarios it necessary to use an additional setup channel, which we call *complementary channel*. This scenario is illustrated the part P2 of Figure 4.1.

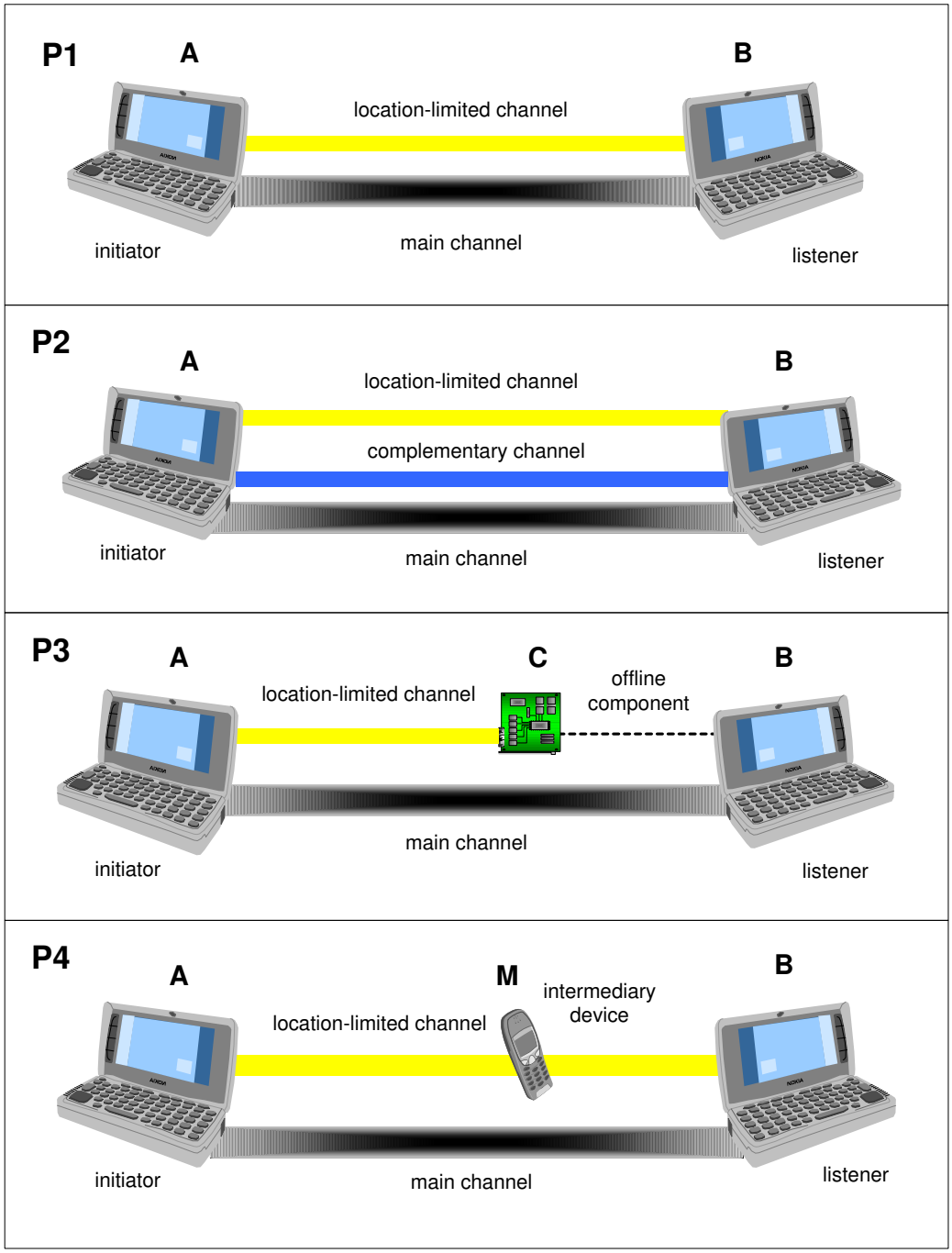

Figure 4.1: Peer-to-peer security initiation models

Even though the complementary channel is conceptually a separate channel from the main communication link, typically both of these channels are implemented using same communication technology. However, in some cases it might be more practical to implement the complementary channel with a different communication technology. For example the complementary channel could be implemented as Bluetooth link and the main communication channel could be a high bandwidth wide area connection.

Two online components	two-way one-way send-only one-way receive-only
Online and offline component	one-way static one-way random one-way timestamp two-way challenge-response
Complementary channel	yes no

Table 4.1: Summary of location-limited channel types

Most of the location-limited channels are formed between two online components. However, some location-limited channels are formed between an online and an *offline* component. This scenario is illustrated the part P3 of Figure 4.1. The offline component can be either static, timestamp, random or challenge-response component.

In some cases the direct location-limited channel communication between the two connecting peers is not very practical. Sometimes a better solution is to use a third device, which we call *intermediary device*, for creating the secure connection between the two communicating peers.

Let us consider a scenario in which two laptop users want to establish a secure connection in a meeting room using an infrared location-limited channel. It is quite difficult to move the two laptops to such positions that the infrared communication becomes possible. A more user friendly solution is to use a third intermediary device, such as mobile phone, for creating the secure connection. This scenario is illustrated in part P4 of Figure 4.1. The location-limited channels between the end point devices and the intermediary device can be any of the previously mentioned type.

Table 4.1 summarizes the different location-limited channel settings. There are three types of location-limited channels formed between two online components. If the other component is an offline component, there are four different kind of channels. In each of these scenarios, the complementary channel may or may not be present. From this reasoning we can see that there are 14 different location-limited channel settings when initiating security directly between the two devices. If an intermediary device is used, there can be 14 different kind of channels between the intermediary device and the first device and 14 different kind of channels between the intermediary device and the seconds device. Thus, the number of possible scenarios shoots up to 196.

4.4 Peer-to-peer protocols

The previous section described the different location-limited channel settings for initiating security between two peers. In this section we define the actual location-limited channel *pre-authentication protocols* that can be used for initiating security in these different settings.

In all the protocols the connection initiator sends an *initiation-request* to the listener. The

LLC :	message is sent using location-limited channel
$Comp$:	message is sent using complementary channel
PK_A :	public key of device A
SK :	shared secret key
$comp_A$:	complementary channel address of device A
$init$:	initiation-request
req :	connection-request
res :	connection-response
$hash(D)$:	hash calculated from data D
$sig_A(D)$:	data D signed with the private key of A
$apps$:	list of supported applications

Table 4.2: Shorthands for pre-authentication protocol descriptions

initiation-request contains the following information.

- *Security objective:* Defines the security objective for this connection. The security objective can be either mutual, listener or initiator authentication.
- *Application ID:* Defines the application that is going to use this connection.
- *Lifetime:* The duration of the created secure connection.

The listener can either accept or reject the connection initiation request according to either its local *security policy* or *user input*. Security policy can for example define which applications require mutual authentication and which applications can be used with unilateral authentication. Alternative approach is that the security initiation system lets the user decide whether he wants to accept the requested connection.

In addition to just exchanging the keys, the pre-authentication protocols should also exchange all the other information that is needed for setting up the secure connection. For this purpose, the connection initiator sends a *connection-request* to the listener. The connection-request contains the following information.

- *Name:* The name of the initiator device.
- *Application data:* The application specific information.
- *Security protocol ID:* Defines the security protocol that is going to be used for securing the actual connection.
- *Security protocol data:* The security protocol specific information.
- *Address information:* The main communication channel address of the initiator.

The listener responds with a *connection-response* that contains its name, application specific data, the security protocol specific information and the address of the listener. The pre-authentication protocols descriptions use shorthands described in Table 4.2

4.4.1 Scenario P1

The first scenario is described in part P1 of Figure 4.1. Both of the devices, called A and B, are equipped with online location-limited components and the location-limited channel is two-way and it can be used to exchange all the needed information.

Protocol PK1:

$LLC : A \rightarrow B : init, PK_A, req$
$LLC : B \rightarrow A : PK_B, res$

Device A sends an initiation-request, his public key and connection-request to B using the location-limited channel. B checks the initiation request. If the local policy of B or the user accepts this connection initiation, then B responds with its public key and connection-response. After these two messages, the peers have exchanged all the needed information for setting up the secure connection.

The protocol fulfills mutual authentication requirements. If the security objective is listener authentication, then only B sends its public key to A, and when the security objective is initiator authentication, then only A sends its public key to B. The initiator always decides the security objective and the listener either replies according to the protocol or rejects the connection initiation request.

Most of the wireless location-limited channels are only receiver location-limited. The sender of the message can be sure that the receiver of the message is in proximity, however, there are no guarantees that the received message was sent within proximity. One might think that the above defined protocol is not suitable to be used with location-limited channels that are not both sender and receiver location-limited. When the location-limited channel is for example only receiver location-limited, device B cannot be sure that the received message was really sent by A and vice versa. It seems that the protocol does not provide any authentication.

However, as Balfanz et al. noted in [16], all that is needed to overcome this problem is one's ability to count. When for example device A receives the location-limited channel message from B, it can be sure that message really comes from B, if the user of device A physically confirms that device B sent a message (for example the traveler at the airport waiting lounge sees that the printer flashes a light), and the security initiation system implementation checks that only one message is received. If two or more location-limited channel messages are received, then something is wrong, and the connection initiation must be aborted.

In some cases, the desired outcome of the message exchange should be a shared secret between the two devices. However, if the devices support public key cryptography, the shared secret should not be transmitted using the location-limited channel, since most of the location-limited channels do not provide confidentiality. A better solution is to carry out a public key exchange as in Protocol PK1 and then send the shared secret encrypted using the complementary channel.

The Protocol PK1 is based on the assumption that both of the devices can do public key cryptography. If either one of the devices does not support public key cryptography, than the we must transmit a secret key using the location-limited channel. The protocol is almost similar:

Protocol SK1:

$LLC : A \rightarrow B : init, SK, req$
$LLC : B \rightarrow A : res$

Also this protocol provides mutual authentication, but confidentiality is required from the location-limited channel. Typically only wired channels support confidentiality, because it is always possible to eavesdrop even small range wireless channels with powerful enough receiver if the location-limited channel is not truly point-to-point channel. For this reason, the public key protocols should be used whenever possible.

4.4.2 Scenario P2

The second scenario is described in part P2 of Figure 4.1. Both of the devices, called A and B, are equipped with online location-limited components, but the location-limited channel has very restricted data transfer capacity and it cannot be used for exchanging all the needed information. The pre-authentication protocol for this scenario is slightly different.

Protocol PK2:

$LLC : A \rightarrow B : init, hash(PK_A), comp_A$
$LLC : B \rightarrow A : hash(PK_B), comp_B$
$Comp : A \rightarrow B : PK_A, sig_A(req)$
$Comp : B \rightarrow A : PK_B, sig_B(res)$

First A sends an initiation-request, hash of its public key and its complementary address using the location-limited channel. If B accepts the connection, it responds with the hash of its public key and complementary channel address. After that, the complementary channel is used to exchange the complete public keys and connection-request and connection-response. The connection-request and connection-response are signed with the private keys of A and B.

The public key exchange on the complementary channel can be verified using the hashes received from the location-limited channel. And both the connection-request and connection-response can be authenticated using the received public keys.

Also this protocol supports mutual authentication. If the security objective is listener authentication, then only B sends information about its public key, and vice versa. Device B decides whether to accept this connection initiation based on the contents of the initiation-request.

The Protocols PK1 and PK2 follow the basic pre-authentication scheme introduced by Balfanz et al. in [16]. The devices exchange commitment to their public keys using the location-limited channel and after that the subsequent messages are authenticated according to the material received from the location-limited channel.

If either one of the devices does not support public key cryptography, than this protocol cannot be used. The alternative would be to transmit the secret key on the location-limited channel, and then finalize the initiation using the complementary channel.

4.4.3 Scenario P3

The third scenario is described in part P3 of Figure 4.1. One of the devices, A, is equipped with a normal online location-limited component and the other device, B, is equipped with an offline location-limited component C. Typically the offline components have restricted data transfer capacity and a complementary channel is used to finalize the security initiation.

The type of the component determines the protocol. If the offline component is *static* component, the location-limited channel is one-way and the following pre-authentication protocol can be used:

Protocol PK3.1:

$LLC : C \rightarrow A : hash(PK_B), comp_B$
$Comp : A \rightarrow B : init, req, comp_A$
$Comp : B \rightarrow A : PK_B, sig_B(res)$

The static offline component C is the representative of device B and it contains the hash of B's public key and complementary channel address. The device A reads the contents of that component using the one-way location-limited channel. After that, complementary channel is used to finalize the security initiation.

This protocol provides only listener authentication. After the message exchange, A can recognize the public key of B, but not the other way round. B cannot verify that the complementary channel message really comes from A, and there is no use sending the public key of A with the initiation-request and connection-request.

One alternative to achieve mutual authentication is to equip both of the devices with a static component. In this case, the devices can read the static components of each other and then use the complementary channel to finalize the security initiation.

Mutual authentication is also possible when the offline component is a *challenge-response* component that is capable of performing cryptographic operations. The following protocol can be used with challenge-response component:

Protocol PK3.2:

$LLC : A \rightarrow C : PK_A$
$LLC : C \rightarrow A : sig_C(PK_A), PK_B, comp_B$
$Comp : A \rightarrow B : sig_C(PK_A), sig_A(init, req, comp_A)$
$Comp : B \rightarrow A : sig_B(res)$

The offline component C is the representative of device B and these two devices know the public keys of each other. The protocol starts when A sends its public key to the offline component C using the location-limited channel. The offline component signs the public key with its secret key and returns it with the public key of B and complementary channel address of B. After that, A sends the signed public key and signed initiation-request, connection-request and complementary channel address to B using the complementary channel. Device B checks that the public key has been signed by C. Initiation-request and connection-request are authenticated using the public key of A. Finally, B replies with signed connection-response.

Protocol PK3.2 guarantees that A has been in proximity. However, B cannot know that device A is in proximity now. The following attack is possible against Protocol PK3.2.

Alice wants to set up a secure connection with Bob. Alice sends her public key to the offline component that signs it. After that, Alice sends the complementary channel message to Bob. However, there exist Eve, who has had her public key signed by the offline component some time ago. Eve changes the contents of the complementary channel message so, that she replaces the Alice's signed key with her own.

If the offline component is also a *timestamp* component, than the component could include a timestamp to the signature. When device B receives the complementary channel message, it checks the signature is fresh. Old signatures are not accepted.

If either one of the devices does not support public key cryptography and the offline component is static, than security initiation is not possible. There is no point in putting a secret key to the static component. In case of challenge-response or timestamp component, it is possible to achieve mutual authentication using a pre-authentication protocol in which the offline component generates keys and distributes them to the peers in proximity. However, these protocols are rather complex and outside the scope of this thesis.

4.4.4 Scenario P4

The fourth scenario is described in part P4 of Figure 4.1. Both of the devices, called A and B, have an online location-limited component and an intermediary device, here called M, is used in the security initiation.

Protocol PK4.1:

$LLC : A \rightarrow M : apps, PK_A$
$LLC : M \rightarrow B : init, PK_A, req$
$LLC : B \rightarrow M : PK_B, res$
$LLC : M \rightarrow A : PK_B, res$

In this protocol the intermediary device starts the connection initiation process, and the intermediary device M generates the initiation-request and the connection-request. For this purpose, the list of supported applications must be transferred from the device A to the intermediary device M. After this first touch the intermediary device must be moved close to device B, and finally the intermediary device must be moved back to device A. Altogether three touches are required.

Again, this protocol provides support for mutual authentication. If the security objective is only unilateral authentication, than the keys are delivered to only one direction. If the location-limited channel has very low data transfer capacity, only hashes and complementary channel addresses could be exchanged using the location-limited channel, and the initiation could be finalized with the complementary channel as in Protocol PK2.

It should be noted that the first touch has to be done only once between devices A and M. Once the intermediary device knows the public key of device A, secure connections can be initiated with only two touches (first touching device B and then touching device A). Thus the actual number of required touches is 1 + 2. If we assume that also the intermediary device can do public key cryptography operations and there exists a complementary channel between devices A and M, then we can reduce the number of needed touches:

Protocol PK4.2:

$LLC : A \rightarrow M : apps, PK_A, comp_A$

$LLC : M \rightarrow A : PK_M$
$LLC : M \rightarrow B : init, PK_A, req$
$LLC : B \rightarrow M : PK_B, res$
$Comp : M \rightarrow A : sig_M(PK_B, res)$

In this protocol devices A and M exchange public keys during the first touch. In addition, device A sends list of its applications and its complementary channel address. Again, this first touch has to be done only once. After this first touch, secure connections can established with other devices only by touching them with the intermediary device M. The device M generates the initiation-request and connection-request and mediates the response back to device A signature protected using the complementary channel. In this protocol the number of required touches is 1 + 1.

If one of the devices does not support public key cryptography, than we must use the following protocol:

Protocol SK4:

$LLC : A \rightarrow M : apps, SK, comp_A$
$LLC : M \rightarrow B : init, SK, req$
$LLC : B \rightarrow M : res$
$Comp : M \rightarrow A : sig_{SK}(res)$

This protocol requires two touches. First the intermediary device M is moved close to the device A. Device A generates the secret key and sends it to the intermediary device M. This message also contains the list of applications and the complementary channel address. After that, the user of the intermediary device selects the application and the intermediary device is moved close to device B. B provide a response to the received message and the intermediary mediates this response back to device A signature protected using the complementary channel.

4.5 Models for group initiation

In addition to creating secure peer-to-peer connections, location-limited channels can be also used to pre-authenticate secure group connections. Section 2.6 discussed the basics of group security, and as already mentioned, Balfanz et al. defined two group pre-authentication methods.

A centrally managed key distribution protocol could be easily pre-authenticated so that the group controller establishes location-limited channel links with all the group members and carries out one of the two-party pre-authentication protocols defined in the previous section. This method is described in the part G1 of Figure 4.2.

The drawback of this scheme is the poor usability. For example if the devices are laptops and the location-limited channel that is used is infrared, it is quite difficult to move the controller laptop so that infrared links can be established with all the group members. Another alternative would be that an intermediary device, such as mobile phone, acts as the group controller and establishes point-to-point location-limited links with all the group members.

Balfanz et al. also described a method for pre-authenticating key distribution protocols with a broadcast location-limited channel, such as audio. In this scheme all the group

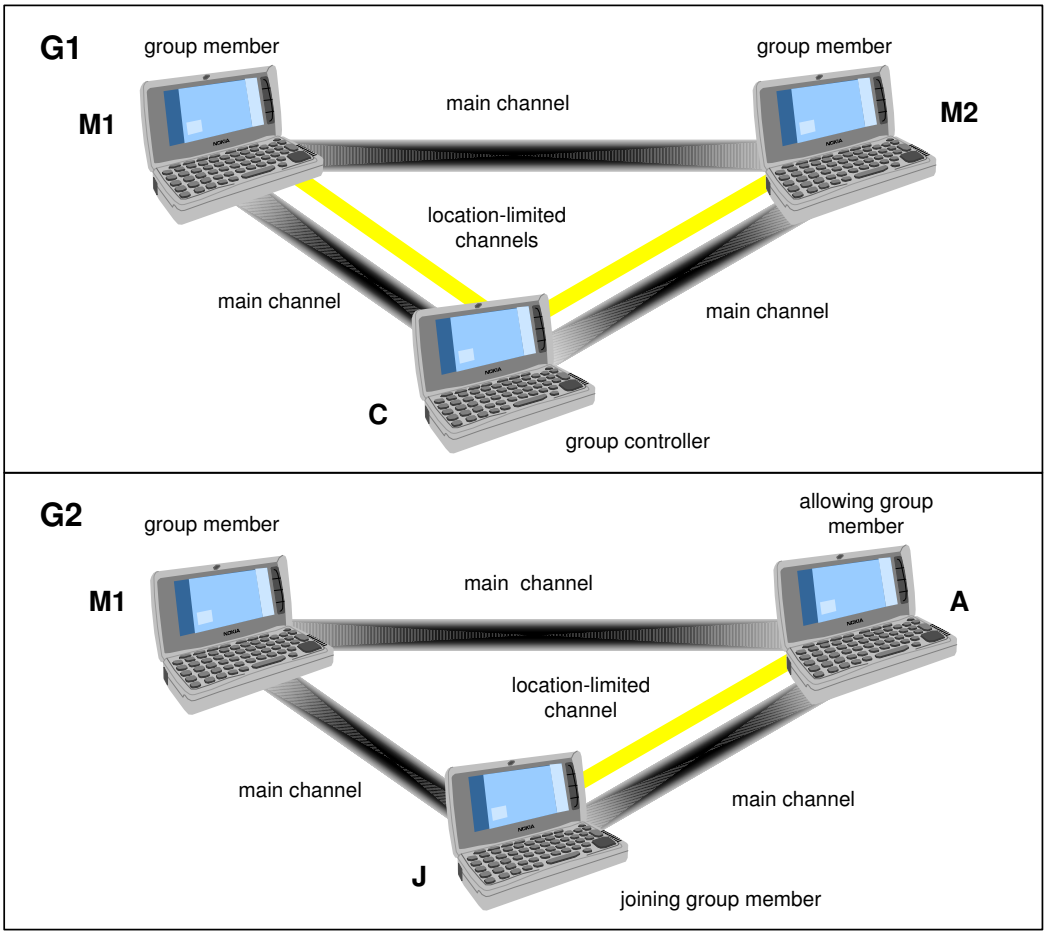

Figure 4.2: Group security initiation models

members broadcast hash of their public keys to other group members using the broadcast location-limited channel. After that, an authenticated key agreement protocol can be started.

We present another method for authenticating key agreement protocols. This method is based on the use of two-party location-limited channel connections and the method is illustrated in the part G2 of Figure 4.2. In this method, the joining group member has to authenticate itself to only one of the group members using a two-party location-limited connection. The benefit of this method is that it is not necessary to implement additional broadcast location-limited channel for creating group connections.

4.6 Group protocols

The previous section illustrated two modes for initiating secure group connections. The purpose of this section is to describe the actual group pre-authentication protocols.

4.6.1 Scenario G1

This scenario is illustrated in part G1 of Figure 4.2. One device, here called C, is assigned to become the group controller and this device carries out one of the two-party pre-authentication protocols with all the other group members, here called M1, M2 and so on. When for example all of the devices are equipped with an online location-limited component and the location-limited channel can be used to exchange all the needed material, we can use Protocol PK1 for the two-party pre-authentication.

Protocol GPK1:

$LLC : C \rightarrow M1 : init, PK_C, req$
$LLC : M1 \rightarrow C : PK_{M1}, res$
$LLC : C \rightarrow M2 : init, PK_C, req$
$LLC : M2 \rightarrow C : PK_{M2}, res$

This protocol provides distribution authentication. After the location-limited channel message exchange, the group controller knows all the public keys of all the other group members and all the other members know the public key of the group controller. After the pre-authentication, the secure group connection can be established using some key distribution protocol. If one of the devices does not support public key cryptography, then we can use Protocol SK1 for the two-party pre-authentication.

Protocol GSK1:

$LLC : C \rightarrow M1 : init, SK_1, req$
$LLC : M1 \rightarrow C : res$
$LLC : C \rightarrow M2 : init, SK_2, req$
$LLC : M2 \rightarrow C : res$

Also this protocol provides distribution authentication. The group controller establishes shared secrets with all the other group members, and after the message exchange some key distribution protocol can be started. The type of the location-limited components and the capabilities of the devices define the two-party protocol that is used between the group controller and other group members.

4.6.2 Scenario G2

This scenario is illustrated in part G2 of Figure 4.2. Whenever a new peer, called *joining member*, wants to join the group, he establishes a location-limited channel connection with only one of the existing group members, called *allowing member*. The allowing member makes the decision, whether the new member is accepted to the group or not.

These two peers first authenticate each other, and after that, the allowing member mediates the keys of all the other group members to the joining member and the key information of the joining member to all the other group members one by one.

Protocol G2:

$LLC : J \rightarrow A : PK_J, comp_J$
$LLC : A \rightarrow J : PK_A$
$Comp : A \rightarrow J : sig_A(PK_{ALL})$
$Comp : A \rightarrow M1 : sig_A(PK_J)$
$Comp : A \rightarrow M1 : sig_A(PK_J)$

This protocol provides agreement authentication. After the message exchange, all the group members are able to recognize the public keys of each other, and some key agreement protocol can be started. The protocol is based on the assumption that all the group members trust the allowing member. The benefit of this protocol is the fact that this protocol requires only one touch. The joining members must establish the location-limited channel connection with only one of the group members, and there is no need to implement additional broadcast location-limited channel for initiating group connections.

4.7 System overview

The previous sections of this chapter have analyzed the different location-limited channels and defined a set of location-limited channel protocols. The purpose of the rest of this chapter is to present a design for location-limited channel based security initiation system.

4.7.1 Selecting the pre-authentication protocol

The first design decision that we have to make is how to choose the right pre-authentication protocol for each security initiation scenario. The characteristics of the used location-limited channel determine the actual protocol, and if the system supports only one location-limited channel, than there is only one possible protocol to use.

But how should the system decide which location-limited channel should be used if several are available? The only possible solution is to leave this decision to the user, since the two devices do not know the addresses of each other before the location-limited channel communication and they cannot negotiate which location-limited channel should be used before the actual location-limited channel communication. All that the security initiation system can do in order to help the user in this decision is to present the user a list of selectable location-limited channel in a preferred order, that is the channel with best security properties is first in the list.

4.7.2 Security initiation processes

The actual security initiation process and the required user interactions depend on the role of the device. This section describes the security initiation processes from the point of view of the user of the initiator, listener and intermediary devices.

Initiator process: The following list presents the required user interactions from the user who starts the security initiation process.

1. User starts the security initiation system.
2. The system user interface presents a list of possible applications. The user selects the application that he wants to use.
3. The system user interface presents a list of possible location-limited channels if more than one is available. The user selects the location-limited channel that he wants to use.

4. The user defines the duration of the connection. The connection can be either one time connection, or the user may define the duration of the connection in days.

5. The system user interface instructs the user to move the device close to the other device and wait for a sound. The user does as instructed.

6. The user hears a sound. The system user interface presents the name of the other party. The user can either accept the connection and save it with the name he chooses, or reject the connection. At this point, the user must visually verify that the correct other peer is truly engaged in the security initiation. The visual verification can be for example checking that the other peer accepts the connection.

After that, the security initiation process is finished. The application is started automatically by the security initiation system, and the user can start using the application. The initiator security initiation process requires six user interactions. The task of selecting the used location-limited channel is skipped if only one location-limited channel is available. Also the task of defining the connection duration can be skipped if some default value, such as one-time connection, is always used.

Listener process: The following list presents the required user interactions from the user who waits for the connection initiation.

1. User starts the security initiation system.

2. The system user interface presents a list of possible location-limited channels if more than one is available. The user selects the location-limited channel that he wants to use. The user is instructed to wait for a sound.

3. The user hears a sound. The system user interface presents the name of the connection initiator and the application that he wants to use. The user can either accept and save the connection, or reject it. Also at this point, the user must visually verify that he is establishing the connection with the correct peer.

After that, the security initiation process is finished. The application is started automatically by the security initiation system, and the user can start using the application. The listener process requires three user interactions. The number of needed user interactions can reduced to one, if the security initiation system is always on and listening to all the supported location-limited channels. However, this is not possible with all the location-limited channels since keeping the location-limited channel always active consumes battery power.

Intermediary process: The required user interactions depend on the details of the selected intermediary device pre-authentication protocol. Protocol PK4.1 requires 1 + 2 touches and Protocol PK4.2 requires 1 + 1 touches. From the point of view of the user of the intermediary device, the security initiation process goes as follows:

1. The user starts the security initiation system.

2. The user is asked to select the used location-limited channel if more than one is available.

3. The user is asked to move the device close to the first device and wait for a sound. The user does as instructed.

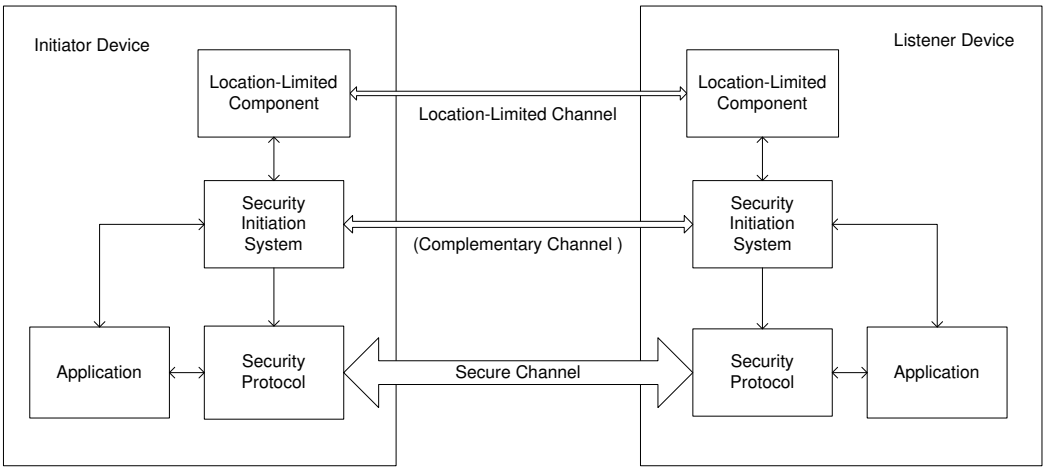

Figure 4.3: System overview in two device scenario

4-5. The user hears a sound. The system present a list of possible applications. The user selects the application and possibly defines the duration of the connection. The user is asked to move the device close to the other device. The user does as instructed.

6. The user hears a sound. At this point the security initiation might be complete, or alternatively the user is still asked to move the intermediary device close to the first device and wait for a sound.

7. The user hears a sound and the security initiation is complete.

4.8 Software architecture

This section describes the internal software architecture of the security initiation system. First we describe the different software components and their relations and then we define the interaction of the different components.

4.8.1 General architecture

Figure 4.3 describes the general architecture of location-limited channel based security initiation system. The security initiation system must be present on both devices. The security initiation system exchanges the needed security initiation information using the selected pre-authentication protocol. The complementary channel is used, if that is required by the protocol. After that, the security initiation system configures the selected security protocol on both devices and launches the applications. The applications start communicating and the communication is secured by the security protocol.

Figure 4.4 describes the general architecture if an intermediary device is used in security initiation. The security initiation system must be present on all three devices. Pre-authentication protocol messages are exchanged using the location-limited channels and the complementary channel is used to complete the initiation if necessary. After the pre-authentication protocol is complete, the security initiation system configures the security

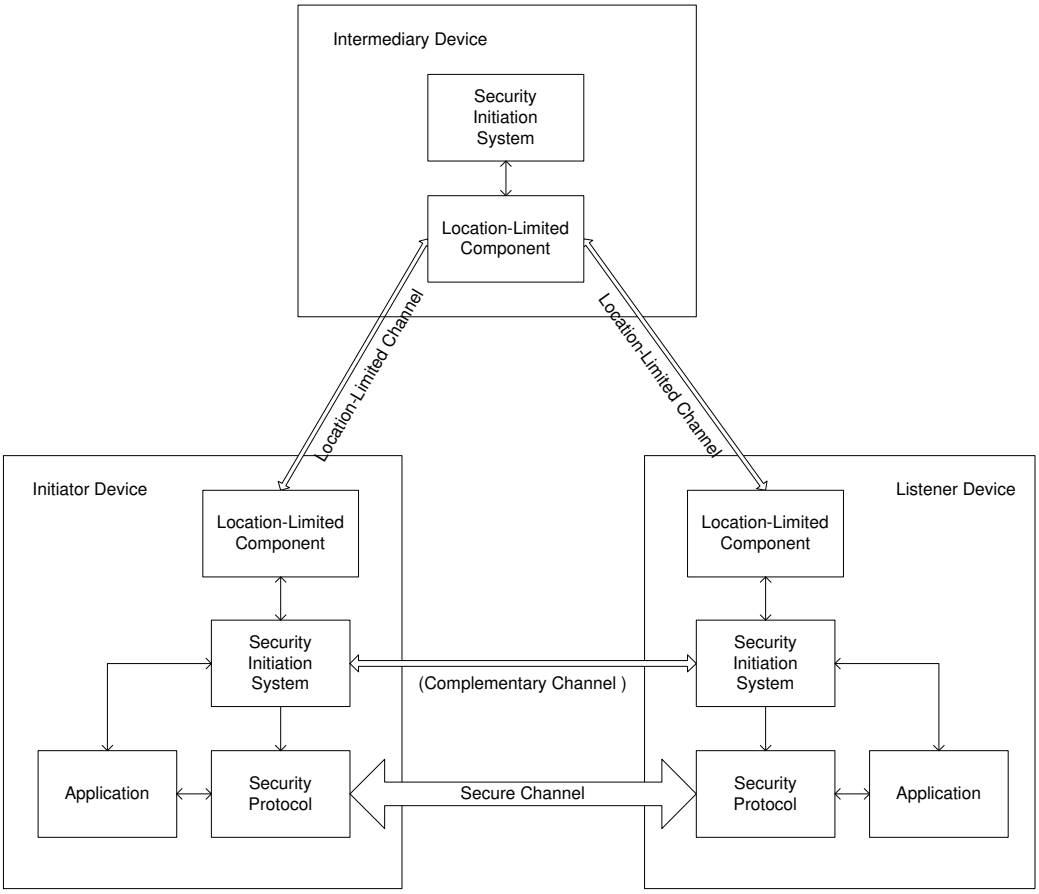

Figure 4.4: System overview in three device scenario

protocols on end point devices and launches the applications. After that the applications running on both end point devices can start communicating over a secure channel.

Initiating group connections can always be done either by using the direct initiation or intermediary initiation. The security initiation system has to be present on all the devices and the complementary channel can be used between the group members to finalize the security initiation. After the security initiation is done, the security initiation system configures the security protocols on all devices and the secure group communication can start.

4.8.2 Internal components

Figure 4.5 illustrates the different internal software components of the security initiation system and the relations of these components.

Initiation Software: The core of the security initiation system is a software component called Initiation Software. This component is responsible for creating and controlling the other components and it has the runtime control during the whole security initiation process.

The Initiation Software implements the pre-authentication protocols. It sends and receives messages using the Location-Limited Components and the Complementary Components.

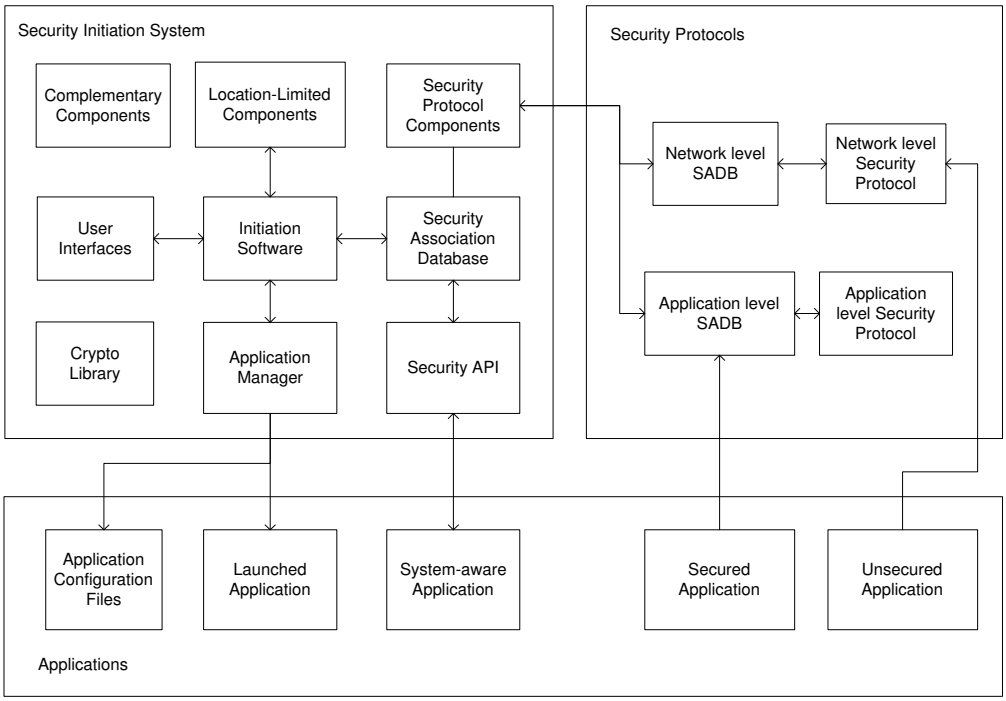

Figure 4.5: Security initiation system software architecture

The Initiation Software also handles the user interaction events received from the User Interface of the system. The Initiation Software is also responsible for creating the security associations and storing them into the system internal Security Association Database. The Initiation Software also controls the Security Protocol Components that configure the different security protocols.

Location-Limited Component: For each supported location-limited channel technology there should be one Location-Limited Component that controls the communication on that physical location-limited channel. The Location-Limited Component takes care of setting up location-limited channel connections and sending and receiving messages.

The different Location-Limited Components of the same physical location-limited component type should all provide similar interface to the Initiation Software. This means that it should possible to replace one Location-Limited Component with another component of similar type without any modifications to other software.

Complementary Component: For each supported complementary channel technology there should be a Complementary Component that controls the complementary communication. Also this component should provide a uniform interface so that all the different complementary channels can be used with similar function calls from the Initiation Software.

User Interface: The system needs also different user interfaces. Separate User Interface components should be implemented for connection initiator, listener and mediator. The purpose of the User Interface component is to get input from the user and pass it to the Initiation Software component. The User Interface component also guides the user during the security initiation process.

Security Association Database: The database component is used to store the established security associations. The Security Association Database module provides an interface for adding, getting, updating and removing database entries. Each SADB entry contains the following information:

- *Peer name:* Human readable name given by the user for this connection.
- *Address type:* The type of the peer address. This can be IP address, Bluetooth address, or any other address type.
- *Address:* The network address of the other peer.
- *Application ID:* Defines the application that uses this security association.
- *Application data:* Application specific data, such as port numbers.
- *Security protocol ID:* The security protocol that is used to protect the connections of this application.
- *Security protocol data:* Security protocol specific parameters such as the used algorithms.
- *Key type:* Defines the algorithm and key length.
- *Key:* The actual key.
- *Lifetime:* Defines how long the security association is valid.

Security Protocol Component: For each supported security protocol, there should be one Security Protocol Component that takes care of configuring the external security protocol with the internally saved security associations.

Security API: The system also contains a component called Security API. The purpose of this module is to provide security services for system-aware applications. The Security API should provide the following functions:

- *Start initiation:* Starts the security initiation process. This function brings up the initiator user interface.
- *Wait for initiation:* Sets the system waiting for security initiation requests on given location-limited channel. This function brings up the listener user interface.
- *Get all database entries:* Returns a list of all database entries in the system internal Security Association Database.
- *Get database entry by name:* Return the database entry that was stored with the given name.
- *Encrypt data by name:* Encrypts the given data with the key that is associated with the given name in the SADB. If the entry that matches the name contains a public key, then the data is encrypted with that public key. If the entry contains a shared secret, then the data is encrypted with that secret key.

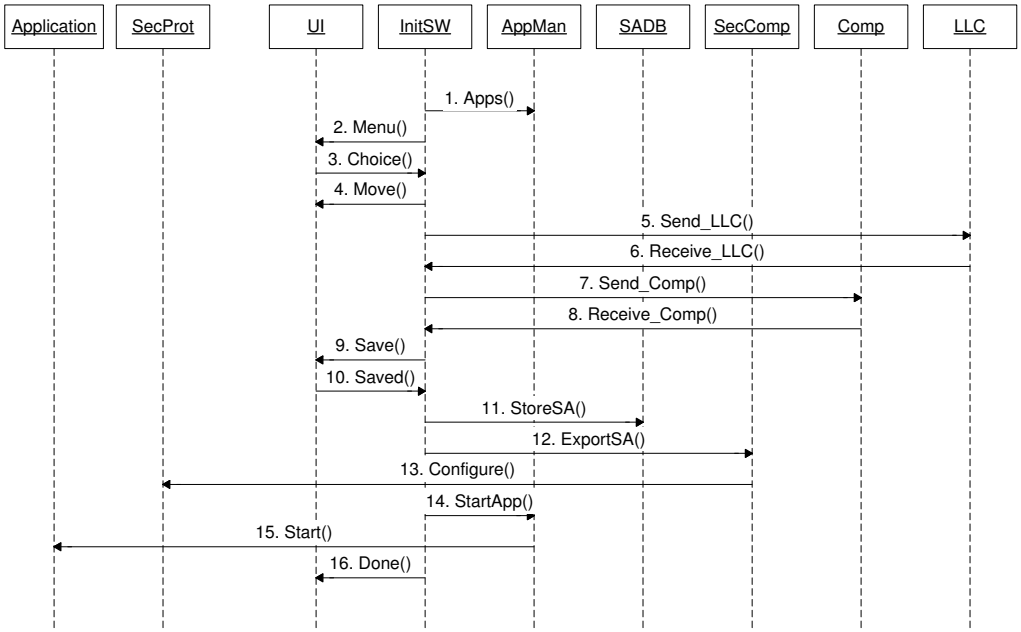

Figure 4.6: Security initiation system component interaction

- *Decrypt data by name:* Decrypts the given data with the key that is associated with the given name in the SADB.
- *Sign data:* Signs the given data with the private key of this device.
- *Verify signature by name:* Verifies that the given signature matches the key that is associated with the name in the database.

Crypto Library: Both the Security Initiation software and the Security API need to do various cryptographic operations, such as encrypt and decrypt data, calculate hashes and verify signatures. Crypto Library is the software component that provides these general crypto functions.

Application Manager: The Application Manager component knows which applications are supported by the security initiation system and what are the security objectives for these applications. The Application Manager module also takes care of starting the applications. The Application Manager gains this information by reading application configuration files that define how each application should be treated.

4.8.3 Component interaction

The previous section described the different internal software components. The purpose of this section is to describe the interactions of these software components. Figure 4.6 presents a sequence diagram that explains the component interactions of the initiator device.

1. The security initiation process begins when the user starts the security initiation system. At first, the Initiation Software module queries the Application Manager module for a list of locally supported applications. The Application Manager module returns the list of supported applications and their security objectives.

2. The Initiation Software presents a list of selectable applications to the User Interface module of the system.

3. The user selects the application that he wants to use and the location-limited channel type. The user may also define a duration for the connection, or alternatively a default value is used for the duration. The selected initiation parameters are returned back to the Initiation Software component.

4. The Initiation Software selects the used pre-authentication protocol according to the selected application, the security objective set for that application and the selected location-limited channel type. The Initiation Software tells the User Interface to instructs the user to move the device close to the other device and wait for a sound.

5-6. The Initiation Software generates the pre-authentication protocol message and sends it using the selected Location-Limited Component. After that the Initiation Software starts waiting for a reply and processes the reply when it is received.

7-8. If the selected pre-authentication protocol requires the use of a complementary channel, then the message exchange is finalized using the Complementary Component. Again, the Initiation Software generates and processes the sent and received messages.

9. Now the pre-authentication protocol message exchange is ready. The Initiation Software makes a sound and tells the User Interface that the user should be asked to save the accepted connection.

10. The user either saves the initiated connection with a name he chooses or rejects the connection. If the connection is rejected, than the security initiation processes stops.

11. All the initiated material is now ready and the user has assigned a name to this connection. The Initiation Software creates a database entry and saves this entry to the Security Association Database.

12-13. The Initiation Software tells the selected Security Protocol Component to configure the security protocol with the initiated information.

14-15. The Initiation Software tells the Application Manager module to start the selected application.

16. Finally, the User Interface shows to the user that the security initiation process is now complete.

The component interactions are somewhat different for listener and intermediary devices, however, the basic scheme is the same. The Initiation Software controls the security initiation process by sending and receiving messages according to the selected pre-authentication protocol. After the pre-authentication protocol message exchange, the Initiation Software stores the initiated security associations, configures the security protocol and starts the application if necessary.

Chapter 5

Implementation

The previous chapter presented the security initiation system design. The purpose of this chapter is to describe the prototype implementation of the security initiation system. The rest of this chapter is organized as follows. Section 5.1 gives and overview of the implementation and Section 5.2 describes the implemented protocols. Section 5.3 presents the implemented software components and Section 5.4 describes the implemented demonstrations. Finally, Section 5.5 discusses implementation findings.

5.1 Implementation overview

This section briefly explains the selected implementation environment and the scope of the implementation.

The implementation was made for three different platforms. The main implementation platform was chosen to be a standard Nokia 3650 phone. The Nokia 3650 phone runs on Symbian OS v6.1. The Nokia 3650 has an internal infrared port and support for Bluetooth, GSM data and GPRS communication. The Nokia 3650 has a relatively powerful CPU, so public key cryptography is reasonable with this phone.

The two other implementation platforms were chosen to be standard IBM Thinkpad laptops running on Linux RedHat 8.0 and MS Windows 2000. The laptops have internal infrared ports and support for Bluetooth and WLAN technologies. Naturally, also the laptops provide support for public key cryptography.

For our implementation, we chose infrared as the location-limited channel technology. Infrared forms a two-way location-limited channel and the infrared port is an online location-limited component. Infrared location-limited channel has relatively high data transfer capacity so that it can be used to exchange all the needed information on the location-limited channel.

The main reason for choosing infrared was the fact that all the selected implementation platforms provided built-in support for infrared communication. Also many other mobile devices, such as PDAs and other mobile phones, support infrared.

The general software architecture is designed so that it can be used in different ad hoc environments with different security protocols. For our implementation, we chose to provide support for IP based communication with IPsec as the security protocol. The standard

Symbian OS v6.1 does not provide support for IPsec. Therefore we used an IPsec implementation [54] for Symbian OS v6.1 that has been developed as part of 6PACK project in Nokia Research Center. For the RedHat Linux laptops we used FreeS/WAN [2] IPsec implementation and for MS Windows 2000 laptops we used the IPsec implementation that comes with the operating system.

The implementation was demonstrated in three different scenarios. In the first demonstration scenario, the target was to set up a secure FTP connection between a Symbian phone and a Linux laptop. In the second demonstration scenario, the target was to initiate a secure NetMeeting connection between two Windows laptops using a Symbian phone as the intermediary device. In the third scenario, the target was to demonstrate the secure communication of system-aware applications with the help of the Security API.

The implementation work was done in co-operation with Sampo Sovio and Kalle Kuismanen. The author was responsible for all the Symbian side implementation. This implementation was done in C++ language using the Symbian 6.1 SDK. Sampo Sovio provided the Linux implementation and Kalle Kuismanen was responsible for the Windows implementation. The system-aware application used in the third demonstration was designed and implemented by Seamus Moloney and Tapio Suihko for another Nokia Research Center project called MyPocket.

This chapter focuses on the Symbian OS implementation that was done by the author. For the Symbian implementation, the functionality of the security initiation system was divided into three separate Symbian applications:

- *Initiator application:* This application is started by the user, when he wants to start initiating a secure connection with another peer in proximity. This application provides a user interface for the initiator process described in Section 4.7.2.
- *Listener application:* This application listens for incoming connection initiations. This application provides a user interface for the listener process.
- *Mediator application:* This application is used when the security initiation is done using a third intermediary device. The application provides a user interface for the intermediary process.

5.2 Protocol implementation

The purpose of this section is to describe the protocol implementation. We describe which protocols were implemented and the general structure of the pre-authentication protocol messages.

5.2.1 Implemented protocols

In the scope of this thesis, we implemented only peer-to-peer protocols. Section 4.3 describes the different peer-to-peer models for using location-limited channel in security initiation.

For Scenario P1, in which the security initiation process happens directly between the two peers, we implemented Protocol PK1. This is the most simple pre-authentication protocol,

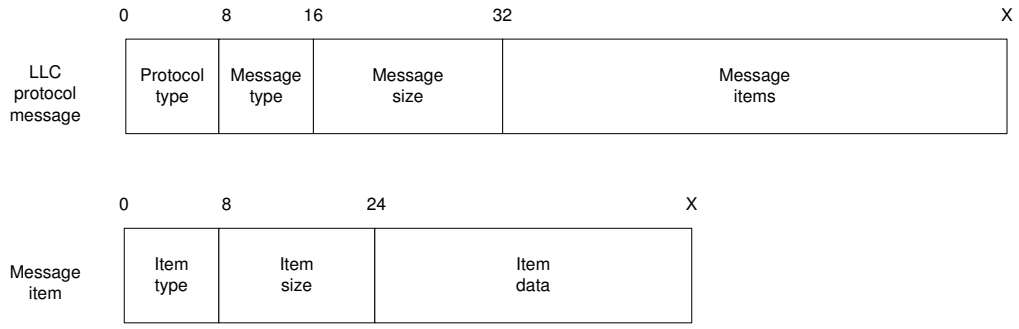

Figure 5.1: Message structure

in which only two messages are exchanged directly between the two peers.

For Scenario P2, in which the complementary channel is used to finalize the security initiation, we implemented Protocol PK2. Because infrared channel has relatively high data transfer capacity, the use of complementary channel is not necessary. However, we decided to implement one complementary channel protocol, just to test and illustrate the usage of complementary channel in security initiation.

In Scenario P3 one of the location-limited components is an offline component. The infrared component is an online component, and for this reason, we did not implement any of the pre-authentication protocols for Scenario P3.

For Scenario P4, in which the intermediary device is used, we implemented Protocol PK4.1. This is the most simple intermediary device protocol, in which the intermediary device does no cryptographic operations.

All of the implementation platforms support public key cryptography, and therefore there was no need to implement the symmetric cryptography variants of these protocols.

5.2.2 Message structure

The pre-authentication protocols were implemented as simple binary protocols. Figure 5.1 describes the overall structure of the pre-authentication protocol messages.

Every message starts with a four byte long header. The first byte of the header defines the protocol type. The protocol type can be any of the protocols defined in Chapter 4. The second byte defines the message type. The two last bytes of the header define the size of the message data.

The actual message consists of *message items*. The protocol type and message type determine the items for each message. Also the item format is similar. The first byte of every item defines the type of the item. The following two bytes define the length of the item. The rest of the item contains the actual data.

5.3 Implemented components

Section 4.8 presented the general software architecture. The purpose of this section is to shortly describe the implemented software components.

Location-Limited Component: The implemented Location-Limited Component provides support for infrared communication. The component uses Tiny-TP protocol [62] that provides TCP like reliable transmission over the infrared link. The Tiny-TP protocol is part of the Symbian OS communication stack. The infrared component provides functions for waiting for connections, starting connections, sending and receiving messages and closing the connection.

Establishing an infrared link and receiving messages from the infrared link are operations that are not likely to happen immediately. Therefore, these functions were implemented as asynchronous operations using the Symbian OS Active Objects framework [61] and callback functions.

Complementary Component: The implemented Complementary Component provides support for TCP/IP communication using the TCP/IP socket interface of Symbian OS. The actual bearer of the complementary channel is GSM data connection. Also this component provides functions for waiting connections, starting connections, sending and receiving messages and closing the connection. Complementary channel communication uses special port number 555.

Initiation Software: The Initiation Software is the core of the security initiation system. This component takes care of creating the other components and controlling the security initiation process. The implemented Initiation Software provides support for Protocols PK1, PK2 and PK4.1. The pre-authentication messages are generated and processed according to the protocol definitions. After the initiation process, the Initiation Software creates security association entries and saves them to the Security Association Database.

Security Association Database: The Security Association Database was implemented using a standard SQL database provided by the Symbian OS. The Security Association database contains only one SQL table, which contains all the security association entries. The peer name is the key of the table, so two security association entries can not have the same name. The Security Association Database module contains interface for adding, querying and deleting the stored entries using the standard SQL queries.

Security Protocol Component: The implemented Security Protocol Component configures IPsec connections. The IPsec configuration is done by manually loading security associations into the Symbian OS kernel side IPsec module using the PFKEY API [47]. PFKEY is a socket interface that allows the manipulation of kernel based IPsec Security Association Database with socket calls. The PFKEY communication happens between the kernel level PFKEY listener and the user level Security Protocol component. The Security Protocol Component creates a socket connection to the PFKEY listener and manipulates the SADB by sending messages to the socket.

The IPsec Security Protocol Component has two main tasks. The first task is to generate a suitable IPsec policy. The policy is generated according to the following basic principles: Traffic to port 555 is permitted from all IP addresses. The purpose of this rule is to allow complementary channel communication with all devices. Traffic to all the remote addresses that use IPsec according to the system internal Security Association Database

must be protected with IPsec ESP. The used encryption algorithm is 3DES in CBC mode [8] and the used authentication algorithm is MD5 [52] with HMAC [44]. All the other traffic is permitted without any encryption or authentication.

The policy is generated by checking from the Security Association Database which IP addresses require IPsec protection. After that, the policy is loaded to the kernel using the socket connection with the PFKEY listener. After the policy is loaded, the IPsec implementation protects all the requested IP connections with ESP.

The second task of the Security Protocol Component component is to generate the IPsec security associations. The IPsec security associations are generated according to the information in the system internal Security Association Database. The following information is set for the IPsec security associations:

For each connection, the connection type is set to ESP. The Security Parameter Index (SPI) is also defined. The connection initiator decides the SPI before the security initiation process and transfers it to the connection listener using the message item that contains security protocol specific data. The SPI number is then stored to the Security Association Database and the IPsec Security Protocol Component reads the SPI from the database. The SPI must be unique for each IPsec connection. In our implementation the connection initiator chooses the SPI number randomly.

For each created connection, the IPsec Security Protocol component sets the encryption algorithm to be 3DES-CBC and the authentication algorithm to be HMAC-MD5. The source and destination IP addresses are set according to the address found from the database entry and the address of this device. For each connection, we also must set the symmetric authentication and encryption keys. These keys are derived from the shared secret that is stored in the database entry. The used authentication key is 128 bits long and the used encryption key is 192 bits long.

The IPsec security associations are unidirectional, and therefore two separate IPsec security associations must be defined for each connection. After the IPsec security associations have been created, the associations are added to kernel by sending two PFKEY SADB_ADD messages to the PFKEY listener.

Application Manager: The implemented Application Manager module is a simplified version of the designed one. The Application Manager module just contains hard coded information about predefined set of applications, application types and security objectives.

Crypto Library: As Crypto Library module we used an existing crypto library implementation by Sampo Sovio. This C implementation was ported to Symbian/C++ environment. The Crypto Library provides support for RSA encryptions, decryptions, signings, verifications and key generations. In addition, it supports AES encryption and decryption in ECB and CBC modes. The Crypto Library also supports SHA-1 and HMAC-SHA1 digests.

Security API: The Security API was implemented as a separate Symbian DLL. Section 4.8 defined the services that the Security API should provide. The following list describes the Security API functions that were actually implemented:

- *Get all database entries:* Returns the list of all database entries in the system internal Security Association Database.
- *Get database entry by name:* Returns the database entry that was stored with the given name.

- *Encrypt data by name:* Encrypts the given data with the key that is associated with the given name in the SADB. The RSA algorithm is used, if the public key of the other peer is stored to the database. In case of shared secret, the data is encrypted with the AES in CBC mode.
- *Decrypt data by name:* Decrypts the given data with the key that is associated with the given name in the SADB. The RSA algorithm and the own private key is used, if the public key of the other peer is stored to the database. In case of shared secret, AES is used.
- *Sign data:* The data is signed with the private RSA key of this device.
- *Verify signature by name:* The data is verified with the private key of the other device that is stored in the SADB.

User Interface: Three different user interface components were implemented. One for each of the Initiator, Listener and Mediator applications. The Initiator application user interface is used to select the application, the duration of the connection and the used location-limited channel. After the connection initiation, this user interface is used to give a name for this connection.

The Listener application user interface is used for accepting incoming connection initiation requests. When the Listener application notices a connection initiation request, it shows the name of the initiator and the application. The user can either accept this connection with the name he chooses, or alternatively reject the connection.

The Mediator application user interface is used to select the application, the duration of the connection and the used location-limited channel type. After that, the user interface guides the user through the intermediary initiation process.

5.4 Implemented demonstrations

The prototype implementation was tested in three separate demonstrations. This section describes those demonstrations.

Demonstration 1: The target of the first demonstration was to establish a secure FTP connection between a Symbian phone and a Linux laptop. A normal FTP client was installed to the phone and a normal FTP server was installed to the laptop. The main communication channel between these two devices was a GSM data connection.

In this demonstration, the Symbian phone acts as the connection initiator. The security initiation process starts when the user of the phone starts the Initiator application and the user interface of the initiator application is brought up. This user interface is illustrated in Figure 5.2.

The user is presented a list of supported applications. The user selects that he wants to use FTP (Step 1 in Figure 5.2). After that, the user is asked whether this is a one-time connection, or if the user wants to initiate a longer lasting connection (Step 2). The user wishes to initiate the connection for 30 days and he sets the number of days to the user interface (Step 3).

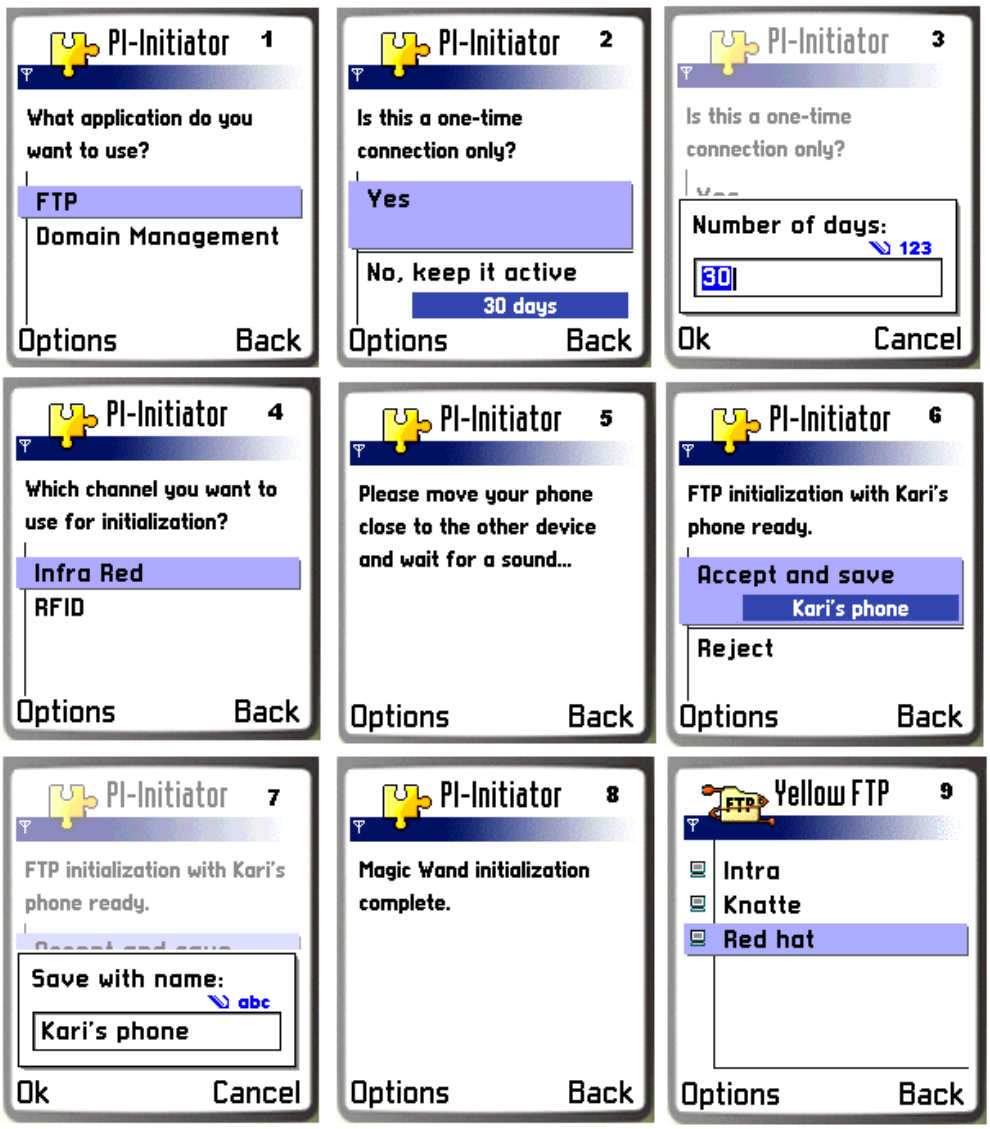

Figure 5.2: Initiator application user interface in FTP demonstration

The next step is that the user interface asks the user to select the location-limited channel that is going to be used for the security initiation (Step 4). The user selects infrared and the Initiator application starts establishing infrared connection. After that, the user interface tells the user to move the phone close to the other device and wait for a sound (Step 5).

The security initiation system implementation on the Linux laptop is always listening for infrared connections. As the user moves the phone close to the infrared port of the Linux laptop, the infrared connection is established and the Symbian side Initiator application starts the pre-authentication protocol. In this demonstration the devices use Protocol PK1, in which public keys and all the other information are exchanged directly between the Symbian phone and the Linux laptop. After the public key exchange, the phone generates a shared secret and sends this to the laptop encrypted with the public key of the laptop.

After the pre-authentication protocol message exchange, the Initiator application makes a sound and the user interface tells the user that FTP initialization is ready. At this point the user either saves the connection with the name he chooses, or rejects the connection (Steps 6). In this case, the user saves the connection (Step 7). After that, the user interface shows that connection initiation is ready (Step 8).

The Initiator application configures the Symbian IPsec implementation with the initiated material. On the Linux side, the security initiation system implementation configures the FreeS/WAN IPsec implementation. The Initiator application starts the FTP application, and the user can start to use the FTP server on the Linux laptop with the IPsec ESP secured communication over the GSM data connection (Step 9).

Demonstration 2: In the second demonstration a secure NetMeeting connection was established between two Windows laptops using a Symbian phone as the intermediary device. The main communication link between the two phones was WLAN connection.

This demonstration was based on Protocol PK4.1, but instead of public keys we used self signed certificates for Windows IPsec compatibility. The security initiation process begins, when the user starts the Mediator application on his Symbian phone and is instructed to touch his own device. The user brings the phone close to his laptop, and the laptop sends its self signed certificate and list of possible applications to the phone.

From the application list the user selects that he wants to use Netmeeting and the user interface instructs him to touch the other device. The user does as instructed, and the intermediary device mediates the certificate along with the other needed information to the other device. The other device responds with its self signed certificate, which is mediated back to the own device, with the third touch.

After that, the security initiation system Windows implementations configure IPsec on both Windows laptops and start the NetMeeting applications.

Demonstration 3: The purpose of the third demonstration was to initiate security between two Symbian phones. After that, the established security associations were used by system-aware domain management applications.

Also this demonstration was based on Protocol PK1. The initiation process starts when the user of the initiator device starts the Initiator application and selects the system-aware domain management application. The user of the listener device starts the Listener application, and the phones are brought close to each other.

The devices exchange public keys using the infrared channel, and the users are asked to save this connection with a name. After that, the security initiation system generates and saves the security associations to the Security Association Database, and starts the system-aware domain management applications on the phones.

The purpose of the domain management applications is to control and manage the parties that are allowed to access certain files in a public directory. In this demonstration the initiator adds the listener to its domain by encrypting the domain key with public key of the listener. The domain application does the encryption by calling the encryption function of the Security API with the name of the listener.

5.5 Implementation findings

The most difficult part of the implementation turned out to be the IPsec Security Protocol Component. The standard Symbian OS 6.1 does not support IPsec. For our implementation we used the Symbian IPsec implementation that was developed in another project at the Nokia Research Center. The IPSec implementation was done for Symbian OS 7.0 version, and we had to port it for Symbian OS 6.1. The porting and the integration of the Symbian IPsec implementation into our development environment took a lot of time and effort.

The compatibility of the Symbian IPsec implementation with the Linux FreeS/WAN IPsec implementation turned out to be another big problem. It took us weeks to find out the suitable connection parameters, that is the used authentication and encryption algorithms, the key sizes etc., to get the IPsec connection working between a Symbian phone and a Linux laptop.

Chapter 6

Evaluation

This chapter contains the evaluation of this work. This work is evaluated according to the evaluation criteria, presented in Chapter 1, and the lower level requirements, presented in Chapter 3. For easy access, the evaluation criteria is restated in Section 6.1, and the design is evaluated in Section 6.2. The focus of this evaluation is on the design, however, also the implementation is briefly evaluated in Section 6.3. In addition, Section 6.4 analyses applications for security initiation system and Section 6.5 presents a comparison with related work. Finally, Section 6.6 discusses future development.

6.1 Evaluation criteria

In Section 1.3 we defined the following evaluation criteria:

- *Security:* What are the security properties of the system? How do the security properties of the system vary in different usage scenarios and environments? How do the user interactions affect the security of the created connections?
- *Usability:* What kind of user interactions does the use of the system require? Is the security initiation process intuitive and user friendly. How many user interactions are needed to set up a secure connection?
- *Generality:* Is the system architecture applicable to a wide range of different usage scenarios and environments? Can the system be used with a variety of existing security protocols and applications. Can the system be extended easily without major modifications to existing design?
- *Simplicity:* Is the system design simple, easy to understand and implement?

6.2 Evaluation of design

The purpose of this chapter is to evaluate the security initiation system design. The evaluation is based on the evaluation criteria and the lower level requirements.

6.2.1 Security

Requirement 1: A set of pre-authentication protocols is needed for different location-limited channels.

In this thesis we have identified different location-limited channel settings and designed suitable pre-authentication protocols for the these different settings. In the scope of this thesis, it is not possible to describe a pre-authentication protocol for each setting explicitly. However, we have covered the most important scenarios.

Requirement 2: The location-limited channel protocols should initiate security associations that fulfill the requested security objective if that is possible.

The security objective for a peer-to-peer connection can be either mutual authentication, listener authentication or initiator authentication. The designed protocols achieve these requirements in most of the scenarios. The only exception is the case in which the other peer is equipped with a static offline component. In this case, only unilateral authentication can be achieved with the designed protocols.

The type of the location-limited channel affects the security properties of the initiated connection. If the used location-limited channel is only receiver location-limited than the user must verify that the right other peer sent the location-limited channel message and the implementation must check that only one location-limited channel message was received. Otherwise we cannot be sure that the received message came from the right sender.

The security objective for group initiation scenario is either distribution authentication or agreement authentication. The designed protocols fulfill these objectives.

Requirement 3: The pre-authentication protocols should exchange all the other information that is needed for establishing complete security associations.

The designed pre-authentication protocols exchange information about the identity of the other peer, the network address, the application and the security protocol. This information is sufficient for establishing security associations and the designed protocol is extensible so that new information can be added to the protocol messages if needed.

Requirement 4: To provide security for unsecured applications, the security initiation system should configure a network level security protocol.

Requirement 5: To protect the communication of secured applications, the security initiation system should provide the needed security association for the particular security protocol that the application uses.

The security initiation system architecture provides support for both unsecured and secured applications. In case of unsecured application, the system configures a selected network level security protocol. In case of secured application, the system configures the security storage that the application uses with the initiated security association. For each different security protocol and security storage, a different Security Protocol Component will be designed and implemented.

Requirement 6: To provide security for system-aware applications, the security initiation system should provide a security API.

The security initiation system architecture contains a Security API, which provides security services for system-aware application developers. The Security API hides the details of actual key management and lets the application developer to concentrate on the application

itself.

Requirement 7: The user of the security initiation system should be able to define the duration of the created secure connection.

In the designed security initiation processes, the connection initiator can define the duration of the connection, or alternatively some default duration, such as one-time connection, can be used. The listener peer cannot affect the duration of the connection, he just either accepts or rejects the connection initiation attempt. If an intermediary device is used, the user of the intermediary device defines the duration of the connection.

Requirement 8: The security initiation system should store the initiated security associations.

The security initiation system has an internal database for storing the initiated material. A separate database entry is created for each initiated connection.

Requirement 9: The user of the security initiation system should be able to give a name to each initiated secure connection.

In the designed initiation process both parties, the initiator and the listener, give a name for the connection.

Requirement 10: The level of security of the initiated connection should not depend on the security awareness of the user.

In our solution, the only security relevant decision that the user has to make is whether to accept the incoming connection request. If the user sees that a legitimate party in proximity is initiating the connection, he should accept the connection. If not, the connection request should be rejected. This verification is intuitive and does not require any security awareness from the user. The user does not have to understand any technical terminology or security relevant details in order to make the correct decision.

Overall, the security initiation system design fulfills all the security requirements. The requested security objectives can be achieved if that is possible with the used location-limited channel setting and the required user interactions do not require security awareness from the user.

6.2.2 Usability

Requirement 11: The security initiation process should be integrated to the normal use of applications.

Our solution fulfills this requirement. When the user wants to use some application securely, he will start the security initiation system and create a secure connection for this particular application. We feel, that this kind of security initiation process is intuitive and natural for the user.

However, this approach has also some disadvantages. Because the secure connections are application specific, that is the some particular security protocol is configured for that application, sometimes several secure connections must be set up between the same two devices. This is, of course, not very user friendly.

Requirement 12: The number of needed user interactions should be small.

We have defined three different security initiation processes, one for connection initiator,

listener and mediator. The initiator process requires six user interactions. If there is only one location-limited channel alternative and the default value is used for the connection duration, then the number of required user interactions can be reduced to three. A security initiation process that requires up to six user interactions is probably too cumbersome, and the user is tempted to skip to whole process. However, four interactions can be considered acceptable.

The listener process requires only one to three user interactions, which is also acceptable. The mediator process requires six or seven user interaction, depending on the used pre-authentication protocol. Also in this process, the number of required user interactions can be reduced by using some default values.

Requirement 13: The user interactions should be reversible.

The user should be able to change his mind after any security relevant decision is made. Our design allows the user to abort the security initiation during the process.

Requirement 14: The initiation process should be understandable.

We feel, that the designed initiation processes are understandable. The user does not have to type in PIN codes or passwords or perform any other cumbersome tasks . The user just selects the application he wants to use, selects the location-limited channel and defines the duration of the connection and then brings the two devices close to each other. All of these tasks should be very intuitive.

Overall, the usability of our solution is quite good. No difficult user interactions, such as typing long and random PIN codes, are required from the user. However, we feel that the security initiation process should be made even more simple, since the user is tempted to skip the whole process if it requires too many interactions. The ultimate goal would be, that to initiate security, the user would only have to bring the two devices together. The user would not have to choose applications, channels or any other connection parameters.

6.2.3 Generality

Requirement 15: The software architecture should be based on separate and interchangeable software components, that can be added to the system later on.

The software architecture fulfills this requirement elegantly. Support for different location-limited channels, complementary channels and external security protocols can be added by implementing separate software components for each specific channel or protocol. Only minor changes are required to the existing software.

Overall, the system design is very general. The architecture is not restricted to any particular ad hoc environment or location-limited channel type, and the system can be used with different type of applications.

6.2.4 Simplicity

The designed pre-authentication protocols are relatively straightforward. What makes the security initiation system complex, is the variety of the different location-limited channel settings and the number of the needed pre-authentication protocols.

The software architecture is also quite simple and easy to understand. The responsibilities

between the different software components are clearly divided and the interactions between the different components are well defined.

The downside of the software architecture is the fact that the architecture is partially described on very high level. For example the description of the Security Protocol Component only states that this component should configure the secure protocol with the initiated security association. Implementing a Security Protocol Component on the basis of this description is far from trivial task.

6.3 Evaluation of implementation

The implementation is intended to be a proof of concept, not a ready product. Therefore, no exhaustive testing or performance evaluation were carried out. Instead, in this evaluation we concentrate on the security and usability aspects of the implementation.

Infrared was chosen as the implemented location-limited channel technology. The communication in infrared channel is typically restricted to few meters and the infrared channel is receiver location-limited. Using infrared as the location-limited channel bearer does not provide the best possible security. Eavesdropping the communication is be possible within meters and sending illegitimate messages is possible from distance. For these reasons, no secret keys should be transferred using the infrared channel, and the implementation should always check, that only one location-limited message is received.

Our implementation does not check, that only one location-limited channel message is received. This is serious defect, since the infrared in only receiver location-limited.

All the implemented pre-authentication protocols are based on public key cryptography. This means, that eavesdropping the infrared channel message exchange does not benefit a possible attacker. When shared secrets are needed, they are transferred encrypted on the complementary channel.

The implementation configures IPsec connections. The IPsec connections are established by manually loading the IPsec security associations into the kernel of the operating system. One defect of this scheme is the fact that the same keys are used for the whole connection. Typically, it is good idea to periodically refresh the keys. Another defect is the fact that the established IPsec connections are bound to the IP addresses of the devices. If one of the devices changes its IP address during the connection lifetime, the IPsec connection does not work any more. Especially mobile phones that access Internet using GSM data or GPRS are likely to get different IP addresses every time they access the network.

The usability of the infrared technology is somewhat worse than with most of the other wireless location-limited channel technologies. The two devices must be positioned so that the infrared ports are against each other in order to establish an infrared connection. This is not very easy, especially with devices that have infrared ports in awkward places. A more usable solution would be to use a location-limited channel that is not directed. The devices would only have to be brought close to each other and the user would not have to worry about the alignment of the devices.

The implemented user interfaces are quite simple and easy to use. The user interfaces are constructed from standard Symbian Series 60 user interface components, and therefore a user who is accustomed to use other Symbian Series 60 applications, should not have any

problems with our security initiation system user interfaces.

6.4 Applications for security initiation

Section 3.2 presented three security initiation examples. These examples illustrated how the security initiation system could set up secure connections in ad hoc environments. In this section we will look at some alternative usage models for location-limited channel based security initiation system.

6.4.1 Secure email

One good application for the security initiation system would be secure email. Many would wish to protect the emails they send to their friends or business associates, but very few actually do that. The reason for this is simple: sending secure email is too difficult for normal people.

There are two methods for email protection that have gained some popularity. The first method is to acquire certificates from a mutually trusted certification authority and install these certificates to the used email clients. This is way too difficult, since most of the normal users do not even understand what certificates are. In addition to that, the acquisition of certificates costs money and it typically does not even provide authentication, since the certification authorities do not verify the identities of certificate applicants properly.

The second method is to use PGP. To protect the sent emails with PGP, the user must exchange public keys with all the people he wants to exchange mail with, and in addition, the public key exchanges must be verified for example by checking the fingerprints on telephone. This is even more cumbersome than the use of certificates.

The security initiation system could be used to protect the email traffic in more usable manner. We will clarify this with an example. Let us assume that two business men meet in some conference. They want to exchange email addresses, and they do this by bringing their mobile phones close to each other. The security initiation systems on the phones exchange public keys with the email addresses.

When the business men get back to work they touch their computers with their phones. The security initiation systems on the computers generate certificates from the public keys and configure the address books with the email addresses and the certificate stores with the generated certificates. The business men can now send protected email to each other, and they do not have to understand any of the underlying security details.

This same approach could be applied to any other peer-to-peer communication. For example instant messaging or voice over IP communication could be secured using the same approach.

6.4.2 Webs of trust

The group security initiation protocol GPK2 defines a method for transferring keys of all group members to the one joining members. Using this protocol the security initiation system could also build PGP like webs of trust.

Let us consider the following scenario. Two friends meet at the street and they both have a mobile phone with them. The friends trust each other and they exchange their public keys by bringing their phones together. In addition to the public keys of the two friends, the phones could also exchange all the other public keys that they have previously received via the secure location-limited channel. This means that when deciding to trust a certain person, one also trusts all the people that are trusted by that person.

6.4.3 Connections without security

The security initiation system could also be useful when setting up connections that do not necessarily need any security. Typically creating the connection requires that the user types the address of the other device to the application. This address is often not known by the user and he has to ask the other user for the address. Some times even the other user does not know his own address and he has to start figuring out what is his address so that he can tell it to the other user.

The security initiation system could make this whole process much more simple. One of the users just selects the application from the security initiation system menu and touches the other device. The security initiation system takes care of configuring all the connection parameters.

6.5 Comparison with related work

The Resurrecting Duckling model of Stajano and Anderson and the work of Balfanz et al. are the most relevant work done in this field. Stajano and Anderson introduced the idea of using a physically secure connection for bootstrapping security in wireless ad hoc environments. Balfanz et al. extended this work by presenting better terminology and introducing the usage of public keys and wireless location-limited channels.

This thesis has extended the work of Balfanz et al. We have presented a more through analysis on the different properties of the location-limited channels, identified new peer-to-peer and group models for using location-limited channels in security initiation, and designed new pre-authentication protocols that can be used in these new security initiation scenarios. In addition, while Balfanz et al. designed only protocols for pre-authenticating security protocols, we have designed a complete application based security initiation system with real user interfaces and implemented a prototype of this system.

Compared to Bluetooth pairing, our security initiation process is more user friendly. The user does not have to type in PIN codes or do any other difficult tasks. In Bluetooth pairing the security of the initiated connection depends on length and randomness of the PIN code. In our solution the user must verify that the right peer is sending the location-limited channel message which is much more intuitive.

6.6 Future work

This work could be continued in many ways. Support for new location-limited channels, pre-authentication protocols and security protocols could be implemented. The user inter-

faces could be refined to make the security initiation process even more user friendly. Or perhaps even totally different alternatives for the usage of location-limited channels could be considered. This section presents some ideas for future development.

6.6.1 RFID support

The current implementation provides support only for infrared location-limited channel. As discussed above, the security of infrared is imperfect because the communication range of infrared is quite long. A better solution would be to use some other location-limited channel technology with shorter communication range.

RFID is a very promising location-limited channel technology. Low power RFID communication is typically restricted to short range which means that in practice the user has to touch the other device when initiating a secure connection. This means better security, and in addition, touching the RFID tag with the RFID reader is more intuitive for the user than trying to place the devices so that an infra-red channel can be established.

The RFID tags and readers are also becoming increasingly popular. In near future RFID tags and readers are expected to become embedded into various devices such as PDAs and mobile phones. For these reasons we are planning to implement an RFID location-limited channel and the required pre-authentication protocols for RFID based security initiation.

6.6.2 Automatic keying with IPsec

The current implementation creates the IPsec connections only using manual keying. A better approach would be to use some automatic key negotiation protocol such as IKE. IKE would refresh the keys automatically and help with the problem of changing IP addresses. When using manual keying the end points of the IPsec connection must be presented with IP addresses. If the security association were negotiated using IKE, domain names could be used instead of IP addresses and IKE would take care of resolving the IP addresses that correspond the domain names. This is naturally possible only in environments where DNS service is available.

6.6.3 Automated Bluetooth pairing

In the current implementation the only configured security protocol is IPsec and thus the only supported communication protocol is IP. Although IP is becoming increasingly popular in ad hoc environments, not all the communication is IP based. Bluetooth is also very popular ad hoc networking technology and the security initiation system would be more useful if it could also set up secure Bluetooth connections.

In the normal Bluetooth pairing procedure the users of the devices type in a short PIN code to both of the devices and the devices generate the shared secret using that PIN code. The security initiation system could be used to generate a sufficiently long and random PIN code and it could be transferred to the other device using a selected location-limited channel. After that the security initiation system would set up the secure Bluetooth connection with the shared PIN code.

6.6.4 Security initiation with service discovery

We are also planning to integrate our security initiation system into some service discovery system. In the current design, the user selects the application he wants to use from a list that is based on the locally supported applications. The other party might not support this application and the connection initiation request would fail. If the security initiation system was integrated to some service discovery system, the information about the remotely supported applications could come from the service discovery middleware.

Let us consider the following scenario. The traveler walks into the airport waiting lounge. The service discovery system reports that printing service is available. If the user wants print the document securely, he touches the printer and our security initiation system initiates and creates the secure connection.

In the current design, the user is also responsible for selecting the used location-limited channel. If the security initiation system was used with some service discovery system, the information about the remotely supported location-limited channel alternatives could come from the service discovery system, and the best possible location-limited channel could be selected automatically without any required user interactions.

Chapter 7

Conclusion

This chapter presents a summary and conclusions of this work. In this thesis we have addressed the problem of user friendly security initiation in ad hoc environments. The ad hoc environments are in many ways different from traditional network environments. From the point of view of this thesis, the most relevant differences are the lack of support infrastructure and the different usage model.

In ad hoc environments we are often talking to peers with no prior security context, such as shared keys or mutually trusted certificates. The existing security initiation methods, such as Bluetooth pairing, have proved to be either insecure or difficult for the user. What is needed, is a user friendly method for initiating security between strangers.

Our solution is based on the use of location-limited channels, a technique originally introduced by Stajano and Anderson in [59] and further developed by Balfanz et al. in [16] The purpose of this thesis has been to design a user friendly location-limited channel based security initiation system and to implement a prototype of the system.

In Chapter 3 we presented the high level goals for the security initiation system. In this chapter we also identified the lower level security, usability and generality requirements.

Chapter 4 contained the core of this thesis. This chapter was divided into three parts. In the first part we presented an analysis on the properties of location-limited channels and different location-limited channel technologies. In the second part we identified different ways for using the location-limited channels in security initiation and defined a set of location-limited channel based pre-authentication protocols. Both peer-to-peer connections and group scenarios were considered separately. In the last part we designed the actual security initiation processes and presented the software architecture for location-limited channel based security initiation system.

In Chapter 5 we presented the implementation work. Symbian OS was selected as the main implementation platform, infrared as the location-limited channel technology and IPsec as the supported security protocol. The implementation covered only a selected subset of designed protocols and features, but nonetheless it proved, that the designed security initiation system can be implemented.

In Chapter 6 we evaluated the design according to the evaluation criteria and the lower level requirements. In addition to that, we also compared our solution with related work and discussed future development. The design fulfilled most of the lower level requirements and our security initiation system was considered to be more user friendly and more secure

compared to alternative solutions.

As a conclusion, this work has shown, that location-limited channels provide an intuitive way of bootstrapping security in ad hoc environments. No support infrastructure is needed, which makes the use of proximity based security initiation an attractive solution for many different scenarios.

Bibliography

[1] ECMA International Website. http://www.ecma-international.org/.

[2] FreeS/WAN project: Home page. http://www.freeswan.org/.

[3] Infrared Data Association. http://www.irda.org/.

[4] NTRU GenuID. http://www.ntru.com/products/genuid.htm.

[5] The Official Bluetooth Membership Site. https://www.bluetooth.org/.

[6] Universal Serial Bus home page. http://www.usb.org/home.

[7] The Working Group for Wireless LANs. http://grouper.ieee.org/groups/802/11/.

[8] Data Encryption Standard (DES). FIPS PUB 46-3, U.S. Department of Commerce/National Institute of Standards and Technology, October 1999.

[9] Radio Frequency Identification (RFID) - A basic primer. White paper, Automatic Identification Manufacturers, September 1999. Available from http://www.aimglobal.org/technologies/rfid/resources/papers/rfid_basics_primer.htm.

[10] Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Standard 802.11, ANSI/IEEE, 1999.

[11] Near Field Communication. Technical Report TC32-TG19, ECMA, December 2003. Available from http://www.ecma-international.org/activities/Communications/2003tg19-012.pdf.

[12] Wireless medium access control (MAC) and physical layer (PHY) specifications: Medium access control (MAC) security enhancements. Standard 802.11i Draft 7.0, ANSI/IEEE, October 2003. Work in Progress.

[13] Jari Arkko and Pekka Nikander. How to authenticate unknown principals without trusted parties. In *Proceedings of Security Protocols Workshop 2002, Cambridge, UK*, April 16-19 2002.

[14] N. Asokan and Philip Ginzboorg. Key agreement in ad-hoc networks. *Computer Communication Review*, 23, November 2000.

[15] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik. Authenticated group key agreement and friends. In *Proceedings of the 5th ACM Conference on Computer and Communications Security, San Francisco, CA*, November 1998.

[16] Dirk Balfanz, Diana Smetters, Paul Stewart, and Hao Chi Wong. Talking to strangers: Authentication on ad-hoc wireless networks. In *Symposium on Network and Distributed Systems Security, San Diego, California, USA*, February 2002.

[17] Mark Baugher, Ran Canetti, Lakshminath Dondeti, and Fredrik Lindholm. Group key management architecture. Internet draft, IETF MSEC Working Group, February 2002. Work in Progress.

[18] Steven Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, California, USA*, May 1992.

[19] Stefan Brands and David Chaum. Distance-bounding protocols (extended abstract). In *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway*, May 1993.

[20] Raffaele Bruno, Marco Conti, and Enrico Gregori. Wlan technologies for mobile ad hoc networks. In *Proceedins of 34th Hawaii International Conference on System Sciences, HICSS 34, Maui, Hawaii*, January 2001.

[21] Laurent Bussard and Yves Roudier. Authentication in ubiquitous computing. In *Proceedings of UbiComp 2002: Ubiquitous Computing, 4th International Conference, Göteborg, Sweden*, September 2002.

[22] Nancy Cam-Winget, Russ Housley, David Wagner, and Jesse Walker. Security flaws in 802.11 data link protocols. *Communications of the ACM*, 46(5), May 2003.

[23] Catharina Candolin. Security issues for wearable computing and Bluetooth technology. Manuscript, October 2000. Available from http://www.tml.hut.fi/~candolin/Publications/BT/btwearable.pdf.

[24] Tim Dierks and Christopher Allen. The TLS protocol version 1.0. RFC 2246, IETF, January 1999.

[25] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, (6), November 1976.

[26] Laura Marie Feeney, Bengt Ahlgren, Assar Westerlundy, and Adam Dunkels. Spontnet: Experiences in configuring and securing small ad hoc networks. In *Proceedings of The Fifth International Workshop on Network Applicances (IWNA5), Liverpool, UK*, October 2002.

[27] Simon Garfinkel. *PGP: Pretty Good Privacy*. O'Reilly and Associates, November 1994.

[28] Christian Gehrmann, Chris Mitchell, and Kaisa Nyberg. Manual authentication for wireless devices. *Cryptobytes*, 7(1), 2004. Forthcoming.

[29] Stephen Gill. PGP key verification. Technical report, qOrbit Technologies, August 2002. Available from http://www.qorbit.net/documents/pgp-key-verification.pdf.

[30] Philip Ginzboorg. Personal communication.

[31] Thomas Hardjono and Brian Weis. The multicast group security architecture. Internet draft, IETF, January 2004. Work in progress.

[32] Dan Harkins and Dave Carrel. The Internet Key Exchange (IKE). RFC 2409, IETF, November 1998.

[33] Kipp Hickman and Taher Elgamal. The SSL protocol. Internet draft, IETF, September 1995. Work in Progress.

[34] Lars Erik Holmquist, Friedemann Mattern, Bernt Schiele, Petteri Alahuhta, Michael Beigl, and Hans-Werner Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *Proceedings of Ubicomp 2001: Ubiquitous Computing, Third International Conference Atlanta, Georgia, USA*, September 2001.

[35] Russell Housley, Tim Polk, Warwick Ford, and David Solo. Internet X.509 public key infrastructure certificate and certificate revocation list CRL profile. RFC 3280, IETF, April 2002.

[36] Mike Just and Serge Vaudenay. Authenticated multi-party key agreement. In *Advances in Cryptology - ASIACRYPT '96, International Conference on the Theory and Applications of Cryptology and Information Security, Kyongju, Korea*, November 1996.

[37] Stephen Kent and Randall Atkinson. IP Authentication Header. RFC 2402, IETF, November 1998.

[38] Stephen Kent and Randall Atkinson. IP Encapsulating Security Payload (ESP). RFC 2406, IETF, November 1998.

[39] Stephen Kent and Randall Atkinson. Security architecture for the Internet Protocol. RFC 2401, IETF, November 1998.

[40] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Simple and fault-tolerant key agreement for dynamic collobarative groups. In *Proceedings of 7th ACM Conference on Computer and Communications Security, Athens, Greece*, November 2000.

[41] Tim Kindberg and Kan Zhang. Context authentication using constrained channels. Technical Report HPL-2001-84, Hewlett-Packard Laboratories, April 2001.

[42] Tim Kindberg and Kan Zhang. Validating and securing spontaneous associations between wireless devices. Technical Report HPL-2002-256, Hewlett-Packard Laboratories, September 2002.

[43] John Kohl and Clifford Neuman. The Kerberos Network Authentication Service (V5). RFC 1510, IETF, September 1993.

[44] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-hashing for message authentication. RFC 2104, IETF, February 1997.

[45] Sami Lais. Optical character recognition. *Computerworld*, July 2002.

[46] Christina Lopes and Pedro Aguiar. Aerial acoustic communications. In *Proceedings of the 2001 IEEE Workshop on Applications of Signal Processing to Audio and Acousitcs, New Pfaltz, New York, USA*, October 2001.

[47] Daniel McDonald, Craig Metz, and Bao Phan. PFKEY Key Management API, version 2. RFC 2367, IETF, July 1998.

[48] Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Carlisle Adams. X.509 Internet public key infrastructure online certificate status protocol - OCSP. RFC 2560, IETF, June 1999.

[49] Charles Perkins. *Ad Hoc Networks*. Addison Wesley, December 2000.

[50] Local Wireless Communication project team. Bluetooth threats and security measures. Technical report, Bundesamt für Sicherheit in der Informationstechnik, 2003. Available from http://www.bsi.de/english/brosch/B05_bluetooth.pdf.

[51] Matthias Ringwald. Spontaneous interaction with everyday devices using a pda. September 2002.

[52] Ronald Rivest. The MD5 message-digest algorithm. RFC 1321, IETF, April 1992.

[53] Sanjay Sarma, Stephen Weis, and Daniel Engels. Radio-frequenct identification: Security risks and challenges. *CryptoBytes*, 6(1), 2003.

[54] Markku Savela. Personal communication.

[55] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, 2nd edition, 1996.

[56] Jean-Marc Seigneur, Stephen Farrell, and Christian Damsgaard Jensen. Secure ubiquitous computing based on entity recognition. In *Proceedings of UbiComp 2002: Ubiquitous Computing, 4th International Conference, Göteborg, Sweden*, September 2002.

[57] Frank Stajano. The resurrecting duckling – what next? In *Proceedings of the 8 th International Workshop on Security Protocols, Lecture Notes in Computer Science, Springler-Verlag, Berlin, Germany*, April 2000.

[58] Frank Stajano. *Security For Ubiquitous Computing*. John Wiley and Sons, February 2002.

[59] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for adhoc wireless networks. In *Proceedings of the 7 th International Workshop on Security Protocols, Lecture Notes in Computer Science, Cambridge, UK*, April 1999.

[60] William Stallings. *Cryptography and Network Security*. Prentice Hall, 2nd edition, 1999.

[61] Martin Tasker. Active objects. Technical Report Revision 1.0, Symbian OS, May 1999. Available from http://www.symbian.com/developer/techlib/papers/tp_active_objects/active.htm.

[62] Stuart Williams, David Suvak, Paul McClellan, and Frank Novak. Tiny TP: A flow-control mechanism for use with IrLMP. Technical Report 1.1, Infrared Data Association, October 1996. Available from http://www.irda.org/standards/pubs/Tinytp11.PDF.

[63] Thomas Wu. The Secure Remote Password protocol. In *Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium, San Diego, CA*, March 1998.

[64] Ka-Ping Yee. User interaction design for secure systems. In *Proceedings of 4th International Conference on Information and Communications Security, ICICS 2002, Singapore*, December 2002.

[65] Tatu Ylönen, Tero Kivinen, Markku-Juhani Saarinen, Timo Rinne, and Sami Lehtinen. SSH protocol architecture. Internet draft, IETF, September 2003. Work in Progress.

[66] Phil Zimmerman. *The Official PGP User's Guide*. MIT Press, May 1995.