NTNU
Innovation and Creativity

Using GSM SIM Authentication in VPNs

Torstein Bjørnstad

Master of Science in Communication Technology
Submission date: May 2007
Supervisor: Van Thanh Do, ITEM
Co-supervisor: Ivar Jørstad, Ubisafe AS

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Until now, GSM SIM authentication has mostly been used in the authentication process of native mobile telecommunication services like voice telephony and SMS. Solutions for SIM-based authentication of Wireless LAN based on EAP-SIM are currently emerging. However, it is possible to reuse the same authentication mechanism for any type of distributed, Internet-based service. The goal of this thesis is to continue the work performed in a student project and complete the design of a solution which allows a user to employ the standard GSM SIM authentication mechanism toward a Virtual Private Network (VPN), thereby achieving three important goals:

- Simplifying the authentication process for the user
- Making it easier for VPN administrators/operators to increase the strength of the VPN authentication procedure, and
- Reducing administrative costs of the VPN.

Assignment given: 17. January 2007
Supervisor: Van Thanh Do, ITEM

Abstract

With the growth of the Internet a lot of different services has emerged. These services are often accompanied by some kind of security system. Since most of these services are stand-alone systems, a whole range of different authentication systems have been developed. Each using one of several kinds of authentication, with one or more proofs of identity. The SIM card used in mobile phones is an identifying token, containing strong authentication mechanisms. If services could utilize the SIM for authentication it would provide both a more secure solution, in addition to increased simplicity for the user.

This master thesis builds on a project that investigated how the security properties of a system can be improved by adding an extra factor to the authentication process – something the user has, or more specifically the GSM SIM card. That project concluded by suggesting an overall design for a VPN Authentication System based on the security mechanisms in GSM. This thesis continues that work by analyzing that design, and describing the implementation of a prototype utilizing the mechanisms available.

Preface

This Master's thesis has been written as a continuation of a project performed in cooperation with Telenor R&I during the 9th semester, autumn 2006. It is part of the Master of Technology program at the Norwegian University of Science and Technology, Faculty of Information Technology, Mathematics and Electrical Engineering. It was written in the 10th semester, spring 2007, at the Department of Telematics, with help from Telenor R&I.

I would like to thank my supervisor Do van Thanh at Telenor R&I and co-supervisor Ivar Jørstad at Ubisafe AS for valuable help and support during my work with this thesis. Their comments and suggestions, especially during the final weeks, were highly appreciated.

Trondheim, May 31st 2007

Torstein Bjørnstad

Contents

List of Figures

List of Tables

Abbreviations

AES	Advanced Encryption Standard
AKA	Authentication and Key Agreement
APDU	Application Protocol Data Unit
API	Application Programming Interface
ATR	Answer To Reset
AuC	Authentication Center
BSC	Base Station Controller
BSS	Base Station System
BTS	Base Transceiver Station
CA	Certificate Authority
CDMA	Code Division Multiple Access
CHV	Card Holder Verification
CPE	Customer Provided Equipment or Customer Premises Equipment
CRC	Cyclic Redundancy Check
DSL	Digital Subscriber Line
EAP	Extensible Authentication Protocol
EIR	Equipment Identity Register
ESP	Encapsulating Security Payload
HLR	Home Location Registry
ICC	Integrated Circuit Card
iDEN	Integrated Digital Enhanced Network
IKE	Internet Key Exchange
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
IrDA	Infrared Data Association
ISAKMP	Internet Security Association and Key Management Protocol
LAN	Local Area Network
ME	Mobile Equipment
MPLS	Multi Protocol Label Switching
MS	Mobile Station
MSC	Mobile Switching Center
OSA	Open Systems Authentication

PC	Personal Computer, a laptop or stationary computer)
PCMCIA	Personal Computer Memory Card International Association
PDA	Personal Data Assistant
PIN	Personal Identification Number
PKI	Public Key Infrastructure
QOS	Quality Of Service
RF	Radio Frequency
SA	Security Association (IPSec)
SAP	SIM Access Profile
SKA	Shared Key Authentication
SLA	Service Level Agreement
SRES	Signed Response
SS	Switching System
TLS	Transport Layer Security
TMSI	Temporary Mobile Subscriber Identity
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
USIM	Universal Subscriber Identity Module
VLR	Visitor Location Registry
VPN	Virtual Private Network
WAN	Wide Area Network
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network

Chapter 1

Introduction

1.1 Motivation

The past few years, the way people connect to the Internet has changed. For several years, dial-up connections have been the most common way of connecting to company networks, but as people are getting fixed line connections such as xDSL and Cable at home, the use of dial-up connections is declining. This introduces an increased need of securing the connection, as the data path from the user to the company network goes through the public Internet. This is where Virtual Private Networks, or VPNs, come in. VPNs have been in use for several years, but mostly for connecting geographically separated sites.

As more users now have to create their own VPN connections from their home computer to the company network, several approaches have been investigated to be able to provide a more user-friendly, but still secure way of authenticating the users.

The use of smart cards has proven to be a good solution to the authentication problem, but this usually requires the users to obtain a new type of device. A better solution is to look at authentication mechanisms already available to the users, and find ways to enable the use of these mechanisms for authentication in other solutions. The SIM card in the mobile phone is also a smart card, and if VPNs could use the SIM authentication mechanisms for authenticating users, it would be both convenient and more secure than just having a password. The system has many existing users, and the administration techniques are well defined. Also, a single token to authenticate the users for several systems make things easier for the users. SIM is already being used in some WLAN and application authentication schemes. The main motivations for using the SIM as an authentication token are:

1. Many existing users (large user base)
2. Established way of deploying tokens (simple administration)
3. User friendly (own device)

The goal of this thesis is to propose a design for a system that allows the authentication of VPN clients based on information contained in the SIM card, as well as develop an example implementation of such a system. The thesis builds on the work done in a

student project by the author, where an overall architecture of a VPN authentication system based on GSM was proposed.

1.2 Problem Definition

The following problem definition has been formulated in cooperation with the supervisor:

Until now, GSM SIM authentication has mostly been used in the authentication process of native mobile telecommunication services like voice telephony and SMS. Solutions for SIM-based authentication of Wireless LAN based on EAP-SIM are currently emerging. However, it is possible to reuse the same authentication mechanism for any type of distributed, Internet-based service. The goal of this thesis is to continue the work performed in a student project and complete the design of a solution which allows a user to employ the standard GSM SIM authentication mechanism toward a Virtual Private Network (VPN), thereby achieving three important goals:

1. Simplifying the authentication process for the user
2. Making it easier for VPN administrators/operators to increase the strength of the VPN authentication procedure, and
3. Reducing administrative costs of the VPN.

1.3 Challenges

More and more companies are setting up VPN access servers in order to better control access to their internal networks (intranets). While this makes the job easier for the system administrators as well as improve overall network integrity, it makes it harder for employees to do their job. To maintain the security of the access control mechanism an authentication mechanism with strong security needs to be used. Unfortunately, strong security usually means making things more complicated for the users.

One way of achieving improved security without the need for extra training for the users is to apply the principle of generalization - use already existing and well known systems, and extend them to cover new areas. The SIM used in mobile phones is getting more and more used, and could also be a good way of authenticating VPNs. However, while using the SIM can make life easier for the users, it also introduces a few challenges;

- Computers usually do not come equipped with SIM readers
- Accessing the SIM card functions is complicated
- Validating the user requires cooperation from the network operator
- Connecting other systems to the authentication system

It will be a challenge getting the SIM authentication mechanism to provide authentication information to a VPN system.

1.4 Objectives

The main problem of this thesis is to find out how the GSM SIM and the GSM authentication mechanism can be used to authenticate a user when establishing a VPN tunnel.

The following sub-problems have also been defined:

a. How can an authentication client communicate with the SIM card?

b. How can authentication be performed between this client and an Authenticator/Identity Provider?

c. How can an Authenticator/Identity Provider verify the identity of the User?

c. How can the VPN server verify the authentication status of the User?

In order to complete the task at hand, solutions to this set of problems have to be found.

1.5 Related Work

Different systems have for a long time been using system-specific authentication solutions. The past few years people have started to focus more on the usability of the systems, realizing that more passwords not exclusively give better security [RSA05]. In addition to this, as more confidential information is communicated across the Internet, higher levels of security are requested.

The use of physical items in authentication has already been in use for a few years. Sun Microsystems introduced their Sun Ray thin client in 1999 [Sun06], which included a smart card reader. By inserting a smart card linked to the user's account, the user is automatically logged onto the system, without getting prompted for a username or password. Similar systems for logging onto Microsoft Windows have also been around since Windows NT 4.0 and Windows 95 [Mic00].

Focus has later shifted to authenticating users for access control to other systems. Several companies now offer solutions that use smart cards for network based authentication. Gemalto[1] offers a security solution using smart cards and USB devices along with an authentication server. This system can be used for authentication both on web sites and network locations. RSA Security offers a similar solution with their Smart Card and Smart Key solutions. Wireless LAN authentication using the 802.1x standard can authenticate using smart card solutions supporting EAP-SIM (which will be described in more detail in section 3.9).

A generic authentication system based on SIM has been presented in a master thesis by Lars Lunde and Audun Wangenberg [LL05], which allows web applications to authenticate users based on the GSM SIM.

[1] Previously Axalto and Gemplus International, before a merger in June 2006

1.6 Methodology

The methodology used in this project can be described as "Design Research". The topic Design Research is explained by the Association for Information Systems in [Vai04], and this information is used in this section. Design Research is a combination of using research, which can be described as *an activity that contributes to the understanding of a phenomenon* combined with design, which means to *invent or bring into being.*

Design Research can be thought of in two ways. First you have the gathering and research of existing knowledge in order to design, and then you have the new knowledge acquired from the design which again can be researched. This is a circular process, as illustrated in figure 1.1.

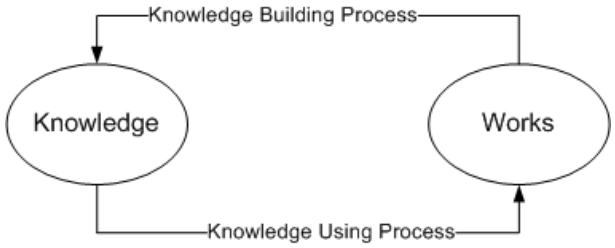

Figure 1.1: The circular process of Design Research.

The whole process can be summed up the in five steps listed below, also illustrated in figure 1.2.

1. One has to *be aware* of a problem, or situation that needs improving or changing.
2. One have to research the area of the problem, in order to *suggest a solution* to how the problem can be solved or mitigated.
3. After the suggestion is made, *development* of the suggested artifact can take place.
4. The developed artifact needs to be *evaluated*, in order to be able to make a proper conclusion.
5. The results from the evaluation provides information to use in a *conclusion.*

This thesis focuses on the three last steps of the process, ending with a summary of the results of the prototype testing.

1.7 Project Outline

A project outline is shown in figure 1.3.

Depending on the familiarity the reader has with the field, three different paths can be chosen. The chapter on Terminology introduces a few relevant areas, and can be skipped if desired. The next two chapters, Background and Existing Systems describe the design and functionality of the systems involved in the VPN GSM SIM authentication system, and an understanding of these chapters are required to understand the last parts of the

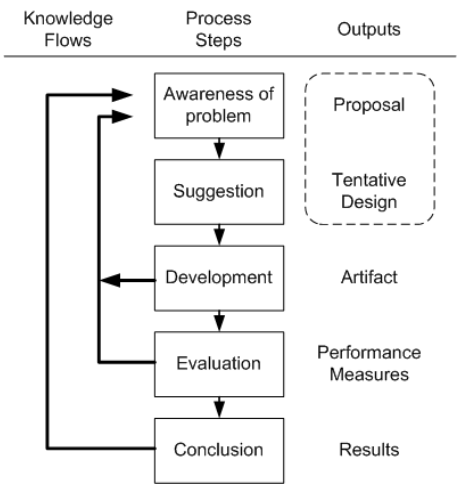

Figure 1.2: The General Methodology of Design Research (from [Vai04]).

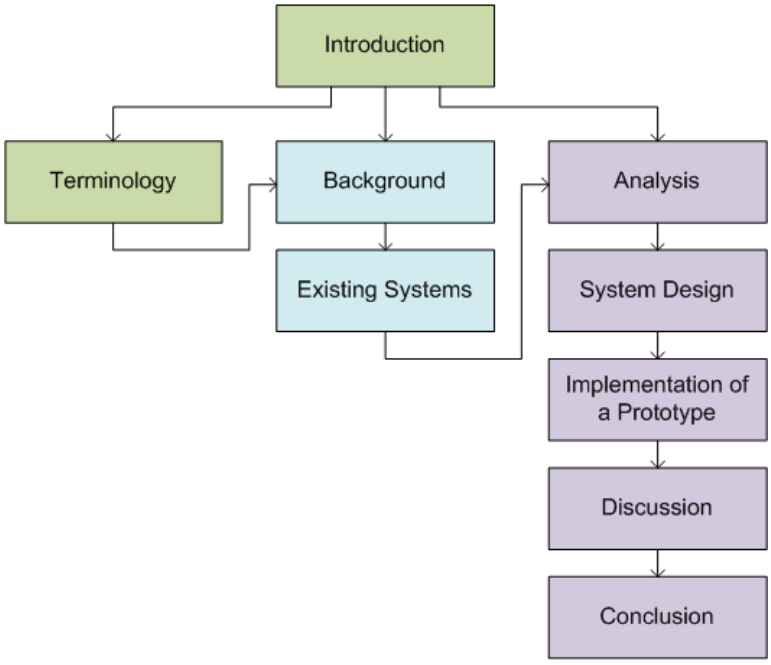

Figure 1.3: An outline of the project

thesis. The last five chapters describe the analysis, design and implementation of the authentication system, and concludes with a discussion of the results and a conclusion.

Chapter 2

Terminology

The goal of this chapter is to introduce a few general terms that will be used throughout this thesis. It will focus mainly on the principles of secure computing, and the requirements of a secure computing system.

2.1 Digital Identities

In today's society, heavily influenced by information technology, computers and the services they offer, the importance of digital identities is increasing. More and more web sites offer various options of customization for people to design their own preferred start pages with news feeds, daily cartoons and other more or less useful content[1]. Much like todays newspaper, only it is made by and for yourself – and it gets updated continuously.

Services like this, along with other personal or personalized services on-line requires some form of identity association. The service provider has to identify you to find out what you should and should not get access to.

The Internet Identity Gang[2] consists of a group of individuals and organizations volunteering with a common mission:

> *To support the ongoing conversation about what is needed for a user-centric identity "metasystem" that supports the whole marketplace*

One of the steps to such a system is to establish a common terminology in the field, so everyone has a common understanding of what is said and done. A list of central concepts are given on their website, in their *lexicon*[The06].

The Internet Identity Gang defines a *Digital Subject* as *An Entity represented or existing in the digital realm which is being described or dealt with.* Continuing, every digital subject is defined as having a finite, but unlimited number of *Identity Attributes.* These identity attributes describe the subject in a way as to be able to identify the person, or subject. Every digital subject also has a set of *Digital Identities*, which is the information used in transactions with different service providers, or *Parties.* These

[1] e.g http://www.google.com/ig
[2] http://identitygang.org

digital identities consist of *Claims* defined by one or more parties, for example saying that that subject has a specific student number. When the user wants to authenticate, one can view it as the digital subject performs a transaction with a *claimant*, using one of it's digital identities. The claimant is a digital subject representing the party that makes a claim. The relations described here are illustrated in figure 2.1.

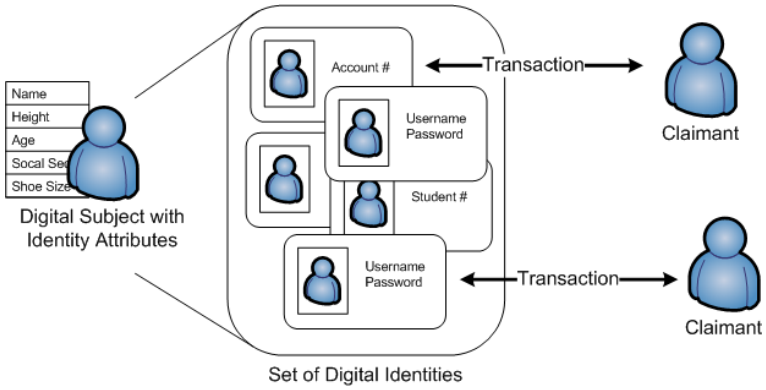

Figure 2.1: Illustration of a Digital Subject and associated Digital Identities

2.2 Cryptography

Cryptography is the art of keeping information secret. For many years, cryptography has been used to provide solutions for the following challenges:

Authentication Ensuring someone is who they claim they are

Confidentiality Protecting data/information from unauthorized parties

Integrity Ensuring that data/information has not been changed or altered in any way

Non-repudiation Ensuring that a contract made cannot be denied by any of the parties

2.2.1 Challenges

Authentication

Authentication is the process of determining the authenticity of something or someone, or whether the claims made about the object are true [Wik06a]. When talking about authentication in computer security, one refers to the process of verifying the digital identity of the person. As mentioned in section 2.1, with most online services today you have a profile identifying you as a user. When you want to access that profile, you need to get authenticated, so the system knows if you are who you claim to be. This way, other users will not get access to your profile. Another situation using authentication is for instance when you want to restrict access to a system, for instance a bank application or personal computer.

Authentication in computing can be performed in several ways. One common method is known as the challenge-response authentication method, and is a system many people

use every day. An example of this is the computer telling you to insert your password, or an ATM asking for your PIN code. It first gives a challenge ("Insert password/pin") and requests a response. After you provide it with your secret key, it lets you into the system. This is based on the fact that only you know the PIN, so unless you have told someone else, the system can be fairly certain you are who you claim to be.

Confidentiality

Confidentiality is the principle of keeping information secret, and many different methods for doing this has been invented through the years. For instance, the Romans shaved the heads of their messengers before tattooing secret messages onto their heads. After the hair had grown back out, they could safely travel to the intended recipient of the message, before shaving off their hair so the message could be read again [Sin99]. Confidentiality in the modern age is mostly achieved through encryption, where messages are concealed by mathematical operations, combined with a secret key. This is explained more in section 2.2.2.

Integrity

Integrity is about knowing whether a message is in its original form or if it has been tampered with on its way to the destination, which is very important in many areas. For instance, you would not want someone else changing the account number when you transfer money between bank accounts. Integrity is often achieved through the same methods as confidentiality, but also usually helped by some form of checksum. A checksum can be viewed as a very short summary of the message transferred, containing small pieces of all the information in the original message. If the message has been changed, it can be discovered by creating a new checksum and comparing the two. Public Key cryptography is often used for integrity checking.

Non-repudiation

When you send a normal letter, that piece of paper can be seen as a proof you actually sent the letter. When it comes to e-mail and other information on the Internet, it is often possible to claim someone else posing as you sent the e-mail or posted the message to the bulletin board. Non-repudiation is a principle meaning that people should not be able to deny sending or agreeing to something. This can also be achieved through Public Key cryptography.

2.2.2 Encryption

Encryption is the actual methods used to solve the challenges presented. It is often based on mathematical principles, or otherwise hard calculations, and usually includes some sort of key in order to have some information that can be shared between the parties.

Symmetric Encryption

The principle of symmetric encryption is that the same key is used for both encryption and decryption. This is why it is also commonly referred to as secret-key, single-key or shared-key encryption. The user who encrypts and the one who decrypts needs to have a common key, so this form of encryption requires some form of key distribution system. This distribution can be performed in three ways:

- Manual distribution, where user A physically delivers the key to user B, or having a third party deliver the keys to both parties.
- Automated distribution, where both user A and B have a connection to a Key Distribution Center
- Public-key encryption, where user A can use the public key of user B to encrypt the secret key

The actual encryption functions will not be described here, but the basic functionality of many of them is that they take the key and the original message as input, and output a decrypted message, looking like a seemingly random sequence of letters and numbers. If someone wants to decrypt the message they need the same key used to encrypt it, along with the decryption function. The encryption and the decryption functions are considered to be known to everyone, so the strength of symmetric encryption lies in the key, and keeping the key secret.

Asymmetric Encryption

This encryption scheme is also referred to as public-key encryption. It is based on a principle where you have key pairs consisting of one private key and one public key. The keys are related mathematically, and messages encrypted using one of the keys can only be decrypted using the other. This means the public key can be made available for everyone, without the encryption losing its strength. Messages encrypted using the public key can only be decrypted by using the private key, ensuring confidentiality, while messages encrypted with the private key can only be decrypted using the public key, ensuring authenticity and integrity. By combining the use of person A's private key and person B's public key when sending a message from A to B, both confidentiality, authenticity and integrity can be ensured. It is possible to sign a message without encrypting the message itself by generating a checksum/hash of the message and then encrypting this instead of the message (thus ensuring integrity and authenticity), and attaching it to the message. This is a common way of using digital signatures [Sch96].

Chapter 3

Background

This chapter will introduce different technologies that are involved in a SIM based VPN authentication system, as well as how the authentication mechanism is performed in different systems. The first part focuses on components in the GSM and UMTS networks, and the rest of the chapter introduces some other relevant areas.

3.1 GSM/UMTS Subscriber Equipment

In the GSM/UMTS network, subscriber equipment consists of a SIM/USIM Card and a Mobile Station, or Mobile Equipment. The two technologies differ on a few points, and these differences will be highlighted in the following sections.

Figure 3.1: The Subscriber Equipment consists of the Mobile Station and a SIM card.

3.1.1 SIM Card

The main purpose of the SIM is to provide a compact and secure storage of the components required for the GSM/UMTS authentication scheme. They are the International Mobile Subscriber Identity (IMSI), the Subscriber Authentication key (Ki) and the authentication and key generation algorithms. The SIM is mostly used in GSM phones, but the modules used in UMTS phones (USIM) and in Integrated Digital Enhanced Network (iDEN) phones are similar.

The SIM is a logical application/module that runs on an Integrated Circuit Card (ICC), but it is commonly referred to as a SIM card. The specification refers to two types of physical characteristics of the SIM card, the "ID-1" and the "Plug-in SIM". The specification of the SIM and the handset was considered part of the GSM standard. The interface ME to SIM is described in GSM 11.11, and GSM 11.14 describes the interface SIM to ME. The newest versions of the specifications are standardized by the 3GPP as TS 51.011 and TS 51.014 [3GP05b] [3GP04].

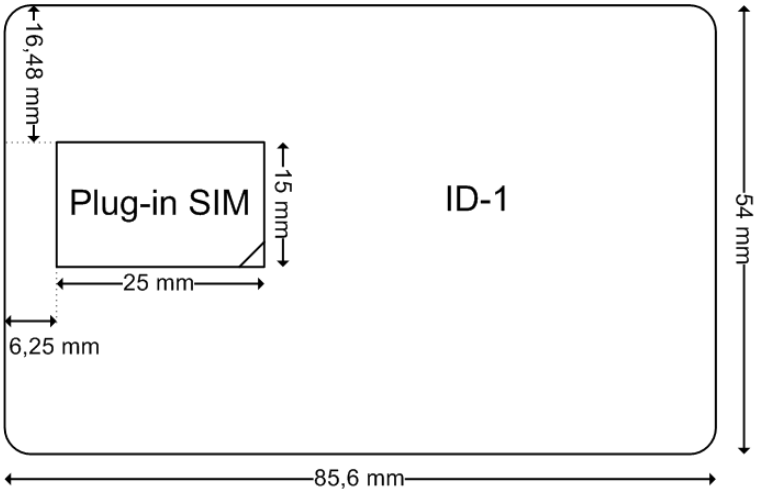

Figure 3.2: An overview of smart card ID-1 and Plug-in SIM dimensions.

The Smart Card on which the SIM is realized is based on microprocessors specifically designed to be tamper-resistant. This means that secret information such as the Subscriber Authentication key and the authentication and key generation algorithms should not be possible to extract from the card. The smart card relies on the reader for its energy, and can, using its on board CPU, calculate and return values based on provided input values. This way, the Subscriber Authentication key is never read, not even by the phone.

Authentication components of the GSM system are, as mentioned, stored in the SIM card. They are as follows:

International Mobile Subscriber Identity (IMSI)

The purpose of the IMSI is to have a unique identifier for every subscriber in the world. This is achieved by a hierarchical organization of the countries, and the mobile operators within that country, as described in the ITU E.212 recommendation, "'The

international identification plan for mobile terminals and mobile users". The IMSI is usually 15 decimal digits long, with the three first digits being the country code, the next two or three digits the network (operator) code, and the remaining part of the number being a unique subscriber number within the specific network [Uni05].

Subscriber Authentication key (Ki)

The Subscriber Authentication key is a randomly generated 128-bit number stored on each SIM card and in the AuC. The key Ki never leaves either of the locations, and authentication of the user is based on a system that checks whether the user has access to Ki. The authentication mechanism in the GSM/UMTS networks are based on challenges that use this key to compute a response. This is described in detail in section 3.2.

Authentication and Key Generation algorithms

As the GSM/UMTS authentication scheme is based on keeping the Subscriber Authentication key secret, a card with computational capability was used in order to perform calculations on the card. The authentication scheme used in the GSM system is based on two algorithms; the A3 algorithm for authentication, and the A8 algorithm for key generation.

The A3 algorithm is a one-way function, with the task of generating the 32-bit Signed Response (SRES) required in the SIM authentication scheme. Its inputs are the 128-bit Subscriber Authentication key (Ki), already stored in the SIM, along with a 128-bit Random Challenge Number (RAND) generated by the HLR in the subscriber's home network. The A8 algorithm is another one-way function, with the task of generating a temporary Cipher key (Kc). This key is used to encrypt phone calls on the radio interface, through the GSM symmetric cryptography algorithm A5. A8 takes the same input parameters as A3.

Since the two algorithms A3/A8 take the same input, they are usually implemented together. The operators choose which type of algorithm is to be implemented on the SIM, but the COMP128 algorithm, developed by the GSM association, is the most common. This algorithm performs the A3 and the A8 algorithm in the same stage, as shown in figure 3.3.

A major flaw in the COMP128 algorithm was that its functionality was not only based on having a secret key. It was also partially dependent on the algorithm being secret. As should have been predicted, the algorithm ended up in the public through a combination of leaked documents and reverse engineering. It has been shown in [JRRT02] how COMP128 can be broken in less than a minute. The algorithm was renamed to COMP128-1, and enhanced variants named COMP128-2 and COMP128-3 have later been developed. They are also partially based on the secrecy of the algorithm.

Card Holder Verification information (CHV) information

Every SIM needs to contain Card Holder Verification (CHV) information to provide protection against unauthorized use. This can also be complemented with a second

CHV (CHV2), used to control access to specific optional features such as call forwarding. Both CHVs are a number consisting of 4 to 8 decimal digits. When subscribers acquire a new SIM, they are usually delivered with preset CHV codes. The CHV is used to authenticate the user to the specific SIM, to hinder the usage of stolen cards. This is both in the interest of the user, who pays the bills, and the operator, who wants to maintain a level of control regarding access to the network. The CHV check can usually be disabled by the user, unless the operator has specified otherwise. The CHV2 check should not be possible to deactivate.
If the CHV check is enabled, the user is prompted for the CHV number (often referred to as a Personal Identification Number (PIN)) when the ME is turned on. The same holds for the CHV2 number when special options limited by the CHV2 are accessed. On correct CHV presentation, the ME performs the requested functions. If an incorrect CHV or CHV2 is presented, the user should be informed of this. After three consecutive incorrect entries the specific CHV is blocked.
Unblocking of blocked CHV numbers can be done by entering CHV Unblocking Keys, also usually delivered with the SIM but not changeable by the user. The SIM contains both a CHV and a CHV2 Unblocking Key, used to unblock the relevant CHV. It should not be possible to read the CHV or the Unblock CHV numbers.

3.1.2 USIM Card

The USIM is the UMTS equivalent of the GSM SIM card, as it has the same purpose in UMTS networks as the SIM has in the GSM network. The USIM is equipped with hardware capable of performing the algorithms required for the AKA algorithm (described in 3.4), and it has the same dimensions and basically the same functionality. The requirements for USIM is described in TS 31.102 [3GP05a].

3.1.3 Mobile Station (MS)

This is the actual mobile equipment used by the user. This is usually a mobile phone, but could also be a Personal Data Assistant (PDA) or a Personal Computer (PC). A special requirement of the two latter is that they are equipped with SIM reading capabilities. The purpose of the MS is to provide a radio interface for communication between the SIM and the network, and also the user and the network.

3.2 GSM Authentication

The GSM authentication procedure is the first authentication system that will be described. All the required components for this authentication system has already been described in the previous chapter, but this section will focus on the order of the various messages, and which messages are communicated where. The authentication protocols are described in more detail in 3GPP TS 43.020 [3GP06].

As already described in sections 3.1.1 and 3.3.1, the authentication mechanism in GSM is based on a unique subscriber identity value (IMSI) and a secret key (Ki) stored only in the SIM Card and in the AuC in the home network. The order of the authentication messages are as follows:

1. First the user has to be identified in the currently serving network. This happens by transmitting the IMSI stored in the SIM to the VLR or SGSN. If the user has an established TMSI with the VLR, this is used instead.
2. As the user is not currently identified as a valid user, the VLR/SGSN has to contact the user's AuC, which may reside in another network, and request authentication triplets.
3. The AuC generates a set of authentication triplets (1-5 triplets), consisting of a random number (RAND) an expected response (RES) and a temporary encryption key (Kc). The encryption key is generated from the permanent key Ki and the RAND value. These values are sent back to the VLR/SGSN.
4. As the VLR/SGSN receives the triplets, it selects one of the triplets, and sends the RAND value to the ME. This is the challenge part of the GSM authentication system.
5. When the ME receives the RAND, a function is called on the SIM to generate a response using that random number.
6. The SIM performs the A3 algorithm, with RAND and Ki as input, and returns the calculated signed response (SRES) to the ME (The A8 algorithm is also called, to generate Kc. This is described in section 3.1.1). This is illustrated in figure 3.3.
7. The ME sends SRES to the VLR/SGSN, which compares the calculated SRES value with the SRES contained in the selected authentication triplet. If the two values match, the ME is granted access to the network. If the two values do not match, the ME is denied access.
8. If the ME is granted access, a new TMSI value is sent form the VLR/SGSN to the ME, to be used for later signaling requiring a IMSI/TMSI to be used.

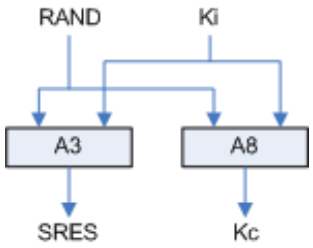

Figure 3.3: Generating the SRES and Kc in GSM

3.3 GSM/UMTS Network Components

3.3.1 GSM/UMTS Switching System (SS)

The Switching System encompasses the main components providing the core functionality of the GSM/UMTS networks. [GSM03]

Home Location Registry (HLR) The Home Location Registry is a database owned and maintained by the mobile operator. It stores data about subscribers of that

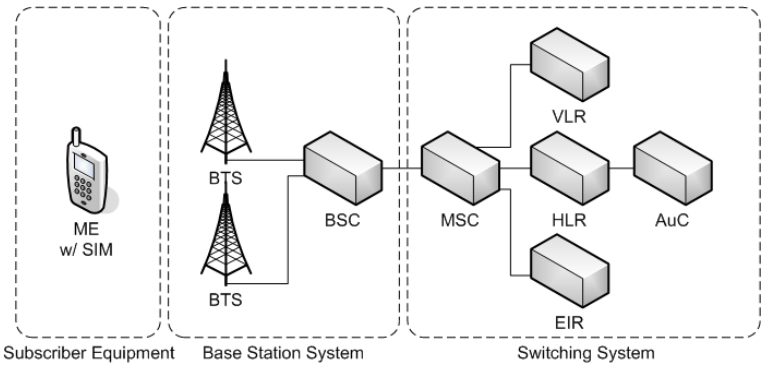

Figure 3.4: The GSM network components and their connections

particular operator, linking subscriber information like the IMSI, the subscribers calling number and the associated authorizations. It also has a reference to where in the network the subscriber is located, so the customer can be reached when roaming.

Authentication Center (AuC) The Authentication Center plays a major part of the authentication procedure in GSM/UMTS networks. It contains the IMSI of the SIMs, along with the corresponding Authentication Key and encryption algorithms (A3 and A8, described in 3.1.1). Its task is to provide the MSC with so-called authentication triplets (RAND, RES, Kc), which are needed to authenticate the user. The AuC is often implemented together with the HLR.

Mobile service Switching Center (MSC) The Mobile service Switching Center performs the telephony switching functions of the GSM network.

Visitor Location Registry (VLR) The Visitor Location Registry is a database like the HLR, and it stores much of the same information as the HLR. Instead of storing information on the mobile operator's own subscriber, it keeps track of roaming subscribers and stores temporary information about them. It assigns a Temporary Mobile Subscriber Identity (TMSI) to the subscribers currently in the network.

Equipment Identity Register (EIR) The Equipment Identity Register is a database of International Mobile Equipment Identities (IMEI), and is used to prevent calls being made from stolen, unauthorized or defective mobile stations. If a mobile phone is registered stolen or missing, its IMEI is tagged in the database and is effectively rendered unusable.

3.3.2 GSM/UMTS Base Station System (BSS)

At the border of the GSM networks we find the Base Station Systems. They provide the interface between the Mobile Station and the core network components. The BSS consists of two parts in the GSM system, these are the Base Transceiver Station and the Base Station Controller.

Base Transceiver Station (BTS) The Base Transceiver Station is for the users the most visible component of the GSM/UMTS system. They are the antennas

mounted on buildings and radio masts which provide radio coverage for the network, and which the Mobile Stations connect to.

Base Station Controller (BSC) The Base Station Controller provides the physical links between the MSC and the BTSs. It provides additional functions like hand overs, cell configuration data and control of radio frequency (RF) power levels in BTSs.

3.4 UMTS Authentication

The basic functionality of the authentication in UMTS is similar to as in GSM, with a few exceptions. UMTS also bases its security on a master key K, shared between the USIM and the AuC. The length of this is 128 bits. The authentication mechanism in UMTS is called the Authentication and Key Agreement (AKA) protocol, and will be described in more detail below. The information in this section is based on [NN03].

AKA was designed by combining the GSM authentication and key agreement mechanism (described in TS 43.020 [3GP06]), and an ISO standard for authentication mechanism based on sequence numbers.

1. Initially the user has to identify itself, and this is used by sending the IMSI, the TMSI or the Packet TMSI to the VLR or SGSN.
2. The VLR or SGSN then sends an authentication data request, containing the IMSI, to the AuC/HLR in the home network.
3. Based on the IMSI, the AuC generates authentication vectors containing a random number (RAND), an authentication token (AUTN), an expected response (XRES) and a cipher key (CK). The cipher key is derived from the permanent key K and the RAND. These vectors are then returned to the VLR/SGSN in the authentication data response.
4. The VLR/SGSN chooses a vector to use, and stores the others for later use. Then it sends RAND and AUTN to the ME.
5. When the ME receives the values, it calls functions on the USIM, to calculate the values RES, CK, IK and XMAC as illustrated in figure 3.5. As an extra security measurement, this algorithm gives the USIM the ability to verify if the AUTN value was generated in the AuC, and that the authentication procedure is not part of a replay attack.
6. If the AUTN parameter is valid, the ME sends the RES back to the VLR/SGSN, which compares the calculated RES value to the expected response XRES. If they match, the ME is authenticated, and can be granted access to the network.

3.5 WLAN Authentication using SIM

Local Area Networks (LANs) and Wireless LANs (WLANs) are two areas that are beginning to apply SIM authentication to existing infrastructure. It is still mostly used for Wireless LANs, as traditional LAN access usually require physical access to an

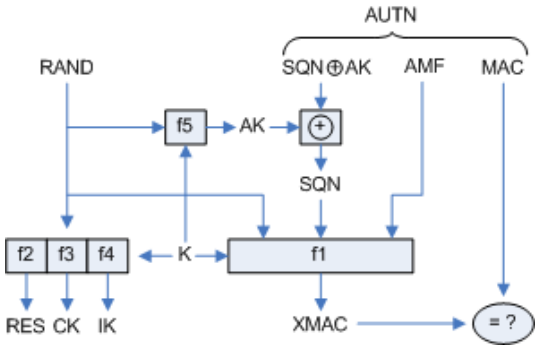

Figure 3.5: Generating the RES, CK and IK in UMTS

area, and is therefore not as exposed as its wireless version. This section will focus on WLAN authentication mechanisms, and how SIM authentication is being used for this purpose.

WLAN is, as the name indicates, a wireless version of the traditional Local Area Networks. Its main difference is that it uses spread spectrum technology based on radio waves instead of a cable as the physical layer. This allows WLAN enabled devices to move around and communicate with other devices in a limited area. When the term WLAN is used, it usually refers to the IEEE 802.11 set of standards [IEE03], also often named Wi-Fi. The 802.11 specification family currently includes six modulation techniques, all using the same protocol, but differing in data rates, ranges and frequencies. The properties of the most common techniques are summarized in 3.1.

Standard	Release Date	Frequency	Max Rate	Range
Legacy	1997	2.4 GHz	2 Mbit/s	<50 m
802.11a	1999	5 GHz	54 Mbit/s	50 m
802.11b	1999	2.4 GHz	11 Mbit/s	100 m
802.11g	2003	2.4 GHz	54 Mbit/s	100 m
802.11n	2006 (draft)	2.4 GHz or 5 GHz	540 Mbit/s	250 m

Table 3.1: A summary of the different properties of the most common WLAN modulation techniques (The ranges are indoor ranges). Modified from [Wik06b].

Devices can be used in two different modes of operation:

Ad-Hoc or peer-to-peer mode, in which two or more computers set up a temporary wireless network between them, without the need for fixed infrastructure like wireless access points, usually to share files or information. Mobile Ad-hoc Networks (MANETs) can also be expanded and used in larger scale situations, as in the battlefield, or along roads. The MANET working group of IETF[1] is working on improving routing protocols for use in ad-hoc networks.

Infrastructure mode is when the wireless network has an access point as the central point of communication, which all packets pass through. The coverage area of such an access point is often referred to as a *Basic Service Set* (BSS), which is a basic building block of a 802.11 wireless LAN. Several of these BSS' are usually

[1]MANET WG: http://www.ietf.org/html.charters/manet-charter.html

connected via a backbone or distribution network, and the total coverage area is referred to as an *Extended Service Set* ESS. The relation between these are shown in figure 3.6.

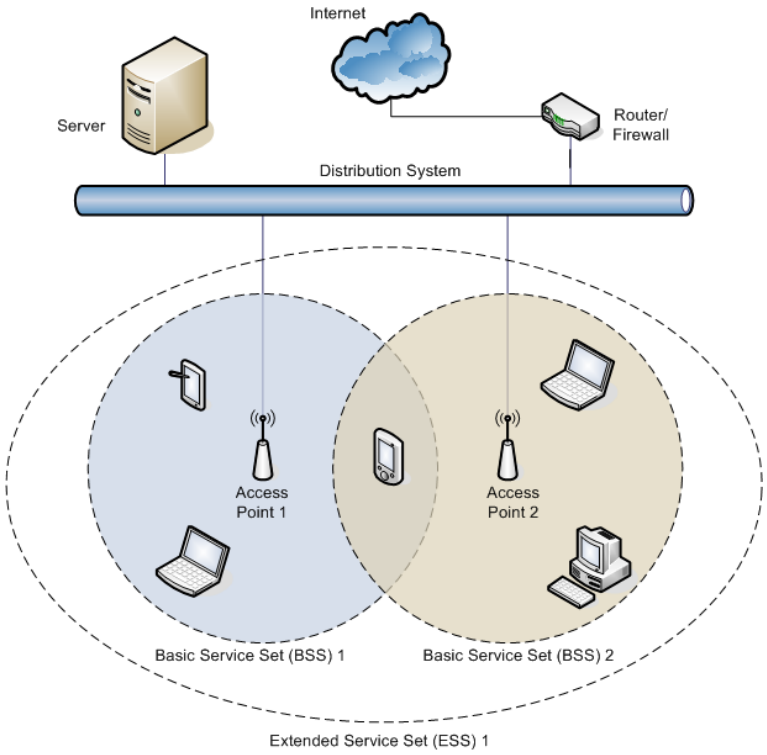

Figure 3.6: An example of a typical WLAN Architecture using an infrastructure BSS, modified from [Wik06b].

3.5.1 Standard WLAN authentication

To accept connections, Access Points need some kind of authentication, and the original 802.11 standard specifies two different authentication options.

Open System authentication (OSA) basically means no authentication. Any device requesting authentication using OSA gets authenticated if the authenticating device is set up to accept OSA and no other restrictions like MAC-address filtering is enabled. OSA is in most cases the default authentication algorithm.

Shared Key authentication (SKA) in the original standard requires the use of the Wired Equivalent Privacy (WEP) mechanism. WEP, as the name implies, was designed to provide wireless connections with security properties equivalent to those of a wired connection. WEP provides mechanisms for authentication requiring a pre-shared key, confidentiality using the RC4 stream cipher and data integrity by encrypting a CRC value of the original message. All these three mechanisms have later been proven to be vulnerable to attack, and thus useless. For instance, the encryption key used in WEP can be cracked in just a couple of hours [Ada02].

As the WEP mechanisms were found to be vulnerable to a series of attacks, an improved security mechanism was needed, quickly. The Wi-Fi alliance released the design for Wi-Fi Protected Access (WPA) in 2003. WPA was meant to be an intermediate solution to the WEP problem, until the 802.11i standard was finished. WPA was designed to function on the same hardware as WEP, only requiring a software upgrade on the equipment.

The major improvement in WPA over WEP is the Temporal Key Integrity Protocol (TKIP), which changes keys continuously as the system is used. This, combined with a better handling of Initial Vectors (IVs) makes the WPA key immune to the key attacks used against WEP [EA05].

However, as mentioned, WPA was only meant as a temporary solution, and WPA2 was described in the 802.11i standard that was finished in 2004. WPA2, which is designed from the ground up, introduces the Advanced Encryption Standard (AES) as the encryption algorithm, which is considered fully secure.

Wireless networks supporting WPA2 now had secure methods for authentication, confidentiality and data integrity, with full support of the 802.1X mechanisms. This included support for EAP as an authentication mechanism, which introduces methods as described in section 3.9.

3.5.2 SIM-based WLAN Authentication

As Wi-Fi hotspot services are getting more and more common, alternative authentication methods using EAP are being explored. One example is a solution HP and Intel, together with Axalto[2], have developed for GSM operators who want to provide simplified Wi-Fi hotspot authentication for their users [HP03].

The background for this SIM-based WLAN Authentication solution is that providing Internet access through Wi-Fi hotspots can be viewed as a natural evolution of the GSM operators' business. As many of them already have both user databases and existing infrastructure along with existing roaming agreements, setting up and operating Wi-Fi hotspots is something they can do easily. The architecture suggested in the HP/Intel/Axalto solution is shown in figure 3.7.

The steps for connecting and authenticating to the network are as follows:

1. The user has to be able to read the SIM card. Axalto provides USB SIM readers for this purpose, either connected through USB, PCMCIA or Bluetooth.
2. When roaming in hotspot coverage, the Axalto Client supplicant on the laptop detects signals from surrounding WLANs and initiates the login procedure.
3. The laptop acquires the login-information needed from the SIM, namely the IMSI, which in turn is sent to the network via 802.1X and the EAP-SIM method.
4. The challenge-response authentication described in the EAP-SIM method (see section 3.9) is initiated, and the messages involved are passed from the computer,

[2] Axalto Homepage: http://www.axalto.com

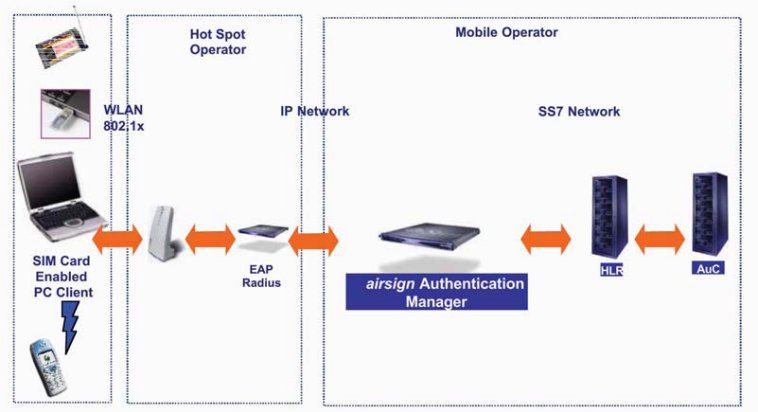

Figure 3.7: An example solution for SIM-based WLAN Authentication by HP and INTEL [HP03].

via the hotspot, to an Authentication Server (here called the Authentication Manager), communicating with the GSM network.

5. When the process is finished, the authentication manager decides whether the user is allowed access to the network.

3.6 SIM/USIM Readers

As explained in section 3.1.1, the GSM/UMTS authentication system is based on a challenge-response system, where the SIM does the calculation without the user ever getting access to the actual Authentication key. This means that to implement a similar authentication system, using the same mechanisms as in GSM/UMTS, the user has to be able to communicate with the SIM in order to read values like the IMSI, call functions, and to get responses generated by the SIM. This requires a SIM reader.

In the GSM system this SIM reader is built into the mobile station. This is also the case for many newer PDAs, as these also come with phone functionality or GSM/UMTS data support. If we want to use a device without a built in reader, like a PC, we need some other way of doing this. There are basically three ways to communicate with a SIM today.

USB or PC/PCMCIA cards Since USB was introduced ten years ago, most computers in use today have USB support [Koo05]. This makes USB a natural choice for connecting external devices, like a plug-in SIM, to a computer. A SIM reader is fairly cheap, and needs minimal setup on the user's computer. Laptop users also often have a PCMCIA card slot, which can give the same functionality, without the reader having to stick out of the computer. For smart card sized SIMs, a smart card reader can be used.

One can also acquire a PCMCIA card or USB dongle with SIM card built in, which will basically work the same way, except the user doesn't have to remove the SIM card from the mobile phone.

Using the Mobile Station as a reader Most newer mobile phones have some way

Figure 3.8: Equipment that can be used for connecting a SIM Card to a computer without a built-in SIM Reader

of connecting to a computer, either via cable or a wireless connection, using Bluetooth or IrDA. This can allow the computer to communicate with the SIM card, given that the mobile phone has a SIM access server. This is illustrated in figure 3.9.

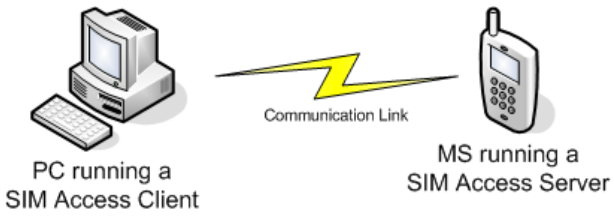

Figure 3.9: SIM Access Client and Server configuration

Phone access over Bluetooth is getting more common, as more and more phones are Bluetooth enabled. However, to communicate with the SIM, the mobile phone also has to have an implementation of the Bluetooth SIM Access Profile (SAP) [Blu05]. SAP is a description of how the SIM access server is made available through the Bluetooth interface, and how a client can communicate with the SIM access server. In SIM access the client can initiate a set of operations on the server (containing the SIM):

Manage Connection Allows the client to establish and terminate a SIM Access Profile connection.

Transfer APDUs The client application sends "Application Protocol Data Unit" (APDU) commands and the server responds with APDU responses.

Transfer ATRs Sends the Answer to Reset (ATR) messages from the server to the client. This is a string of bytes containing information about the SIM status and implementation.

Control the SIM Lets the client turn the SIM on or off.

A requirement for using Bluetooth communication to communicate with the SIM is to first authenticate both of the communicating parties, preferably through the use of a long passkey. This is known as pairing in Bluetooth. After the devices

are paired, an encrypted connection can be set up. This should be a requirement for any SIM Authentication System.

Built-in SIM Reader Some computers come with a built-in SIM reader, which makes it practically similar to a mobile phone. It is still very rare for normal laptop computers, but it is getting more and more common in smaller size PDAs. In a situation where the SIM reader is located in the device, the SIM client can communicate directly with the SIM.

3.7 Virtual Private Networks

There is currently a large deployment of VPNs across the world, though not as big as it could have been. This is caused by a lack of consensus regarding the definition and scope of VPNs, and a confusion over the different types of solutions described as VPN. In [GLH+00], a VPN is defined as

> *Emulation of a private Wide Area Network (WAN) facility using IP facilities.*

And this explains some of the confusion surrounding VPN solutions, and their lack of interoperability.

A VPN is simply a way to connect hosts. Two or more hosts can connect to a VPN, and several VPNs can be interconnected through a host, just like we are used to in physical networks. An example of a simple VPN configuration is shown in figure 3.10. There we see a network of computers, with a VPN established between three of the connected computers. The fourth computer is still connected to the physical network and can communicate with the three others, but the virtual private network does not include the fourth computer.

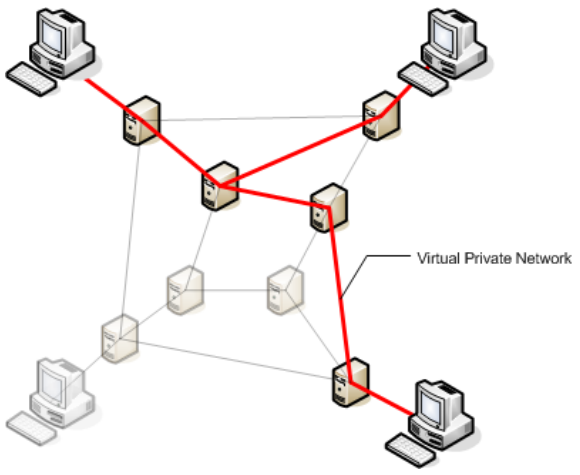

Figure 3.10: Example of a simple VPN configuration

3.7.1 Requirements

There are various types of VPNs, ranging from site-to-site VPNs for tunneling all the traffic between the sites, to simple application tunnels tunneling the data from one port on one computer, to another port on another computer. They all have a set of defined requirements:

Opaque Packet Transport The traffic carried through the VPN may have no relation to the other traffic carried on the network. It may be a different addressing scheme, or a different protocol altogether.

Data Security VPN security is based on different trust models. One model is where the VPN customer does not trust the service provider to provide any security, and uses Customer Provided Equipment (CPE) with firewall functionality that are connected using secure tunnels. The service provider here provides only the basic IP packet transport service.
The other model is where the customer trusts the service provider to provide a secure managed VPN service. This usually involves keeping the packets inside the provider network, as well as not looking at or modifying the transmitted data.

Quality of Service Guarantees There are often QOS guarantees on VPN services provided by an external provider. This usually encompasses both bandwidth, latency and availability guarantees. While these requirements depend greatly on the services provided by the underlying IP backbone, the VPN service must also be able to meet the demands by the customer with the added complexity of connection establishment and potential encryption.

Tunneling Mechanism The two first requirements imply that the VPN connection must be implemented through some form of IP tunneling mechanism, in such a way that the original packets can be transferred unmodified through the existing physical network. These tunnels can also be realized through some other IP traffic management mechanisms, such as MPLS.

3.7.2 Configurations

The various VPN implementations are usually based on CPE, with the VPN tunnel being created and terminated at or inside the customer's firewall, as illustrated in figure 3.11. The VPN gateways can either be managed by the company itself, or by a third party provider. This solution is totally transparent to the users inside the network, and as far as they can see, the two sites connected are on the same local area network (LAN). The connection establishment and authentication is on site-level, and is performed when the links are first established, and is of no concern to the users.

A variant of the Site-to-Site VPN configuration is the Computer-to-Site VPN configuration. This is usually applied when connecting a single computer to the corporate network when outside the network, for instance at home or on the road. To accomplish this the computer needs to run VPN client software, configured to connect to a VPN concentrator located inside the corporate network. This could be viewed as an equivalent to users dialing in to the network using a modem and a telephone line. An

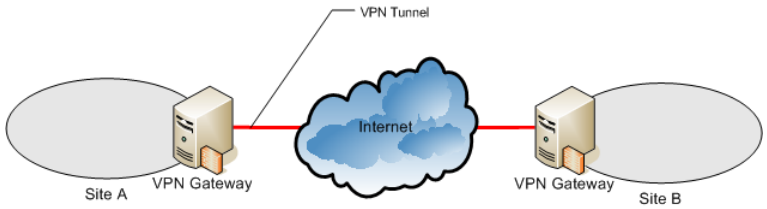

Figure 3.11: A Site to Site VPN, with the VPN gateways residing at the edges of the private networks and the VPN tunnel being through the public Internet.

illustration of this configuration is shown in figure 3.12.

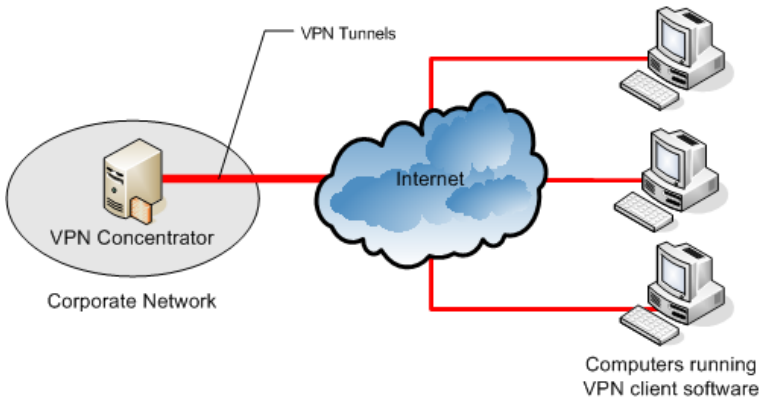

Figure 3.12: A Computer to Site VPN, with the computers running VPN clients connected to a VPN concentrator inside the corporate network.

When establishing the VPN tunnel the VPN client software initiates an authentication procedure with the VPN concentrator, and once authenticated the users are usually authorized access to the corporate network as if they were inside the walls of their office. This includes getting an internal IP-address, assigned by the DHCP server. The fact that the user gets elevated access to the internal resources on the network addresses the need for secure authentication mechanisms, in addition to a user-friendly way of providing user credentials.

The third VPN configuration is using a trusted network like a leased line, or a part of the Internet controlled by a trusted third party and routing the traffic via a "safe" path through that network (figure 3.13). This requires guarantees that the integrity of the data transferred is preserved.

The configuration with the strongest requirements regarding secure credentials management and ease of connection establishment is the Computer-to-Site configuration, and this thesis will primarily focus on that type. Unless specified otherwise, the term VPN will from here on refer to this configuration.

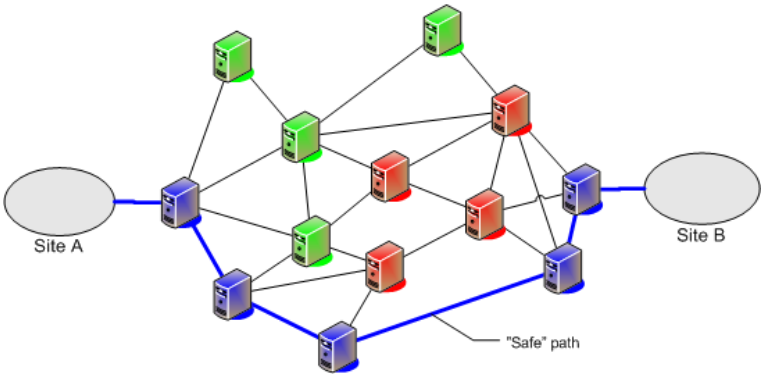

Figure 3.13: Routing traffic through a part of the network controlled by a trusted provider (blue), while avoiding untrusted parts.

3.7.3 VPN Components

This section will describe the components needed for a basic PC to Site VPN, introduced in section 3.7.2. As already mentioned, this type of VPN consist of a computer (a client) and a VPN server located inside or at the border of the company network.

VPN Client The VPN client is the entity establishing the VPN connection. This client can either be purely a piece of software running on the computer wishing to establish the VPN connection, or a dedicated network component referred to as Customer-premises or Customer-provided equipment (CPE). The most common for personal computers is the software type, as it does not have to be brought along when moving the computer.

To the operating system, the VPN client is often set up so it can be viewed as a virtual network adapter. This way, traffic that are to be tunneled to the remote site is sent onto the network using the virtual adapter, after the VPN tunnel is established. The virtual adapter is set to forward the traffic, now tunneled, through a normal network adapter, onto the network. This is illustrated in figure 3.14

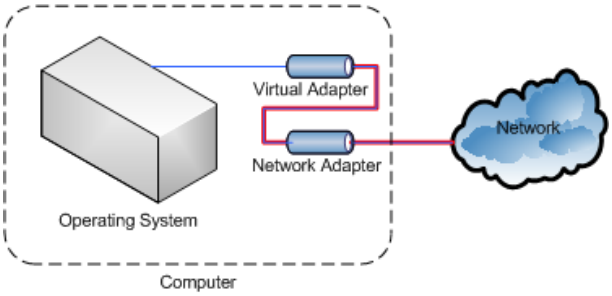

Figure 3.14: A network adapter forwarding traffic (blue) through a VPN Tunnel (red).

VPN Server To establish a tunnel, you need two end points. The first end point is of course the computer initiating the VPN connection. The second is a VPN server usually situated inside the company network as illustrated in figure 3.12. The VPN server is also often referred to as a concentrator or Network Access Server (NAS). The VPN server has to have a connection to the VPN clients, as well as

a connection to the local network. It is also often connected to an authentication server, as shown in figure 3.15.

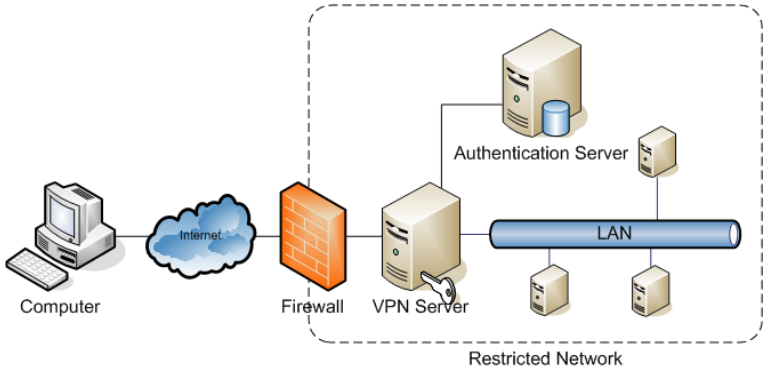

Figure 3.15: Example of a typical VPN infrastructure. An authentication server is set up to handle authentication requests.

The VPN server has two main jobs. In chronological order; first it has to accept incoming connection, figure out what type of authentication protocol the client wants to use and forward the request to the appropriate authentication server. Second, after getting a signal from the authentication server saying the authentication process was successful, it has to act as the termination point for the VPN tunnel.

Authentication Server The authentication server's job is to take some of the load off of the authenticator, in this case the authenticator part of the VPN Server, by being in charge of the main authentication-process. This gives us a clearer division between the actual data tunneling, performed by the VPN server, and the authentication procedure, performed by the Authentication Server.

3.7.4 Authentication

Users wanting to establish a VPN tunnel connection from a VPN Client to a VPN Server have to go through a process of authentication and authorization. Especially when the VPN Server is used as an access control point at the border of an intranet. In many cases, when users are connected to the VPN Server they have the same restrictions, or lack thereof, as they would have sitting in their office. Authentication for VPN connections is usually based on one or more of the following;

Username/Password The username of the user is transferred in some way to the VPN server (or forwarded to the authentication server), and the password is used to authenticate the user. This can be either by sending the password to the server, which is considered very insecure, or using it to calculate some sort of response to a given challenge (more secure).

Certificates Every client has a certificate, which consists of a public and a private key. This is based on the systems described in section 2.2.2. When authenticating, the server sends a challenge to the client, which is then encrypted using the client's private certificate. If the server is able to decrypt the message using the client's public certificate, it knows the client is who he claims to be (as long as the private

certificate is not compromised). This authentication mechanism is stronger and more secure than normal username/password authentication, but requires trusted distribution of the keys in advance.

Pre-Shared Keys In systems using Pre-Shared Keys for authentication, a single key is used to encrypt the channel, or derive an encryption key for the channel. This system functions as described in section 2.2.2. This system has perfect strength in theory, but it requires a complicated system for key distribution – ideally, an encrypted channel. If they key should be compromised, the security is non-existent. Also, earlier communication using the same key would no longer be protected.

3.7.5 Access Mechanisms

As the user wants to establish a connection to a VPN server somewhere, an Internet connection is assumed to already be available. As neither the authentication mechanism or the VPN connection itself require much bandwidth, the user is free to use any connection available to realize the authentication system. The only requirements are:

- The connection should be stable, as the VPN connection has to be kept alive. If the connection goes down, so does the VPN connection, and the user has to get re-authenticated.
- The bandwidth of the VPN tunnel is of course restricted by the bandwidth of the channel the VPN tunnel goes through. To be able to use services with high bandwidth requirements, such as streaming video, the user's Internet connection must have sufficient bandwidth.

There are many types of access mechanisms available, and the connection used does not restrict the system in any ways, except for bandwidth limitations. The most common connection types today are:

DSL Digital Subscriber Line

Cable Modem

LAN Local Area Network

WLAN Wireless Local Area Network

ISDN Integrated Services Digital Network

The access mechanisms will not be discussed any further in this thesis, as the solution only requires some type of Internet connectivity, and the type of the physical transport is irrelevant.

3.8 Diffie-Hellman Key Exchange

Diffie-Hellman is a cryptographic protocol for calculating a shared key between two parties over an insecure communications channel. The two parties need not have prior knowledge of each other, and they do not need to have any shared secrets in advance.

After the key exchange is completed both parties share a key, which then can be used to secure subsequent communication using symmetric cryptography, for instance in a VPN tunnel.

The key exchange algorithm was first presented by Whitfield Diffie and Martin Hellman in 1976, but it is said to have been discovered by GCHQ, the British signals intelligence agency, a few years earlier. The protocol gets its security from the difficulty of calculating discrete logarithms in a finite field.

The protocol goes as follows:

1. Alice chooses a random large integer x and sends Bob
$$X = g^x \bmod n$$
2. Bob chooses a random large integer y and sends Alice
$$Y = g^y \bmod n$$
3. Alice computes
$$k = Y^x \bmod n$$
4. Bob computes
$$k' = X^y \bmod n$$

Now we know that both k and k' equals $g^{xy} \bmod n$. As only n, g, X and Y can be discovered by listening to the channel, the key can not be found – unless the discrete logarithm is solved and x and y is recovered. At the moment this can not be done if both g and n are chosen appropriately. The number $(n-1)/2$ should be a prime, and n should be large. Any g, such that g is primitive $\bmod n$, can be chosen [Sch96].

3.9 Extensible Authentication Protocol (EAP)

The Extensible Authentication Protocol (EAP) is an authentication framework with support for multiple authentication methods. EAP is defined in RFC 3748 [ABV+04], and this section is based on this.

EAP is designed to run directly over data link layers such as Point-to-Point protocol or IEEE 802, without requiring IP. As EAP is often used for network access authentication, this is a requirement as IP is not always available. EAP is designed to be flexible, and can be used as transport protocol for other authentication methods, such as EAP-SIM (SIM card), EAP-AKA (UMTS Authentication and Key Agreement) and EAP-TLS (Wireless authentication). These are called EAP Types.

An illustration of the EAP multiplexing model is shown in figure 3.16

The EAP multiplexing model shows how EAP authentication is performed. On the bottom we have the *lower level*, which is the layer responsible for transmitting and receiving EAP frames between the peer and the authenticator. This can be for instance PPP or L2TP. The *EAP layer* deals with duplicate detection and retransmission, and

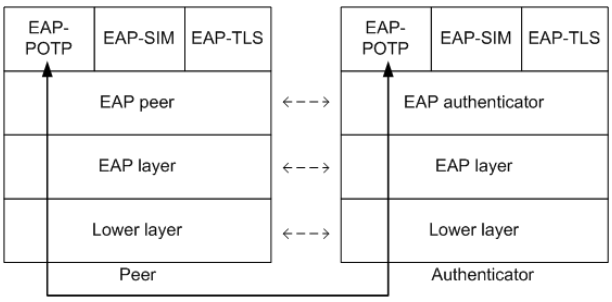

Figure 3.16: The 4-layer EAP Multiplexing Model shows the layers involved in EAP authentication

transmits EAP packets between the EAP peer/auth. and the lower layer. A host typically implements either of the *EAP peer and authentication layers*, but in the case of a pass-through authenticator, both layers can be present. The *EAP method layers* implement the authentication algorithms and receive and transmit the appropriate messages during an EAP session. The EAP methods relevant to this thesis are explained in the next sections.

3.9.1 EAP-SIM

EAP-SIM is an EAP mechanism for authentication and session key distribution using the GSM SIM card. EAP-SIM is developed by 3GPP and is described in RFC 4186 [HS06].

The EAP-SIM mechanism specifies enhancements to GSM authentication and key agreement by enabling the authentication mechanism to use multiple authentication triplets. This offers authentication responses and session keys of greater strength than can be achieved using single triplets. In addition the mechanism includes network authentication, identity protection, result indications and a fast re-authentication procedure.

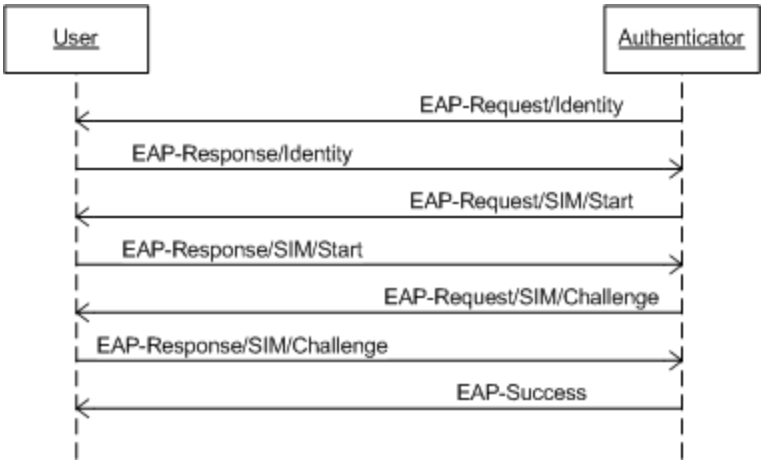

Figure 3.17: EAP-SIM Full Authentication Procedure

3.9.2 EAP-AKA

EAP-AKA was developed by 3GPP, and is defined in RFC 4187 [AH06]. It is an EAP mechanism for authentication and session key distribution using the Authentication and Key Agreement (AKA) mechanism used in UMTS (and CDMA2000) networks. AKA uses symmetric keys, and is usually implemented in the USIM.

EAP-AKA is the 3rd generation counterpart EAP-SIM, and allows the use of 3rd generation mobile network authentication infrastructure in authentication mechanisms.

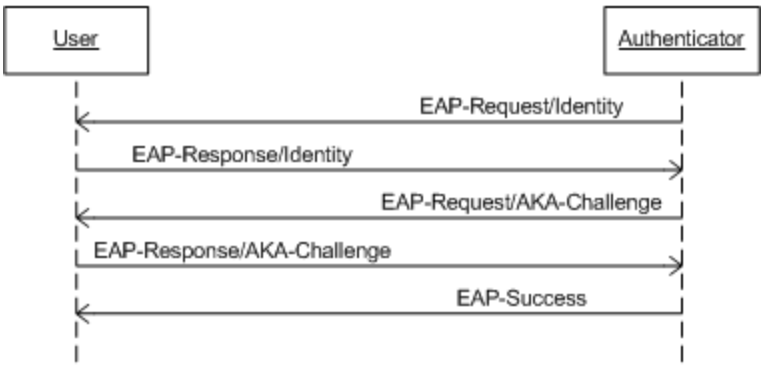

Figure 3.18: EAP-AKA Full Authentication Procedure

EAP-POTP

EAP-POTP is an EAP method for the Protected One Time Password-protocol, for use in OTP authentication systems. It is described in a draft by RSA Security [Nys06].

It is designed to meet the needs of organizations that wants to use OTP to authenticate users over EAP. The method is not designed around any particular OTP system, and is therefore possible to adopt to different systems.

3.9.3 EAP-TLS

The EAP-TLS protocol describes how to use the mechanisms of Transport Layer Security (TLS) within EAP. These mechanisms are mutual authentication, integrity-protected cipher suite negotiation and key exchange. The EAP-TLS method gives EAP the possibility to transfer both user and server certificates between parties. EAP-TLS is defined in RFC 2716 [AS99].

3.9.4 EAP-SC

The EAP-SC method is at the time of writing a draft by Pascal Urien of ENST[3] [Uri06]. It describes a standard interface to an EAP implementation embedded in a smart card.

[3]Ecole Nationale Supérieure des Télécommunications, France

An EAP smart card implements one or several EAP methods, and uses a smart card interface entity that sends EAP messages to and from the device. The EAP smart card can for instance implement the EAP-SIM, EAP-AKA and the EAP-POTP methods, and these methods can be called through a simple interface.

3.9.5 Support for EAP

The use of EAP is getting more and more widespread, and the protocols supported increases by each release of the supplicants. Microsoft has since Windows 2000 been including a native supplicant in Windows, with support for the most common EAP types. This client also supports EAP-plugins developed using the EAP API, found on Microsoft's MSDN pages[4]. For Linux, the Open1X project is working on a supplicant with support for most of the EAP types. The EAP types supported by three different clients at the time of writing are listed in table 3.2.

	Windows 2000	Windows XP, 2003	Open1x 1.2 (Linux)
EAP-AKA			+
EAP-MD5	+	+	+
EAP-MSCHAP v2			+
PEAP		+	
PEAP-MS-CHAP v2		+	+
EAP-TLS	+	+	+
PEAP-TLS		+	
EAP-TTLS			+
EAP-GTC			+
EAP-OTP			+
LEAP			+
EAP-SIM			+
Plug-Ins	+	+	

Table 3.2: An overview of EAP support in common supplicants.

3.10 Summary

This chapter has described the components used in both GSM networks and in existing VPN network implementations, and is a good starting point for finding out what is needed for a combined solution.

A combined solution will need to include parts from three different domains, shown in figure 3.19. We have the User Equipment, communicating with the VPN network, requiring authentication, and the GSM network, which has methods for authenticating

[4]Microsoft Developer Network: http://msdn.microsoft.com

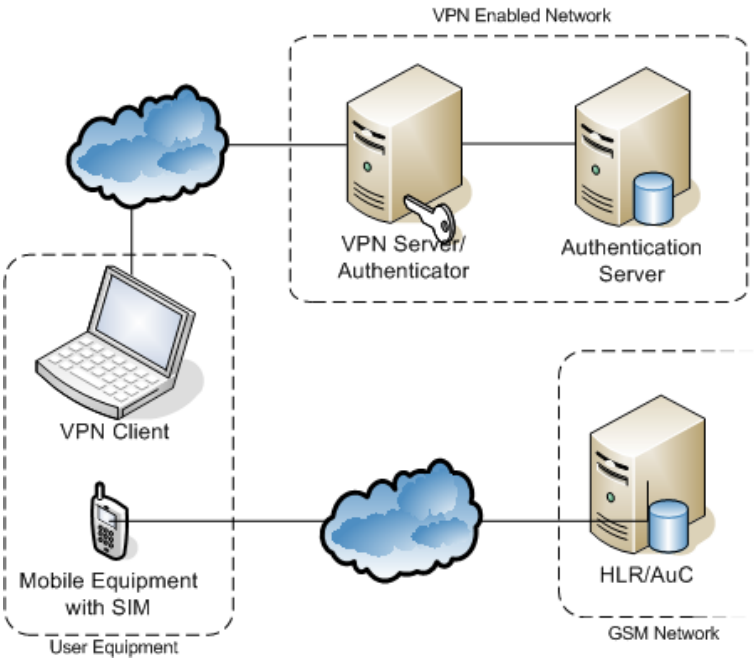

Figure 3.19: An overview of today's situation with three separate domains, divided by the Internet and GSM networks.

the user. The challenge is communicating authentication information between the two external domains in a secure way.

Chapter 4

Existing Systems

4.1 OpenVPN

OpenVPN is, as explained on its home page[1], a

> full-featured SSL VPN solution which can accommodate a wide range of configurations, including remote access, site-to-site VPNs, Wi-Fi security, and enterprise-scale remote access solutions with load balancing, failover, and fine-grained access-controls

OpenVPN offers OSI layer 2 or 3 security by using the industry standard SSL/TLS protocol, and offers several types of authentication methods as well as user or group-specific access control policies.

4.1.1 Functionality

One big difference between OpenVPN and several other VPN solutions is that OpenVPN bases its functionality on something called *tun* and *tap* interfaces. Tun and tap interfaces are virtual network adapters that looks like point-to-point network hardware to the operating systems and applications running on it, but lets a software receive its output and can process it before sending it out the "real" network adapter. This software, for instance SSH, can then encrypt and decrypt the data going between the tun/tap adapter and the real adapter, adding security to the data flow. This without any applications having to support encryption themselves. The difference between tun and tap interfaces is that tap emulates ethernet, while tun emulates point-to-point connections.

One advantage of using the tun/tap model is that OpenVPN is user-space. Users can run their own VPN client and connect to VPN servers in different places. The alternative IPSec, requires a modification to the IP stack itself and listens to any passing packets and determines whether they need encryption or decryption, and which security association it needs to use to do so.

[1] OpenVPN Home Page: http://openvpn.net

4.1.2 Security

OpenVPN offers different methods of keying when establishing connections;

Static, pre-shared keys This method lets you configure the VPN Client and the VPN Server with a pre-shared key which is used to encrypt the tunnel. This simplifies the configuration of the VPN endpoints, and makes setting up the tunnel a fast process. The static key contains 4 independent keys: HMAC send, HMAC receive, encrypt and decrypt. By default, both hosts use the same key.

RSA PKI By supporting RSA Public Key Infrastructure, the VPN infrastructure can be configured using certificates and private keys for authenticating the endpoints and communicating connection parameters in a secure way.

SSL/TLS For initial authentication and symmetric key exchange SSL and TLS can be used. This method uses certificates in order to authenticate the endpoints.

4.1.3 Management

OpenVPN offers a management interface in order to control and manage OpenVPN remotely. This interface can be used to develop a front-end to the OpenVPN client, or supply the client with parameters prior to or during connection. When enabling remote management, a TCP server is established on a pre-set port, which allows remote control using for instance a telnet client. The management interface is described on the OpenVPN home page [2].

4.1.4 Compatibility

OpenVPN runs on all major operating systems, including Linux, Windows 2000/ XP/Vista and Mac OS X.

4.2 OpenSwan

The information in this section is based on the OpenSwan Wiki[NM] and the OpenSwan documentation that follows the downloadable source code.

OpenSwan is based on the FreeS/WAN product, which ended with a final release in April 2004 [Wik]. OpenSwan is mainly a set of tools for performing IPsec functions on Linux operating systems. It consists of three tools;

- Configuration Tools (also known as ThoseAwfulScripts)
- Key Management Tools (aka pluto)
- Kernel Components (KLIPS and 26sec)

OpenSwan with IPsec provides encryption and authentication at the IP level. It is designed to either create permanent secure tunnels or dynamic tunnels for people connecting from laptop machines or on the road, referred to as Road Warriors. One goal

[2] OpenVPN Management Interface Documentation: http://openvpn.net/management.html

of the OpenSwan project is to simplify the setting up of tunnels, so they can be created on demand without co-ordinating with another site administrator. This is referred to as opportunistic encryption.

4.2.1 Functionality

OpenSwan uses IPsec, and this requires cooperation with the operating system kernel. KLIPS, or Kernel IPsec Support is the kernel portion of OpenSwan and what used to be FreeS/WAN. With the release of the Linux 2.6 kernel, the 26sec components (called NETKEY) are included in the kernel, and partially replace KLIPS.

It is important to note that IPsec is designed to secure IP links between machines, and its authentication is based on authenticating machines, not users. IPsec does not have any concept of user ID, so IPsec can not be used to control which *users* get access to your server. To do this, non-IPsec mechanisms are needed.

When setting up a connection OpenSwan first uses the Internet Key Exchange (IKE) protocol. In phase one the two gateways negotiate and set up a two-way ISAKMP (Internet Security Association and Key Management Protocol) Security Association (SA) for handling phase two negotiations. In phase two the ISAKMP SA is used to negotiate IPsec SAs. These SAs are unidirectional and are negotiated in pairs to handle two-way traffic. After these two phases are complete, the IPsec tunnel can be established.

4.2.2 Security

Authentication in OpenSwan is usually key-based. It supports manual keying (pre-shared) where the keys can be manually set with the connection definitions before connection establishment, and automatic keying where OpenSwan negotiates key using the IKE protocol. The automatic method is preferred as it is more secure, and it also re-keys the connection periodically.

OpenSwan does let you authenticate in "normal" fashion, with username and password during connection establishment, when using XAUTH. XAUTH extends phase 1 of the IKE to include additional user authentication exchanges. To complete the tunnel establishment, the client has to pass the XAUTH verification in addition to the normal IKE authentication.

4.2.3 Compatibility

OpenSwan runs on Linux.

4.3 Cisco VPN

Cisco offers two different types of VPN Clients; the hardware based Cisco VPN 3002 which is a dedicated hardware component that acts as an endpoint for a VPN tunnel, and the Cisco VPN Client software which is the one that will be described in this

section. This is similar to the permanent and dynamic tunnels of OpenSwan, described in 4.2.

This information is from the Cisco VPN Data Sheet [Sys].

The Cisco VPN Client, according to Cisco,

> allows organizations to establish end-to-end, encrypted VPN tunnels for secure connectivity for mobile employees or teleworkers.

Cisco's VPN Client is a commercial product, and it is not open source. It is also primarily designed to connect to Cisco's own VPN Server products;

- Cisco 6500 / 7600 IPSec VPNSM and VPN SPA IOS Software Release 12.2SX and later
- Cisco VPN 3000 Series Concentrator Software Version 3.0 and later
- Cisco IOS Software Release 12.2(8)T and later
- Cisco PIX Security Appliance Software Version 6.0 and later
- Cisco ASA 5500 Series Software Version 7.0 and higher

There exists other VPN Server solutions that supports the Cisco VPN Client, such as OpenSwan[3].

4.3.1 Functionality

The Cisco VPN Client uses IPsec tunneling to create VPN connections, employing Encapsulating Security Payload (ESP) to tunnel the original packets. ESP provides origin authenticity, integrity and confidentiality protection of packets and has support for the DES, 3DES and AES encryption algorithms. Cisco VPN Client sets up the VPN tunnel in the same way as OpenSwan, described in section 4.2, going through the different IKE phases..

4.3.2 Security

As Cisco VPN Client uses IKE for establishing the VPN Connection, it supports using keys (pre-shared and dynamic) as well as XAUTH. This is also similar to OpenSwan. It also supports RADIUS authentication, using token cards, Kerberos/Active Directory authentication, NT Domain authentication and other common authentication methods.

4.3.3 Compatibility

Cisco VPN Client supports all Windows versions as well as Linux, Solaris and Mac OS X.

[3] OpenSwan: http://www.openswan.org

4.4 Summary

The VPN Client solutions described have different properties when it comes to mode of operation, functionality and ease of use. Based on all the evaluated features, the OpenVPN solution appears to be the one most suited for integration with this prototype.

- The unique mode of operation lets the user(s) run the VPN Client as a normal application which allows establishing of different VPN tunnels for different requirements.
- It is open source, which allows for custom modifications if needed.
- It runs on all major operating systems.

Chapter 5

Analysis

In this chapter an analysis of the SIM based VPN Authentication System will be performed. The first section presents a few possible usage scenarios, the next gives an overview of the user experience. Then a set of system requirements are given, both in textual and in use case form, before Message Sequence Diagrams will be presented and described.

5.1 Authentication Scenarios

5.1.1 Scenario 1: Two-Factor Authentication Security

Frank works as a professional salesman for a major pharmaceutical company, and often has to visit customers across the country to present new products. Even though he usually travels by air, these business trips often take as much as a day or two if the presentation is to be held for more than one audience. In these cases Frank has a lot of spare time in between presentations where he would like to do some other work, or maybe prepare another presentation. To be able to work efficiently, Frank is dependent on resources located on isolated servers inside the company's network. Isolated, in the sense that they are inaccessible from the outside world. To connect to this server Frank has to use a VPN client he has installed on his laptop computer. The VPN client used to prompt the user for a password, but after an incident where a competing company gained access to the confidential resources a year ago, they now use two-factor authentication – PIN and SIM. When Frank clicks *connect* on his computer, he is asked to enter the 8-digit PIN for the SIM he has in his Bluetooth enabled company mobile phone, before he is authenticated and allowed access to the company network. If someone should steal his SIM to get access, it is deactivated after three unsuccessful login attempts, if Frank has not already notified the IT-department and blocked his IMSI.

5.1.2 Scenario 2: Ease of Use

Evelyn is an example of your average student. As most people working with computers every day she has to remember a lot of different passwords. She uses one to

log onto computers in the computer rooms, another to log onto the mail-system to check her email, and a third for VPN access to the school intranet. Most users keep these passwords the same so they will not forget them, lowering the overall security of the system. Evelyn has also considered this, despite the IT-department suggesting otherwise. During the past few months her university, in cooperation with the local telephone operator, has started to test SIM authentication of services. The goal is to improve the security of the university-provided services, while at the same time making life easier for the students. This started with SIM-authentication for WLAN services using 802.1x, expanding to simplified login to the university computers using a SIM Generic Authentication System. Now they also have enhanced the VPN access solution with SIM-authentication to enable access to limited intranet services, like access to the personal home directory and software distribution shares.

Most of the students at the university already have a laptop and a Bluetooth enabled mobile phone. Those without these are offered a USB SIM-reader, or a USB-device with a built in SIM so they can connect their SIM to other computers. This way everyone can benefit from the new simplified solution.

5.1.3 Scenario 3: Simplified Roll out

In Dans Delivery Company everyone has their own PDA with built in GSM phone. This way all the drivers can bring up a map of where the next delivery is, periodically updated from central administration. To prevent people from accessing sensitive data, the company has adopted two more elements of security. First, the company firewall blocks all connections from the outside world, except VPN connections. Second, the VPN server only supports authentication using 802.1x. This means that everyone connecting to the central server has to use a VPN client, which also gives added security to the data transmissions to and from the PDA. Just by adding VPN client software to the standardized PDAs during the nightly update schedule and setting up the PDA to connect to the VPN server when connected to the network, the drivers did not notice any difference the next day (except the yellow padlock appearing in the tool bar of the system software when connected).

5.2 Overview of the User Experience

A goal of the SIM-based VPN Authentication System is to add features to the already existing VPN authentication mechanisms. As an important feature of them is that the VPN should be relatively transparent to the user, this is also emphasized in this system.

As the user chooses to establish a VPN connection, or such an establishment is automatically initiated by another application, the authentication mechanism is started. For the authentication to succeed there are two prerequisites;

The SIM Card must be accessible by the SIM Client in the device, as described in section 3.6. If this is not the case, the user is prompted to connect the SIM in some way. After the SIM Reader is detected, and the SIM is identified, the authentication continues.

The PIN associated with the connected SIM Card has to be input by the user. A prompt asking the user to input the PIN should be shown on the screen.

After the prerequisites have been met, the authentication mechanism should complete without any more user interaction, provided the SIM is valid and the PIN is correct. If there for some other reason was a problem establishing the connection, the user should be notified of this. The user should also be given some sort of acknowledgment when the connection is established.

5.3 High-Level Requirement Analysis

The overall system requirement analysis is split up into two parts, the functional and the non-functional requirements. The functional requirements specify the intended behavior of the system, while the non-functional requirements specify properties the system should satisfy.

5.3.1 Functional Requirements

Bredemeyer[1] defines functional requirements as follows:

> *Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks or functions the system is required to perform [MB99].*

A list of functional requirements of this system, based on the scenarios described, are shown in table 5.1.

	Functional Requirement
1	The system should not be restricted to a single VPN client or server
2	The system should not be restricted to a single type of authenticator
3	The user should be able to use different SIM readers
4	Existing VPN Requirements should still be met:
4.1	– Opaque Packet Transport
4.2	– Data Security
4.3	– Quality of Service Guarantees
4.4	– Tunneling Mechanism
5	The system should not interfere with normal GSM use
5.1	– Not cause unnecessary load on the system
5.2	– Not introduce security risks

Table 5.1: A list of the functional requirements of the system

5.3.2 Non-Functional Requirements

Bredemeyer defines non-functional requirements as:

[1] www.bredemeyer.com

> *Non-functional requirements include constraints and qualities. Qualities are properties or characteristics of the system that its stakeholders care about and hence will affect their degree of satisfaction with the system. Constraints are not subject to negotiation and, unlike qualities, are (theoretically at any rate) off-limits during design trade-offs [MB01].*

Availability

In telecommunications, the term availability means the proportion of time the a system is in functioning condition. In this case, it is the probability that the user is able to use the system at any time. That is, authenticate and connect to the VPN server. Because of this two part division, we get two limitations. The availability of the system is not better than the combined availability of the authenticator, the authentication server and VPN server (and the user's Internet connection). If any of these fails, the system is, from a users perspective, unavailable. The availability is usually specified in a Service Level Agreement (SLA). .

Reliability

Reliability is related to the availability goal, in that both are related to the system functioning as it should. But while availability focuses on whether the system is accessible at a given time, reliability is a measurement of how reliable and stable the system is – once the user is connected. As the authentication part of the system is performed once, and completes after a very short period of time, the major focus point when it comes to reliability is the reliability of the VPN connection.

Once the user is connected, the connection should not be terminated unless the user explicitly requests it (this includes turning off the user's computer or terminating the Internet connection). The reliability of the system is therefore almost identical to the reliability of the VPN server, which should be specified in some sort of SLA.

Compatibility

Compatibility means the system should be compatible with both other systems and older systems. It should, as mentioned in the functional requirements, be a goal to make it possible to extend existing and new versions of both clients and servers to support the functionality described. The system itself should use well documented protocols and mechanisms, to avoid any ambiguities, as well as give a well documented view of its own architecture and construction.

Performance

Performance is a measurement of how long the system takes to respond when an action is initiated. In this case it is how long it takes for a user to get authenticated when an authentication request is sent. This should be at most a few seconds, but preferably as fast as possible. This can be achieved by using an efficient protocol and transmitting

only the information necessary for the task to be completed. A protocol with few messages back and forth is preferred.

Security

As the system expands already existing security mechanisms, the security in the system itself should be as strong or stronger than the systems it communicates with, in order to prevent holes that could compromise the entire system.

Usability

As one of the goals listed in section 1.2 is that the system should simplify the authentication process for the user, usability should be one of the main functional requirements for this system. Bad usability is often an obstacle when implementing new systems, as non-technical users tend to prefer the most non-intrusive and simple systems, without considering the security parameters.

5.4 Use Case Analysis

To get a better overview of the requirements of the system, a UML use case analysis will be performed in the following section. The use cases will describe the actions performed from a user's perspective, and the sequence of events following. An overview of the use case scenario can be viewed in figure 5.1.

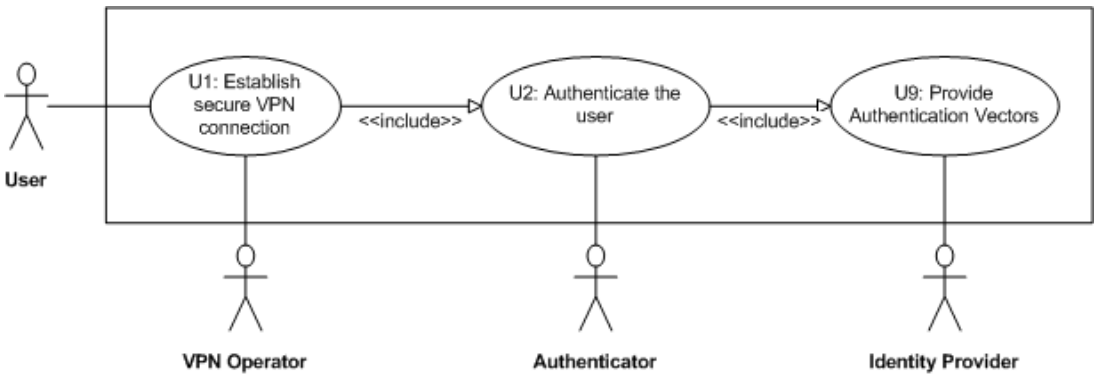

Figure 5.1: An overview of the three main areas the use cases can be divided into.

Here we see the two main tasks performed in the system, as well as one additional task requiring a third party. One task is initiating and setting up the actual VPN tunnel, where the user communicates with an *actor* we here refer to as the VPN Operator, and the other task is authenticating the user, performed together with the Authenticator and requiring help from the Identity Provider.

The VPN part of the system doesn't introduce many new situations, as a main goal of the system is to make things easy for the users.

Use Case	1. Establish VPN Tunnel
Description	Start the VPN Client, perform authentication to the VPN server, and establish VPN tunnel.
Actors	VPN Client, VPN Server
Assumptions	The VPN Client is installed and configured correctly The SIM Server is running
Steps	1. The user starts the VPN client, and the VPN client starts the authentication mechanism (UC 2) 2. When the authentication is complete, the VPN client connects to the VPN server and gives its credentials 3. The VPN server validates the user (UC 3) 4. The VPN tunnel is established
Variations	#1 VPN client not responding, RETURN error. #1 Authentication fails, RETURN error #2 VPN server not responding, RETURN error #3 Access denied, RETURN error #4 Connection fails, RETURN error
Issues	Should the user be able to use different VPN clients? How?

Table 5.2: Use Case - Establish VPN Tunnel

Use Case	2. Perform Supplicant Authentication
Description	The user is not yet authenticated with the Authenticator, and it requests the supplicant to perform authentication.
Actors	VPN Client, Supplicant, SIM, Authenticator
Assumptions	The Authentication server is running The supplicant is configured correctly
Steps	1. The VPN Client contacts the Supplicant, and calls the Authenticate method 2. The Supplicant starts communication with the SIM (UC 4) 3. The Supplicant verifies the PIN (UC 5) 4. The Supplicant performs authentication with the Authenticator (UC 6) 5. The result is returned to the VPN client
Variations	#1 SIM communication fails, RETURN failure #2 PIN verification fails, RETURN failure #4 Authentication fails, RETURN failure
Issues	

Table 5.3: Use Case - Perform Supplicant Authentication

Use Case	3. Validate User
Description	The VPN Server receives an incoming authentication request, has to validate the user.
Actors	VPN Client, VPN Server, Authenticator
Assumptions	"Perform Supplicant Authentication" was successful
Steps	1. The VPN Server receives a connection request from the VPN Client 2. The VPN Server contacts the Authenticator, and asks for the authentication status for the connecting user (UC 11) 3. The VPN Server receives authentication status from the Authenticator 4. The VPN Server establishes the VPN Tunnel with the VPN Client
Variations	#2 The Authenticator does not respond, RETURN failure #3 Unknown user or user not authenticated, RETURN failure
Issues	Which format should the username be in? And the keys?

Table 5.4: Use Case - Validate User

Use Case	4. Start SIM communication
Description	The supplicant wants to start SIM communication, and needs to check the SIM.
Actors	Supplicant, SIM
Assumptions	The SIM card is connected to the SIM reader
Steps	1. The communication with the SIM is started 2. Read ATR to verify the card 3. RETURN success
Variations	#2 The card is not supported, RETURN error
Issues	

Table 5.5: Use Case - Start SIM communication

Use Case	5. Verify CHV
Description	Verify the user's PIN code (CHV) with the SIM.
Actors	Supplicant, SIM
Assumptions	"Start Sim Communication" was successful
Steps	1. Ask the Supplicant for the PIN code 2. Send the "Verify CHV" APDU, containing the PIN code to the SIM 3. Verify the response from the SIM 4. The Supplicant is authenticated to the Card, RETURN success
Variations	#1 PIN protection is disabled on the card, enable the PIN and continue. #3 Wrong PIN code provided, retry up to two times before PUK is needed #3.1 Wrong PUK code provided, retry up to nine times before card is deactivated
Issues	Should the software ask the user for PUK?

Table 5.6: Use Case - Verify CHV

Use Case	6. Authenticate
Description	The VPN Server receives an authentication request, and contacts the Authenticator in order to infer on the GSM authentication.
Actors	VPN Client, VPN Server, Authenticator
Assumptions	The VPN Server is configured with the correct Authenticator
Steps	1. The VPN client contacts the VPN server and requests tunnel establishment 2. The VPN server contacts the Authenticator, and receives a confirmation of the client's authentication status 3. The VPN server completes the tunnel establishment with the VPN client
Variations	#2 The user is not yet authenticated, perform supplicant authentication #2 VPN server not responding, RETURN error #3 Authentication fails, RETURN error
Issues	

Table 5.7: Use Case - Authenticate

Use Case	7. Perform Authenticator Authentication
Description	The supplicant performs authentication with the Authenticator.
Actors	Supplicant, Authenticator, IdP, SIM
Assumptions	
Steps	1. The supplicant contacts the Authenticator and requests authentication (EAP-Request) 2. The supplicant reads the IMSI from the SIM (UC 8) 3. The supplicant provides the Authenticator with the IMSI read from the SIM (EAP-Request) 4. The Authenticator fetches authentication vectors from the IdP (UC 9) 5. The Authenticator returns a challenge (EAP-Response) 6. The supplicant asks the SIM to run the GSM algorithm (UC 10) 7. The result is returned to the Authenticator (EAP-Request) 8. The authentication is completed (EAP-Success) 9. The Authenticator returns the authentication status to the IdP
Variations	#4 The IdP denies the authentication, RETURN failure #7 The result is invalid, authentication fails, RETURN failure
Issues	

Table 5.8: Use Case - Perform Authenticator Authentication

Use Case	8. Get IMSI
Description	Read the IMSI from the connected SIM card. The IMSI is stored in a file on the card, which need to be selected.
Actors	Supplicant, SIM
Assumptions	"Start Sim Communication" was successful "Verify CHV" was successful
Steps	1. Select the file DF_{GSM} 2. Select the file EF_{IMSI} 3. Read the binary data from EF_{IMSI} 4. Convert the data to a decimal IMSI 5. Return IMSI
Variations	
Issues	Is the IMSI stored in the same way on all SIMs?

Table 5.9: Use Case - Get IMSI

Use Case	9. Request Authentication Vectors
Description	The Authentication queries the IdP for authentication vectors after receiving an authentication request from the Supplicant.
Actors	Authenticator, IdP
Assumptions	The Authenticator is configured with a valid IdP, and a trust relationship between them exists
Steps	1. After receiving the IMSI from the Supplicant the Authenticator contacts the IdP 2. The Authenticator requests authentication vectors for the IMSI 3. The IdP returns a set of authentication vectors to the Authenticator 4. The Authenticator stores the authentication vectors
Variations	#3 The IdP denies the authentication, RETURN failure
Issues	

Table 5.10: Use Case - Request Authentication Vectors

Use Case	10. Run GSM Algorithm
Description	The Identity Provider issues RANDs as a challenge. The Supplicant has to run the GSM algorithm on the SIM, to calculate the correct SRES values.
Actors	Supplicant, SIM
Assumptions	"Start Sim Communication" was successful "Verify CHV" was successful
Steps	1. Select the file DF_{GSM} 2. Send the "Run GSM Algorithm" APDU, containing the RAND values, to the SIM 3. Send the "Get Response" APDU to the SIM 4. Read the $SRES/K_C$ value from the response 5. Return $SRES/K_C$
Variations	
Issues	Do we need the Get Response APDU?

Table 5.11: Use Case - Run GSM Algorithm

Use Case	11. Process Authentication Status Request
Description	The VPN Server needs to authenticate a connecting user, requests authentication details from the Authenticator.
Actors	VPN Server, Authenticator
Assumptions	"Perform Supplicant Authentication" was successful
Steps	1. The Authenticator is contacted by the VPN Server 2. The VPN Server queries the status of a user 3. The Authenticator looks up the user status 4. The Authenticator returns the user status to the authentication server
Variations	
Issues	What should be returned to the VPN server?

Table 5.12: Use Case - Process Authentication Status Request

5.5 Message Sequence Diagrams

In order to better be able to select the components and design the structure of the authentication system, the message sequence diagrams of the system should be analyzed. This will better show which components communicate, and the order of the messages sent between the different components.

Message Sequence Diagrams show which messages are sent between the components, and which messages trigger the processing and sending of other messages. The diagram shown in figure 5.2 show the messages sent between the Supplicant and the Authenticator when the VPN Client sends the initial authentication request.

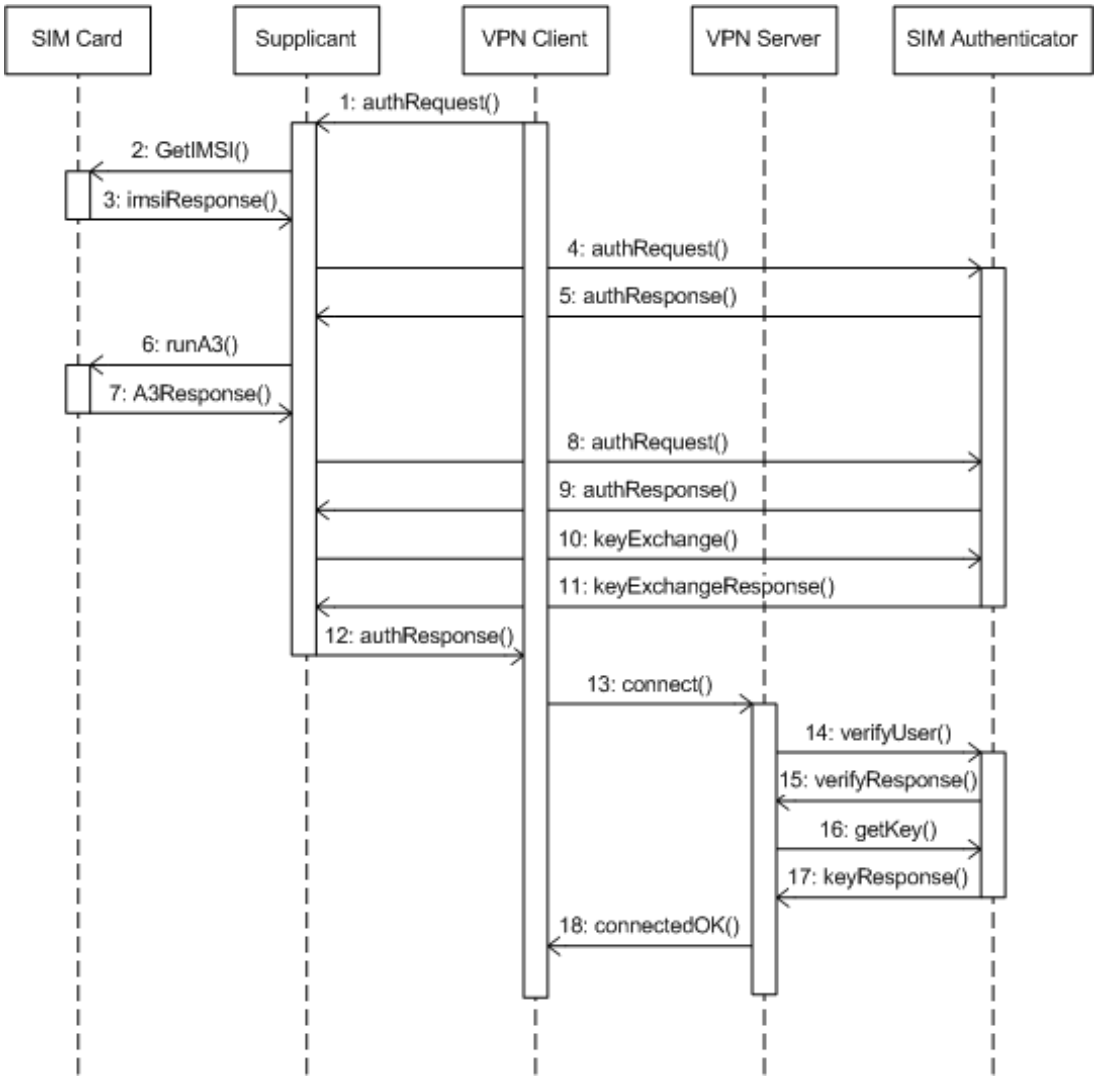

Figure 5.2: Message Sequence Diagram - Full Authentication process between the Supplicant and the SIM Authenticator

We see the main message exchange is between the Supplicant and the Authenticator, and these messages is a variant of the EAP-SIM messages shown in figure 3.17, with the occasional request sent to the SIM in order to acquire the IMSI and the SRES responses. Finally, a message is sent to the VPN Client with the key to use for encrypting the

tunnel.

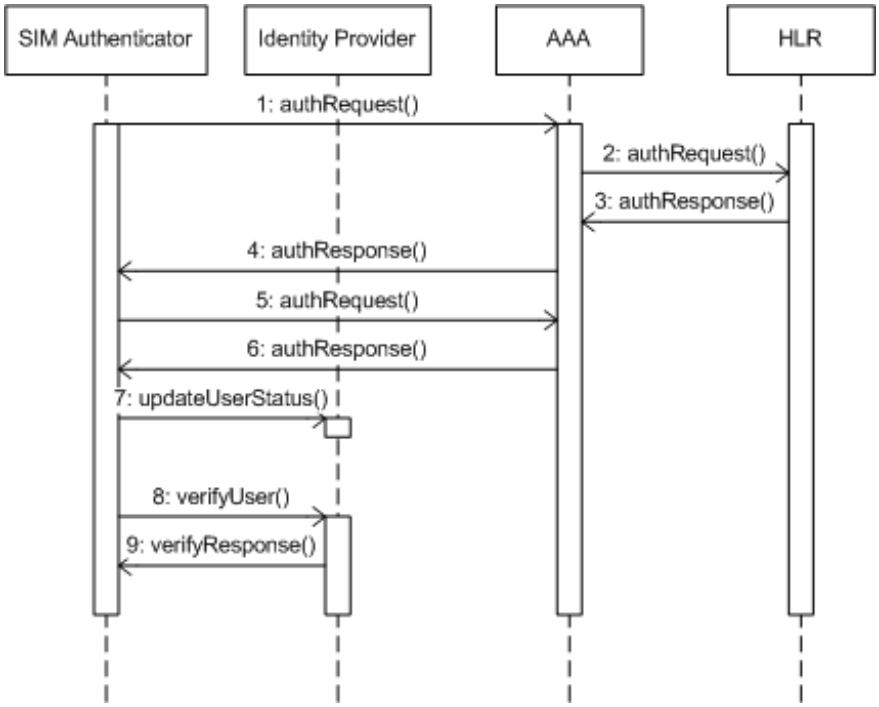

Figure 5.3: Message Sequence Diagram - Authentication process between the Authenticator and the HLR

The message sequence chart in figure 5.3 shows the message exchange between the Authenticator and the HLR of the GSM network. The first 7 messages are related to the initial authentication and the user registration in the Identity Provider. The exchange of the last 2 messages is triggered when the VPN Server request user authentication when the VPN Client requests establishment of the VPN tunnel.

Chapter 6

System Design

The system design will be described in this chapter. It will describe the software components included in the system, and the corresponding packages and classes.

6.1 Components

This section will describe the components of the VPN SIM authentication system. The main task of the system is split up into several sub-tasks, and these are performed by the different components described here.

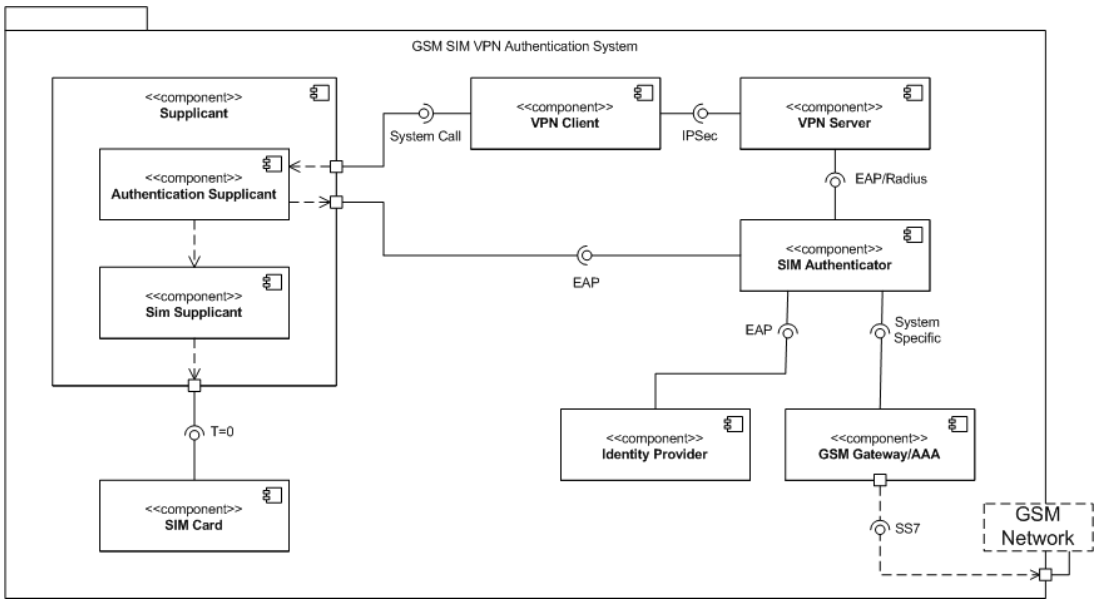

Figure 6.1: High Level Component Diagram

6.1.1 Supplicant

None of the components of the Supplicant currently exist, and will have to be implemented for the prototype.

The overall task of the Supplicant, shown in figure 6.2, is to receive authentication requests and to perform authentication toward the Identity Provider or AAA-server, via the Authenticator. To perform the latter, it needs to communicate with the SIM using APDUs and run the A3 authentication algorithm. These tasks are split into two subtasks, handled by two components; the Authentication Supplicant and the SIM Supplicant.

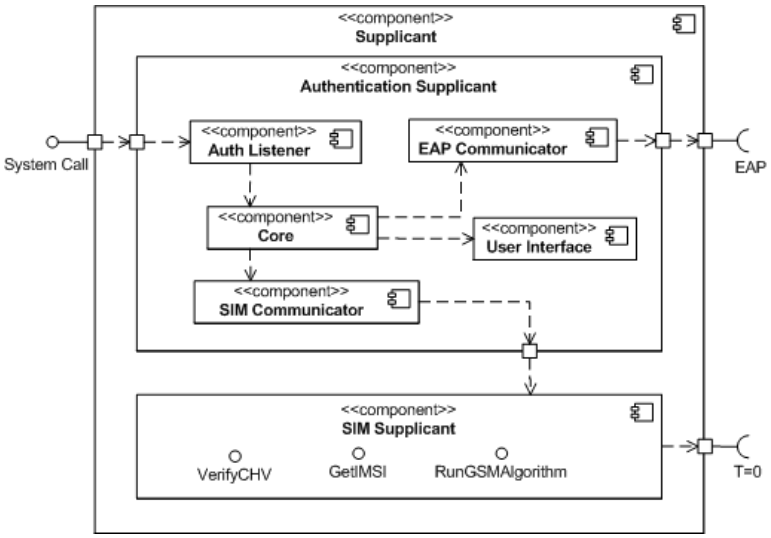

Figure 6.2: Component Diagram - Supplicant

Authentication Supplicant

The Authentication Supplicant is the part of the Supplicant that is the entry point for authentication requests from the VPN Client. It is initiated when the VPN client requests authentication, and terminates when the authentication has been performed against the Identity Provider.

The Authentication Supplicant consists of five subcomponents; Authentication Listener, EAP Communicator, SIM Communicator, Core and User Interface.

The Authentication Listener listens for incoming authentication requests. In this context they come from the VPN client.

The EAP Communicator handles the communication with the Authenticator, sending and receiving messages during authentication.

The SIM Communicator handles the communication with the SIM Supplicant, sending and receiving messages during authentication.

The Core is the main component, which performs calculations and handles communication between the other components as well as with the SIM Supplicant.

The User Interface is responsible for displaying messages to the user, and requesting information when needed.

The Authentication Supplicant communicates with the Authenticator via a pre-defined protocol, for instance EAP-SIM.

SIM Supplicant

The SIM Supplicant is responsible for providing an interface for the Authentication Supplicant. The SIM Supplicant can either be located together with the Authentication Supplicant and communicate with the SIM via a SIM reader connected to the PC, or it can be located on a mobile phone, communicating directly with the SIM. The communication between the Authentication Supplicant and the SIM Supplicant depends on the location of the SIM Supplicant. If it is located on a different device, they communicate over a communication protocol like for instance Bluetooth, and if they are co-located, direct method calls can be used.

The communication with the SIM is performed by sending APDUs defined in GSM 11.11 [ETS95] over T=0.

6.1.2 SIM Card

The SIM Card exists, and is defined in GSM 11.11 [ETS95].

The interfaces of SIM Cards are described in GSM 11.11. There are several interfaces, for performing various calculations and reading data, but the SIM Supplicant only requires a subset of them. Figure 6.3 shows the needed interfaces provided by the SIM.

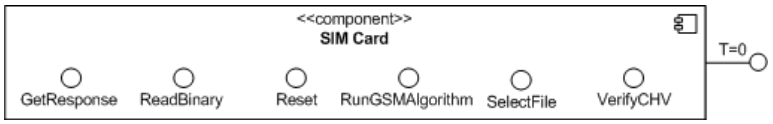

Figure 6.3: Component Diagram - SIM Card

6.1.3 (SIM) Authenticator

None of the components of the SIM Authenticator currently exist, and will have to be implemented for the prototype.

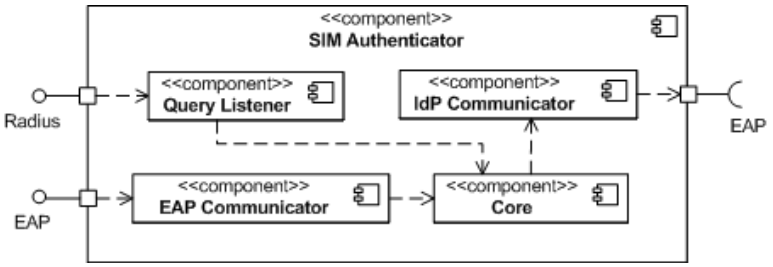

Figure 6.4: Component Diagram - SIM Authenticator

The SIM Authenticator is responsible for receiving authentication requests from the SIM Supplicant. This component exists in order to limit the number of trust relationships the Identity Provider (IdP) or AAA-server has to manage. Without this component every SIM Supplicant needs to establish a trust relationship with the (or

every) IdP/AAA, which primarily does not scale but also introduce potential security risks. By hiding the IdP/AAA behind an Authenticator, the number of connections and trust relationships it has to manage is reduced to a minimum. It is also configured to receive connections from the VPN Server, in order to verify the authentication of users and to provide the encryption key for the VPN tunnel

6.1.4 VPN Client

A goal is that it should be up to the user to choose which VPN solution to implement, and that the GSM SIM authentication should be easy to add. To achieve this, the VPN Client has to support some sort of scripting or plugin which allows it to request VPN tunnel encryption keys before establishing the connection.

6.1.5 VPN Server

The VPN Server needs to support querying the Authenticator for authentication information. Most VPN servers solutions support the RADIUS protocol for querying Authentication Servers, so this is the preferred way of communicating with the Authenticator.

The authentication mechanisms supported in VPN server solutions varies a lot. The simplest authentication mechanisms use username/password combinations, while the more advanced support Smart Cards or One-Time-Passwords (These are described in section 3.7.4). One challenge of adding SIM authentication as an option is to not interfere with existing authentication mechanisms.

One approach is to integrate the authentication with the establishment of the Point-to-Point Tunneling Protocol (PPTP). EAP is for instance supported by Windows XP out of the box, and allows developers to add different modules based on the desired authentication mechanism. However, severe security flaws have been found in the Microsoft implementation of PPTP [Sch], so other solutions have to be investigated.

Another possibility is to integrate SIM authentication with the open source VPN solution OpenVPN, described in section 4.1, and this is the solution which has been investigated in this thesis. OpenVPN has support for running scripts before channel establishment, on both ends, which allows providing "pre-shared keys" to be used for tunnel encryption. This does not require any rewrite of the OpenVPN code base.

6.1.6 Authentication, Authorization and Accounting (AAA)

The AAA-server is a server located with the mobile operator, and is used for authenticating and authorizing users in addition to registering the services used in order to provide accurate accounting. The AAA-server is connected to the mobile network through an SS7 MAP gateway, which provides protocol conversion between the Internet Protocol domain and the Signaling System 7 domain used in mobile networks like GSM. The AAA-server can be used to authenticate users for the authenticator, but is often placed behind an Identity Provider.

6.1.7 Identity Provider

In order to allow a more transparent authentication through the AAA, but still keep control of user authorization, an Identity Provider (IdP) is introduced. The Identity Provider offers a simple interface for authenticating users and acts as a proxy and forwards authentication requests on to the AAA-server. It can also include federation and SSO functionalities, and will in many cases be located closer to the service provider/company than the AAA.

6.2 Interfaces

All of the communication performed between the different components of the systems has to meet strong security requirements. The main goal of the system is to establish a VPN connection with strong security characteristics, and this directly defines the required minimum security of the other links in the system. This security does not necessary involve encryption, as some protocols allow for secure exchanges over public links like for instance Diffie-Hellman, described in section 3.8.

The different interfaces of the system are described below, along with their different security parameters.

6.2.1 VPN Client – VPN Server

The currently existing interface between the VPN Client and the VPN Server remains unchanged. The goal is to allow existing VPN Client/Server implementations to be reused, so only the authentication part of these systems should be altered.

6.2.2 VPN Client – Supplicant

This interface does not exist today, and no standardization of this interface exists. It must at least, as indicated by the message sequence diagrams in 5.5 support the method *authRequest()* which initiates the authentication of the user, and signals the SIM Supplicant to contact the SIM Authenticator. The Supplicant receives the required data, and answers the method call from the VPN Client with the encryption key to use.

This interface is between two applications running on the same system, which greatly reduce the attack surface. Measures must however still be taken to minimize exposure of the encryption key, as some applications could be malicious, like viruses and spyware.

6.2.3 SIM Supplicant – SIM

The interface between the SIM Supplicant and the SIM Card is based on Smart Card APDUs. These messages are sent to a SIM Card Access Device using either a physical connection, like USB, or across a Bluetooth links for mobile phones supporting SIM Access Profile. These access devices are described in section 3.6. This interface must support the method *verifyPIN(PIN)* which takes a PIN code and verifies it towards

the PIN on the SIM card in order to enable other methods on the SIM card. Other methods needed are *getIMSI()* which returns the IMSI value of the SIM card and *runGSMAlgorithm(RAND)* which calculates and returns the signed response (SRES) to the challenge random number provided as described in 3.2.

If a physical connection to the SIM is used, the interface can be assumed secure. When a Bluetooth connection is used both authentication and strong encryption have to be enabled, and the Bluetooth link has to use a 16 byte passkey.

6.2.4 SIM Supplicant – SIM Authenticator

The link between the SIM Supplicant and Authenticator is a link with strong security requirements. No secret information should be exchanged across this interface, as any eavesdroppers are likely to listen to this connection. The EAP-SIM protocol is used here, and this protocol does not exchange any secret information. It should still preferably be performed over a secure link, protected by SSL or TLS, to minimize the risk of man in the middle-attacks. After the authentication phase is performed, the Diffie-Hellman key exchange algorithm is used to calculate a secret key on both ends.

6.2.5 VPN Server – SIM Authenticator

This interface exists to let the VPN Server query the authentication status of the user, as well as to exchange the secret key calculated for that specific user. This interface has to use encryption, as this key can not be compromised.

6.2.6 SIM Authenticator – Identity Provider

When authenticating the user, authentication can be performed towards the Identity Provider instead of directly to the AAA. This interface also lets the SIM Authenticator set the authentication status of the user, to allow for Single-Sign-On solutions. This is however not used in this system.

This interface has the same security requirements as *SIM Supplicant – SIM Authenticator*, described in section 6.2.4.

6.2.7 SIM Authenticator – AAA

This interface uses the EAP-SIM protocol, and is similar to the one described in section 6.2.4.

6.3 Class Diagrams

The design of the system starts with an UML analysis of the system going to be built. The package diagram, shown in figure 6.5 shows the main components of the system, which will be used as a basis for the class diagrams that will be introduced below. Note that the following diagrams are initial designs, and will be modified when needed

during the implementation phase. Especially the class diagrams will be extended, after the need for more methods is identified.

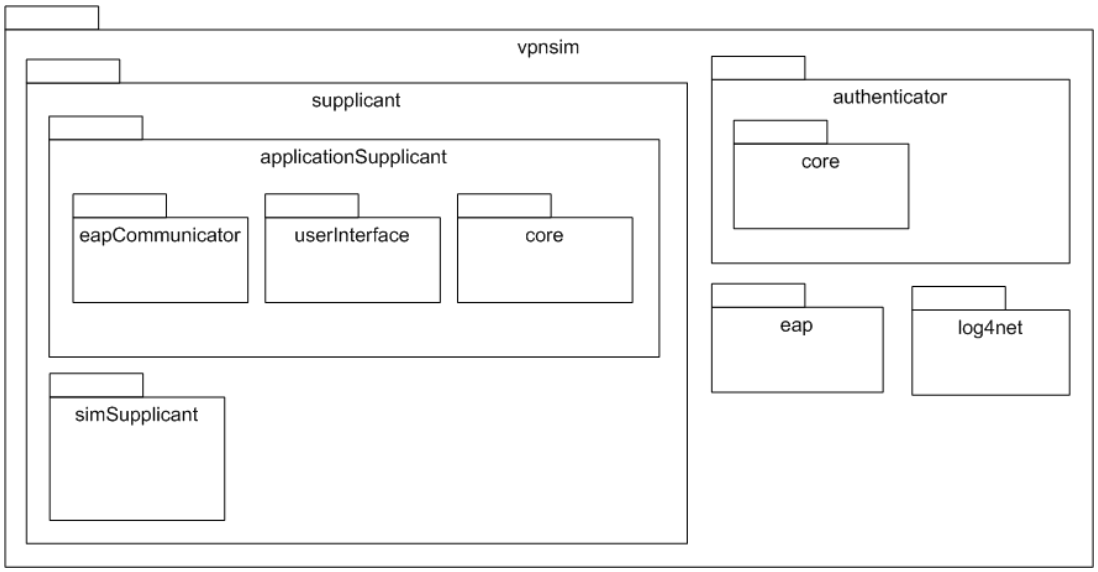

Figure 6.5: Package Diagram, an overview of the packages of the system

The package diagram shows the overall design of the system, and is designed after the component analysis in the preceding chapter. The major components of the system are described in class diagrams, describing the connections between the different classes of the system.

The design of the *SIM Supplicant* is presented in figure 6.6 which show the classes in the *simSupplicant* package.

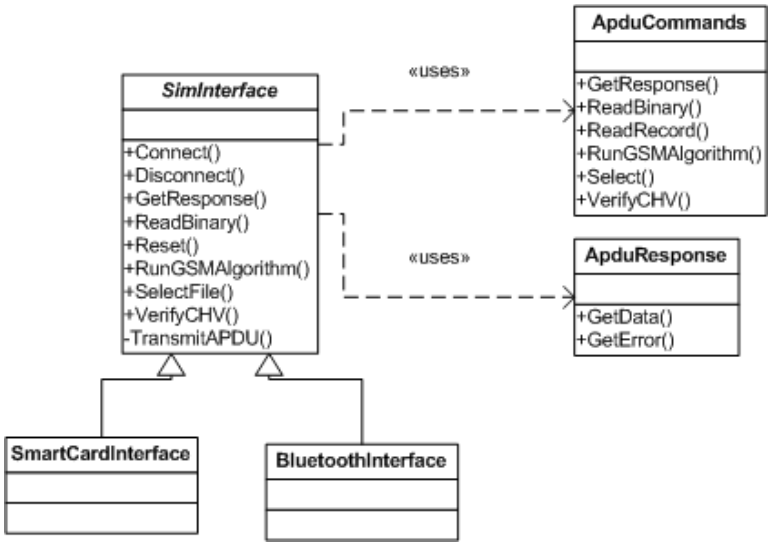

Figure 6.6: Class Diagram of vpnsim.supplicant.simSupplicant

The main component here is the *SimInterface* abstract class, which is used as a foundation for the *SmartCardInterface* class and the *BluetoothInterface* class. These classes provide functions for communicating with the SIM via the different interfaces. Addi-

tional classes can be added in the same way, to allow for any other SIM Interfaces that might emerge. The SIM-Interface classes provide a range of methods, shown in the Class Diagram. When one of the public methods are called, the SimInterface create an instance of the ApduCommand class, which reflect the APDU commands defined in GSM 11.11 [ETS95]. These ApduCommand objects are given as a parameter to the *TransmitAPDU* method which should be overridden in the classes that extend SimInterface, as the way of transmitting APDUs are different for each interface. The TransmitAPDU method sends the byte array representing the APDU to the SIM.

When the SIM received a valid APDU, it returns a byte array response, which is used to create an *ApduResponse* instance, which can be returned to the calling class.

The *Application Supplicant* is the main part of the Supplicant application, and its design is shown in figure 6.7. This diagram shows the classes contained in the *vpnsim.supplicant.applicationSupplicant* package.

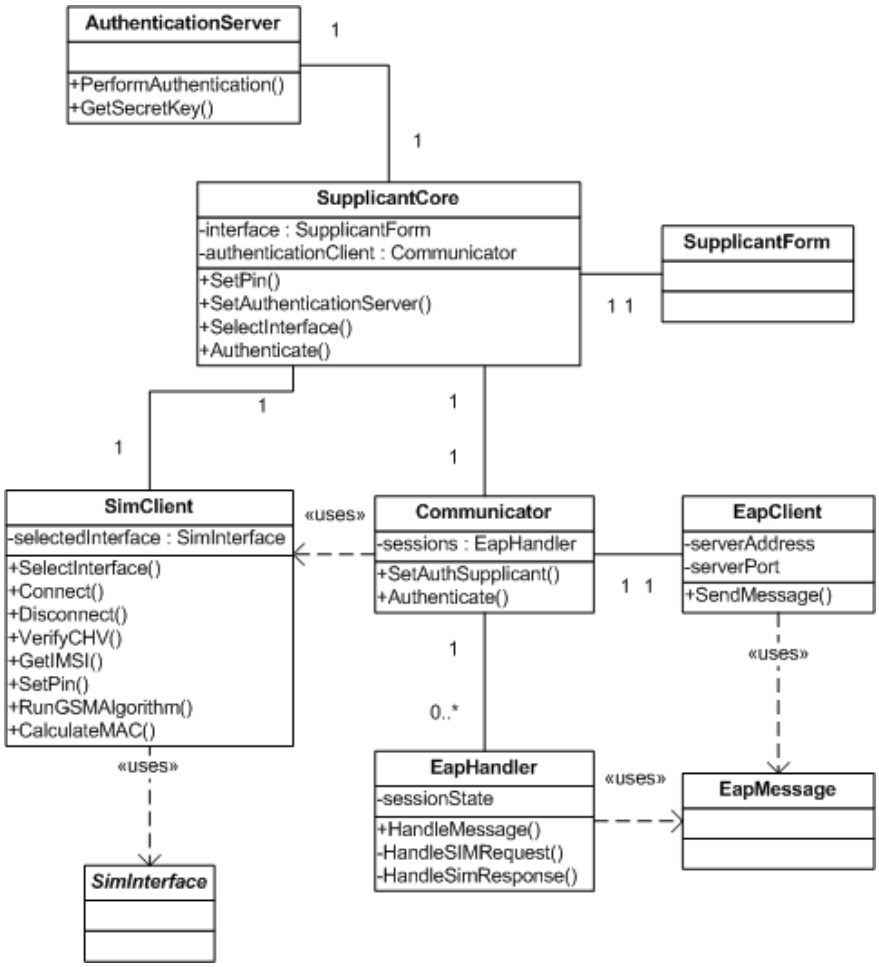

Figure 6.7: Class Diagram of vpnsim.supplicant.applicationSupplicant.*

The main component of the Application Supplicant is the *SupplicantCore*, and this is the class that is initially instantiated. This class then creates an instance of *AuthenticationServer* in order to be able to respond to incoming authentication requests, an instance of *SimClient* in order to query the SIM for the required information, an instance of *Communicator* to be able to perform authentication with the authenticator

and an instance of SupplicantForm to allow the user to for instance set PIN code and configure the Authenticator IP address.

When authentication is requested, through the AuthenticationServer, the Authenticate method in the Communicator object is called. This creates an instance of the *EapHandler*, in order to create the responses based on the requests sent by the Authenticator. Any SIM communication that needs to be performed is performed by the Sim Client.

The EapClient sends *EapMessages* to the Authentication Server, and this class is described below.

The *Authenticator* is contained the package *vpnsim.authenticator* and is described in figure 6.8.

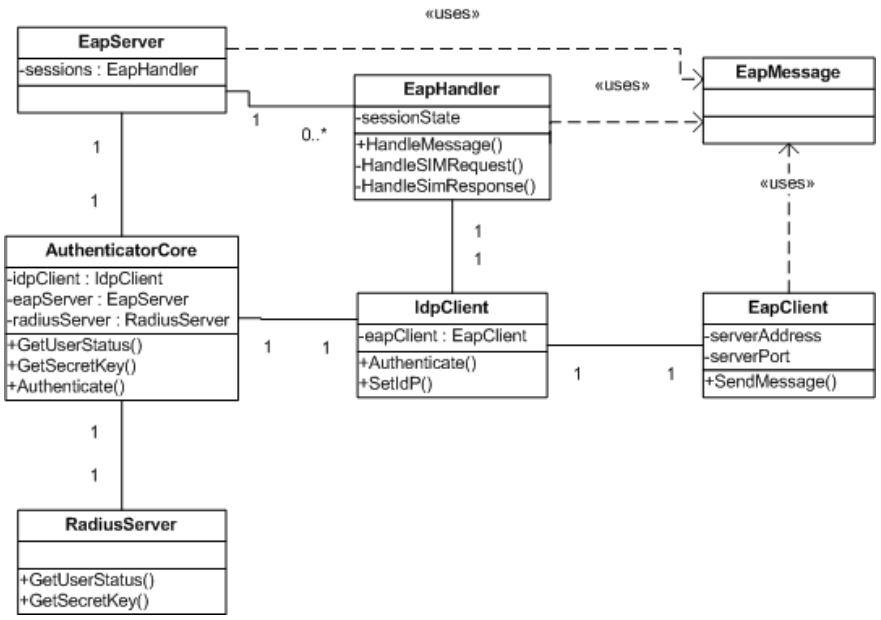

Figure 6.8: Class Diagram of vpnsim.authenticator.*

Its main component is the *AuthenticatorCore* class. This class is instantiated on startup, and creates an instance of *RadiusServer* in order to receive UserStatus requests, an instance of *EapServer* in order to process incoming authentication requests and an instance of *IdpClient* in order to forward these request to the Identity Provider or AAA. The EapServer creates instances of EapHandler when an incoming authentication EapMessage is received, and the EapHandler processes and replies with an appropriate EapMessage.

The IdpClient works much in the same way as the Communicator class of the *vpnsim.supplicant.eapCommunication* package, except the IdpClient doesn't query the SIM for the required information, but usually just forwards the package to the Identity Provider.

Figure 6.9 shows the supporting classes, which are used by the different components of the system. The EapMessage is used in both the supplicant and the authenticator to send EAP-Requests and Responses back and forth. It contains the fields described in RFC 4182 [HS06] and can contain any number of EAP-Attributes.

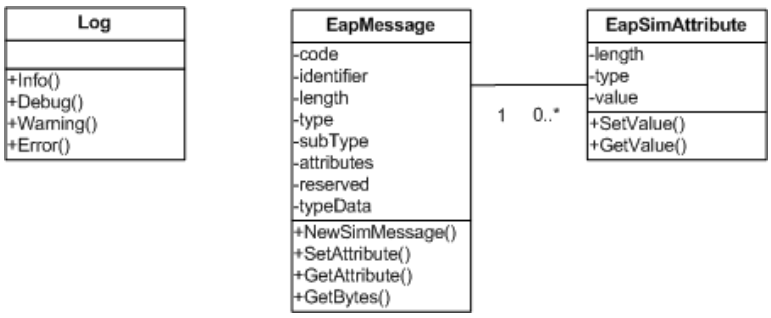

Figure 6.9: Class Diagram of Supporting Classes

The *Log* class is used to gather and filter log data, and is used by most of the other classes. It allows for easy debugging, as well as provide additional information to the user.

6.4 Summary

The design has shown that by using existing protocols and technologies, a system SIM authentication mechanism for VPNs can be created. To have a functioning prototype, the Supplicant and the Authenticator have to be created. The VPN Client and Server also need plugins to retrieve the encryption keys from the Supplicant and Authenticator, respectively.

Establishment of a VPN connection consists of two steps; authentication and establishment of the secure tunnel. In this system, authentication starts when the user requests the establishment of the secure tunnel. The VPN Client sends an authentication request to the Supplicant, and the Supplicant performs the authentication towards the Authenticator, which relays the authentication to the Identity Provider or the AAA. The authentication finishes by generating a key using Diffie-Hellman, and the Supplicant and the Authenticator ends up with the same secret key. The key is then returned to the VPN Client, and the VPN Client can continue the tunnel establishment. As the VPN Server receives this establishment request, it asks the Authenticator for the authentication status of the connecting user, and receives the secret key if the user is authenticated. Now the VPN Server and the VPN Client can establish a symmetric encrypted tunnel using the secret key as a pre-shared encryption key.

The component diagrams give us an idea of how a prototype can be composed, and the class diagrams provide instructions on how to implement the classes. The process of implementing the prototype will show if (and how) the class diagrams need to be modified. The implementation is described in the following chapter.

Chapter 7

Implementation of a Prototype

This chapter will present the implementation of the prototype, and will describe the configuration of the components involved.

7.1 Deployment Diagrams

A deployment diagram was created to give an overview of the system components and their interconnections. It shows which components reside on which devices, as well as the requirements for this system. This diagram is shown in figure 7.1.

7.2 Implementation

A description of the SIM Supplicant implementation is described here.

7.2.1 SIM Communication

The SIM supplicant's task is to relay information from the connected SIM Card to the other parts of the system. This can be performed using two methods; the built in SIM reader in the phone, and the Smart Card reader connected to the computer. The SIM interface is already defined, so the communication with these two readers is done as described in GSM 11.11.

Smart Card Reader

The Smart Card reader was designed as the main form of communication with the SIM, as the readers are easy to get hold of, and provide a good level of security. The C# implementation bases itself on the WinScard.dll, which is a native Windows library. A Smart Card library named *SCLib*, written by Digvijay Chauhan, was used to simplify the process of connecting to the Smart Card reader and querying the SIM Card.

The libraries let the supplicant query the Operating System for any connected Smart Card readers. A dialog is presented to the user, allowing the user to choose which card

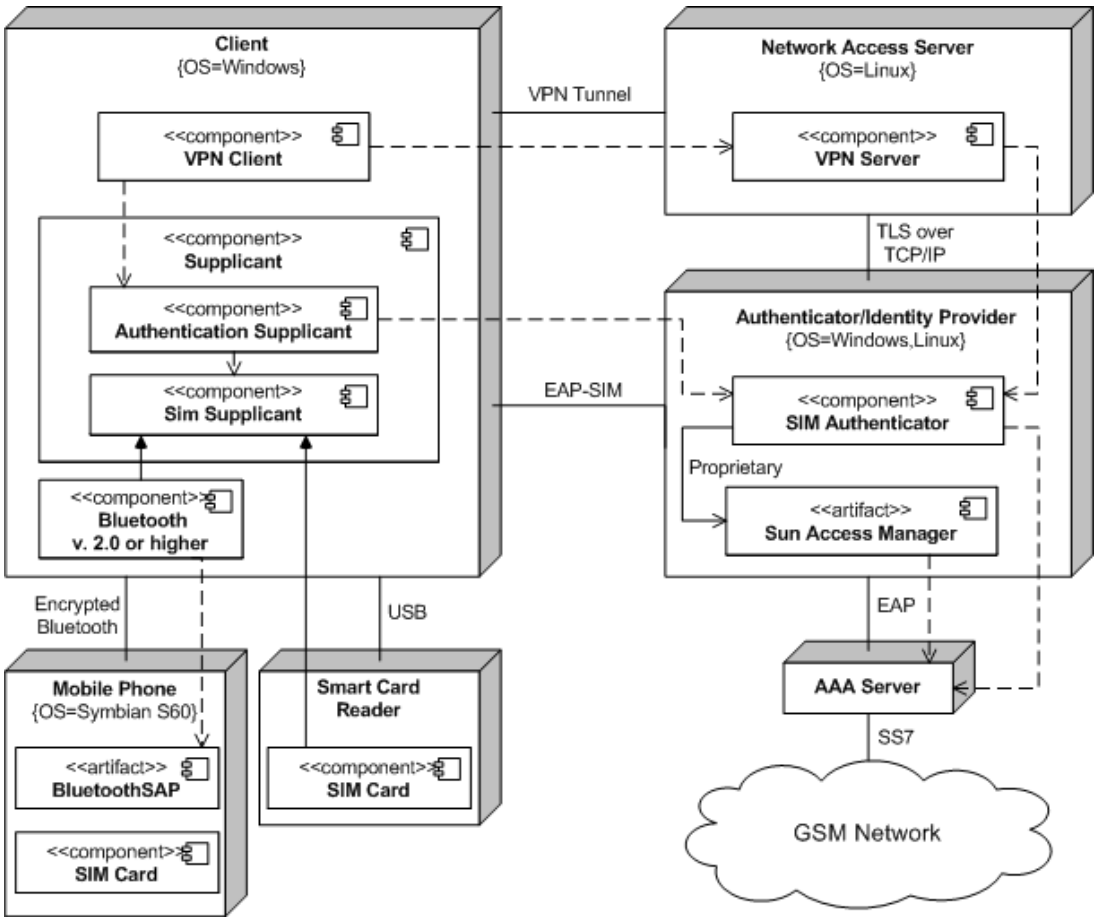

Figure 7.1: High Level Deployment Diagram

reader to use for communicating with the SIM. If a SIM is present in the card reader, the Supplicant queries the card, and tries to authenticate using the PIN code provided by the user.

When a authenticated connection to the card is established, the supplicant can transmit APDUs to the card to perform any of the functions supported by the Smart Card, described in GSM 11.11.

Built in SIM Reader

By using a phone supporting the Bluetooth SIM Access Profile, or SAP, the computer can establish a wireless connection to the SIM Card inserted in the phone. After activating Bluetooth on the phone and setting the phone to discoverable mode, the SIM Supplicant can find the mobile phone when scanning for nearby Bluetooth devices. This of course requires the computer to support Bluetooth.

After the Phone is connected, the Supplicant requests the connection to the SIM Access Profile. If the phone does not support this profile, the connection request is denied. If the profile is supported, a passkey must be provided to complete the connection. This passkey has to be 16 digits.

Only newer phones support the SIM Access Profile. Windows can show a list of the supported profiles by going to the property page of the Bluetooth Device while holding down the *shift* key.

After the connection is established, APDUs can be sent to the device, the same way as when using the Smart Card reader. This allows the computer to view the established Bluetooth connection in much the same way as the Smart Card Reader.

Windows XP service pack 2 and newer includes a native Bluetooth stack which allows communication with Bluetooth hardware, but only complex interfaces are visible. The SIM supplicant uses the shared-source library 32feet.NET developed by In The Hand[1], which simplifies setup and configuration of Bluetooth communication links.

7.2.2 Network Communication

The main communication protocol used in the VPN authentication system is EAP, or Extensible Authentication Protocol. The implementation of this protocol is described in this section. Finally, the communication between the VPN Server and the Authenticator is described.

EAP

EAP is the most important protocol of this system, as it handles all the communication between the SIM Supplicant and the SIM Authenticator. As the signaling between these two components are those of a normal GSM authentication dialog, EAP-SIM was an obvious choice of protocol. Unfortunately, an implementation of EAP, or EAP-SIM, did not exist at the time of implementation so an EAP library was written in C#. The EAP library includes support for all the types used during normal EAP-SIM communication, as well as two extra messages which will be described below.

The supported messages of the EAP library are the basic *EAP-Identity*, *EAP-Request*, *EAP-Response*, *EAP-Failure* and *EAP-Success* messages. In addition, the following EAP-SIM messages are implemented: *EAP-Request/SIM/Start*, *EAP-Response/SIM/Start*, *EAP-Request/SIM/Challenge* and *EAP-Response/SIM/Challenge*.

Two extra EAP-SIM messages were also added to the ones specified in the protocol, to allow for a Diffie-Hellman key exchange over EAP-SIM. These are the *EAP-Request/SIM/KeyExchange* and *EAP-Response/SIM/KeyExchange* messages. The Subtype value of these messages is set to 255, which is for Experimental use. They take the attributes AT_KEYPARAMG, AT_KEYPARAMP and AT_PUBLICKEY, with types 253, 254 and 255 respectively. The format of the packet is illustrated in figure 7.2.

The *Code* field of the packet is set 1 or 2, specifying whether it is the request or response message. The request message take all three parameters, to allow for sending of the g and p parameter as well as the Diffie-Hellman public key (described in section 3.8). The response message only take the AT_PUBLICKEY attribute. The *Length* of the packet is $8 + p$ with p being the total length of the Diffie-Hellman attributes.

[1] 32feet: http://www.32feet.net

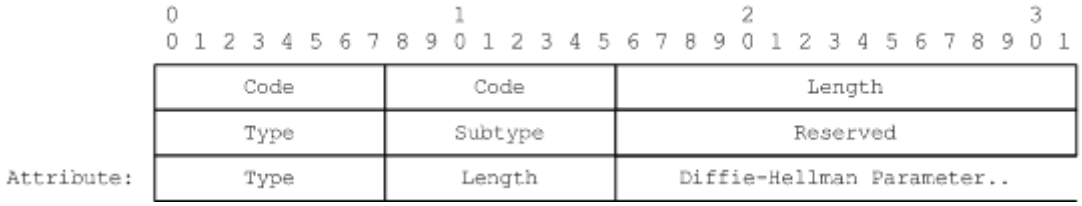

Figure 7.2: EAP SIM Key Exchange Packet Format

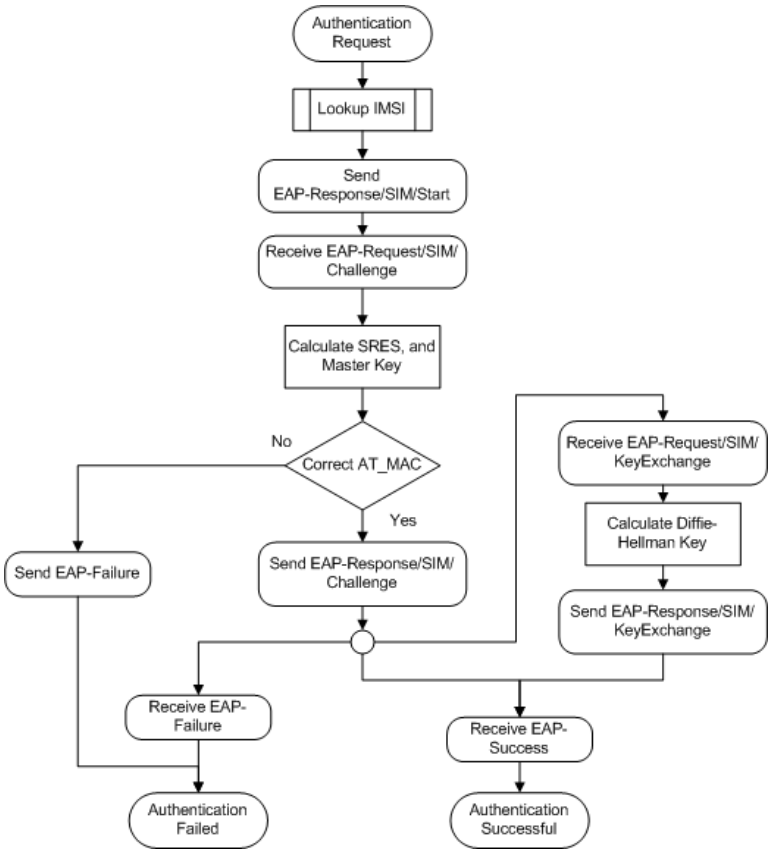

Figure 7.3: EAP Message Exchange State Chart, Client Side

An EAP protocol handler is also included, which handles the processing of EAP messages, and creates the appropriate replies. After a successful EAP-SIM authentication, the EAP-Request/SIM/KeyExchange message is sent, containing the first message of a Diffie-Hellman key exchange. When this is received by the client, the Diffie-Hellman response message is created, and sent to the server in an EAP-Response/SIM/KeyExchange. Then the SIM authentication is finalized by sending a normal EAP-Success message to the client.

The EAP Message exchange is described in two state chart diagrams, Figure 7.3 and 7.4.

> Note: As the goal of this prototype was to evaluate VPN authentication via GSM SIM, only a subset of the EAP spec was implemented in the current version of the EAP library. Specifically, the encryption and hash algorithms used in the EAP specification has been simplified. Dummy functions are

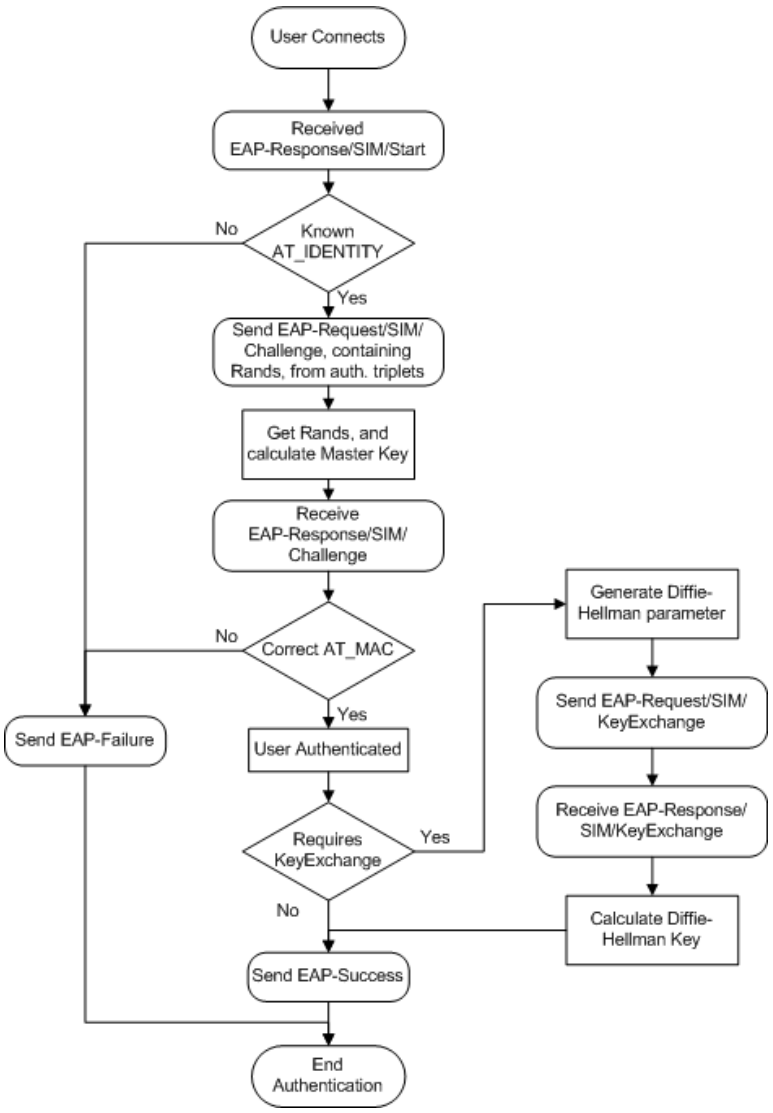

Figure 7.4: EAP Message Exchange State Chart, Server Side

used, so this functionality can be added later if desired. Also, the key derivation mechanisms in the EAP-SIM RFC [HS06], using the PRNG specified in NIST Federal Information Processing Standard Publication 186-2 [oST00], has been simplified. Instead of deriving the Transient EAP Keys for protecting the EAP-SIM packets according to the RFC, a variant of the Master Key is used. This simplification was made to save time, as implementing the full protocol was not part of the assignment.

EAP over UDP

In this prototype, the EAP messages between the Supplicant and the SIM Authenticator is sent over UDP, the User Datagram Protocol. UDP is not as reliable as TCP (Transmission Control Protocol, RFC 793 [Pos81]), but is easier to implement and allows fast and efficient message exchanges which makes it possible to perform the EAP

authentication quickly. For an implementation that requires better reliability, TCP should be considered. TCP also allows the use of security features like SSL and TLS. However, the GSM authentication mechanism is intended to provide security by itself, using the secret key stored in the SIM.

The Authentication Server by default listens for incoming UDP connections on port 7746, but this can be customized. The Supplicant has an option for specifying the IP and the port number of the Authentication Server. When an incoming connection is received, the EAP-handler is called and the message processing is initiated

Authenticator Key Query

When the VPN Server receives an authentication request from the user, it has to query the Authenticator. This process was intially planned to use Radius, but because of poor Radius support in .NET, a custom message exchange was used.

To fetch the user key, the VPN Server sends the username to the Authenticator on a dedicated port using UDP, and the Authenticator returns the key of the user, or an empty string to indicate unknown or unauthenticated user.

7.3 Configuration

7.3.1 VPN Client

The prototype uses the OpenVPN client to initiate VPN connections. There are several clients that could have been used, but OpenVPN has good support for remote management, which enables easy integration with the Application Supplicant of the system.

Before the VPN Client is started, it has to be properly configured. A sample configuration file is shown in appendix A.2. Also, the Certificate Authority (CA) certificate has to be placed in the OpenVPN config directory. A step by step guide to creating the CA certificate is shown in section A.1.

When the VPN Client is started it creates a Management Interface that listens on local IP 127.0.0.1, TCP port 4593. The Application Supplicant connects to this port as soon as it is opened. On this channel OpenVPN issues a *PASSWORD* request, which the Application Supplicant interprets as an SIM authentication request. When the SIM authentication is complete, the Application Supplicant writes the username and password to this channel, and the VPN Client has what it needs to establish a connection with the VPN Server.

When connecting to the VPN Server a VPN tunnel is first created, based on the server certificate. By validating the certificate against the CA certificate, the user can authenticate the VPN Server. After the tunnel is established the VPN Client sends its username/password combination. The VPN Server then accepts or denies the authentication request, as described in the next chapter.

The username supplied to the VPN Client over the Management Interface is the IMSI retrieved from the SIM card. The password is an MD5 hash of the key that was derived

using Diffie-Hellman, as described in section 7.2.2. To the VPN Client, this is just a normal username/password pair.

7.3.2 VPN Server

The VPN server used in this prototype is the server part of the OpenVPN package (actually, the same program as the VPN Client, but with different parameters). It is easy to configure, and has plugin support and can be extended with authentication scripts, which allows for easy integration with the Authenticator.

Before the VPN Server can be started, a few steps have to be completed. First, a CA certificate and a server public/private key pair have to be created as shown in appendix A.1, and the configuration file has to be created. A sample configuration file can be found in appendix A.3. Also, the user authentication script has to be put in the *scripts* directory. This script is shown in appendix A.4. The script requires perl support in the operating system.

When the VPN Client connects to the VPN Server, an encrypted tunnel is created between the client and the server. Over this tunnel, the Client sends its username (the IMSI) and its password (the MD5-hashed key) to the server. When the server receives the authentication request, a script is called with the IMSI and the key-hash as parameters. This script connects to the Authenticator and asks for the key associated with that specific user. If the user is not authenticated, a null value is returned, and the VPN Server disconnects the client. If the user has authenticated at the Authenticator, the key byte sequence is returned to the VPN Server. The hash of this byte sequence is then compared with the hash received from the client. If they are equal, the server has proof that the client is the same as the authenticated user.

7.4 Deviations from the Original Design

In the prototype implementation, a few deviations from the original design had to be made. Some because of time constraints, and other for practical reasons.

7.4.1 EAP key-derivation algorithms

As mentioned in section 7.2.2, one of the algorithms used in the generation of EAP messages has not been created according to the RFC (RFC 4186 [HS06]). This is the pseudo-random number generator function used to derive the Transient EAP Keys, the Master Session Key and the Extended Master Session Key from the Master Key. The Master Key is created according to the RFC, but is used directly for calculating the MAC. This should be redesigned in a non-prototype implementation.

This was done because of the complexity of the algorithm, and because implementing the EAP protocol was not part of the assignment this part was given low priority. Implementing this function later is possible.

7.4.2 Creating SIM Authentication Triplets

Because of time constraints, integration with an Identity Provider or AAA server was not implemented in the Authenticator. Instead the authentication triplets are generated in advance, and stored in a file. During authentication the Authenticator reads this file, filters out the triplets with the IMSI equal to that of the user, and selects three random ones to use.

Before trying to authenticate the SIM, it is vital to the success of the authentication that the Triplet file is first generated on the Supplicant, then copied from the Supplicant to the Authenticator. If the file already contains Triplets with the IMSI of the SIM used, this step can be skipped. If the dummy interface is used, this must be done every time the Supplicant is restarted, because of the random IMSI of the dummy interface.

Integrating the Authenticator with an Identity Provider later should not be too difficult. An alternative TripletSource class forwarding the EAP requests from the supplicant to the Identity Provider could be written and included in the Authenticator.

7.4.3 Verifying User Authentication Status

It was initially planned to use Radius as a way to query the Authenticator for the user authentication status. Because of a lack of Radius libraries for .NET, this was not implemented. Instead, a simple query/response mechanism is used to send the key of the connecting user, described in section 7.2.2.

Radius should have been used here, in order to improve compatibility with other VPN Server authentication mechanisms, but writing a Radius implementation would have taken too much time.

7.4.4 Authentication Key Usage

In this implementation, the key derived from Diffie-Hellman during the EAP authentication is not actually used as the VPN tunnel encryption key, as originally planned. This is due to the strict authentication sequence OpenVPN uses. First it establishes an encrypted tunnel based on the server certificate, to validate the server side of the connection. This sequence may also take a client certificate to authenticate the client, but this is not used in this configuration. Then the username and password is sent to the server over the encrypted tunnel (not required if client certificates are used). This is used to authenticate the client.

It was originally intended to use a static "pre-shared" key for channel establishment, as described in section 6.1.5. However, the pre-shared key solution starts encrypting the data from the first bit, without performing any parameter exchange in advance. This means that the VPN Server has no way of finding the username without first having the pre-shared key, as the incoming data stream would only look like random bits. Requiring that the VPN Server should have this pre-shared key in advance would defeat the purpose of this system.

One solution to this could be altering the connection establishment procedure when using secret keys, so the VPN Client could inform the VPN Server of the username

before enabling encryption on the channel. This way the VPN Server could look up the secret key when the client connects, and retrieve the "pre-shared" key before ordering the VPN Client to enable encryption.

7.5 User Manual

This section will describe the usage of the Supplicant.

7.5.1 Requirements

In order for the application to start, the *.NET framework version 2.0* has to be installed on the computer. This is a free download from Microsoft and can be found easily by searching on their web site[2]. Also, a SIM Reader should be installed on the computer. The application runs without a SIM reader, but only the Dummy interface will then be available. Usage of the dummy interface is explained in the next section.

To be able to use Bluetooth or Smart Card readers, make sure to have the correct device drivers installed. Bluetooth communication only works if the mobile phone supports the Bluetooth SAP profile. Also, the Bluetooth radio in the computer has to support 128bit encryption of the radio link.

7.5.2 Application Supplicant

When the application supplicant is started for the first time, the settings window is displayed. This window is shown in figure 7.5.

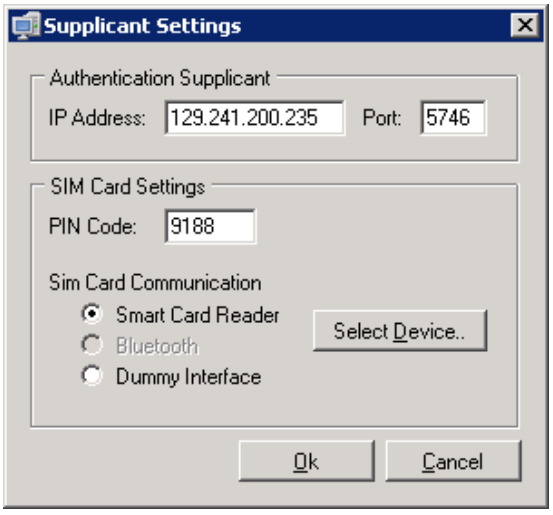

Figure 7.5: Screenshot of the Supplicant Settings window

This window lets the user specify the IP address and port of the authentication supplicant, the PIN code of the user's SIM, as well as the device the user want to use to

[2]Microsoft website: http://www.microsoft.com

communicate with the SIM. The available devices depend on which devices exist on the user's computer.

Smart Card Reader If a Smart Card reader is connected to the computer, this option should be available. If not, verify that the drivers of the Smart Card reader are correctly installed.

Bluetooth If a Bluetooth Radio is available on the computer, this option is available. If not, verify that the drivers for the Bluetooth device are correctly installed.

Dummy Interface This is an interface that is for testing purposes, and simulates a SIM interface. On selection, it generates a random IMSI. Any PIN code can be used.

When either the *Smart Card Reader* or *Bluetooth* device is selected, clicking *Select Device* will display the dialogs shown in figure 7.6 or 7.7, respectively.

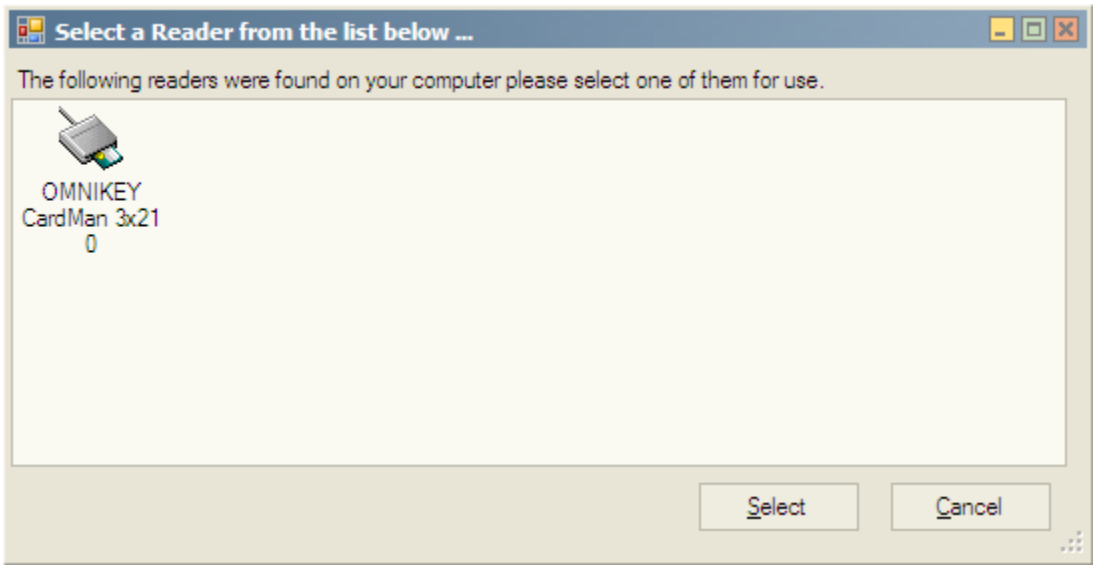

Figure 7.6: Screenshot of the Smart Card selection window

> **Note:** When selecting a Bluetooth phone, this phone has be paired with the computer in advance. This is due to limitations in the Bluetooth library used. Look for *Bluetooth Devices* in control panel. When pairing, make sure a passkey of **16 digits** is used.

After clicking OK, the application should minimize to the system tray, and the Supplicant starts listening for the OpenVPN Management Interface (described in section 7.3.1). If authentication is initialized, and the Supplicant is authenticated, the tray icon will switch to reflect this status. The normal and the authenticated icon is shown in figure 7.8).

Right clicking on the system tray icon will display the menu shown in figure 7.9. This menu lets you open up the settings window to change configuration, as well as display the log window in order to view the messages sent, and any error messages. A debug-menu is also added to this version, in order to easier test the authentication supplicant. This menu is described in section 7.5.2.

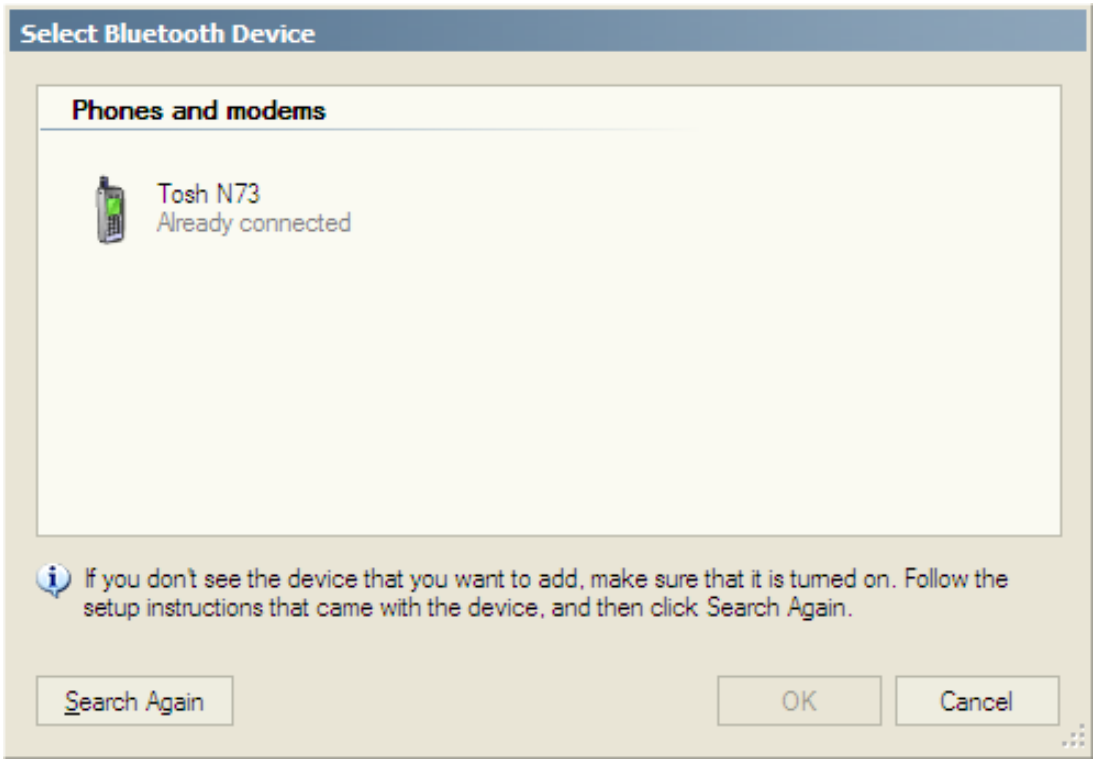

Figure 7.7: Screenshot of the Bluetooth selection window

Figure 7.8: Screenshot of the system tray icons, normal to the left, and authenticated to the right

A log window is also included, to show the details of the authentication process. This is shown in figure 7.10.

Debug Mode

The debug menu option has two sub-options. *Authenticate* and *Generate Triplets*.

Authenticate The Authenticate option starts the authentication with the Authentication Server in order to authenticate the supplicant without being initialized by the OpenVPN Management Interface. Open the log output window to view the communication with the Authenticator.

Generate Triplets As this prototype does not communicate with the AuC in the GSM network, the triplets used for authenticating the user has to be created in advance (to have the same SRES and Kc value as the GSM Algorithm generates). By selecting *Generate Triplets* the SIM is asked to run the GSM algorithm on three random numbers RAND, and returns the correct SRES and Kc response. These triplets are then written to a file together with the IMSI. An example of a triplet file is shown in appendix B. This file must then be copied to the *database*

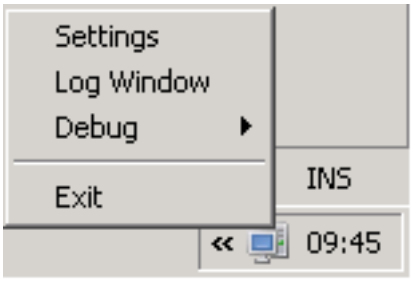

Figure 7.9: Screenshot of the right click menu

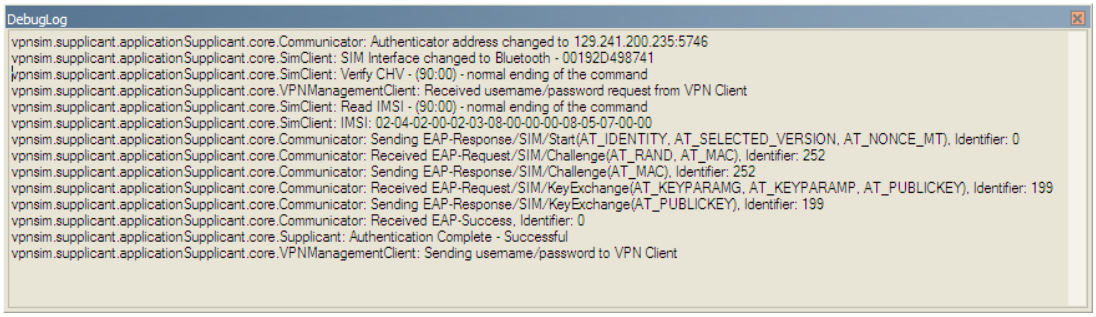

Figure 7.10: Screenshot of the Log Window

directory in the authentication server directory (overwriting any existing ones) in order for it to authenticate the SIM.

7.5.3 Authenticator

When the Authenticator is started, it displays a basic command line window, showing the messages received and the events triggered. Any error messages during SIM authentication is also displayed here. An example of this window is shown in figure 7.11

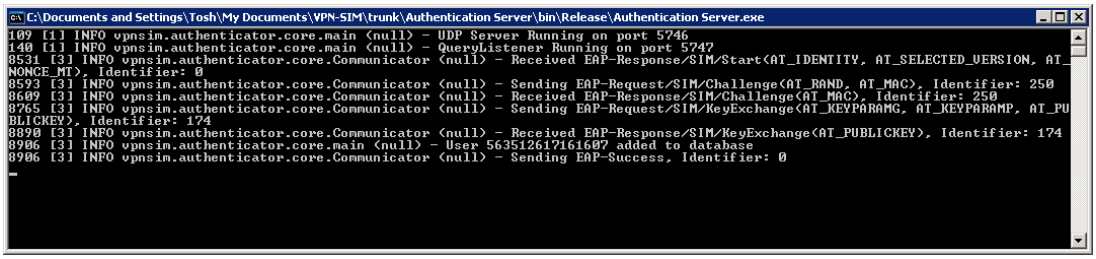

Figure 7.11: Screenshot of the Authenticator

Chapter 8

Discussion

8.1 Technology Choices

When choosing the communication protocols to use for a system like the one presented, an analysis of the different options had to be performed. Considering the task at hand, EAP was chosen for the actual authentication messaging. This since EAP already has support for the messages used for GSM authentication, as well as the other alternative methods suggested. In addition to this, EAP provides an easy extendable mechanism, in order to both modify or develop new authentication methods. The prototype uses EAP-SIM with two custom messages, which allowed both using a standardized way of performing authentication as well as an asymmetric key exchange already proven to be secure.

The tunneling protocol used for the VPN has not been discussed much in this paper. But as the initialization and creation of the VPN tunnel is performed by the VPN client independent of the authentication and key exchange performed in advance, the type of tunnel is not relevant. The only prerequisite is that the VPN solution has to support creation of a tunnel based on static pre-shared keys, instead of exchanging key information across the VPN tunnel.

The language used for implementing the Supplicant was C#.NET. The .NET framework offers excellent integration with Windows and its built in functions, so enabling communication over both Bluetooth and Smart Card Reader was fairly straightforward. Unfortunately, a lack of third party libraries turned out to be a drawback, compared to Java. The .NET framework has no built in EAP functions, and no third party libraries could be found, so an EAP library had to be written to enable the EAP-SIM communication. The development and debugging of this introduced quite a bit of extra work.

8.2 Choice of Identity Provider

The chosen Identity Provider should be a trusted source, as it handles information regarded by the users as secret, namely the IMSI. A malicious Identity Provider could in theory perform a man in the middle attack on users connecting, and acquire secret

information without the users knowing. It could also get access to the information later encrypted in the VPN tunnel, if the keys used for encryption is derived from authentication information.

The most natural Identity Provider is a GSM operator, but it could also be a trusted third party providing the service for the GSM operator. When dealing with multiple networks and more than one GSM operator, the third party solution would probably be the best for all parties.

8.3 Security Considerations

As the background for this thesis is to suggest an improvement to the security properties of a system already considered secure, the resulting security properties of the final system must not be inferior to the original system. This means maintaining the integrity and the confidentiality of both the data used or contained in the different components as well as the data transmitted between two components in the system. As the total security of a system usually boils down to the security of the weakest link, every part of the system is equally important.

The weak points of systems like this are usually found in the interfaces or connections between the different components. Some may be vulnerable to eavesdropping, while others may open up for man in the middle attacks. To evaluate the security characteristics of this system, an evaluation of the interfaces are needed. In the proposed solution, there are two different types of connections. Some connections are long lasting, and are often referred to as *static connections*. These are often controlled by service-level agreements. The shorter, on-demand connections are referred to as *dynamic connections* and have a lower level of control.

The static connections are between the entities listed below:

- SIM Supplicant and SIM Authenticator
- VPN Server and SIM Authenticator
- SIM Authenticator and Identity Provider
- SIM Authenticator and AAA
- AAA and HLR

All these connections are secured with their own security measures, i.e., certificates and SSL/TLS security, and can in this context be ignored when discussing weak points. The connection between the user computer and the SIM card can be either via cable or through Bluetooth. When Bluetooth is used, the devices must be paired, and the pairing is secured using a strong 128 bit user-selected key. This pairing is considered to be a static, long-lasting connection as it is usually performed only on the first connection.

The dynamic connections in this system is limited to the connection between the VPN Client and the VPN Server. Connections secured using a long pre-shared key are considered to be secure as long as the key is kept secret from third parties. As this key is negotiated between the SIM Authenticator and SIM Supplicant over an assumed secure connection, this connection is also secure.

Most of the functionality of this authentication system is based on standardized protocols which have all gone through detailed analysis, and any strengths and weaknesses are already known and asserted by the community. The session key agreement of EAP-SIM has been shown to be faulty when not taking particular precautions [Pat03], but is strong enough when used correctly. This system exchanges the encryption keys using an asymmetric key exchange algorithm, so this weakness is not an issue in this context.

Chapter 9

Conclusion

This master thesis has investigated methods for implementing GSM SIM mechanisms in VPN authentication systems. Underlying protocols and support systems have been analyzed, and a prototype has gone through the phases of software design, implementation and analysis. The system has been tested end to end, though without communication with the GSM infrastructure, and has shown that authenticating the user during VPN tunnel establishment using tokens negotiated through the GSM authentication is indeed feasible.

The core of the SIM Authentication System design is not directly connected to the VPN authentication mechanism, and can in fact be used in a more general setting than described by this thesis. By adding the required key request plug-ins to application requiring authentication, this system can also authenticate other systems. This however requires an additional security evaluation.

By authenticating VPN users using the GSM SIM authentication mechanism the initial goals of the thesis, listed in 1.2 can be met:

- The authentication process for the user is made easier, as it is no longer necessary to remember an additional username/password combination.
- The SIM is an authentication token already owned by most people. By adding this token to the authentication mechanism of the VPN, a VPN administrator can effectively increase the VPN security without having to purchase and distribute new authentication tokens.
- Adding new users to the VPN boils down to adding the IMSI of the new user to the list of approved users. No extra password distribution is required.

9.1 Results

The system presented and analyzed in this thesis presents how a VPN connection between two endpoints can be authenticated by means other than the traditional VPN authentication mechanisms.

A complete system design has been made, and can be found in the design chapter (chapter 6). This design describes the components of the final prototype, and their

role in the authentication process. The prototype implementation (described in section 7) has a few deviations from the original design but none of these are considered to be of major concern. These deviations are listed in section 7.4.

A prototype was implemented in C# to demonstrate the authentication mechanism. The implementation of the prototype is documented, and detailed descriptions of any additions to existing protocols are provided – with accompanying diagrams when needed. A user manual with screenshots is also created, to allow for easy testing of the application. The VPN software, namely the VPN Client and the VPN Server, was evaluated against alternative solutions and the chosen solution was considered to be the most suitable for this prototype. With the appropriate modifications, other VPN solutions could also be used, but this process is not described in this thesis.

The source code of the Supplicant and the Authenticator as well as compiled binaries should be accompanying this thesis, but can also be obtained by containing the Department of Telematics, NTNU and asking for a version of this thesis with accompanying code. The source code contains full documentation of the classes and their public methods, and the binaries should run on Windows XP with the .NET framework..

The project has shown that VPN authentication can be improved by using features of GSM SIM, and that such a system can be implemented using existing technologies by modifying the existing VPN authentication mechanisms.

Based on this master thesis, a paper called *Securing Virtual Private Networks with SIM Authentication* has been written. This paper has been submitted to the *WiMob 2007* conference, held in New York, USA October 8-10. 2007, and is currently under review. This paper is included in appendix C.

9.2 Future Work

Extensions to the Authenticator should be written in order to integrate the presented solution with existing Identity Provider solutions. This will allow for authentication with the GSM network, as initially planned for this prototype. This will also enable federation of identities which lets an authenticated user use multiple services after authenticating once, called Single Sign-On.

A complete EAP library for .NET should be developed and made available. This will greatly ease the implementation of similar authentication solutions, as well as help make EAP an authentication standard. Also, a .NET Radius library would improve the stability of the communication between the VPN Server and the Authenticator.

A standardized interface for providing encryption keys to the VPN Client and Server should be created. As opposed to Radius, which allows 3rd party verification of username/password combinations, it should be made possible to retrieve certificates or keys for authenticating parties and encrypting the initial tunnel.

The SIM Authentication Supplicant should be standardized, and be made into a service that could run on multiple platforms in order to provide SIM authentication capabilities to a range of existing applications.

Bibliography

[3GP04] 3GPP. TS 51.014 V4.5.0 - Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface, 2004.

[3GP05a] 3GPP. TS 21.111 V3.5.0 - Universal Mobile Telecommunications System (UMTS); USIM and IC card requirements, 2005.

[3GP05b] 3GPP. TS 51.011 V4.15.0 - Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface, 2005.

[3GP06] 3GPP. TS 43.020 V5.2.0 - Digital cellular telecommunications system; Security-related network functions, 2006.

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

[Ada02] Adam Stubblefield, John Ioannidis, Aviel D. Rubin. Using the Fluhrer, Mantin, and Shamir Attack to Break WEP, 2002.

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

[AS99] B. Aboba and D. Simon. PPP EAP TLS Authentication Protocol. RFC 2716 (Experimental), October 1999.

[Blu05] Bluetooth Special Interest Group. SIM Access Profile, version 1.0, 2005.

[EA05] Jon Edney and William A. Arbaugh. *Real 802.11 Security, Wi-Fi Protected Access and 802.11i.* Addison Wesley, 2005.

[ETS95] ETSI. GSM TS 11.11 V5.0.0 - Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface, 1995.

[GLH+00] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis. A Framework for IP Based Virtual Private Networks. RFC 2764 (Informational), February 2000.

[GSM03] GSM Security. GSM Security FAQ. `http://www.gsm-security.net/faq/gsm-network.shtml`, 2003.

[HP03] INTEL HP. Wi-Fi Hotspot Authentication using GSM phone SIMs. `http://www.hpintelco.net/pdf/solutions/telecom/mobile/wifi_hotspot_authentication_blueprint.pdf`, 2003.

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

[IEE03] IEEE. 802.11, 1999 Edition - Wireless LAN Medium Access Control and Physical Layer Specifications. http://standards.ieee.org, 2003.

[JRRT02] Helmut Scherzer Josyula R. Rao, Pankaj Rohatgi and Stephane Tinguely. Partitioning Attacks: Or How to Rapidly Clone Some GSM Cards. http://www.research.ibm.com/intsec/gsm.html, 2002.

[Koo05] John Koon. The USB Vision: 10 Years Later. http://www.everythingusb.com/timeline.html, 2005.

[LL05] Audun Wangensteen Lars Lunde. Using SIM for strong end-to-end Application Authentication. http://www.lundeweb.com/studier, 2005.

[MB99] Ruth Malan and Dana Bredemeyer. Functional Requirements and Use Cases. http://www.bredemeyer.com, 1999.

[MB01] Ruth Malan and Dana Bredemeyer. Defining Non-Functional Requirements. http://www.bredemeyer.com, 2001.

[Mic00] Microsoft Technet. Smart Cards. http://www.microsoft.com/technet/prodtechnol/windows2000serv/maintain/security/smtcard.mspx, 2000.

[NM] NTA-Monitor. OpenSwan - NTA-Wiki. http://www.nta-monitor.com/wiki/index.php/OpenSwan.

[NN03] Valtteri Niemi and Kaisa Nyberg. *UMTS Security.* Wiley, 2003.

[Nys06] M. Nystroem. The Protected One-Time Password Protocol (EAP-POTP). ftp://ftp.rsasecurity.com/pub/otps/eap/draft-nystrom-eap-potp-07.txt, September 2006.

[oST00] National Institute of Standards and Technology. Federal Information Processing Standards (FIPS) Publication 186-2, January 2000.

[Pat03] S. Patel. Analysis of EAP-SIM Session Key Agreement. http://wwws.ietf.org/proceedings/03jul/slides/eap-11.pdf„ July 2003.

[Pos81] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFC 3168.

[RSA05] RSA Security. RSA Security Survey Reveals Multiple Passwords Creating Security Risks and End User Frustration. http://www.rsasecurity.com/press_release.asp?doc_id=6095, 2005.

[Sch] Bruce Schneier. Frequently Asked Questions - Microsoft's PPTP Implementation. http://www.schneier.com/pptp-faq.html.

[Sch96] Bruce Schneier. *Applied Cryptography.* Wiley, 1996.

[Sin99] Simon Singh. *The Code Book: The Evolution Of Secrecy From Mary, To Queen Of Scots To Quantum Cryptography.* Doubleday, 1999.

[Sun06] Sun Microsystems. Sun History. http://www.sun.com/aboutsun/coinfo/history.html, 2006.

[Sys] Cisco Systems. Cisco VPN Client Data Sheet. http://www.cisco.com/en/US/products/sw/secursw/ps2308/products_data_sheet0900aecd801a9de9.html.

[The06] The Identity Gang. Lexicon. http://identitygang.org/Lexicon, 2006.

[Uni05] International Telecommunication Union. E.212: The international identification plan for mobile terminals and mobile users. http://www.itu.int, 2005.

[Uri06] P. Urien. EAP-Support in Smartcard. Internet-Draft, August 2006.

[Vai04] Vaishnavi, V. and Kuechler, W. Design Research in Information Systems. http://www.isworld.org/Researchdesign/drisISworld.htm, 2004.

[Wik] Wikipedia. Wikipedia FreeS/WAN. http://en.wikipedia.org/wiki/FreeS/WAN.

[Wik06a] Wikipedia. Authentication. http://en.wikipedia.org/wiki/Authentication, 2006.

[Wik06b] Wikipedia. Wireless LAN. http://en.wikipedia.org/wiki/Wireless_LAN, 2006.

Appendix A

VPN Configuration

A.1 Certificates

For the VPN Client/Server authentication, a set of certificates needs to be created.

For Public Key Infrastructure (PKI) management, a set of scripts bundled with OpenVPN is used. After downloading the OpenVPN package, the *easy-rsa* directory can be found. Edit the *vars* file in this directory and set the *KEY_COUNTRY*, *KEY_PROVINCE*, *KEY_CITY*, *KEY_ORG* and *KEY_EMAIL* parameters.

Next, initialize the PKI;

On Linux:

```
. ./vars
./clean-all
./build-ca
```

On Windows:

```
vars
clean-all
build-ca
```

The final command (build-ca) builds the Certificate Authority (CA) certificate and key by invoking *openssl*

```
# ./build-ca
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
```

```
Country Name (2 letter code) [NO]:
State or Province Name (full name) [ST]:
Locality Name (eg, city) [TRONDHEIM]:
Organization Name (eg, company) [NTNU]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:VPN-CA
Email Address [tosh@tosh.no]:
```

The default values of the queried parameters were set in the vars file, and the only one that has to be explicitly entered is *Common Name*. This is set to VPN-CA in the example above.

Next we have to build the certificate and private key for the server.

On Linux:

```
./build-key-server server
```

On Windows:

```
build-key-server server
```

Use the default values for the different parameters, and enter *server* when asked for the common name. The two last questions, "Sign the certificate?" and "1 out of 1 certificate requests certified, commit?" should be answered with *y*.

Next, the Diffie-Hellman parameters need to be generated.

On Linux:

```
./build-dh
```

On Windows:

```
build-dh
```

Now the different keys and certificates are found in the *keys* subdirectory. See table A.1 for a short explanation of their purpose.

Filename	Needed by	Purpose	Secret
ca.crt	Server + Client	Root CA certificate	NO
ca.key	Key signing machine	Root CA key	YES
dh{n}.pem	Server only	Diffie-Hellman parameters	NO
server.crt	Server only	Server Certificate	NO
server.key	Server only	Server Key	YES

Table A.1: OpenVPN Certificates and keys

The file *ca.crt* should be copied or downloaded to the client. This file is used to validate the server certificate.

A.2 OpenVPN Client

Following is the configuration file for the OpenVPN Client, client.conf.

```
# Run OpenVPN in Client mode
client

# Enable a TCP server on IP:port to handle daemon management
# functions
management 127.0.0.1 4593

# Query management channel for private key password and
# --auth-user-pass username/password.
management-query-passwords

# Authenticate with server using username/password.
auth-user-pass

# Use tun device
dev tun

# Use udp for VPN tunnel
proto udp

# Connect to server on port
remote vpn.tosh.no 1194

# If hostname resolve fails, retry indefinitely
resolv-retry infinite

# Do not bind to local address and port.  The IP stack will
# allocate a dynamic port for returning packets.
nobind

# Try to preserve some state across restarts.
persist-key persist-tun

# Certificate authority (CA) file in .pem format, also
# referred to as the root certificate.
ca C:\\Program\ Files\\OpenVPN\\config\\ca.crt

# Enable compression on the VPN link.  Don't enable this
# unless it is also enabled in the server config file.
comp-lzo

# Set log file verbosity.
verb 3
```

A.3 OpenVPN Server

Following is the configuration file for the OpenVPN Server, server.conf.

```
# Run OpenVPN in Client mode
client

# Enable a TCP server on IP:port to handle daemon management
```

```
# functions
management 127.0.0.1 4593

# Query management channel for private key password and
# --auth-user-pass username/password.
management-query-passwords

# Authenticate with server using username/password.
auth-user-pass

# Use tun device
dev tun

# Use udp for VPN tunnel
proto udp

# Connect to server on port
remote vpn.tosh.no 1194

# If hostname resolve fails, retry indefinitely
resolv-retry infinite

# Do not bind to local address and port.  The IP stack will
# allocate a dynamic port for returning packets.
nobind

# Try to preserve some state across restarts.
persist-key persist-tun

# Certificate authority (CA) file in .pem format, also
# referred to as the root certificate.
ca C:\\Program\ Files\\OpenVPN\\config\\ca.crt

# Enable compression on the VPN link.  Don't enable this
# unless it is also enabled in the server config file.
comp-lzo

# Set log file verbosity.
verb 3
```

A.4 OpenVPN User Authentication Script

Following is the *auth-pam.pl* script, used by OpenVPN server for verifying users.

```
#!/usr/bin/perl -t

#   Return success or failure status based on whether or not
#   a given username/password authenticates using PAM.
#   Caller should write username/password as two lines in a
#   file which is passed to this script as a command line
#   argument.

use POSIX;
```

```
use IO::Socket::INET;
use Digest::MD5 qw(md5_hex);

# Get username/password from file
if ($ARG = shift @ARGV) {
    if (!open (UPFILE, "<$ARG")) {
        print "Could not open user/pw file: $ARG\n";
        exit 1;
    }
} else {
    print "No user/pw file specified on command line\n";
    exit 1;
}

$username = <UPFILE>;
$password = <UPFILE>;

if (!$username || !$password) {
    print "Username/password not found in file: $ARG\n";
    exit 1;
}

chomp $username;
chomp $password;

close (UPFILE);

$HOSTNAME = "yoshi.tosh.no";
$PORTNO = 5747;

$sock = IO::Socket::INET->new(
        PeerPort=>$PORTNO,
        Proto=>'udp',
        PeerAddr=>$HOSTNAME) or die "Can't bind: $@\n";

print "Sending..\n";
$sock->send($username)
        or die "cannot send to $HOSTNAME($PORTNO): $!";
$MAXLEN = 1024;
$authString = "";

$sock->recv($authString, $MAXLEN, 1024) or die "recv: $!";
$host = gethostbyaddr($ipaddr, AF_INET);


if(length($authString)<8){
    print "Auth '$username' failed,
        user not found or key too short\n";
    exit 1;
}
print length($authString);
$hash = md5_hex $authString;

if($hash eq $password){
    exit 0;
```

```
}else{
    print "Wrong password ($hash vs $password)\n";
    exit 2;
}
```

Appendix B

Triplet Example File

```
<Triplets xmlns="http://tempuri.org/Triplets.xsd">
  <Triplet>
    <imsi>242023800085759</imsi>
    <rand>7737F2017F329DB0BA7E4FD31B85B3D1</rand>
    <sres>EA8D8FBF</sres>
    <kc>057D5F2C95C96400</kc>
    <id>0</id>
  </Triplet>
  <Triplet>
    <imsi>242023800085759</imsi>
    <rand>A50AD169B53A1705EB02C47D996CFCA7</rand>
    <sres>2873CEC1</sres>
    <kc>4DE93682CFE3E000</kc>
    <id>1</id>
  </Triplet>
  <Triplet>
    <imsi>242023800085759</imsi>
    <rand>8D6B9E556570374773AF0A1FBCB15661</rand>
    <sres>F9E9F586</sres>
    <kc>8412A9C4EFF2FC00</kc>
    <id>2</id>
  </Triplet>
  <Triplet>
    <imsi>227842834778004</imsi>
    <rand>22E26266A58F24DB7CF71123B767C0CE</rand>
    <sres>70666CAB</sres>
    <kc>272A4DE196F746F3</kc>
    <id>3</id>
  </Triplet>
  <Triplet>
    <imsi>227842834778004</imsi>
    <rand>C41ACD2BBE41855431971CBF1B39EA72</rand>
    <sres>70666CAB</sres>
    <kc>272A4DE196F746F3</kc>
    <id>4</id>
  </Triplet>
```

Appendix C

Paper submitted to WiMob 2007

Securing Virtual Private Networks with SIM Authentication

Torstein Bjørnstad, Ivar Jørstad and Do Van Thanh

***Abstract*— With the ever increasing amount of systems requiring user authentication, users are experiencing an substantial inconvenience. A huge amount of passwords needs to be remembered, which in many cases decrease system security, as users choose to write them down or use the same password for several systems. The number of username/password pairs has simply become unmanageable. Other, stronger authentication mechanisms are added to some systems, which increase administration costs for the enterprises/service providers. There is an immediate need for a stronger, yet at the same time simpler and cost-efficient authentication mechanism which can be reused across many different types of services. This paper proposes a novel VPN solution which employs the GSM SIM authentication mechanism for authenticating users towards VPN networks and for setting up the encrypted tunnel between the VPN client and server. The solution ensures strong security, is user-friendly and cost-efficient.**

***Index Terms*— Authentication, Mobile Phones, Security, Virtual Private Networking**

I. Introduction

The past few years, the way people connect to corporate networks have changed. The trend has moved from dial-up connections terminating inside secure networks to more permanent connections such as xDSL and Cable terminating at the edges of the ISP core networks. As these connections need to transport data through potentially unsecure networks in order to reach their destination, additional security solutions are needed.

Additional security solutions in most cases mean added inconvenience for the end user. Users already have a large number of passwords and certificates to keep track of, and avoiding increasing this number should be a goal. Recent surveys show a large percentage of help desk requests are password related [1], and password management is estimated to cost enterprises $150-350 per user per year [2].

Manuscript received April 15, 2001.

T. Bjørnstad, is a graduate student at the Norwegian University of Science and Technology, Trondheim, Norway

I. Jørstad is with Ubisafe, Oslo, Norway (corresponding author to provide phone: +4793039594; e-mail: ivar@ubisafe.no).

Do van, T. is with the Norwegian University of Science and Technology and Telenor R&I, Fornebu, Norway

II. Background and Related Works

A. Virtual Private Networks

Companies have for many years been leasing dedicated lines in order to connect internal networks of geographically dispersed sites. These lines offered the companies stable, permanent connections with both bandwidth guarantees as well as unique security characteristics. With the explosive growth of Internet coverage and the bandwidth increases over the past decade, companies started to use already existing infrastructure along with security mechanisms provided by for instance IPSec. This created so called Virtual Private Networks.

Another important trend is the increased mobility of users, which requires that they have access to their company resources from anywhere, at any time. This access must also be properly secured, since the data traffic will travel across several unsecured networks. VPN configuration used in highly mobile environments requires adequate authentication mechanisms.

The use of Virtual Private Networks is also being extended to personal computers, to secure connections between home offices and corporate or academic networks. To protect the integrity of the networks and the confidentiality of the data transmitted, access control and key management is of a major concern.

The process of establishing a VPN connection between a client and a server consists primarily of two steps:

1. Authentication of the user
2. Establishment of an encrypted, secure channel

Current VPN architectures usually correspond to the one illustrated in Figure 1. A VPN client on a computer in a remote location connects to a VPN server (often referred to as a VPN concentrator) in the home network or enterprise network. Between these two components, an encrypted tunnel is established as soon as the used has properly authenticated himself towards the system.

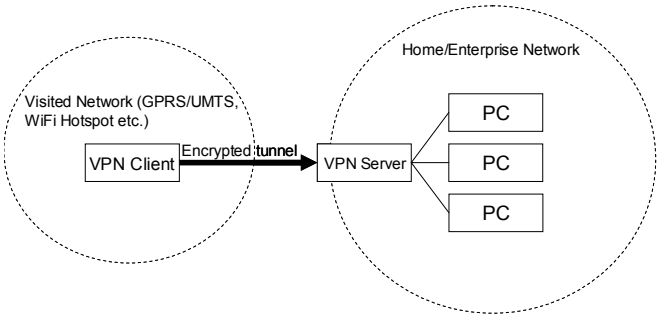

Figure 1 A basic VPN configuration with a client connecting to the home/enterprise network from a remote location

Authentication in VPN is today most often realised using username/password pairs or Smart Card solutions. One-time-password solutions (OTP) are also used many places, sometimes realised using SMS messaging through the GSM network for distribution of one-time-passwords.
Using username/password is a too weak authentication mechanism to be used with VPNs, whereas Smart Card solutions and OTP solutions are strong enough, but not cost-efficient.

B. GSM SIM authentication

The authentication system used in the Global System for Mobile Communications is a Challenge-Response mechanism, based on a secret key (Ki, 128 bit) stored only in the subscriber's home network and on a smart card called the Subscriber Identity Module (SIM).

When the network needs to authenticate the user, a random authentication challenge (RAND, 128 bit) is sent to the user terminal. This value, along with the secret key, is used as input to the A3 algorithm, implemented on the SIM. A signed response (SRES, 32 bit) is generated and returned, and is verified by the home network. [3]

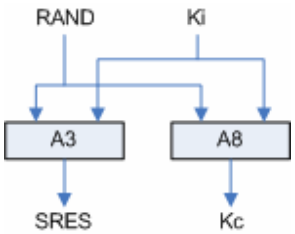

Figure 2: The A3 and A8 algorithms

Access control based on GSM SIM has been applied to other systems, such as Wireless LANs (WLAN EAP-SIM [4]) and web sites [5]. The latter including a third party Identity Provider which implements the Liberty Alliance specifications for identity management [6].

C. SIM access

To be able to perform the SIM authentication, the computer must have access to the user's SIM. This can be realized either through a Smart Card reader, or via a Bluetooth connection supporting the SIM Access Profile (SAP). Both means of connection enables the transmission of SIM APDU-messages, which is used in order to perform the A3 algorithm. The calculated response is returned to the computer using APDU-response-messages.

As these messages contain the basis for establishing the secure VPN connection, the security of the link between the computer and the SIM, regardless of connection type, has to be strong. A Smart Card reader is usually connected by a cable, but the wireless Bluetooth connection has to have strong encryption enabled. When pairing Bluetooth devices, a passkey of up to 16 byte is possible, and should be used for these SAP connections.

III. PROPOSAL OF A SIM-BASED VPN SOLUTION

A. General architecture

The VPN architecture proposed in this paper consists of several new components which do not exist in traditional VPN solutions. The most important of these components are the VPN SIM Supplicant and the VPN SIM Authenticator. The major components (see Figure 3) are now described.

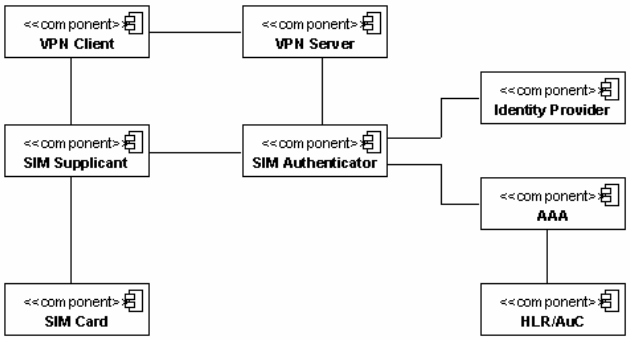

Figure 3: The major components of the VPN solution.

1) SIM Supplicant

This component initiates, on request from the VPN Client, the authentication process towards the VPN SIM Authenticator. It communicates with the SIM card using APDUs in order to properly run the A3 authentication algorithm, and it communicates with the VPN SIM Authenticator using the EAP-SIM protocol.

2) SIM Authenticator

This component performs authentication towards an Identity Provider or AAA-server upon request by the VPN SIM Supplicant or the VPN Server.

3) VPN Client

This is the typical VPN Client software, which has been adapted to be able to communicate with the VPN SIM Supplicant.

4) VPN Server

This is the typical VPN Server software, which has been adapted to be able to communicate with the VPN SIM Authenticator.

Today, there are a wide variety of VPN implementations, and they support many different authentication mechanisms. The mechanisms range from simple username/password, to more sofisticated solutions using Smart Cards or One-Time-Passwords. One of the challenges of adding SIM authentication as an option to existing VPN solutions is to do so without interfering with existing authentication mechanisms.

Integration of SIM authentication with the Point-to-Point Tunneling Protoocol (PPTP) [7] might be the easiest approach. Windows XP supports an EAP API, which allows programmers to add network modules to the configuration. It is thus possible to implement an EAP-SIM module for Windows, and this module will be available for the PPTP client. However, the Microsoft implementation of PPTP (which is the one most commonly used) is known to have severe security flaws [8]and it is therefore necessary to examine other possible solutions.

Another possibility, which has been more thoroughly investigated, is to integrate SIM authentication with the open source VPN solution OpenVPN [9]. It is possible to extend OpenVPN with new authentication mechanisms, and the architecture is open enough to embed also SIM authentication in several different ways, also without requiring any rewrite of the existing code base.

5) Authentication, Authorisation and Accounting (AAA)

The AAA server is connected to the mobile network through an SS7 MAP gateway, which provides protocol conversion between the Internet Protocol (IP) domain and the Signaling System 7 (SS7) domain used in mobile networks like GSM. This AAA server is located with the mobile operator.

6) Identity Provider (IdP)

Since the AAA should be located with the mobile operator, it is possible to perform general level access control there. However, to allow a more transparent authentication through the AAA, but still keep control of user authorization, the Identity Provider is introduced as a component of the solution. This Identity Provider can either be located on-site with the enterprise which uses the VPN solution or it can optionally be located within the telecom operator domain together with the IdP. This depends on the level of control over the solution the enterprise requires. The IdP has been extended with an authentication module specifically designed for handling the SIM authentication. By using such an IdP it is also possible to provide Single-Sign-On (SSO) functionality across a wide range of different services, not only for VPN purposes.

B. Interfaces and protocols

All of the communication between the components in the system has to meet strong security requirements. As the goal of the system is to establish a VPN connection with strong security characteristics, the security characteristics of both the interfaces and the components in the system have to be of at least the same strength as the VPN link. Because of this, all inter-network links have to use encryption, and the integrity of the components has to be guaranteed. The VPN solution employs standard protocol wherever possible.

The interfaces between the various components are now described.

1) VPN client-VPN server interface

The interface between the VPN client and the VPN server remains unchanged. The goal is to reuse both the existing authentication, as well as encryption procedure.

2) VPN client-SIM supplicant interface

This interface is non-existing today, and it has not been subject to any standardisation. This interface must in the proposed solution support at least one method, authRequest(), which initiates the authentication of the user and triggers the SIM supplicant to contact the SIM authenticator. The SIM supplicant will receive required data from the SIM authenticator (random challenge) in order to proceed with the authentication.

3) SIM supplicant-SIM interface

The interface between the SIM supplicant and the SIM card is based on Smart Card Application Protocol Data Units [10]. The messages are sent towards the SIM Card Access Device (CAD) using either a physical connection (e.g. USB) or across a Bluetooth link towards a cellular phone. When using Bluetooth, the SIM Access Profile (SAP) is used. Messages are sent according to the ETSI specification GSM 11.11 [11], and more specifically, messages conform to the RUN GSM ALGORITHM APDUs specified in the same document. The interface supports the following methods:

Response verifyPIN(PIN) – This method takes a PIN code as input and verifies it towards the PIN on the SIM card.

Response runA3(RAND) – This method takes as input a challenge of 128bits and returns the signed response (SRES) generated by the A3 algorithm on the SIM card.

4) SIM supplicant-SIM authenticator interface

All communication in the authentication phase between the SIM supplicant and the SIM authenticator is performed according to the EAP-SIM protocol as defined by IETF [12]. After the authentication phase has succeeded, different solutions can be used to exchange encryption keys between the SIM supplicant and the SIM authenticator; this is discussed in the next section.

5) VPN server-SIM authenticator interface

This interface must fulfill two primary tasks. It must allow the SIM authenticator to directly or indirectly notify the VPN server that a specific user has been authenticated. Second, the SIM authenticator must be able to provide an encryption key to the VPN server, which will be used to secure the connection between the VPN client and the VPN server.

6) SIM authenticator-Identity Provider interface

This interface must allow the SIM authenticator to also notify the Identity Provider that a specific user has been successfully authenticated. In response to this, the Identity Provider will provide the SIM authenticator with a unique token as proof of the registered authentication. In the current solution, this information is not used. However, in a Single-Sign-On (SSO) solution, this token will be used to provide unified authentication across different types of services.

7) SIM authenticator-AAA interface

This interface conforms to the EAP-SIM protocol, and is similar to the interface described in 4) above for the authentication phase.

C. Authentication Process and Establishment of Encrypted Tunnel

As said in the introduction, establishment of a VPN connection consists of two steps; authentication and establishment of a secure tunnel. The complete process is now described in more detail (see **Error! Reference source not found.** and **Error! Reference source not found.** for sequence diagrams).

The first step in the authentication process is for the VPN client to request an authentication from the SIM Supplicant. The SIM Supplicant will then contact the SIM Authenticator, which in turn initiates authentication towards the AAA server. The authentication process between the SIM Supplicant, the SIM Authenticator and the AAA follows the procedure specified by the EAP-SIM protocol.
Upon successful authentication, the IdP is notified with this fact and returns an authentication token to the Authenticator. The Authenticator will forward this token to the Supplicant. When the VPN Client then tries to establish contact with the VPN Server (and presenting the authentication token received from the Authenticator), it will contact the SIM Authenticator to check for a valid authentication. If an IdP is used, the Authenticator will additionally contact the IdP for further check of authorization information.
When the AAA is contacted by the SIM Authenticator, it requests a set of Authentication Triplets from the HLR in the

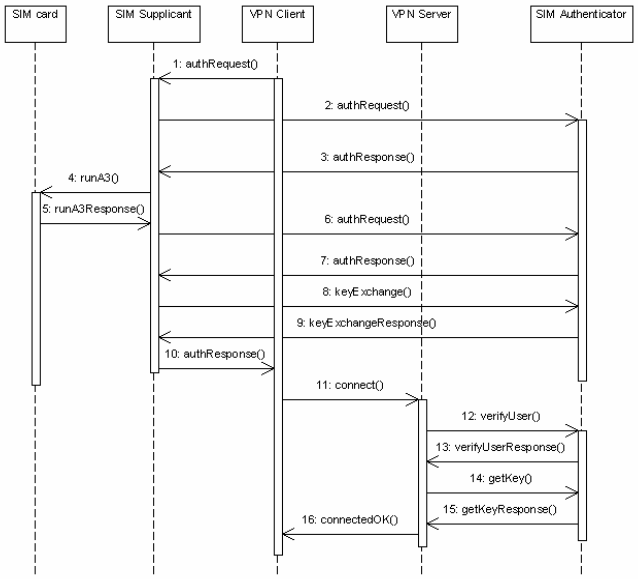

Figure 4 The authentication and key exchange process between the SIM Supplicant and the SIM Authenticator

GSM network. These consist of the random challenge (RAND), the correct (eXpected) response (XRES), as well as a Cipher key (Kc) used in GSM for encrypting calls on the radio interface.

The AAA returns parts of the triplets to the SIM Authenticator, which in turn returns them to the SIM Supplicant (in EAP-SIM messages). As the response is calculated by A3 with the secret key (K_i) as input, only the user holding the correct SIM can generate the expected response.

There are now several options for establishing the encrypted tunnel between the VPN client and the VPN server. One option is to use the ciphering key (K_c) generated by the A3 algorithm for this purpose. Another option is to concatenate several SRES values to construct a large enough encryption key. However, since SRES is not directly carried in the standard EAP-SIM protocol it is not straightforward to gain access to these values by the VPN SIM Authenticator, and eventually the VPN server. However, it is possible to establish a session key by using EAP-SIM and by reusing the K_c.
Another simpler option is to employ a key exchange algorithm between the VPN SIM Supplicant and the VPN SIM Authenticator once the client has been successfully authenticated. For example, both Diffie-Helman [13] and RSA key exchange [14] can be used for this purpose. After successful key exchange using any of these protocols, the

shared keys are provided to the VPN client by the Supplicant and to the VPN server by the Authenticator.

IV. Security Evaluation

Several new components and interfaces are introduced in the proposed solution, in comparison to existing VPN solutions. It is therefore crucial to assert the strength of each new interface and component, and verify if there are any obvious weaknesses or elements which must be strengthened.

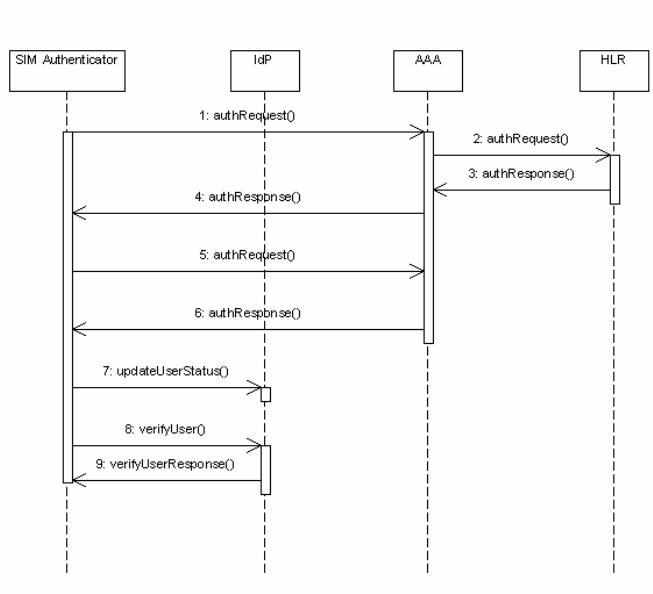

Figure 5 The authentication process between the SIM Authenticator and the HLR

It is possible to differentiate between two different types of connections in the proposed solution. First, there are the *static connections* which are long-lasting, and which in most cases are controlled by service-level agreements. Second, there are *dynamic connections* like the one between the VPN client and the VPN server.

The static connections are the ones between the SIM Supplicant and the SIM Authenticator, between the SIM Authenticator and the Identity Provider, between the SIM Authenticator and the AAA and between the AAA and the HLR. All these connections are secured with their own security measures, i.e., certificates and SSL/TLS security.

As mentioned earlier, the connection between the user computer and the CAD can be either with connections or through Bluetooth. When Bluetooth is used, the devices must be paired, and the pairing will be secured using a 128 bit user-selected key. This connection can be said to be long-lasting, because it is not necessary to do the re-pairing except in very rare-occasions.

Since much of the authentication functionality is based on standardised protocols, the strengths and weaknesses of these are already known and asserted by the community. Session key agreement using EAP-SIM has been shown to be faulty when not taking particular precautions [15], however, if used correctly, it is strong enough. In the proposed solution, this feature of EAP-SIM is not used, and encryption keys are instead exchanged using an asymmetric key exchange algorithm.

V. Conclusion

The paper shows that it is possible to secure a VPN solution using GSM SIM, and that the SIM authentication system can easily be extended to other services than the originally intended ones. By having a generic SIM Supplicant on the user's computer, other applications and services can utilize the SIM authentication mechanism, in order to simplify password management for the users.
Preliminary testing has been performed for VPN SIM authentication, and a more robust prototype of the authentication system is currently under development.

References

[1] RSA Security, The 2nd annual RSA Security Password Management Survey. September 2006. http://www.rsa.com
[2] Aberdeen group, Passwords are Gobbling Up Your Profits, May 2003
[3] Jeremy Quirke, Security in the GSM System. May 2004. http://www.gsm-security.net
[4] Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) SIM (EAP-SIM) – RFC 4186 – IETF
[5] Do van Thanh, Armand Nachef et al., Offering Strong SIM Authentication to Internet Services. February 2006.
[6] The Liberty Alliance Project – http://www.projectliberty.org
[7] Hamzeh, K. et.al. (1999), "Point-to-Point Tunneling Protocol (PPTP)", IETF, July 1999
[8] Schneier, B. "Frequently Asked Questions – Microsoft's PPTP Implementation", available online: http://www.schneier.com/pptp-faq.html
[9] Wikipedia, "OpenVPN", http://en.wikipedia.org/wiki/OpenVPN
[10] ISO, "ISO 7816-4 Smart Card Standard: Part 4: Interindustry Commands for Interchange"
[11] ETSI, "Digital cellular telecommunications system (Phase 2+);Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) Interface (3GPP TS 11.11 version 8.13.0 Release 1999)", ETSI/3GPP, 2005
[12] Haverinen, H. & Salowey, J. (ed.). "Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)", IETF, January 2006, available online: http://www.ietf.org/rfc/rfc4186.txt
[13] Diffie, W. & Helman, M. (1978), "Diffie Helman", available online: http://www.lsv.ens-cachan.fr/spore/diffieHelman.html
[14] Jonsson, J. & Kaliski, B. (2003), "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", IETF; available online: http://tools.ietf.org/html/rfc3447
[15] Patel, S., "Analysis of EAP-SIM Session Key Agreement", Lucent Technologies, available online: http://www3.ietf.org/proceedings/03jul/slides/eap-11.pdf