Remote User Authentication Using Embedded Systems and a Web Service

Martin Aastrup Olsen

Institute for Informatics and Mathematical Modelling

Technical University of Denmark

Kongens Lyngby 2007

IMM-B.Eng-2007-18

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-B.Eng-2007-18

1 Abstract

This final project report analyzes, specifies, designs, implements, and tests a remote user authentication system using embedded systems and a web service. The different choices made at each step are documented

User authentication systems involving only a chip card or a chip card and a PIN are subject to the possible sharing of card and PIN between multiple persons, which will allow multiple persons to be able to authenticate as a single person.

The system proposed and implemented herein is not subject to this through a combination of fingerprints and chip card as a means to authenticate. Since a fingerprint cannot be easily shared like a PIN. This method provides a more secure verification of the actual presence of the authorized person.

2 Preface

This paper is part of the Computer Engineering education at the Institute for Informatics and Mathematical Modelling at Technical University of Denmark (DTU). The report is documentation for the final project, which is credited with 15 ECTS points.

Thanks to Mads Siggaard-Andersen, Steen Jørgen Sannung, Frank Schwartz Christensen, John Erik Johansen, Morten Skjellerup, Mads Pii, and others at Logos Design A/S who provided technical support and more.

Martin Aastrup Olsen

March 2007

3 Contents

4 Introduction

The objective of this project is to implement a system able to perform remote user authentication using biometrics. Specifically fingerprints will be used. To achieve this I have made use of several existing components and integrated them so that I by the end of the report can present a working system.

In Figure 4-1 a picture of the Ni embedded computer with a card reader and fingerprint scanner connected is shown. These are all parts in the remote user authentication system proposed. A server is also part of the system but is not shown in the picture but a RJ45 connector that connects the Ni to the server can be seen.

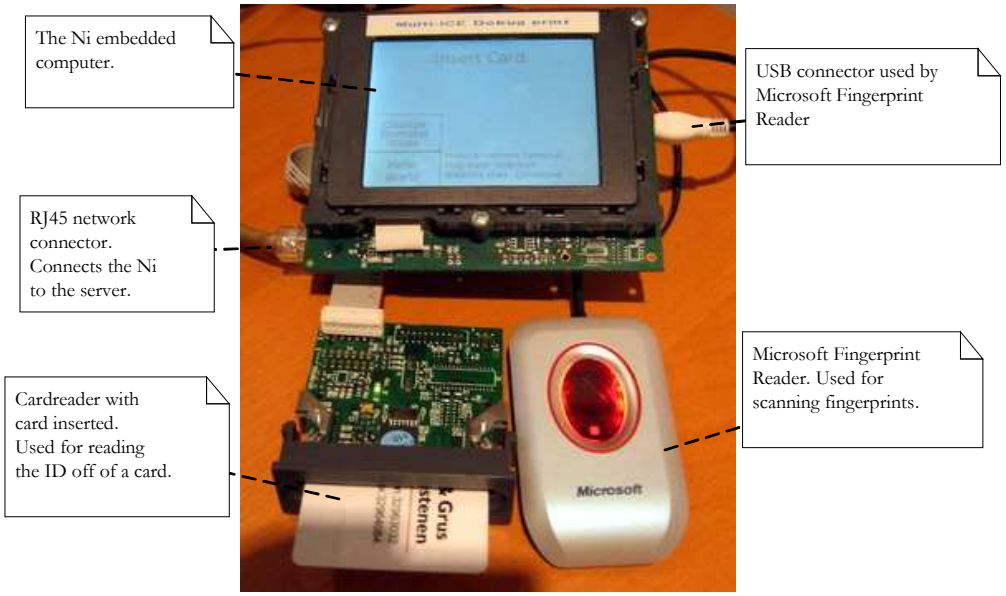

Figure 4-1 Picture of the Ni embedded computer, a card reader and fingerprint reader. All components in the remote user authentication system proposed in this report.

In this report I will document the different parts of the system and how they interact. I will go through the analysis, specification, design, implementation, and test of the remote user authentication system. I will also discuss the usage of fingerprints for authentication. The users manual is included in Appendix VI – Users Manual (Authentication Device) (section 17). This report also includes a CD-ROM (overview of contents is included in Appendix VII (section 18)).

In this chapter I will briefly take a look at biometrics in general and then the Danish Data Retention Law that will apply to this project.

4.1 Biometrics

Merriam-Webster defines Biometrics as "the measurement and analysis of unique physical or behavioral characteristics (as fingerprint or voice patterns) especially as a means of verifying personal identity"[1].

4.1.1 A Short History

According to National Center for State Courts the first known use of biometrics was in China in the 14th century where "Chinese merchants were stamping children's palm- and foot prints on paper with ink in order to distinguish young children from one another." [2]

Outside of China around 500 years should pass before fingerprinting was used for the identification of persons. In 1892 in Argentina the first criminal fingerprint identification was made[3] when a woman was found guilty of murder after a police officer had shown that the bloody fingerprints found at the crime scene were hers.

With the advent of computers came the possibility to make use of a broader range of biometrics such as iris and voice recognition. At the same time it has become easier to take advantage of biometrics in e.g. an access-control system such as the one this report describes in the following chapters.

4.1.2 Different Kinds

There exists a large array of biometrics each having different properties; I will only list two of them here along with their properties and a description.

Fingerprints are a good choice for authentication because they have a high rate of permanence and uniqueness[4] and very fast algorithms (around 10 ms. for verification[5]) exist for fingerprint recognition. Furthermore it is not very intrusive because people put their fingerprints everywhere already.

Iris-recognition is a way to authenticate using pattern recognition algorithms on an image of a person's iris. It is possible to scan the iris from several meters away and this could of course result in the scanning of the iris of people who are not interested in access and therefore need no authentication.

For the remote user authentication system described in this report I will make use of fingerprints for the reasons mentioned above and also because Logos Design A/S (the

[1] http://www.m-w.com/dictionary/biometrics
[2] http://ctl.ncsc.dni.us/biomet%20web/BMHistory.html
[3] http://en.wikipedia.org/wiki/Fingerprint#Timeline
[4] http://www.itsc.org.sg/synthesis/2002/biometric.pdf
[5] http://www.griaule.com/page/en-us/grfinger_fingerprint_sdk#t6

company at which this project is made) has a working implementation of drivers for a fingerprint reader for their in-house developed embedded system Ni.

4.2 Data Retention Law

The remote user authentication system proposed in the report makes use a user's fingerprint in order to authenticate the user. This is personal information that can be used to uniquely identify a person and certain regulations apply when dealing with personal information.

The Danish data retention law[6] defines how personal data must be treated, stored, and when it is allowed to collect this data. The laws § 6 article 1(1) specifies that if a registered person (user) has provided informed consent then the data can be included in a database and used as a basis for identifying the user. This will include authentication of said user.

Once the data is registered it must be stored in a manner which ensures that it is safe and it must not be shared with a 3rd party unless the registered user provides consent.

Datatilsynet is a government entity which advices institutions and corporations on how to apply the data retention law. They have advised a training centre not to use fingerprints for their access-control system because the centre wanted to store the fingerprints in a central database. It should be mentioned that if implemented then it would only be possible to enter the centre using a fingerprint.

The system proposed in this report is similar to the one suggested by the training centre but it has the main difference that it is not necessary to use fingerprints to enter and therefore it is completely voluntarily to make use of the system.

4.2.1 Fingerprint on Chip

The Danish ferry company BornholmsTrafikken makes use of a system that identifies its passengers by using fingerprints. They do so by storing the passengers fingerprint template on a card that the passenger carries and by doing so they have no central storage place. The fingerprint matching (authentication of users) takes place within a terminal, which scans the passengers' finger and matches it with the template stored on the card that the passenger carries. The system eases the boarding procedure for frequent travellers. When using this method the data retention law does not apply in the same way because they do not actually store any personal data but merely process it.

[6] http://www.datatilsynet.dk/lovgivning/personoplysninger/indhold.asp

5 Analysis

In this section I will first describe the identify factors which can potentially jeopardize the success of this project. Then I will present the remote user authentication system in the context of it being installed in an amusement park. Further I will describe the usage of fingerprints for authentication (Section 5.4) and the security of the system (Section 5.5).

5.1 Risk Analysis

This project, like most other projects, was subject to several risks that may have hindered it in achieving its goals. At the outset I identified the following major risks to the project.

A major technological risk in this project is the usage of the third party component Griaule GrFinger Fingerprint SDK Recognition Library. It poses a risk because it will have to be integrated into a managed .NET component that in turn will integrate into Microsoft SQL Server 2005. This can lead to compatibility issues and other problems with the integration. Even though the Brazil-based Griaule company (the makers of GrFinger Fingerprint SDK) can offer technical support, their response times and the thoroughness with which they will investigate any problems which may arise is unknown. Within the private sector it is not unusual to have response times on the order of several weeks when investigating problems of this kind. With the duration of the project being 10 weeks such a response time would make it considerably less likely that the project will achieve its goal of being able to successfully authenticate a user.

A former employee at Logos A/S made as an internship project drivers for the Microsoft Fingerprint Reader for the Ni embedded system that will be used in this project. As that project is undocumented and that no test documents except a demo application exists this must also be considered a major risk.

Another technological risk is the usage of the Microsoft Internet Information Server (IIS) and the IIS extension SQLXML 3.0 for bindings with Microsoft SQL Server 2005. Though, before the start of the project I made a small project to ensure that it was possible to communicate with the SQL Server through the SQLXML 3.0 interface.

Otherwise I have strived to use as much known technology in this project in order to minimize risks associated with the development of this remote user authentication system.

Backups of all project files (e.g. documents and source code) to my assigned development PC at Logos A/S, my PC at home, and to the DTU fileserver will be executed on a daily basis to minimize the risk of data loss.

5.2 Usage

An amusement park has a facility where its visitors can play various kinds of games where they can win money. Everyone can enter the facility but there also exists memberships that offer some advantages for the members. One of these advantages is that a holder of a membership access card is given a bonus every time the holder enters the park. This bonus can be transferred to an account and be used as credits in the gambling area. Even though the card is personal it is easy to share with others because it is not being controlled whether the holder of the card really is the owner of the card. Therefore it is possible for one person to bring several cards and thus collect a bonus for all the cards even though the real cardholders never visited the park that day. This is not profitable for the amusement park.

The problem cannot be solved using a PIN because it is trivial to just share it along with the card. An automated access-control system combining the access card with the cardholders fingerprint allows for a remote user authentication system to be implemented while solving the problem of a person accessing the premises using multiple cards and collecting bonus for all the cards. The amusement park will then only pay bonus to the cardholders who actually visit the park.

Many people visit the park throughout the season and this means that many users (several thousand) will be using the system each and every day.

5.3 Deployment of the Remote User Authentication System

The remote user authentication system has three main components that are interconnected using a network (e.g. LAN):

- At least one authentication device that is placed by the park entrance.
- A server that performs the actual remote authentication and is storing all the enrolled users (cardholders) of the system.
- An enrolment terminal that enrols users in the system.

The authentication device is responsible for collecting the fingerprint and card identification number (CID) from the user and will, depending on the result supplied by the server, insert the bonus on the users account and allow access to the gambling area. The system could also be set to disallow access if the authentication failed.

The server must be capable of receiving a fingerprint and a CID from an authentication device. It will then try to match the received fingerprint with a template retrieved using the CID. The authentication device will then receive the result of the remote authentication and then take the appropriate action and if applicable the server will insert a bonus on the users account.

The enrolment terminal is responsible for creating accounts and linking a CID to a fingerprint so that the server can perform the authentication. This terminal is not dependent on being installed at any particular place in the park from a functional point of view.

The enrolment terminal can even be left out of the system, and replaced by the server inserting and linking the fingerprint to the CID on first usage.

A diagram of the structure of the remote user authentication system is depicted in Figure 5-1.

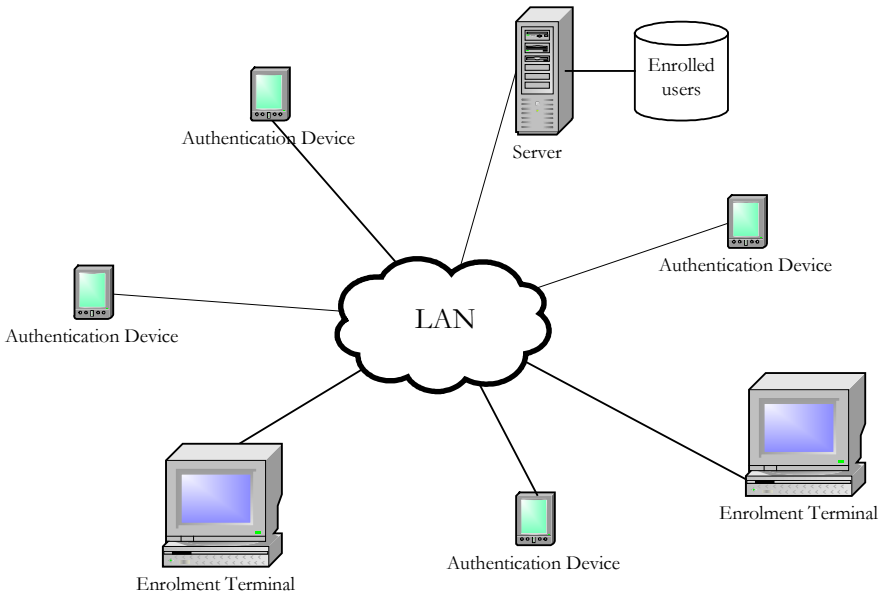

Figure 5-1 An overview of the remote user authentication system.

The authentication device using an external fingerprint reader will capture the fingerprints and read the CID using a card reader.

5.4 Using Fingerprints for Authentication

The remote user authentication system makes use of fingerprints for authentication. In this section I will go through some points on why this works and present two different methods that can be used for authentication.

In order to successfully authenticate it is necessary to have something unique to authenticate and a human fingerprint is just that - unique. A fingerprint consists of ridges and valleys and if one examines a fingerprint one will notice that in several places the ridges will end (known as a ridge ending) and in others they will spilt into two ridges (known as a bifurcation). In Figure 5-2.a I have marked two of such points. The points of ridge endings and bifurcations are also called minutiae and are used as a means to describe a fingerprint and are frequently used by fingerprint recognition algorithms. In Figure 5-2.b the graphical output of Griaule GrFinger

FingerprintSDK Recognition Library[7] is shown and it can be seen that it has extracted all the minutiae points present in the fingerprint.

When these features have been extracted from a persons fingerprint they are stored in a template, which can then be used to authenticate the person against at some later point in time. This template actually consists of all the minutiae points from the fingerprint described as vectors consisting of the minutiae type, its coordinates, the angle, and the weight of the minutiae based on the quality of the image at that particular point.

By storing only the template of the fingerprint, which is described in less than 1 kilobyte, instead of the image of the fingerprint, which can easily take up 100 kilobytes, the retrieval from database and verification is sped up.

There are two ways to employ the authentication – either by identification (1:*N* matching) or by verification (1:1 matching).

Identification is a method where the system takes a fingerprint and tries to find a match within the set of fingerprints belonging to all the enrolled users. This method relies on a fingerprint being the only input and thus it does not rely on the user to carry anything physical such as a CID. This method is viable mostly for small systems such as an access-control system for a family house because as the number of users increases so do the time to identify a specific user. The system simply iterates through all of the fingerprints and tries to match the incoming fingerprint with each of the enrolled fingerprints. Identification will require a higher quality of the fingerprint scans than verification and a higher matching threshold will be necessary because of the likelihood of finding a matching fingerprint increases with the size of fingerprint being matched against.

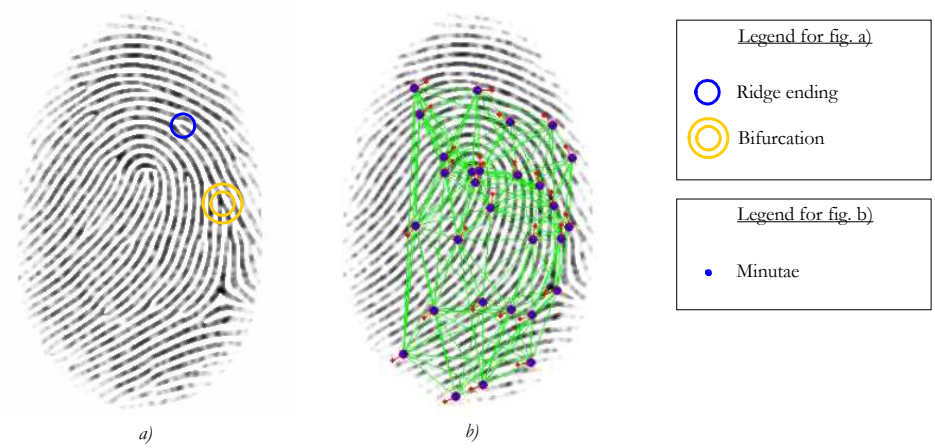

Figure 5-2 Two sample synthetic fingerprint images with features marked.

The verification method does not have this limitation. Because it relies on both a fingerprint and a unique ID it can be executed in constant time regardless of the number of users in the

[7] http://www.griaule.com/page/en-us/grfinger_fingerprint_sdk

system. This gives this method an advantage because it is a less computationally intensive operation to look up one fingerprint based on an ID than to look trough every fingerprint until a match is found. Employing multiple parallel systems, each identifying the incoming fingerprint against a subset of the enrolled users, could speed up the identification but this may not be an economically viable option.

The remote user authentication system proposed here will make use of the verification method because the number of users could be very large and the system should still respond in a quick manner. The Griaule GrFinger Library will be used as a component for verification and extraction of templates in this project.

5.4.1 Challenges and Obstacles When Using Fingerprints

Using a biometric for authentication is generally a very reliable way of authentication but it is not flawless at all. Taking fingerprints as an example, we can see that some types of work will expose the fingerprints to more wear and tear than other types of work. This will cause the fingerprint to change over time such that a person can no longer be identified based on a several years old fingerprint. Burns, cuts, and other temporary or permanent damages will inevitably happen and as such a system relying on fingerprints could fail to authenticate a person whose fingerprints have such damages. Storing the newest fingerprint in a FIFO could solve this problem. Thus fingerprints would be replaced as they where scanned and authenticated. This could lead to another problem regarding declining quality of fingerprints

This method would require significant testing in order to ensure that the quality of the fingerprint does not degrade when replaced. The fingerprints should only be replaced when a fingerprint has been authenticated successfully and achieved a score above a certain threshold.

Another related problem is how to deal with a person who for example does not have any fingerprints at all. The system proposed in this paper does not seek to solve this problem.

Using fingerprints for authentication can feel intrusive to some people because of privacy concerns. An adversary copying fingerprints from a central database and using them to commit crimes is a concern shared by many people. Though, the International Biometric Industry Association[8] (IBIA) lists this as a perceived concern. According to IBIA a very real concern is that the copied fingerprints could be used to commit fraud and theft[9].

5.4.2 Different Fingerprint Readers

Fingerprints can be read using different methods such as optical, ultrasonic, and capacitive sensing. Other kinds of readers exist but will not be mentioned here.

[8] http://www.ibia.org
[9] http://www.ibia.org/biometrics/industrynews_view.asp?id=347

The optical reader is basically a specialized digital camera that captures an image of the fingerprint on the epidermal layer using visible light. The optical reader is susceptible to the problems with damaged fingerprints discussed in 5.4.1.

Ultrasonic readers on the other hand do not read the epidermal layer but rather the dermal layer and are thus not sensitive to how clean the reader surface is or how damaged the epidermal skin is.

Active capacitive readers read the dermal layer by applying a voltage to the skin and then measuring an electrical field that follows the ridges in the dermal skin layer. This method is not susceptible to the problems discussed in 5.4.1.

The remote user authentication system proposed here will make use of an optical reader. Specifically the Microsoft Fingerprint Reader will be used because of its high availability and the existence of drivers for the Ni embedded computer that is used in this authentication system. I will discuss the Microsoft Fingerprint Reader in section 7.4.

5.5 Security of the System

When considering the security of an authentication system, two sides must be considered. On one side we have the user whose main concern would be the protection of personal information such as the biometric, social security number, address, and other sensitive information. On the other side we have the operator and owner of the system whose main concern is that only users who are allowed access actually gets access. The system should strive to satisfy both of these concerns.

One obvious issue is that of the transportation of the fingerprint from a device that captures it to the device that authenticates or stores it. The data could be secured by using for example Secure Socket Layer (SSL). The device itself should also be secured such that it cannot be tampered with easily. This system is meant to be placed within an amusement park where there have been taken security measures against breaking into the park during closed hours.

The system must not expose the matching score of the verification but rather a quantified response. This will make it more difficult for an adversary who is able to capture data hitting the authentication device to see whether his attempts to break the system are getting better or worse. This type of attack using a hill climbing procedure and measures against it has been described by Umut Uludag and Anil K. Jain[10].

A type of attack that the system is susceptible to, is an attack where an adversary uses a gelatine cast of a fingerprint of an already enrolled user. This could be done by covert acquisition (lifting a fingerprint off of a surface) or by cooperation where the adversary and an enrolled user work together to create a cast. This would enable multiple people to use the same card for access thus defeating the authentication and causing the amusement park to loose financially.

[10] http://biometrics.cse.msu.edu/Publications/SecureBiometrics/UludagJain_BiometricAttacks_SPIE04.pdf

Putte and Keuning[11] have fooled five out six fingerprint readers using this method in the first attempt and the last one in the second attempt. This type of attack is difficult to detect and prevent because it occurs at a point where a legitimate user is in contact with the system.

5.5.1 FAR, FRR, EER, and FIR

When specifying a biometric access control system two terms are frequently used:

FAR – **F**alse **A**cceptance **R**ate: The probability that the system will incorrectly identify an individual.

FRR – **F**alse **R**ejection **R**ate: The probability that the system will fail to identify an enrolled individual.

Each of these terms describes a case where the system behaves in an undesirable way. Depending on the system and its use the severity can vary – access to a top secret military facility will strive to approach FAR = 0 and a reasonably low FRR, while a supermarket customer savings account system will strive to approach FRR = 0 and a reasonably low FAR.

Any system which implements some form of biometric security will have to deal with these rates and determine acceptable levels for each of them depending on the purpose of the system.

FAR and FRR are closely related and reducing one of the rates will increase the other. This is due to the fact that in order to get a lower FAR it is necessary to increase the sensitivity of the system, and by doing so the tolerance is lowered causing more false rejects.

At the point where FAR and FRR intersect we have yet another measure of a biometric system: **EER** – **E**qual **E**rror **R**ate. EER is useful because it gives us a measure of how accurate the system is because this is the rate at which both FAR and FRR are minimized.

Besides FAR and FRR there is another relevant rate: **FIR** – **F**alse **I**dentification **R**ate. FIR is a measurement for the probability that an enrolled individual is identified as one or more enrolled individuals. In an access-control system with only one class of users this is of little importance, but it is critical if a user's biometric feature is tied to a specific account such as a bank account[12].

The Griaule GrFinger Library used in this project cites a 1‰ FRR when the matching threshold for verification is set to 30. This seems like a good starting point for a system like this but before deployed it should be tested thoroughly with both higher and lower thresholds to determine what the best setting is.

---

[11] T. Putte and J. Keuning, "Biometrical fingerprint recognition: don't get your fingers burned", Proc. IFIP
TC8/WG8.8, Fourth Working Conf. Smart Card Research and Adv. App., pp. 289-303, 2000.

[12] http://www.bromba.com/faq/biofaqe.htm#FIR

5.6 Identification of Actors

The remote user authentication system only has two actors: Operator and User. In Table 5-1 I have listed the actors and a description of them.

Name	Description
Operator	The Operator has the responsibility of adding and removing users from the system. The Operator is the one who is usually performing tasks on the Enrolment Terminal.
User	The User is the one who will use the system to enter certain areas that require authorization.

Table 5-1 Description of actors in the remote user authentication system and their rights.

5.7 Constraints

The following constraints were imposed on the present work:

The system must be low-cost and the server must run Windows and use Microsoft SQL Server 2005 Express Edition as a backend for storing user data. Microsoft SQL Server 2005 Express Edition has been chosen because it is free, redistributable, and easily upgradeable should the requirements for larger databases arise. Griuale GrFinger FingerprintSDK Recognition Library 4.2 shall be used for extracting features from fingerprints when enrolling users and performing recognition for authentication. The authentication device must use the Ni embedded platform developed by Logos A/S. Logos A/S has also specified that the user authentication should occur within the SQL Server.

6 Specification

In this section I will describe the specifications of the complete remote user authentication system, and then move on to describe each component.

6.1 Remote User Authentication System

The remote user authentication system proposed must be able to authenticate a user enrolled in the system. An operator can enroll a user from a central place where one of the user's fingers will be scanned using an optical reader a number of times, stored, and given a unique ID. An identification card containing this unique ID will be issued to the user.

Because of the relationship between a user's fingerprint and the ID it is possible to authenticate the user at a later time by having the user scan a finger and present the identification card to the system – thus verifying, with great certainty (discussed in section 5.5 above), that the cardholder is the rightful owner of the card. The user can then gain access to an otherwise restricted area.

The system consists of three main components:

- At least one authentication device, running on an embedded system that is able to read an enrolled user's fingerprint and ID card and authenticate against the server.
- At least one enrolment terminal (controlled by an operator) where the user can be enrolled. The enrolment terminal will not be designed as a separate entity but rather have some of its functionality included in the authentication device.
- One server where the enrolled user's information is stored and the user authentication and enrolment occur.

The terminal(s), server, and device(s) are all connected to the same network (e.g. a 100 MBit Ethernet LAN) so they are able to communicate with one another. This network must have a low latency and high bandwidth to ensure a fast authentication. That is, the authentication of a user must during normal conditions occur within 3 seconds.

Each authentication device and enrolment terminal must have a card reader and a fingerprint reader attached along with a display that is used to display information and to indicate what actions the user should take.

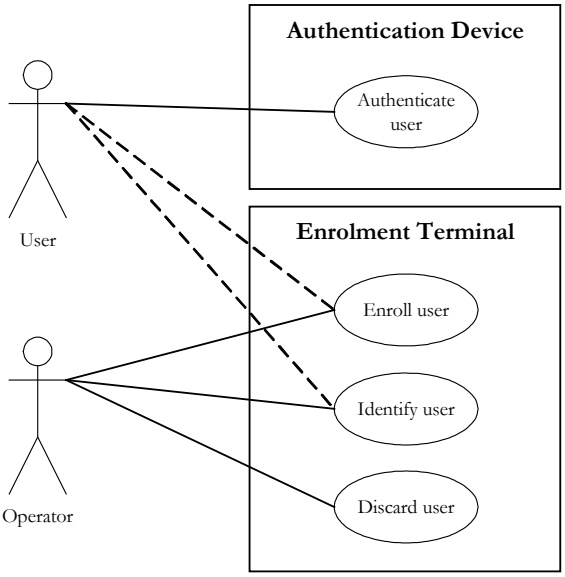

Figure 6-1 Actors and use cases in the authentication system and their relation to one another. All the use cases will in fact invoke procedures on the server to perform most of the actual work related to each use case.

Including the functionality of the enrolment terminal in the authentication device would allow for a completely automatized remote user authentication system where user are enrolled automatically the first time the card is used. This will not ensure that the user who received the card will be the one actually enrolling. Though, it will ensure that once a person is enrolled using a specific card only this user can gain access using that card.

In Figure 6-1 I have depicted the use case diagram for the remote user authentication system. It should be noted that the authentication device and enrolment terminal in the implementation will invoke procedures on the server to perform most of the work involved in the use cases.

No	Use case name	Objective	Priority
1	Authenticate user	The system must be able to authenticate a user and take action based on the outcome of the authentication.	High
2	Enroll user	The system must be able to accept new users so that they can use the remote user authentication system.	High
3	Identify user	The remote user authentication system must be able to create a new ID card for a user if the user has lost the ID card.	Low
4	Discard user	The remote user authentication system must be able to discard users if they no long wish to be part of the system.	Low

Table 6-1 The identified use cases and their objective and priority.

The use cases for the authentication device and enrolment terminal along with their objective and priority is shown in Table 6-1. The use cases can be found in their entirety in Appendix I – Use Cases (Section 12).

6.2 Authentication Device

The authentication device is only used for authentication of the users of the system. It must use the Ni embedded computer system developed by Logos Design. It has a display on which information to the user can be presented and a RJ45 connector that can be used for a network connection to the server. Also a card reader and Microsoft Fingerprint Reader can be connected using a USB connector present on the PCB.

Drivers for the Microsoft Fingerprint Reader have been ported to the Ni and it is therefore possible to control it from software running on the Ni.

The device is responsible for gathering the card ID and fingerprint from the user so the user can be authenticated.

The authentication device always initiates communication between a device and the server.

In case of communication failure between the authentication device and the server the default behaviour should be to keep the entrance locked, but this may not always be desirable. If the device is unable to contact the server for a longer period it would be undesirable to deny access for those who hold membership cards because they may not want to enter the facility without getting their bonus. And each person who does not enter the facility is potentially a loss for the park.

Instead the system should act as if it is functioning correctly and just record the card number and let the person enter. Then, at a later time when the connection to the server has been restored, contact the server and send the card information of the users who entered so they can be given their bonus.

This will keep the customers content but it also offers an opportunity for those who wish to cheat the system and collect extra bonus points, but since the system does not show any direct signs of not working correctly it is highly unlikely that this would be problematic.

The authentication devices are placed at the entrances and control the members' entrance locks. The locks must remain locked until a user is authenticated. Once this happens the user is let in and the entrance will be locked once again.

It should be noted that any user can always gain access to the premises but in order to get a bonus one must enter through the members' entrance.

6.3 Enrolment Terminal

The operator whose primary function is to enroll users operates the enrolment terminal. It can use either a standard PC or the Ni embedded computer but it must have a fingerprint reader connected and a network connection such that it can communicate with the server.

This project mainly deals with the authentication of users and therefore it is not within the scope of the project to describe the enrolment terminal. Nonetheless, it is necessary for test purposes to have a way in which one can enroll users and therefore I will combine a simple version of the enrolment terminal with the authentication device. This means that the Enroll User use case (Use Case 2) will be implemented on the authentication device that will be able to shift between the two modes of operation (i.e. enrolment and authentication).

6.4 Server

The server carries out the authentication and enrolment of users to the database. Communication to and from the server is carried out by a web service interface. The actual authentication and enrolment of users will take place in the FingerprintController which is a .NET CLR running on a SQL server. I will describe the different parts in the following subsections.

6.4.1 Web Service on the Server

The web service is the visible part of the server with which the authentication device and enrolment terminal communicates. It shall expose methods so the use cases shown in Figure 6-1 can be executed. The methods it must expose are enumerated in Table 6-2.

N	Method Name	Use Case	Function
1	CheckUser	Enroll User (Use case 1)	Authenticates a user given a fingerprint and a card ID.
2	InsertUser	Authenticate User (Use case 2)	Enrols a user given a fingerprint and a card ID.
3	IdentifyUser	Identify User (Use case 3)	Identifies a user by with 1:*N* matching using a fingerprint.
4	DiscardUser	Discard User (Use case 4)	Removes a user from the system.
5	WriteLogFromNi	N/A	Allows the Ni to write events and other information to the server log.

Table 6-2 Enumeration of methods exposed by the web service.

Only the methods 1, 2, and 5 from Table 6-2 will be implemented in this project but all of the methods are desirable in a fully implemented remote user authentication system.

The web service must be able to receive requests in the Simple Object Access Protocol (SOAP). This has been chosen because of the SQL Server being able to return results in XML format and the IIS supporting SOAP through an extension. An XML based alternative to SOAP is XML-RPC. Extensions for the IIS exist but to the knowledge of the author these are not easily integrated with the SQL Server.

6.4.2 The Server Database

The database must run on Microsoft SQL Server 2005 Express Edition, which is free to use and has the capability of running .NET CLR assemblies. The relations between users and their cards IDs must be kept here. The database must supply methods such that fingerprint templates can be inserted and retrieved from the database.

The database should allow for multiple fingerprints to be related to one user and a card. This will enable possible future expansions such as the FIFO scheme described in section 5.4.1.

The database must contain a log of access attempts and a log of other actions and events occurring in the remote user authentication system. This will prepare the system for a possible extension such as enabling it to analyze the access patterns and generate statistics so the system can be optimized.

6.4.3 FingerprintController (.NET CLR)

The FingerprintController is a .NET Common Language Runtime (CLR). It must be able to perform the methods exposed by the web service (listed in Table 6-2). By making use of the Griaule GrFinger Fingerprint SDK Library it must be capable of authenticating, identifying, and enrolling users.

The FingerprintController will be responsible for inserting and retrieving fingerprint templates to and from the database.

The FingeprintController is also responsible for logging incoming request parameters. It should allow for various levels of logging such that it can be set to only log errors or request parameters. This will make it easier to debug the authentication system. The log must be written to the database.

6.5 Communication Protocols

The authentication device and enrolment terminal will communicate with the web service using SOAP over HTTP.

Using SOAP over HTTP enables us to freely communicate without routers and firewall interfering as HTTP traffic is almost always allowed.

7 Design

The remote user authentication system consists of several components as mentioned in section 5.3. This design separates itself from other client-server systems in that the server side is not designed in the traditional way where there usually is a business layer in front of the database. Instead I have chosen to integrate the business layer directly in the database as a .NET CLR assembly. There are both pros and cons to this server architecture and I will discuss them specifically in section 7.5. First I will describe the overall design of the remote user authentication system.

7.1 The Remote User Authentication System

The remote user authentication system is designed in such a way that an IIS server receives requests from an authentication device or enrolment terminal and then passes them on to the MS SQL Server 2005. The IIS will be using an extension known as SQLXML 3.0 that enables it to receive and send SOAP messages and communicate with the SQL Server.

An overview of the design, interaction, and integration of the components of the remote user authentication system is depicted in Figure 7-1.

Using the IIS and its ability to receive and direct SOAP requests to the SQL server enables us to concentrate on the business logic, which handles the authentication and registration of users. The SQL server will also contain a database, which is used for storing the users and their associated templates. The reliance of a 3rd party component (e.g. the IIS) in the system can of course be problematic if security holes and bugs are discovered in it but not fixed by the 3rd party. On the other hand the maintainability of the system increases when using a 3rd party component because the amount of code decreases. Furthermore the development time is shortened due to this.

Another 3rd party component in the system is the Griaule GrFinger Fingerprint SDK Recognition Library (noted as GrFingerXLib.dll in Figure 7-1), which implements algorithms for feature extraction and matching. The component will be accessed from the FingerprintController, which feeds it with the necessary data for extraction of templates and verification of fingerprints.

The authentication device interacts with the IIS and acts like a thin client because it does not do any processing of data but merely sends it off to the server which then deals with it.

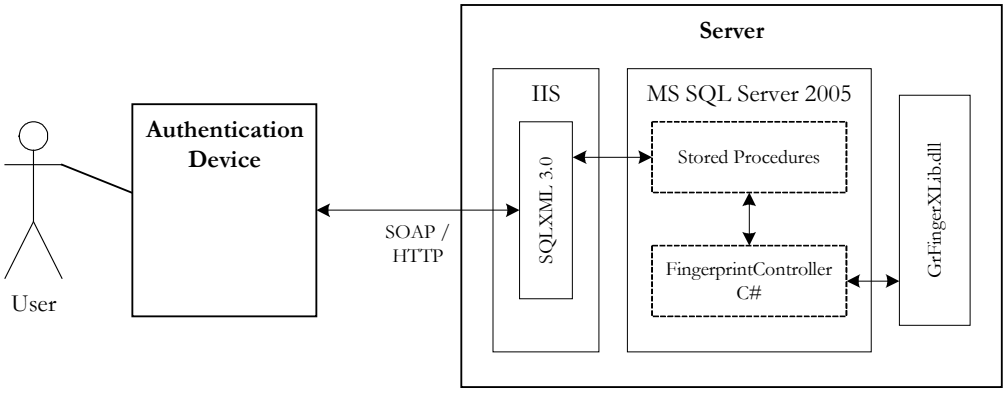

Figure 7-1 An overview of the design and integration of the components in the remote user authentication system.

In the following sections I will go through the dataflow of the system and present a system sequence diagram. Then I will move on to describe the design of the authentication device, enrolment terminal, and finally describe the server and its components (see Figure 7-1). Since the IIS and SQLXML 3.0 only needs to be configured I will not it describe them here but instead refer to Appendix III – Configuration of IIS (Section 14) and Appendix IV – Configuration of SQLXML 3.0 (Section 15).

7.1.1 Dataflow

The dataflow of the system is illustrated using Use Case 1 (Authenticate User) as an example and is depicted in Figure 7-2. The flow starts when an ID card is presented to the system. Once the authentication device receives an ID it initiates a fingerprint scan. When the authentication device has collected both an ID and a fingerprint they are sent to the server for verification. Depending on the outcome of the verification access is allowed or disallowed by the authentication device.

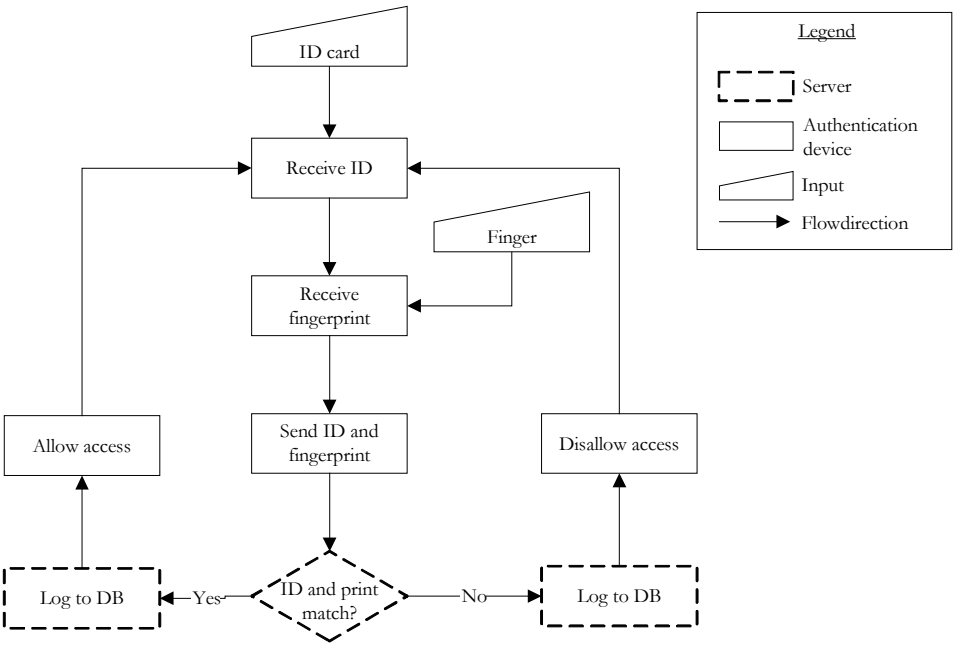

Figure 7-2 The main dataflow of the system.

If a fingerprint is not captured from the fingerprint reader the first time it will default to try and scan again up to three times. Similarly the device will try to resend the ID and fingerprint up to three times if it does not succeed the first time. If the ID card is removed at either the receive fingerprint step or the sending step it should abort its action. If the ID and fingerprint has been sent to the server for authentication before the card is removed it should continue operation as normal. The server should log the result of the authentication.

7.1.2 System Sequence Diagram

In Figure 7-3 I have depicted the system sequence diagram using Use Case 1 (Authenticate User) as an example. The diagram shows that the authentication device collects data from the user and displays and acts upon the result returned from the server. It also shows that the server is taking care of the actual authentication. The IIS, SQL Server 2005, and the FingerprintController will run on the server. The actions taking place on the server are discussed in detail in section 7.5.1.

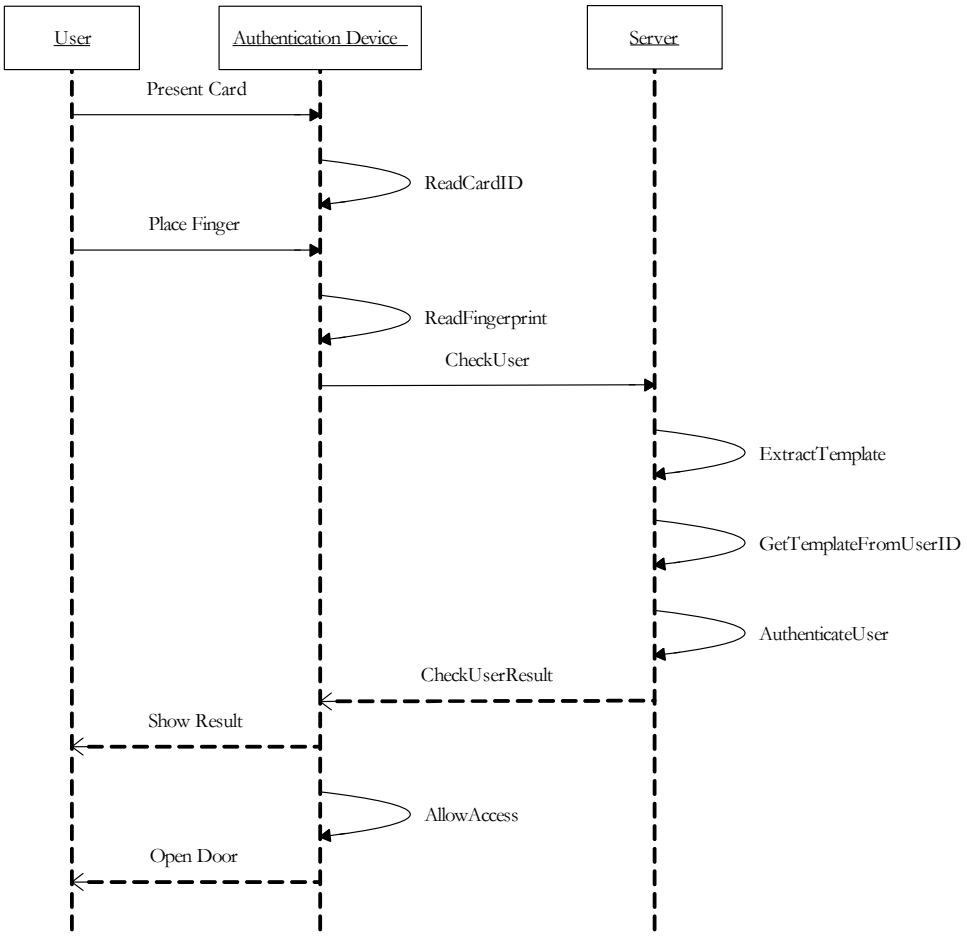

Figure 7-3 System sequence diagram using Use Case 1 (Authenticate User) as example.

7.2 Authentication Device

The authentication device will communicate with the server using SOAP and it should therefore implement the necessary methods for this. The Ni platform already allows for HTTP communication so only the construction of SOAP messages along with the reception need to be implemented. The device also supports the Microsoft Fingerprint Reader allowing for full control of capturing fingerprints.

Numerous card readers are supported by the Ni platform and at first a RFID card reader was chosen but in the last third of the project it was changed to a chip card reader. The change

occurred because the early development version of the Ni I used broke down and without any quick way of fixing it, it was just replaced. The change to a production version of the Ni was relatively smooth except for some complications with the card reader that took a couple of days to sort out. The complications were mainly due to required changes in especially the card handling part of the finite state machine that handles the process of authenticating a user.

A FSM will be used to control the dataflow and I have depicted a version of it in Figure 7-4. The change from the start state to state 1 (AwaitFinger) is triggered when a card is inserted into the card reader and it has been read. In state CheckUser and InsertUser messages are sent to another state machine (which is already implemented in `WebData.c` on the Ni) that handles the sending and reception of the SOAP messages. When received and parsed a signal indicating the result will be sent back to the first FSM so the result of the authentication can be shown to the user.

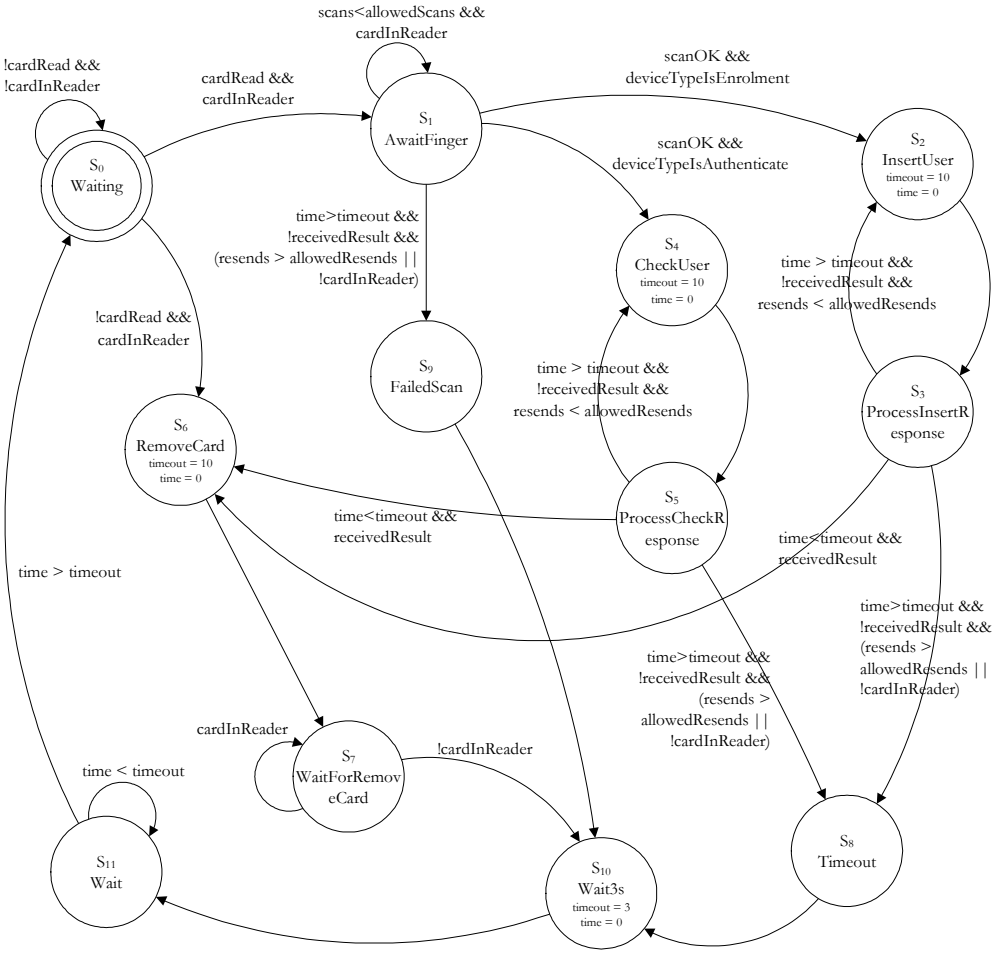

Figure 7-4 The authentication device's finite state machine controlling the interaction with the user and the server.

Due to the change from RFID card to chip cards the complexity of the state machine increased. Specifically the states RemoveCard and WaitForRemoveCard had to be introduced along with some logic to ensure that the card is removed from the card reader once the authentication has completed.

7.3 Enrolment Terminal

As mentioned in section 6.3 the design and implementation is included in the authentication terminal.

7.4 The Microsoft Fingerprint Reader

Both the authentication device and the enrolment terminal use the Microsoft Fingerprint Reader for the scanning of fingerprints. The images scanned are not a 1:1 image of the fingerprint presumably because the actual sensor is tilted slightly in order to capture a larger area of the fingerprint. This results in a scanned image which is slightly non-conformal, making a trapezoid like shape. The effect of this is that the physical direction of the finger on the scanner will negatively influence the result of the authentication.

M. Kiviharju has analyzed[13] the Microsoft Fingerprint Reader and estimated a transformation formula that maps the non-conformal image coordinates into the conformal image coordinates. The formula is shown in Equation 7-1 where the vector (x', y') represents the non-conformal image coordinates and (x, y) are coordinates in the conformal image.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1.153x(1-6.410^{-4}y)^{-1} \\ 0.845y \end{pmatrix}$$

Equation 7-1 The transformation formula which shows the relation between the conformal and non-conformal image coordinates.

Transforming the scanned image using the equation shown in Equation 7-1 will have the effect that the physical direction of the finger on the scanner will be irrelevant in the context of performing the authentication (when the GrFinger Library is configured to allow rotations).

This should only be implemented if time allows it. It is not critical to the system but it will greatly increase the usability of the system because it will be less sensitive to finger placement on the scanner.

[13] http://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-Kiviharju/bh-eu-06-kiviarju.pdf

7.5 Server

The server architecture is heavily dependent on the presence of a SQL server that has support for .NET CLR assemblies (e.g. Microsoft MSSQL Server 2005). Although it is possible to connect to another SQL server from within the assembly it is not possible to run the assembly itself on a non-CLR enabled server.

Running the assembly directly within the SQL server allows for a context connection to be established which offers several advantages such as not having to re-authenticate each time a connection is established and completely bypassing network protocols and the transport layer. This will result in increased performance and less resource use[14].

Microsoft has listed certain limitations[15] when using the context connection but none of them impact this system. The computational requirements for the database server are increased because apart from fetching the records it also has to perform the verification itself. But since verification only requires one record to be fetched from the database, which is a very fast operation for the database engine to do, this will be negligible.

Had the system mainly been using identification as a means to authenticate then it would be desirable to separate the business layer completely from the database in order to employ multiple databases as discussed in section 5.4. As this is not the case with this remote user authentication system and it is a requirement from Logos A/S (see section 5.7) that the business layer is integrated into the SQL Server I will design the system as such.

As mentioned the server consists of three parts. The Web Service, Database, and the FingerprintController (.NET CLR) and in the following subsections I will describe the design of the database and FingerprintController. First I will present a Service-level sequence diagram for the server.

7.5.1 Service-level Sequence Diagram

In Figure 7-5 the service-level sequence diagram for the server is shown using Use Case 1 (Authenticate User) as example. There are two main points to notice in the diagram; the incoming requests are served from the IIS and directed to the SQL Server and then on to the FingerprintController (noted as Controller) and the FingerprintController uses the services provided by the Griaule GrFinger Library (noted as grFingerXLib). The FingerprintController (.NET CLR component) is the assembly that runs within the SQL Server and it handles the processing and authentication of users. Not shown is the IIS, which is responsible for passing the requests on to the SQL Server. This prevents the SQL Server from being exposed directly to the network thus minimizing intrusion risk. The IIS exposes the methods that can be used by the authentication device or the enrolment terminal (listed in Table 6-2). The methods exposed are available to any application that can reach the IIS, and therefore there should be some kind of authentication between the IIS and the devices communicating with it.

[14] http://msdn2.microsoft.com/en-us/library/ms131104.aspx
[15] http://msdn2.microsoft.com/en-us/library/ms131101.aspx

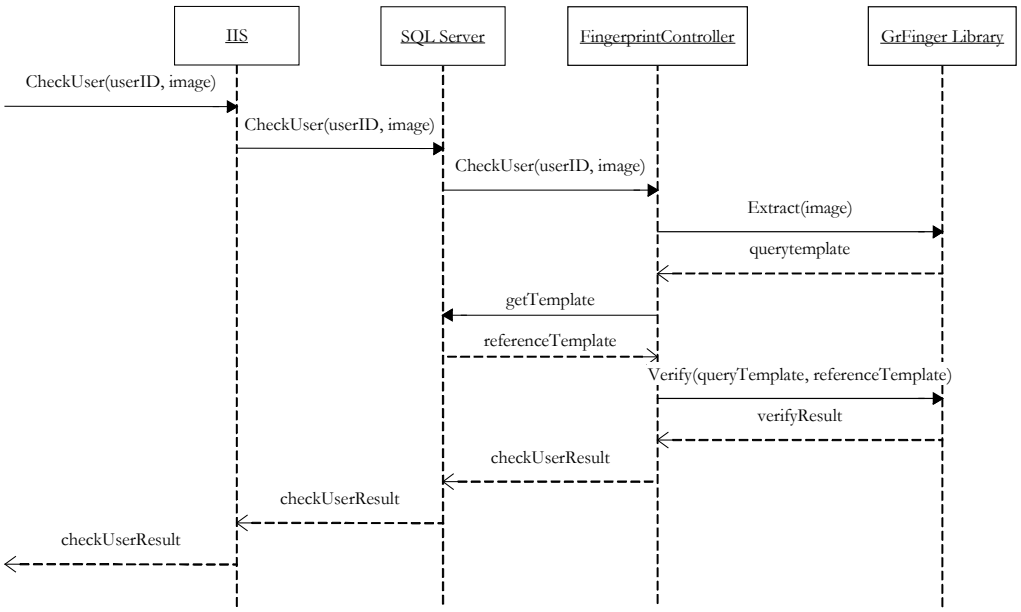

Figure 7-5 Service-level sequence diagram showing the server-side components.

7.5.2 The Server Database

The database plays a central role in this access-control system because this is where the templates and IDs of the enrolled users are stored. Furthermore the database will also hold the log for actions and events taking place on the authentication device and the FingerprintController.

In Figure 7-6 I have depicted a diagram of the showing the tables and their relations in the database. The figure shows that the database has been normalized to 3rd Normal Form.

In Table 7-1 a list of the entities in the database along with a short description of their purpose can be found. Only the tables Templates and Users are actually necessary from a functional point of view because the authentication only depends on the relation between a user and the associated template. The other tables have been added as a step towards a practical system where such features as logging and settings are desirable.

Entity Name	Description
Log	Holds a log of events and actions taking place in the FingerprintController and the Ni.
Log_Severity	Holds definitions of the different log levels. Used to filter the log.

Entity Name	Description
Terminals	Holds a list of the terminals in the system. Used by the FingerprintController to determine if a request is coming from a registered terminal or device.
Access_Attempts	Holds a log of all authentication attempts. Can be used for statistics and for analyzing possible fraud.
Templates	Holds the template associated with a user along with some properties.
Users	Contains users enrolled in the system.
Settings	Contains settings used by the FingerprintController.

Table 7-1 Table of entities in the database and a description of them.

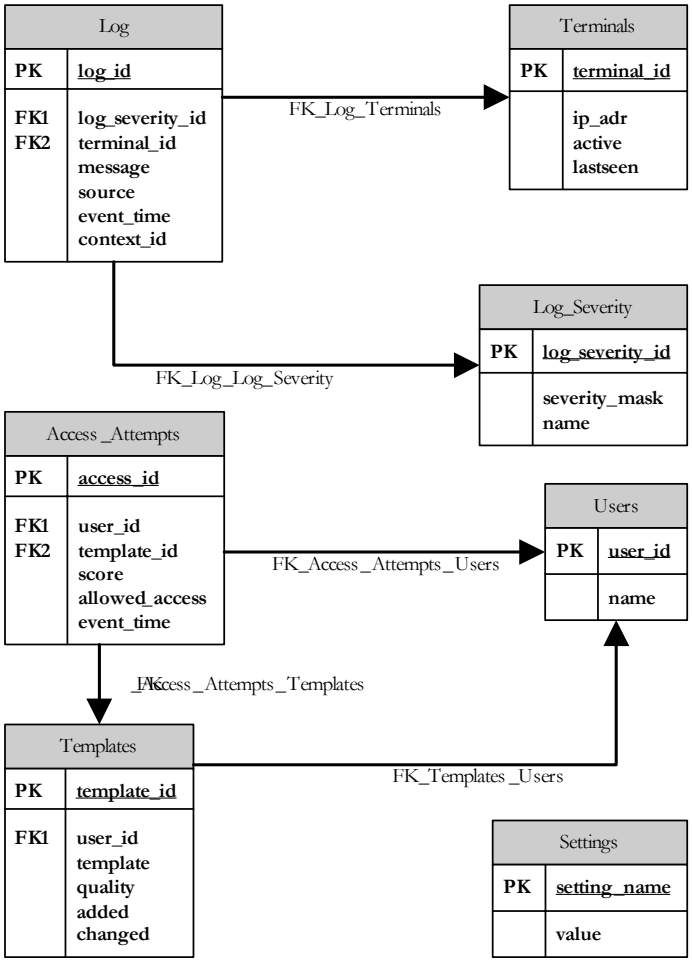

Figure 7-6 Database diagram showing the different entities and their relations. The database has been normalized to 3rd Normal Form.

The SQL Server must not accept any SQL statements from the outside - instead only stored procedures are used. This ensures that SQL and other code is nicely separated and that if the need to change the underlying table structure arises then it can be handled by changing the stored procedures in a central place. Performance is also increased because the SQL statement only has to be parsed and optimized the first time it is run.

The database must have stored procedures to perform task relating to the database. The stored procedures and a description of their functionality and the intended consumer of the procedure are listed in Table 7-2.

Procedure Name	**Description**	**Consumer**
`AddTemplate`	Adds a template to the database linking it with a user ID	FingerprintController
`CheckUser`	Provides the IIS web service with the ability to authenticate a user.	Authentication Device
`FingerprintController_CheckUser`	Authenticates a user.	SQL Server
`FingerprintController_FingerVersion`	Writes the version number of GrFinger to the log. Used for test.	N/A
`FingerprintController_InsertUser`	Inserts a user.	SQL Server
`GetNewestTemplateFromUserID`	Retrieves the template associated with a user. Used when authenticating.	FingerprintController
`InsertUser`	Provides the IIS web service with the ability to insert a user.	Enrolment Terminal
`LogAccessAttempt`	Used for logging of access attempts.	FingerprintController
`tester_FingerprintController_VerifyTemplate`	Verifies two templates. Used for testing.	N/A
`WriteLog`	Writes to the log. Only accessible to the SQL Server.	FingerprintController
`WriteLogFromNi`	Writes to the log. Accessible via the IIS web service.	Authentication Device or Enrolment Terminal

Table 7-2 Stored procedures available in the database.

7.5.3 FingerprintController (.NET CLR)

The FingerprintController will do the actual authentication and enrolling of users in the system. It will integrate into the database as an assembly and make use of the Griaule GrFinger Library and the database.

Class Name	Description
`Controller`	This is the class which the database passes the requests from the IIS server to. It uses the Façade pattern in that it hides all the complexity involved in authenticating and enrolling users.
`DatabaseService`	This class prepares and assembles requests to the database by constructing the list of parameters to the stored procedures being called.
`DatabaseInterface`	Does the work related to opening and closing connections to the database and executing SQL statements.
`GRFingerHandler`	Interfaces with the Griaule GrFinger Library.
`TTemplate`	Class for holding template structures. It has various methods useful for converting the template to and from Base64, which is the format they are stored in, in the database.
`Logger`	Constructs log messages and determine if a message should be written based on the log severity filter level set in the `Settings` class.
`Settings`	Has methods for setting and getting the settings used by the FingerprintController and GrFinger Library. This will allow for changes of settings while the system is running.
`Util`	Different methods for saving the received fingerprint to disk and for checking parameters received.

Table 7-3 Classes in FingerprintController and a description of them.

In Table 7-3 I have described the different classes found in FingerprintController.

The GRFingerHandler class provides the Controller class with lower level methods that access the Griaule GrFinger Library. In particular the methods ExtractTemplateFromBase64 and VerifyTemplate are important to notice. ExtractTemplateFromBase64 extracts the template from Base64 encoded image received from the terminal. Once the image is converted to a raw image it is passed to the GrFinger Library, which does the actual template extraction. The template can then be used for authentication or be inserted into the database.

Figure 7-7 shows the interaction between the classes and their methods using the authenticate user use case as example (Use Case 1). The CheckUser method is called whenever an authentication device initiates the authentication of a user.

Using the Facade pattern the Controller class hides all the steps involving the actual user authentication. Figure 7-7 shows that the procedure that the Controller class follows is to first extract the template from the image received from the authentication device. This template will be the query template. Using the card ID also received from the authentication device it retrieves the reference template associated with this ID. As a final step it passes the query and reference templates to the GrFinger Library in order to verify them. The GrFinger Library then matches the two templates and returns the result of the verification to the Controller. The Controller will then pass the result on to the SQL Server that will then pass it to the IIS server, which then sends the result of the authentication encapsulated in a SOAP message to the authentication device.

The sequence used for user insertion is similar to the one shown in Figure 7-7.

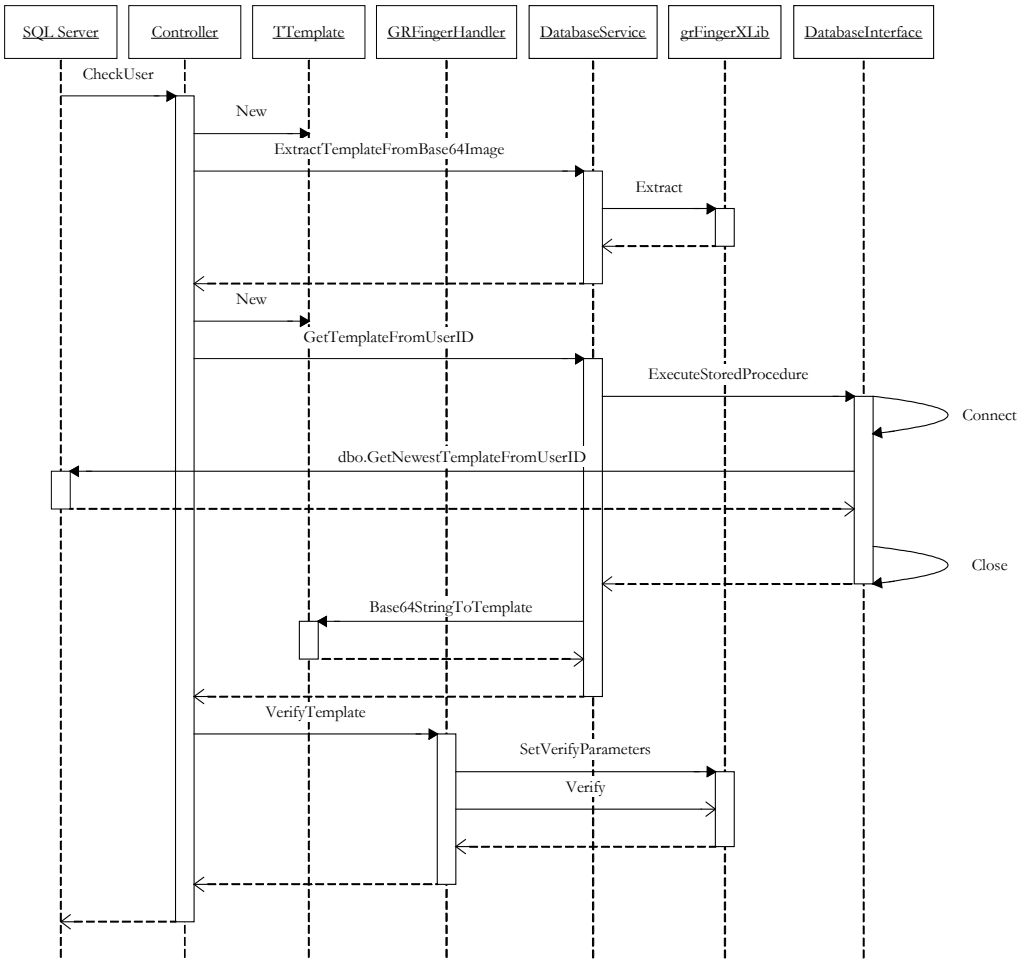

Figure 7-7 Sequence diagram for the FingerprintController.

Interaction with Griaule GrFinger Library

The GrFinger Library exposes several useful methods, which support the realization of the remote user authentication system. The primary methods used in the FingerprintController along with a description of them are listed in Table 7-4.

Function	Description
`Initialize`	Initializes the GrFinger Library. Must be called before any other method in the library can be called.
`Extract`	Given a pointer to a byte array containing a raw image along with its properties and other parameters it will return a template containing the features extracted from the raw image.
`Verify`	Will perform a verification given two pointers to two templates. Returns a value indicating the matching score and a return code of `GR_MATCH` or `GR_NOT_MATCH` indicating if the score is higher than the verification threshold specified using the `SetVerifyParameters` method.
`SetVerifyParameters`	Sets the verification score threshold and the rotation tolerance used when performing the verification.

Table 7-4 Description of the GrFinger Library methods primarily used by the FingerprintController.

7.6 Communication

The communication method chosen is SOAP. It has been chosen because the IIS is able to receive SOAP requests and direct them to an SQL Server using the SQLXML 3.0 component[16] provided by Microsoft. SOAP messages are transferred using the HTTP protocol.

A disadvantage in using SOAP is that binary data such as images cannot be encapsulated in XML because the binary stream may contain XML termination characters. This can be overcome by encoding the binary data in Base64 that only includes the upper- and lower-case Roman characters and the symbols "+", "/", and "=" which can all be placed legally within XML tags. Now, this conversion does not come for free as it increases the size of the data by approximately 33% (excluding padding) because essentially 3 bytes of binary data is converted into 4 bytes of Base64 data.

When the authentication device sends a message to the server to authenticate a user it contains around 150 kB data, which mostly consists of the image encoded in Base64. The image size is 384 x 269 pixels with a depth of 8 bits per pixel.

The message size will not be a problem when the system operates using a fast LAN but if a slow network is used it will be necessary to compress the image and maybe reduce it in dimensions. Using a 100 Mbit/s (12.5 MB/s) LAN the transmission time of a 150 KB message

[16] http://msdn2.microsoft.com/en-us/library/aa286527.aspx

is around 0.012 seconds (see Equation 7-2) assuming no other traffic and no protocol overhead.

$$\frac{0.15MB}{12.5MB/s} = 0.012s$$

Equation 7-2 Calculating the approximate transmission time of 150 KB data over a 100Mbit/s (12.5 MB/s) network.

The format of the SOAP messages used in this system is described in Appendix V (Section 16).

8 Implementation

In this section I will describe the implementation of some of the essential parts that were described in the design (Section 7).

8.1 Authentication Device

As I mentioned in section 7.3 some of the functionality of the enrolment terminal is integrated into the authentication device.

Code 8-1 shows the implementation of the AwaitFinger state, which is responsible for scanning the finger. After a successful scan it will change to state CheckUser or InsertUser based on the terminal type (see Figure 7-4 for a diagram of the state machine). These states will handle the sending of requests to the server. The messages on the display are set using the `GControlSetText` methods.

```
case AwaitFinger:      //1
  cardRead = FALSE;
  if ((++scanCount <= ALLOWED_FAILED_SCANS) && (cardInReader)) {
    if (GetAndWaitFinger(fingerScan)) {
      GControlSetText(psd, Text1, "Scanned...");
      if (TerminalType == ENROLMENT_TERMINAL) {      // Enrol user
        scanState=InsertUser;
      }
      else if (TerminalType == AUTHENTICATION_DEVICE) {//auth. user
        scanState=CheckUser;
      }
    }
    else {
      sprintf(s, "Retries left: %d", ALLOWED_FAILED_SCANS - scanCount);
      GControlSetText(psd, Text2, s);
    }
  }
  else {
    scanState = FailedScan;
  }
break;
```

Code 8-1 AwaitFinger state in the state machine on the authentication device.

8.2 Server

In the following subsections I will describe the implementation of selected parts of the server components.

8.2.1 The Server Database

The FingerprintController component is inserted into the database as an assembly. In Code 8-2 I have shown the SQL statements necessary to create the assembly within the SQL Server. The important thing to notice in the code is that it is create with permission set to `UNSAFE`. This is necessary because the GrFinger Library runs as unmanaged code and references are used within the FingerprintController. The references are used when the GrFinger Library extracts the template from an image and when performing verification or identification.

```
use fingerprint
IF  EXISTS (SELECT * FROM sys.assemblies asms
  WHERE asms.name = N'FingerprintController')
  ALTER ASSEMBLY [FingerprintController]
  FROM 'C:\Program Files\Microsoft SQL
    Server\MSSQL.4\MSSQL\Data\FingerprintController.dll'
  GO
CREATE ASSEMBLY [FingerprintController]
  AUTHORIZATION [dbo]
  FROM 'C:\Program Files\Microsoft SQL
    Server\MSSQL.4\MSSQL\Data\FingerprintController.dll'
  WITH PERMISSION_SET = UNSAFE
  GO
```

Code 8-2 Establishing the link between the SQL Server and FingerprintController assembly.

The binding between the SQL Server and the FingerprintController are made as stored procedures. In Code 8-2 I have listed such a binding. By using the `EXTERNAL NAME` keyword in SQL it is possible to create a binding to a `public` method defined in the assembly.

```
use fingerprint
--- CHECK USER ---
CREATE PROC FingerprintController_CheckUser
  @queryImage nvarchar(max),
  @width int,
  @height int,
  @res int,
  @referenceID int,
  @contextID int,
  @terminalID int
AS
  EXTERNAL NAME FingerprintController.[FingerprintController.Controller].CheckUser
  GO
```

Code 8-3 Linking into the methods exposed by the FingeprintController from the SQL Server.

8.2.2 FingerprintController (.NET CLR)

The FingerprintController is implemented as a .NET CLR assembly using C# .NET version 2.0. It makes use of the external library Griaule GrFinger Fingerprint SDK Recognition Library 4.2. I will start by describing how I use the `Controller` class and then move on to how the FingerprintController interacts with the GrFinger Library.

The Controller Class

The `Controller` class contains the methods that the clients (authentication device and enrolment terminal) of the web service can consume. It is only in this class (and the `TTemplate` class for reasons explained in the section below) that methods are declared as `public`. All other classes have their methods declare as `internal` or `private`. A method declared using the `internal` keyword ensures that it only can be used by other methods within the assembly. This will prohibit unintentional referencing of methods that are only for use within the assembly.

In Code 8-4 a shortened version of the `CheckUser` method is shown. The code shows how a template is first extracted from the incoming image (`queryImage`). Then a template is retrieved from the database based on the card ID (`referenceID`) supplied by the user. Lastly the two templates are matched and a result is returned. Also showing is that at the points where the method exits it writes a message to the access attempt log indicating the result of the attempt.

```
public static int CheckUser(String queryImage, int width, int height,
  int res, int referenceID, int contextID, int terminalID)
{
  // Create template from imagestring
  TTemplate queryTemplate = new TTemplate();
  ret = GRFingerHandler.ExtractTemplateFromBase64Image(queryImage,
    width, height, res, contextID, terminalID, ref queryTemplate);
  if (ret < 0) {  // Error
    Logger.WriteError("Aborting user authentication. Template could not be
      extracted. Ret: " + ret, terminalID, contextID);
    Logger.WriteAccessAttempt(referenceID, matchScore, false, templateID);
    return ret;
  }
  // Find the referencetemplate in DB
  TTemplate referenceTemplate = new TTemplate();
  ret = DatabaseService.GetTemplateFromUserID(referenceID, ref
    referenceTemplate, ref templateID);
  // Compare the two
  ret = GRFingerHandler.VerifyTemplate(queryTemplate, referenceTemplate,
    contextID, terminalID, ref matchScore);
  Logger.WriteAccessAttempt(referenceID, matchScore, ret ==
    (int)GRConstants.GR_MATCH ? true : false, templateID);
  return ret;
}
```

Code 8-4 Example from CheckUser method in the Controller class.

Interaction With the GrFinger Library

The FingerprintController uses the GrFinger Library to perform verification and extraction of templates. In Code 8-5 I have listed a section of a function in the `GrFingerHandler` class. The `ExtractTemplateFromBase64Image` method shown uses the `Extract` function in the GrFinger Library. The code shows that the `Extract` function takes references to the image (noted as `rawImage`) from which the template will be extracted and to the template structure (noted as `temp.tpt`) where the extracted template will be placed. The references along with the `unsafe` property of the method shown make it possible for the GrFinger Library to write to the addresses passed to it.

```
internal unsafe static int ExtractTemplateFromBase64Image(String queryImage, int
  width, int height, int res, int contextID, int terminalID, ref TTemplate
  queryTemplate)
{
  .
  .
  object rawImage = cleanedImage;
  TTemplate temp = new TTemplate();
  temp.size = (int)GRConstants.GR_MAX_SIZE_TEMPLATE;
  ret = grFingerXLib.Extract(ref rawImage, width, height, res, ref temp.tpt, ref
    temp.size, contextID);
  Array.Copy(temp.tpt, queryTemplate.tpt, temp.size);
  queryTemplate.size = temp.size;
  queryTemplate.Quality = ret;
  .
  .
  return ret;
}
```

Code 8-5 Call to the GrFinger Library from the FingerprintController.

9 Test

9.1 Authentication Device

The first functional test of the authentication device involving sending a full authenticate user request to the server revealed an error in the code that receives HTTP messages on the Ni. When the authentication device sends a CheckUser message it sends around 150 KB of data and while it is sending this data the IIS sends HTTP ACK messages back. Now the code receiving the messages was not prepared for the ACK messages having no header and thus a function returned as failed.

By using the Wireshark Network Protocol Analyzer and some debugging messages on the authentication device the offending function that resided in the HTTP response header handling code was found. The function was patched such that it would accept HTTP containing an empty header. Following tests revealed no side effect of this patching.

9.2 Acceptance Test of the Remote User Authentication System

For the acceptance test of the remote user authentication system I will run the test cases shown in Table 9-1. The table lists the test cases and the purpose. The complete test cases can be found in section 13.1 in Appendix II – Test (section 13) which are made from the use cases shown in Appendix I – Use Cases (section 12). The complete test reports are found in section 13.2 in Appendix II – Test (section 13) and a summary of the reports can be found in Table 9-2.

Name	**Purpose**
Enroll User (Test Case 1)	To ensure that the remote user authentication system is successfully able to enroll a user in the system (i.e. complete the main sequence in Use Case 2 - Enroll User (version 1.1).
Authenticate User (Test Case 2)	To ensure that the remote user authentication system is successfully able to authenticate an enrolled user (i.e. complete the main sequence in Use Case 1 – Authenticate User (version 1.2).
Invalid Card Insertion (Test Case 3)	To ensure that if card is invalid display text "Card read error. Remove card." and aborts the enrolment and returns to its base state.

Name	Purpose
Failed Placing Finger on Reader (Test Case 4)	To ensure that if the scanning of a fingerprint fails the enrolment terminal or authentication device will return to its base requiring the user to remove the card. This test the behaviour specified in Use Case 1 – Authenticate User (version 1.2) and in Use Case 2 – Enroll User (version 1.1).
Failed to Place Correct Finger on Reader (Test Case 5)	To ensure that authentication of user fails if the correct finger is not scanned on the authentication device. This test the behaviour specified in Use Case 1 – Authenticate User (version 1.2).

Table 9-1 Summarized test cases and a description of their purpose.

Name	Result
Enroll User (Test Report 1)	The remote user authentication system is able to enroll a user in the system.
Authenticate User (Test Report 2)	The remote authentication system is able to authenticate an enrolled user.
Invalid Card Insertion (Test Report 3)	The device can determine if a card is inserted in an invalid manner respond correctly.
Failed Placing Finger on Reader (Test Report 4)	The device will retry to scan a finger three times after the initial scan has been attempted acting as described by the use case.
Failed to Place Correct Finger on Reader (Test Report 5)	The remote authentication system is able to recognize that another finger than the one used during enrollment was used during the authentication and correctly rejects the users.

Table 9-2 Results from the test reports.

The test has revealed that the remote user authentication works as expected. In Enroll User (Test Report 1) (section 13.2.1) and Authenticate User (Test Report 2) (section 13.2.2) it has been remarked that when performing a user authentication or enrollment the first time right after server start up there is a notable delay of around 5 seconds for the process to complete. This only occurs the first time that either of these methods is called. This indicates that the FingerprintController is not preloaded when the SQL Server is started and that there is a start up cost involved in instantiating it. It should be researched if it is possible to preload the FingerprintController when starting the SQL Server.

Invalid Card Insertion (Test Report 3) (section 13.2.3), Failed Placing Finger on Reader (Test Report 4) (section 13.2.4), and Failed to Place Correct Finger on Reader (Test Report 5) (section 13.2.5) showed that the authentication device works as expected.

The remote user authentication system developed is able to perform basic tasks required by an authentication system. It can enroll users and authenticate them at a later time.

10 Conclusion

The objective of this project was to create a working remote user authentication system using embedded systems and a web service within the projects duration of 10 weeks, and this goal has been fully achieved. It has been shown that using biometrics, as a means to authenticate, is viable in large distributed embedded systems by using verification as the method for authentication.

Informal tests of the system have shown a false rejection rate (FAR) of less than 5% where most of these rejections where due to incorrect placement or pressure of the finger on the fingerprint reader. Implementation of the fingerprint image correction algorithm discussed in the report is expected to decrease the FAR and thus a success rate close to 100% can be achieved. The tests have not shown a single case where a user was identified incorrectly which means a false identification rate of much less than 1% can be assumed. A formal test of the reliability of the remote user authentication system was deemed outside of the scope of the project.

By the end of this project I conclude that none of the risks identified at the beginning of the project became an issue.

The design of the remote user authentication allows for several future extensions such as automatic enrolment of users when they use the system the first time. Also the design allows for adaptation into other forms of access-control systems because the embedded authentication device decides what to do when a user has been authenticated. This would only require minor changes to the authentication device and none to the server.

The remote user authentication system presented herein could, with some modifications, be used in other contexts such as a supermarket payment system or as an access-control system for large corporations.

The future of user authentication systems using biometrics like the one presented herein looks very bright. Some car manufacturers now use biometric authentication systems instead of keys. A similar system is also seeing use in private peoples homes.

Though the future seems bright for finger print biometrics, there is good reason to tread carefully. As a fingerprint is not replaceable (a person cannot be issued a new finger), great care should be taken when processing and storing it. This is largely a privacy issue that probably and hopefully will continue to be discussed in the future.

11 References

1. http://www.m-w.com/dictionary/biometrics
2. http://ctl.ncsc.dni.us/biomet%20web/BMHistory.html
3. http://en.wikipedia.org/wiki/Fingerprint#Timeline
4. http://www.itsc.org.sg/synthesis/2002/biometric.pdf
5. http://www.griaule.com/page/en-us/grfinger_fingerprint_sdk#t6
6. http://www.datatilsynet.dk/lovgivning/personoplysninger/indhold.asp
7. http://www.griaule.com/page/en-us/grfinger_fingerprint_sdk
8. http://www.ibia.org
9. http://www.ibia.org/biometrics/industrynews_view.asp?id=347
10. http://biometrics.cse.msu.edu/Publications/SecureBiometrics/UludagJain_BiometricAttacks_SPIE04.pdf
11. T. Putte and J. Keuning, "Biometrical fingerprint recognition: don't get your fingers burned", Proc. IFIP
12. TC8/WG8.8, Fourth Working Conf. Smart Card Research and Adv. App., pp. 289-303, 2000.
13. http://www.bromba.com/faq/biofaqe.htm#FIR
14. http://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-Kiviharju/bh-eu-06-kiviarju.pdf
15. http://msdn2.microsoft.com/en-us/library/ms131104.aspx
16. http://msdn2.microsoft.com/en-us/library/ms131101.aspx
17. http://msdn2.microsoft.com/en-us/library/aa286527.aspx
18. http://www.wireshark.org

12 Appendix I – Use Cases

12.1 Authentication Device

Unless noted otherwise each use case in this section assumes that the authentication device is in its base state showing a welcome screen. When the device is in the base state it is ready to receive commands from the user and the server.

If an authentication device does not have network access or is unable to contact the server for other reasons it is considered offline.

12.1.1 Authenticate User (Use Case 1)

Use case 1		**Authenticate user**	**Version:**	**1.2**
Objective		The system must be able to authenticate a user and take action based on the outcome of the authentication.		
Priority		High.		
Prerequisites		The Authentication Device is not offline. A user with a valid ID card.		
Action on success		The text "User authenticated. Access granted." is shown on the display.		
Action on error		The text "Access denied." is shown on the display.		
Primary and secondary actors		User (primary) Server (extern)		
Trigger		The user presents the ID card to the system.		
Sequence	Step	Action		
	1	The authentication device reads the ID off of the ID card.		
	2	The display shows the text "Place finger on scanner".		
	3	An image of the fingerprint is acquired within timeout.		
	4	Authentication device scans the finger and the display shows the text "Scanned…".		
	5	Send ID and fingerprint to server		
	6	Fingerprint and ID is authenticated and the display shows the text "Authorized."		

Use case 1		**Authenticate user**	**Version:**	**1.2**
Alternative Sequence	Step	Action		
	3a	User fails to place finger correctly on scanner within timeout		
	3b	The display shows the text "Could not scan finger".		
	5a	The server does not respond within timeout.		
	5b	The display shows the text "Could not process request. Please contact service."		
	6a	Fingerprint and ID does not match so the display shows the text "Not authorized."		
Extensions	Step	Action		
	2	A timer showing time left before timeout.		
Non-functional		Timeout: Timeout should be no more than 15 seconds.		
Issues		N/A		

12.2 Enrolment Terminal

The enrolment terminal must have a display, card writer, and a fingerprint reader connected to it. Unless otherwise is stated each use case in this section assumes that the enrolment terminal is in its basic state showing the program main screen. When the enrolment terminal is in its base state it is ready to receive commands from the user and server.

If an enrolment terminal does not have network access or is unable to contact the server for other reasons it is considered offline.

12.2.1 Enroll User (Use Case 2)

Use case 2	**Enroll User**	**Version:**	**1.1**
Objective	The system must be able to accept new users so that they can use the remote user authentication system.		
Priority	High		
Prerequisites	The Enrolment Terminal is in its base state.		
Action on success	The display shows the text "User successfully enrolled. User ID: {*USER_ID*}".		
Action on error	The display shows the text "User could not be enrolled.		

Use case 2		**Enroll User**	**Version:**	**1.1**
		Error: {*ERROR_DESCRIPTION*}".		
Primary and secondary actors		Operator (primary) User (secondary) Server (extern)		
Trigger		An Operator selects the "Enroll User" option on the Enrolment Terminal.		
Sequence	Step	Action		
	1	The display shows the text "Please present ID card to system."		
	2	The Operator presents the ID card and the terminal reads the ID from it and sends it to the server.		
	3	If card is valid the display shows "Place finger on scanner.".		
	4	User places finger correctly on scanner.		
	5	Finger is scanned and display shows "Finger scanned".		
	6	Enroll user on server.		
	7	If successful enrolment display text "User Enrolled. Remove Card."		
Alternative Sequence	Step	Action		
	3a	If card is invalid (already in use or other) display text "Card read error. Remove card."		
	3b	Return to base state.		
	4a	If scanning is unsuccessful return to step 3. If step 7 has failed a total of 3 times abort the enrolment and return to base state.		
	6a	If the authentication fails to contact the server it will retry up to 3 times. If it has failed all three retries it will display the text "Could not process request. Contact service."		
Extensions	Step	Action		
Non-functional				
Issues				

12.2.2 Identify User (Use Case 3)

Use case 3		**Identify User**	**Version**	**1.0**
Objective		The remote user authentication system must be able to create a new ID card for a user if the user has lost the ID card.		
Priority		Low		
Prerequisites		The enrolment terminal is in its base state.		
Action on success		The display shows the text "User identified as user ID: {*ID*}".		
Action on error		The display shows the text "User could not be identified.".		
Primary and secondary actors		Operator (primary) User (secondary) Server (extern)		
Trigger		An Operator selects "Identify User" option on the Enrolment Terminal.		
Sequence	Step	Action		
	1	The display shows the text "Place finger on scanner".		
	2	User places finger on scanner.		
	3	Finger is scanned and display shows "Finger scanned".		
	4	Identify user on server		
	5	The display shows the text "User identified as user ID: {*ID*}".		
Alternative Sequence	Step	Action		
	5a	If the user was not identified the text "User could not be identified." is displayed.		
Extensions	Step	Action		
Non-functional				
Issues				

12.2.3 Discard User (Use Case 4)

Use case 4		**Discard User**	**Version:**	**1.0**
Objective		The remote user authentication system must be able to discard users if they no long wish to be part of the system.		
Priority		Low		
Prerequisites		The Enrolment Terminal is in its base state.		
Action on success		The display shows the text "User with user ID: {*ID*} has been removed from the system.".		
Action on error		The display shows the text "User with user ID: {*ID*} could not be removed from the system. Error: {*ERROR_DESCRIPTION*}"."		
Primary and secondary actors		Operator (primary) User (secondary) Server (extern)		
Trigger		An Operator selects the "Remove User" option on the Enrolment Terminal.		
Sequence	Step	Action		
	1	The display shows the text "Please present ID card to system."		
	2	The Operator presents the ID card and the terminal reads the ID from it and sends it to the server for deletion.		
	3	If the deletion was successful display the text "User with user ID: {*ID*} has been removed from the system.".		
Alternative Sequence	Step	Action		
Extensions	Step	Action		
Non-functional				
Issues		If a user wants to be deleted from the system he should prove his identity to the system (using a fingerprint).		

13 Appendix II – Test

13.1 Test Cases

13.1.1 Enroll User (Test Case 1)

Test Case 1	**Enroll User**		**Version:**	**1.0**
			Date:	**March 22. 2007**
Purpose	To ensure that the remote user authentication system is successfully able to enroll a user in the system (i.e. complete the main sequence in Use Case 2 - Enroll User (version 1.1).			
Prerequisites	The remote user authentication system is configured as described in 14, 15, and the database is properly setup with tables and a FingerprintController. The enrolment terminal is powered up and connected to the server.			
Test Data	A chip card with an ID *N* and a user which will use the finger *F*.			
Steps	1.	Approach the enrolment terminal and verify that the terminal is displaying the message "Insert Card.".		
	2.	Insert the chip card with ID *N* in the card reader attached.		
	3.	Verify that the terminal displays "Place finger on reader.".		
	4.	Place finger *F* on the reader and allow it to scan the finger.		
	5.	Verify that the terminal displays "Scanned."		
	6.	Verify that the terminal displays "User Enrolled."		
	7.	Verify that the terminal displays... "User Enrolled. Remove Card".		
Expected Result	The terminal displays "User Enrolled. Remove Card" and a user have been created in the database (linked to the chip cards ID *N*).			
Notes				

13.1.2 Authenticate User (Test Case 2)

Test Case 2	**Authenticate User**	**Version:**	**1.0**
		Date:	**March 22. 2007**
Purpose	To ensure that the remote user authentication system is successfully able to authenticate an enrolled user (i.e. complete the main sequence in Use Case 1		

	authenticate an enrolled user (i.e. complete the main sequence in Use Case 1 – Authenticate User (version 1.2).	
Prerequisites	The remote user authentication system is configured as described in 14, 15, and the database is properly setup with tables and a FingerprintController. The authentication device is powered up and connected to the server. A user is enrolled using card ID *N* and finger *F*.	
Test Data	An enrolled user with card ID *N* and finger *F* (follow the steps presented in Test Case 1 - Enroll User (version 1.0) to enrol a user properly.	
Steps	1.	Approach the device and verify that it displays "Insert card."
	2.	Insert card with ID *N*.
	3.	Verify that the device displays "Place finger on reader."
	4.	Place finger *F* on reader.
	5.	Verify that the device displays "Scanned…"
	6.	Verify that the device displays "Processing authentication".
	7.	Verify that the device displays "Authorized. Remove card".
Expected Result	The device displays "Authorized. Remove Card". The database table Access_Attempts will contain a record of the authentication performed.	
Notes		

13.1.3 Invalid Card Insertion (Test Case 3)

Test Case 3	**Invalid Card Insertion**	**Version:**	**1.0**
		Date:	**March 22. 2007**
Purpose	To ensure that if card is invalid display text "Card read error. Remove card." and aborts the enrolment and returns to its base state.		
Prerequisites	The remote user authentication system is configured as described in 14, 15, and the database is properly setup with tables and a FingerprintController. The enrolment terminal is powered up and connected to the server.		
Test Data	A chip card with an ID *N*.		
Steps	1.	Approach the enrolment terminal and verify that the terminal is displaying the message "Insert Card.".	
	2.	Insert the chip card with ID *N* with the chip facing downwards in the card reader attached.	
	3.	Verify that the terminal displays "Card read error. Remove card.".	

	4.	Remove the card.
	5.	Verify that the terminal displays "Insert Card."
Expected Result	The terminal has returned to is in its base state.	
Notes		

13.1.4 Failed Placing Finger on Reader (Test Case 4)

Test Case 4	**Failed Placing Finger on Reader**	**Version:**	**1.0**
		Date:	**March 22. 2007**
Purpose	To ensure that if the scanning of a fingerprint fails the enrolment terminal or authentication device will return to its base requiring the user to remove the card. This test the behaviour specified in Use Case 1 – Authenticate User (version 1.2) and in Use Case 2 – Enroll User (version 1.1).		
Prerequisites	The remote user authentication system is configured as described in 14, 15, and the database is properly setup with tables and a FingerprintController. The enrolment terminal or authentication device is powered up and connected to the server.		
Test Data	A chip card with an ID *N* and a user which will use the finger *F*.		
Steps	1.	Approach the enrolment terminal and verify that the terminal is displaying the message "Insert Card.".	
	2.	Insert the chip card with ID *N* in the card reader attached.	
	3.	Verify that the terminal displays "Place finger on reader.".	
	4.	Allow the device to make three retries. (After each retry it must display the number of retries left.)	
	4.	After three retries the terminal should display "Could not scan finger..." for a couple of seconds.	
	5.	The terminal displays "Card read error. Remove card"	
	6.	Remove card.	
Expected Result	The terminal has returned to its base state.		
Notes			

13.1.5 Failed to Place Correct Finger on Reader (Test Case 5)

Test Case 5	**Failed to Place Correct Finger on Reader**	**Version:**	**1.0**
		Date:	**March 22. 2007**
Purpose	To ensure that authentication of user fails if the correct finger is not scanned on the authentication device. This test the behaviour specified in Use Case 1 – Authenticate User (version 1.2).		
Prerequisites	The remote user authentication system is configured as described in 14, 15, and the database is properly setup with tables and a FingerprintController. The authentication device is powered up and connected to the server. A user is enrolled using card ID *N* and finger *F1*.		
Test Data	A chip card with an ID *N* and a user which will use the finger *F2* for the scan.		
Steps	1. Approach the enrolment terminal and verify that the terminal is displaying the message "Insert Card.".		
	2. Insert the chip card with ID *N* in the card reader attached.		
	3. Verify that the terminal displays "Place finger on reader.".		
	4. Place finger *F1* on the reader		
	4. Verify that the device displays "Scanned…"		
	5. Verify that the device displays "Processing authentication".		
	6. Verify that the device displays "Not Authorized. Remove card".		
Expected Result	The device displays "Not Authorized. Remove Card". The database table Access_Attempts will contain a record of the authentication attempt performed.		
Notes			

13.2 Test Reports

13.2.1 Enroll User (Test Report 1)

Test Report 1	**Enroll User**	**Version:**	**1.0**
		Date:	**March 23. 2007**
Test Case Used:	Enroll User (Test Case 1) (Section13.1.1)	Version:	1.0

Summary	The terminal acted as stated in the test case. Checking the database revealed that a new user had indeed been created. The message written to the server log was: "A new user has been created with UserID: 101" and a new template had been inserted relating to the user with ID 101.
Result	The remote user authentication system is able to enroll a user in the system.
Remarks	When the server has just been started the response time to enroll the user is longer (up to 5 seconds). This only happens when the enroll user is executed as the first method call after server startup. This indicates a startup cost occurring at the server. This can result in the first user not being enrolled.

13.2.2 Authenticate User (Test Report 2)

Test Report 2	**Authenticate**	**Version:**	**1.0**
		Date:	**March 23. 2007**
Test Case Used:	Authenticate User (Test Case 2) (Section 13.1.2)	Version:	1.0
Summary	The terminal acted as stated in the test case. The Access_Attempts table in the database showed the authentication was successful and access was allowed for the user.		
Result	The remote authentication system is able to authenticate an enrolled user.		
Remarks	When the server has just been started the response time to authenticate the user is longer (up to 5 seconds). This only happens when the authentication user is executed as the first method call after server startup. This indicates a startup cost occurring at the server. This can result in the first user not being authenticated.		

13.2.3 Invalid Card Insertion (Test Report 3)

Test Report 3	**Invalid Card Insertion**	**Version:**	**1.0**
		Date:	**March 23. 2007**
Test Case Used:	Invalid Card Insertion (Test Case 3) (Section13.1.3)	Version:	1.0
Summary	The device acts promptly with an error message when a card is inserted as described in the test case.		

	as described in the test case.
Result	The device can determine if a card is inserted in an invalid manner respond correctly.
Remarks	N/A

13.2.4 Failed Placing Finger on Reader (Test Report 4)

Test Report 4	**Failed Placing Finger On Reader**	**Version:**	**1.0**
		Date:	**March 23. 2007**
Test Case Used:	Failed Placing Finger on Reader (Test Case 4) (Section 13.1.4)	Version:	1.0
Summary	The device acts as described in the test case.		
Result	The device will retry to scan a finger three times after the initial scan has been attempted acting as described by the use case.		
Remarks	The device should abort the retries if the card is removed from the card reader. It is an annoyance because the next user in line cannot use it until it has finished the three retries.		

13.2.5 Failed to Place Correct Finger on Reader (Test Report 5)

Test Report 5	**Failed to Place Correct Finger on Reader**	**Version:**	**1.0**
		Date:	**March 23. 2007**
Test Case Used:	Failed to Place Correct Finger on Reader (Test Case 5) (Section 13.1.5)	Version:	1.0
Summary	The authentication device processes the authentication as stated. It correctly displays "Not authorized. Remove card.".		
Result	The remote authentication system is able to recognize that another finger than the one used during enrollment was used during the authentication and correctly rejects the users.		
Remarks	N/A		

14 Appendix III – Configuration of IIS

Here I will show how to configure the IIS so that it can receive requests from the authentication device and pass them to the SQL Server. It is assumed that the IIS is installed.

Using the Windows Control Panel navigate to Administrative Tools and open Internet Information Services. A window similar to the one shown in Figure 14-1 will appear.

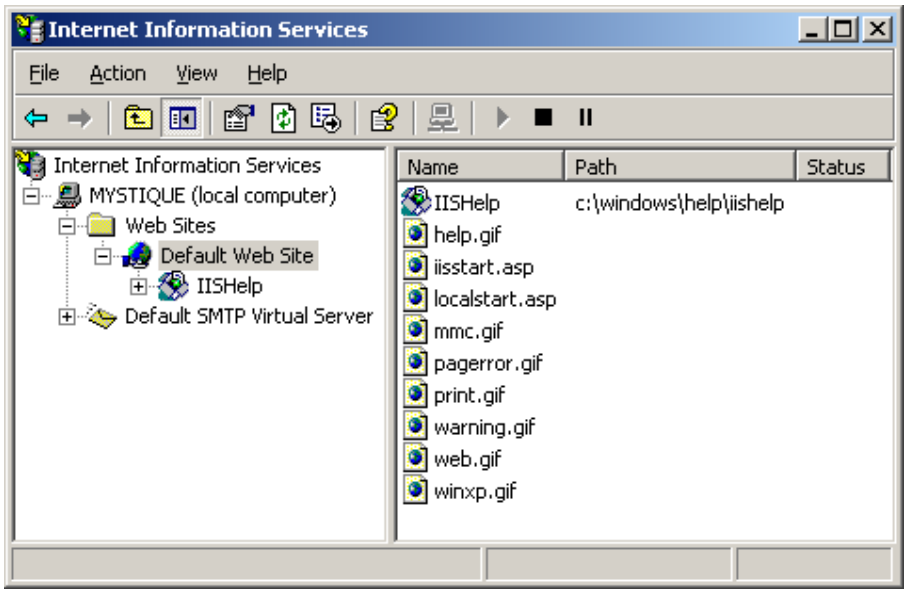

Figure 14-1 The Microsoft Internet Information Services (IIS)

Right-click "Default Web Site" and select "Properties". Select "Directory Security" and then click "Edit" in the "Anonymous access and authentication control" section. A window similar to the one shown in Figure 14-2 will appear. Make sure that "Anonymous access" and "Allow IIS to control password" is checked.

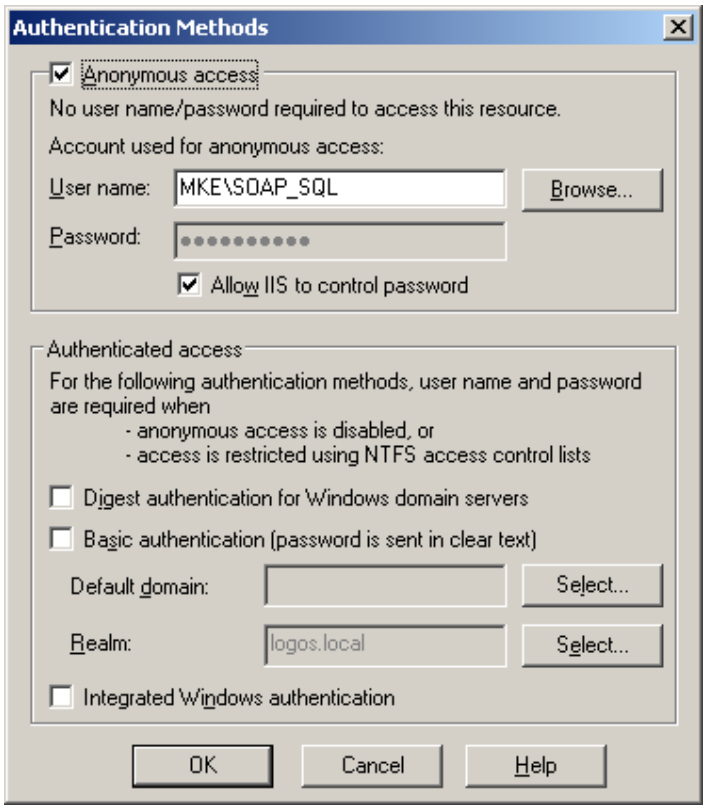

Figure 14-2 Default Web Site Authentication Methods settings.

Right-click "Default Web Site" and select "New"->"Virtual Directory...". A Virtual Directory Wizard will appear. Click Next.

The wizard will request an alias for the virtual directory. This must be "FingerPrintService". Click Next.

Now enter a path to the directory of your choice. This directory can be any directory. An empty directory is preferable. Click Next.

Now the wizard will ask you to set the access permissions. We will configure the IIS later so just click Next. Then Finish.

A new item called FingerPrintService will now appear under the Default Web Site. Right-click it and select properties. The properties shown should match the ones shown in Figure 14-3 (the "Local path" field may differ). The IIS is now correctly configured.

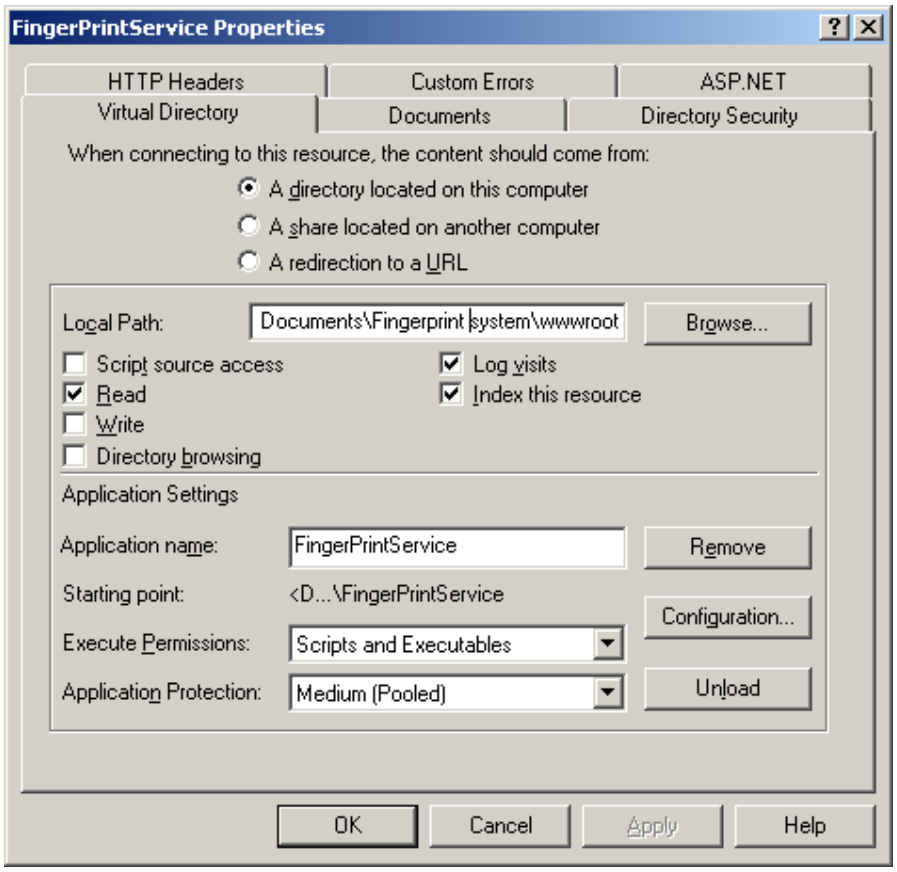

Figure 14-3 The properties of the FingerPrintService virtual directory.

15 Appendix IV – Configuration of SQLXML 3.0

The configuration of SQLXML 3.0 assumes that the IIS has been configured following the steps presented in Appendix III – Configuration of IIS (section 14) and that SQLXML 3.0 is installed and Microsoft SQL Server 2005 is installed and setup correctly.

The configuration of SQLXML 3.0 can be started from the Windows start menu.

In the SQLXML utility right-click "FingerPrintService" and select "Properties".

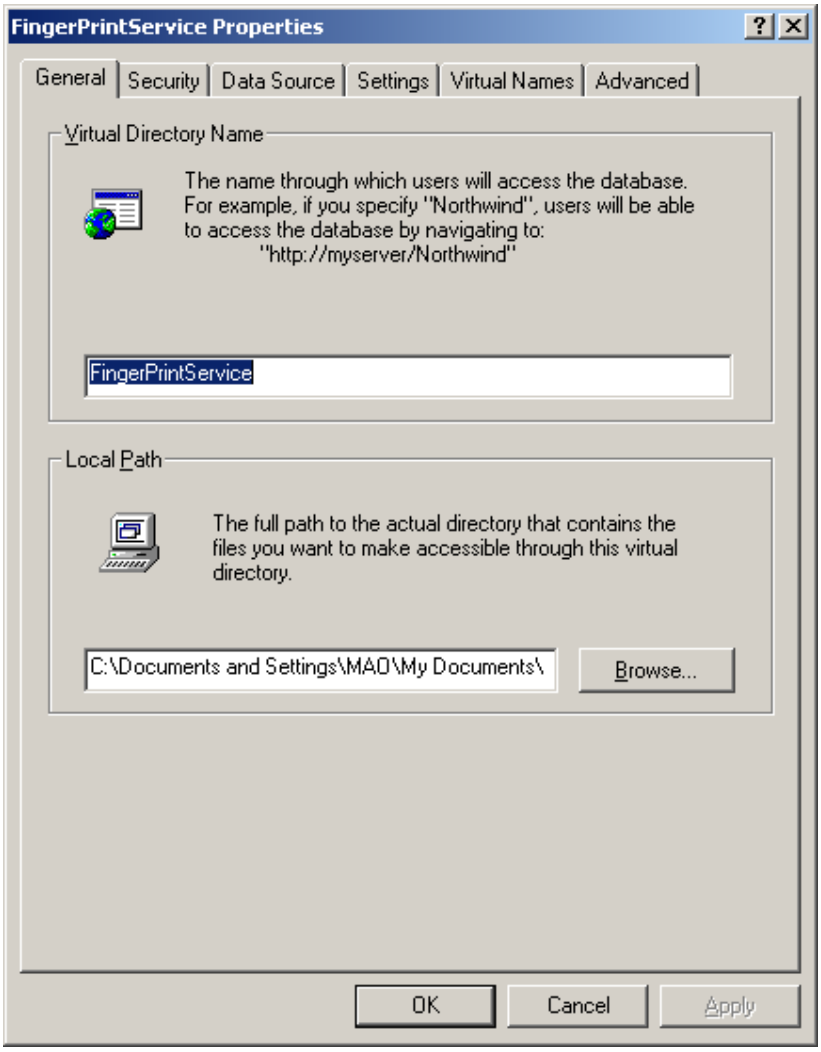

Figure 15-1 The SQLXML 3.0 properties window for FingerPrintService.

Figure 15-1 shows the window that will appear when entering properties for FingerPrintService. The correct settings for the tabs will be shown in the following figures.

The "Security" tab settings should reflect those shown in Figure 15-2. The User name and password may differ.

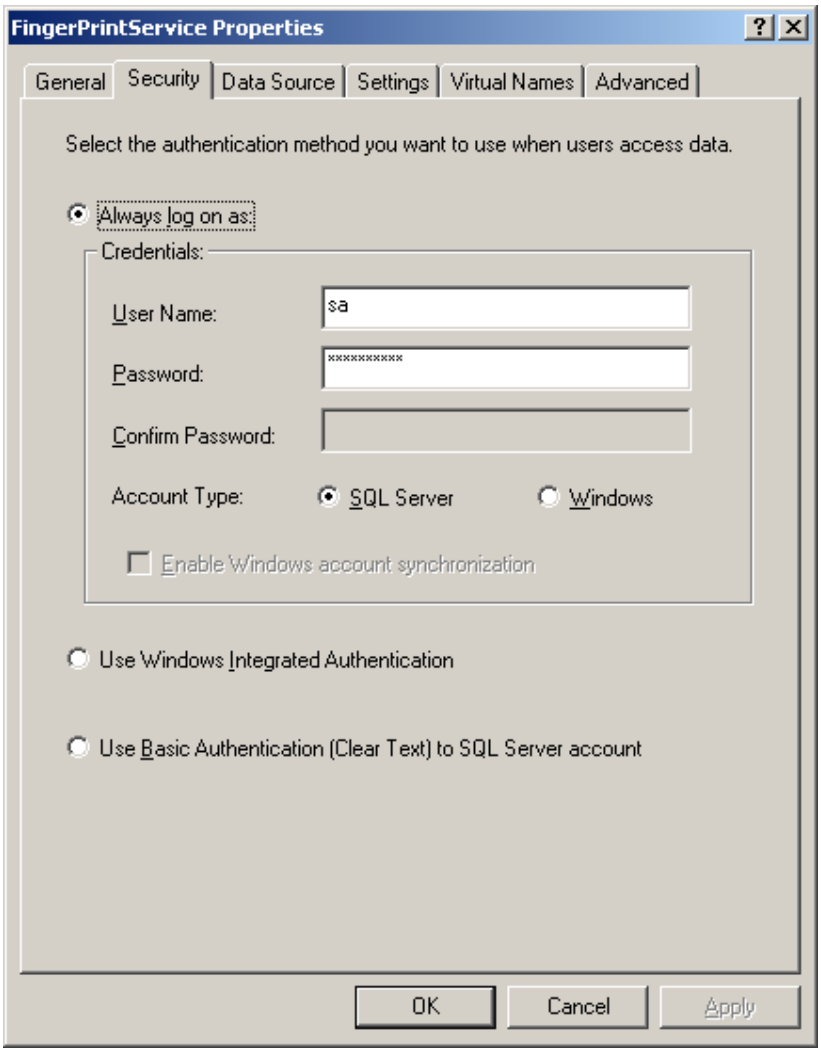

Figure 15-2 Settings for the Security tab in FingerPrintService properties.

The "Data Source" tab settings should reflect the ones shown in Figure 15-3. The name of the SQL Server may differ.

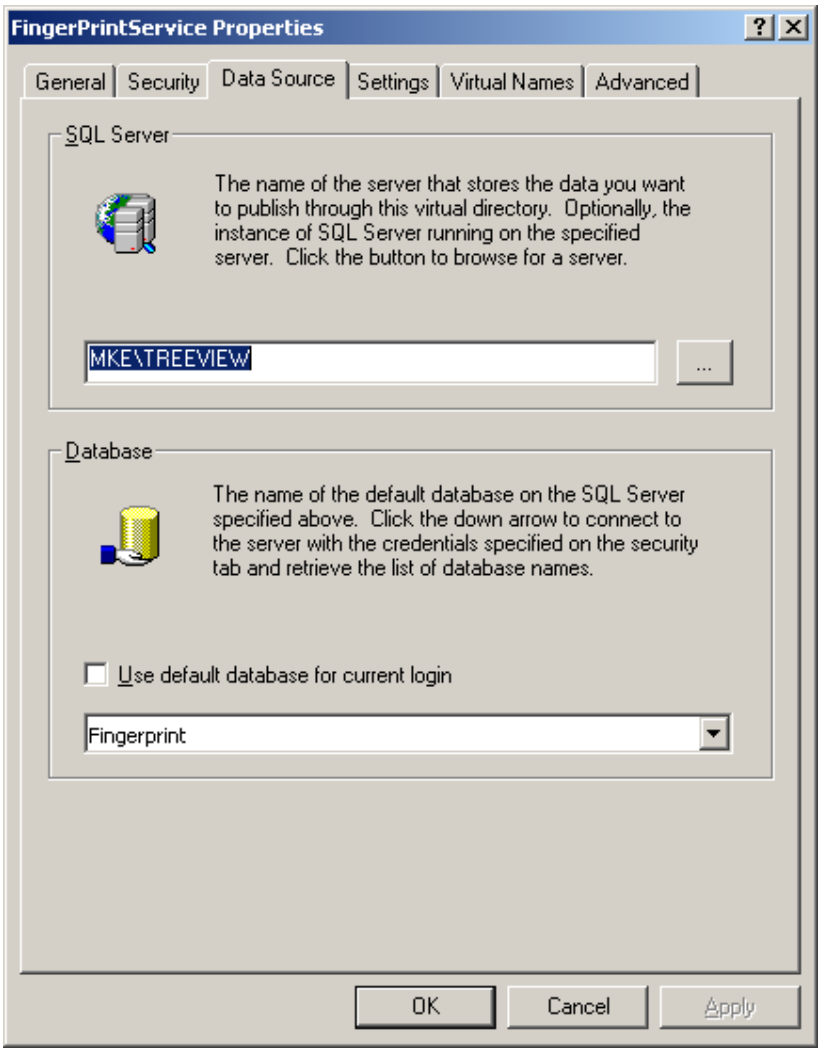

Figure 15-3 The Data Source tab in FingerPrintService properties.

The "Settings" tab settings should reflect the ones shown in Figure 15-4.

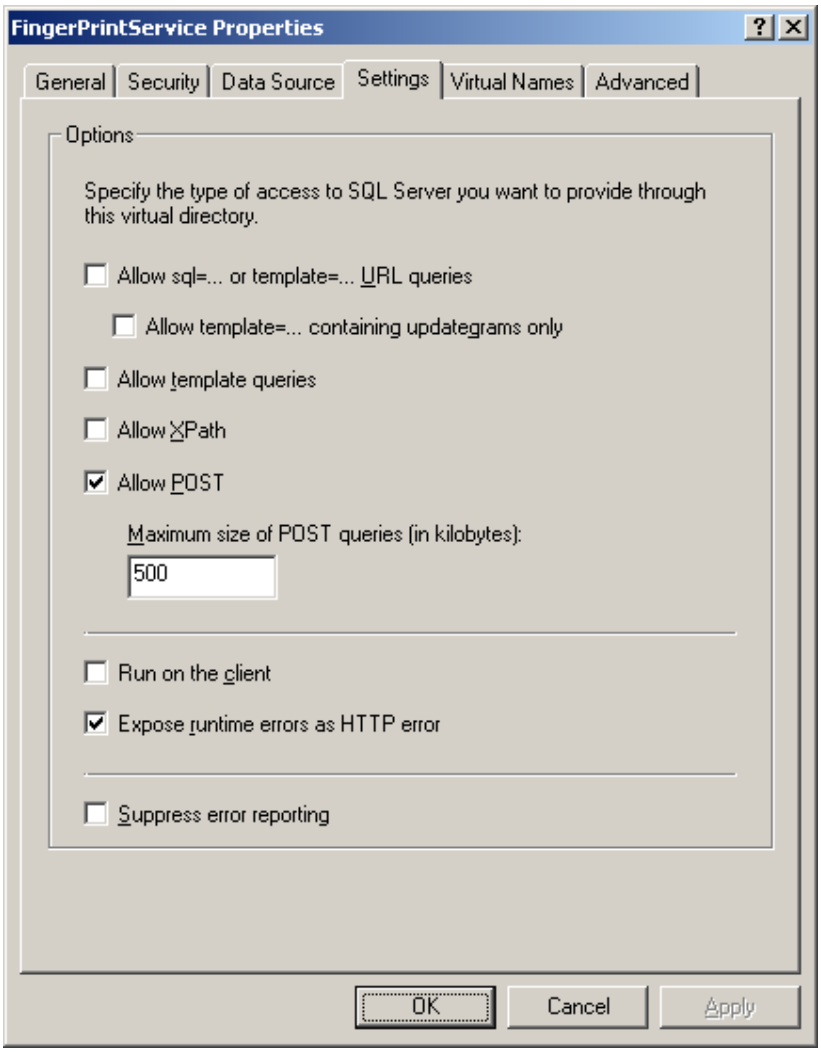

Figure 15-4 The Settings tab in FingerPrintService properties.

In the "Virtual Names" tab a new virtual name must be created. "Name" must be FingerprintController. "Type" must be "soap". "Path" can be chosen to be the same selected when configuring the IIS (Appendix III – Configuration of IIS (section 14)). The settings should reflect the ones shown in Figure 15-5.

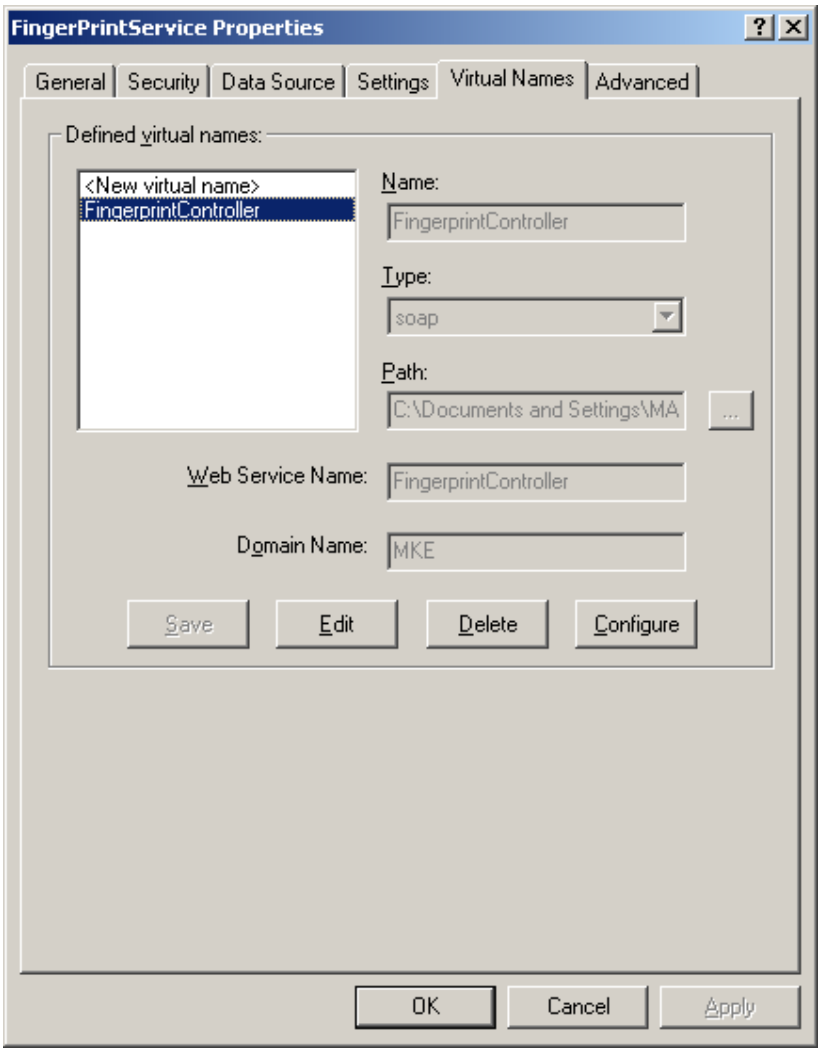

Figure 15-5 The Virtual Names tab in FingerPrintService properties.

Now press the "Configure" button in the "Virtual Names" tab. A window similar to the one shown in Figure 15-6. New methods can be mapped to the SQL Server by chosing "<New method mapping>". The methods "CheckUser" and "InsertUser" must be added. The method "SimpleLog" can be added for test purposes (enabling the authentication device to write to the log in the database).

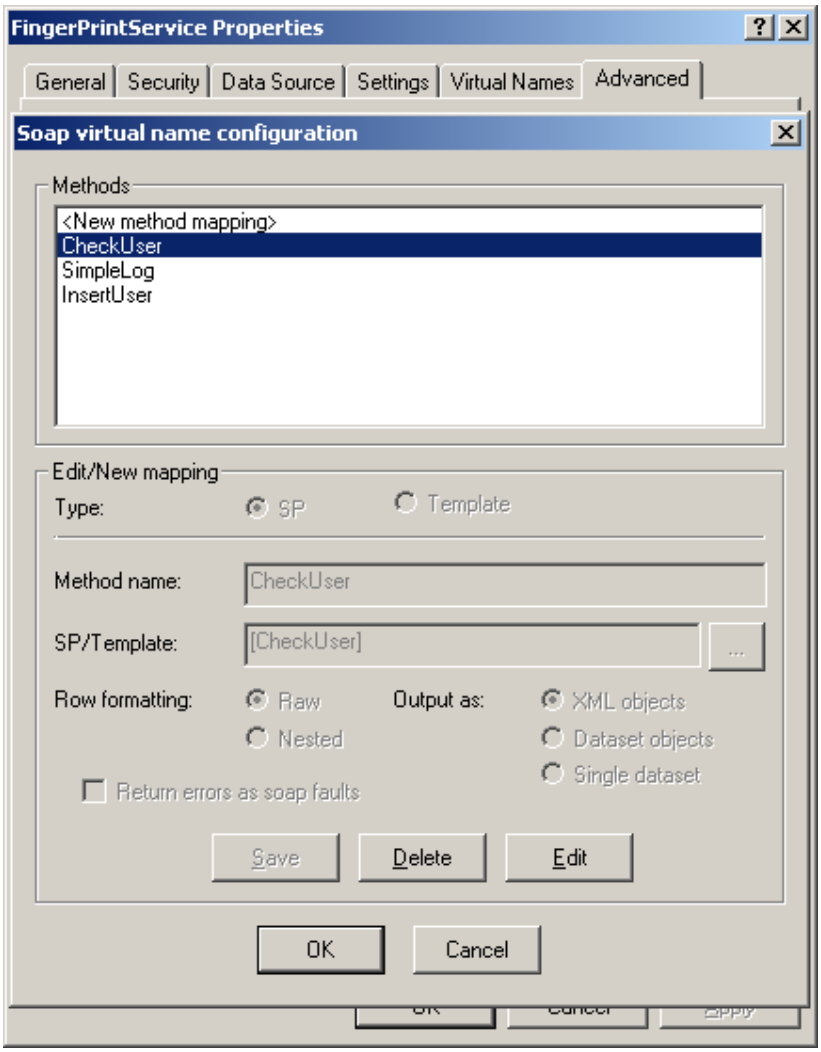

Figure 15-6 Soap virtual name configuration in FingerPrintService.

As a final step make sure that the "Advanced" tab settings reflect the ones shown in Figure 15-7. The SQLXML 3.0 is now correctly configured and the server is now ready to be used by the authentication device and the enrolment terminal.

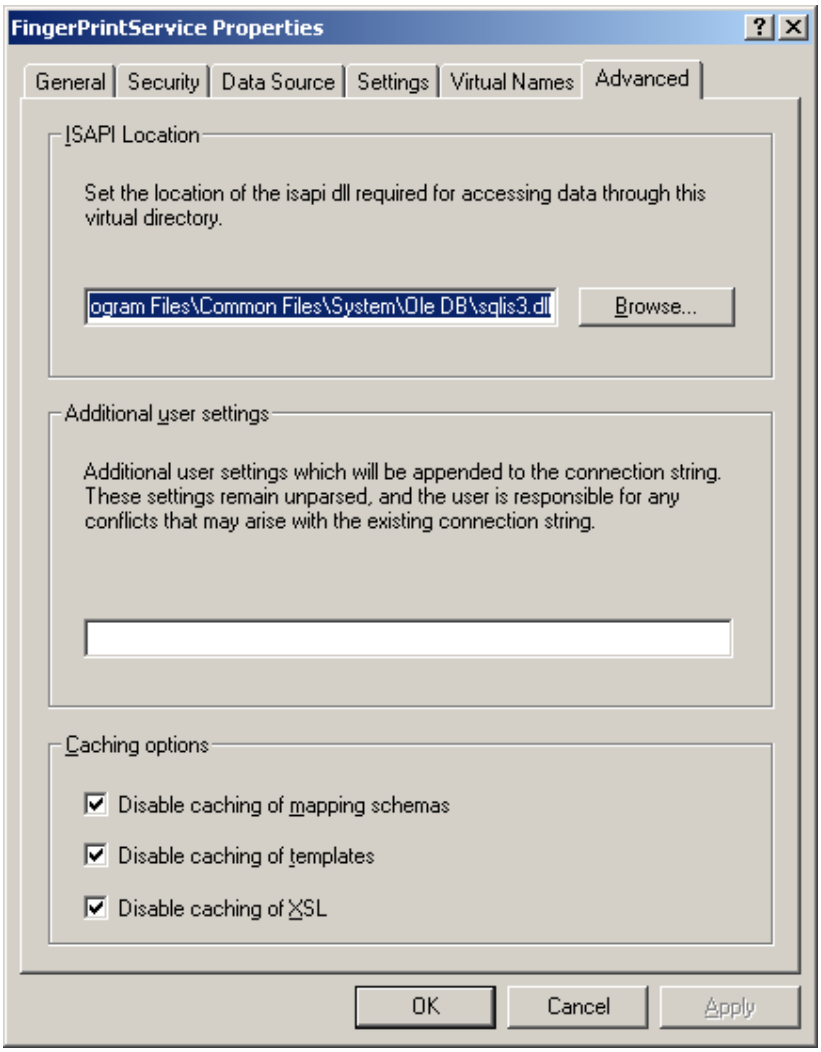

Figure 15-7 The Advanced tab settings in FingerPrintService.

16 Appendix V – SOAP Message Format

The examples shown in Code 16-1 and Code 16-2 have been captured from the network using Wireshark Network Protocol Analyzer[17].

A request for authentication sent to the server by the authentication device can be found in Code 16-1. The contents within the "image" tags are reduced in size (normally around 184 kB of data will reside here). The code shows that the parameters to the CheckUser method exposed by the IIS server (/FingerPrintService/FingerprintController) are encapsulated within the CheckUser tags. The reference tag contains the ID that the device read off of the card which the inserted. The sending of this message to the server will result in the server sending a response. This response is shown in Code 16-2.

```
POST /FingerPrintService/FingerprintController HTTP/1.1
Host: NiFinger
Content-Type: text/xml
Content-Length: 185014

<?xml version="1.0" encoding="utf-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <CheckUser xmlns="http://MKE/FingerPrintService/FingerprintController">
      <image>tWWs0</image>
      <imWidth>384</imWidth>
      <imHeight>269</imHeight>
      <imRes>500</imRes>
      <reference>101</reference>
      <context>0</context>
      <terminal>0</terminal>
    </CheckUser>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Code 16-1 Example of request from authentication device.

[17] http://www.wireshark.org

Code 8-2 shows the response from the server after it received the request shown in Code 16-1. The real result of the authentication is between the CheckUserResult tags. The result (1) shows that the user has been successfully authenticated.

A similar message is returned as result to an InsertUser request.

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
Date: Wed, 21 Mar 2007 15:23:59 GMT
Connection: close
Content-type: text/xml; charset=utf-8
Expires: -1;

<?xml version="1.0" encoding="utf-8" ?>
<SOAP-ENV:Envelope xmlns:xsd='http://www.w3.org/2001/XMLSchema'
 xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
 xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
 xmlns:sqltypes='http://schemas.microsoft.com/SQLServer/2001/12/SOAP/types'
xmlns:sqlmessage='http://schemas.microsoft.com/SQLServer/2001/12/SOAP/types/SqlMessa
ge'
xmlns:sqlresultstream='http://schemas.microsoft.com/SQLServer/2001/12/SOAP/types/Sql
ResultStream'
 xmlns:tns='http://MKE/FingerPrintService/FingerprintController'>
   <SOAP-ENV:Body>
     <tns:CheckUserResponse>
       <tns:CheckUserResult xsi:type='sqlresultstream:SqlResultStream'>
         <sqlresultstream:SqlXml xsi:type="sqltypes:SqlXml">
           <SqlXml>
             <row>
                <CheckUserResult>1</CheckUserResult>
              </row>
            </SqlXml>
          </sqlresultstream:SqlXml>
        <sqlresultstream:SqlResultCode xsi:type='sqltypes:SqlResultCode'>0
        </sqlresultstream:SqlResultCode>
      </tns:CheckUserResult>
    </tns:CheckUserResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Code 16-2 Example of response from the server.

17 Appendix VI – Users Manual (Authentication Device)

This manual describes how to use the authentication device. It is assumed that the server is configured correctly and that the device has a network connection available.

When the device is powered up and ready it will display the image shown in Figure 17-1. At this point the device is ready to read a card inserted by the user.

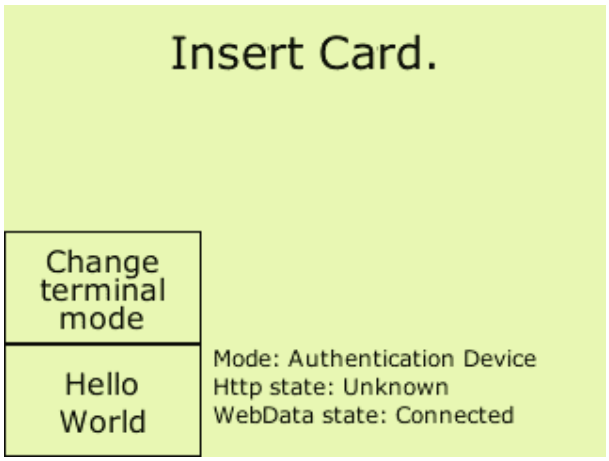

Figure 17-1 The authentication device is ready to authenticate.

If an invalid card is inserted or the device could not read the card correctly the device will display the image shown in Figure 17-2.

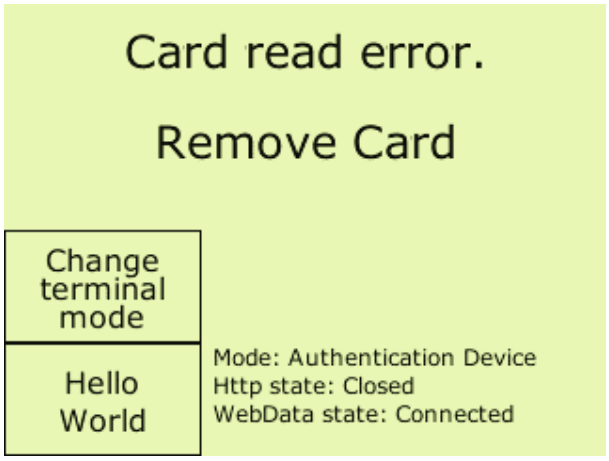

Figure 17-2 The authentication device encountered an error when reading the card.

When a valid card is inserted into the card reading slot the device will display the image shown in Figure 17-3. At this point the device is ready to read a fingerprint using the fingerprint reader connected. The fingerprint reader will display a bright red light.

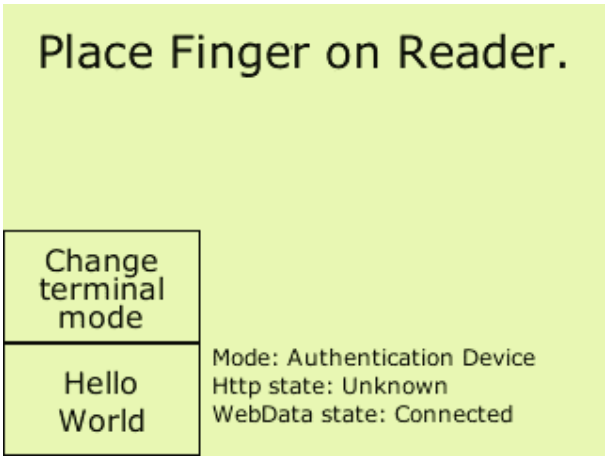

Figure 17-3 The authentication device has successfully read a card and is awaiting the placement of a finger on the fingerprint reader.

Placing a finger on the reader will cause the authentication device to scan the fingerprint. Once it has done this it will display the image shown in Figure 17-5. If the users fails to place his finger on the fingerprint reader within the time out end the three retries the device will display the image shown in Figure 17-4.

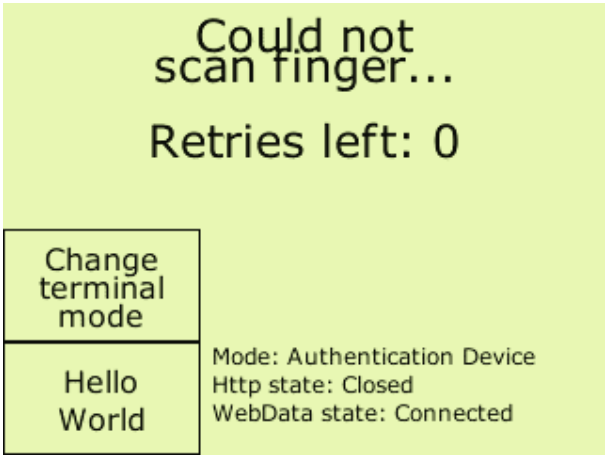

Figure 17-4 The authentication device failed to successfully scan a fingerprint.

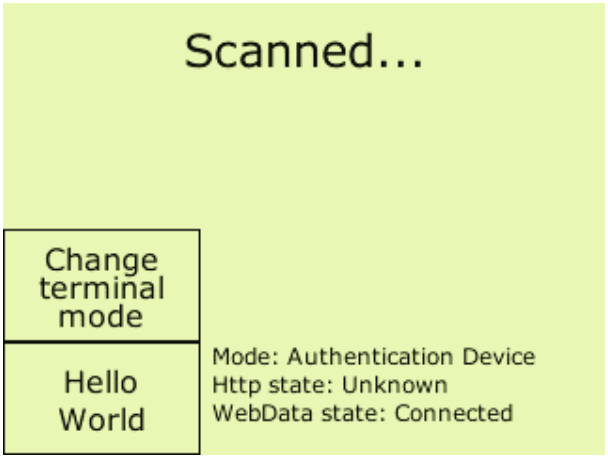

Figure 17-5 The authentication device has successfully scanned a fingerprint.

When a fingerprint has been scanned the authentication device will send the image of the fingerprint and the card ID to the server for verification and authentication of the user. While the authentication device is waiting for the result from the server it will display the image shown in Figure 17-6.

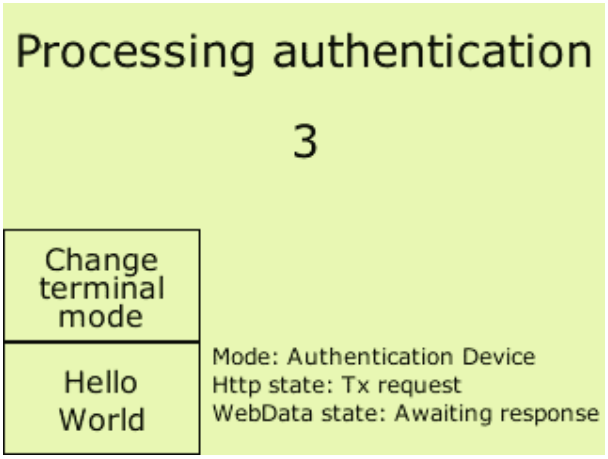

Figure 17-6 The authentication device is awaiting the result of the authentication from the server.

When the server has processed the authentication and the authentication device has received the result it will display the image shown in Figure 17-7 if the authentication was successful or the image shown in Figure 17-8. If communication with the server fails then the authentication device will display the image shown in Figure 17-9. This will indicate to the user that contact to the staff should be made.

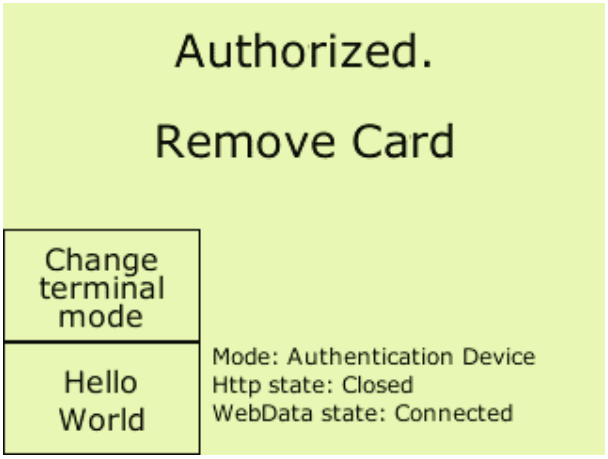

Figure 17-7 The server has successfully authenticated the user. The doors are now unlocked.

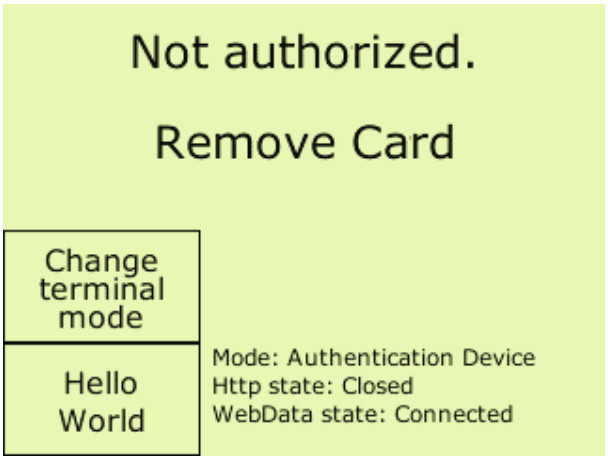

Figure 17-8 The server could not authenticate the user because the card ID and the fingerprint did not match.

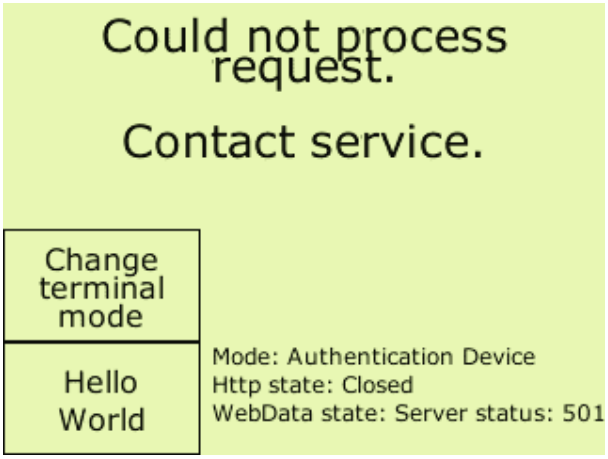

Figure 17-9 The authentication device could not reach the server.

The authentication has now been completed and the authentication device will be ready to authenticate the next user and will once again display the image shown in Figure 17-1.

18 Appendix VII – CD-ROM contents

The CD-ROM included contains this report in PDF-format, source code for the FingerprintController, SQL used in the database, and source code for the Ni (authentication device).

An overview of the contents of the directories found on the CD-ROM is shown in Table 18-1.

Directory	**Contents**
/Report	This report in PDF-format.
/SourceCode/FingerprintController	The C# source code for the FingerprintController.
/SourceCode/AuthenticationDevice	The C source for selected parts of the Ni.
/SourceCode/DatabaseSQL	SQL source files used to establish the database and all stored procedures.

Table 18-1 Overview of the contents of the directories found on the CD-ROM.