An authentication mechanism for nomadic computing users

Kenni Lund

Kongens Lyngby 2011
IMM-MSC-2011-23

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

Summary

Hospitals and nomadic computing environments in general are known to have issues with password-based authentication, as the authentication method isn't suitable for the environment. When moving between many different computers throughout a workday, it has a negative impact on the usability.

In this project I'll propose a solution to the usability issues seen in nomadic computing environments in relation to authentication. The proposed solution is a multi-factor authentication system that requires minimal interaction from the users, and supports session migration. The system is using RFID as the ownership-based authentication factor and a webcam for face recognition, as the inherence-based authentication factor. The proposed solution will be implemented as a prototype in a Microsoft Windows environment, based on the technologies available in Windows 7 clients and Windows 2008 Server.

Preface

This thesis was prepared at DTU Informatik as part of the requirements for acquiring the M.Sc. degree in engineering at the Technical University of Denmark.

I would like to thank my supervisor Associate Professor Christian D. Jensen for support and guidance throughout the project. I would also like to thank Chief Physician Niels Løkkegaard from Holbæk Sygehus, who kindly allocated time in his tight schedule, to introduce me to their IT environment and their daily workflow with IT systems and applications.

Lyngby, April 2011

Kenni Lund

Contents

Chapter 1

Introduction

Using password-based authentication in large computing environments presents several challenges. Users are often required to choose long and complex passwords in order to fulfill the password policy of the organization, which has a negative impact on the usability. This is especially true in large nomadic computing environments, where users are working on a number of different systems throughout a workday. Every time they move to a new system, they have to remember to logout of their previous system, before leaving it. If they don't, the system will be at risk of abuse. On the new system they have to spend time entering the credentials and logging on once again. When the user struggles to remember his credentials, time is wasted on failed login attempts as well as requests to the IT department on resetting the password. To avoid this, the user might write down his password, select a weaker password or simply avoid logging out of systems - all of which affects the security level of the organization. The Danish hospitals are known to have extensive problems with password and authentication handling, due to many heterogeneous systems and applications. Research shows that doctors, nurses and other employees at the Danish hospitals are expected to spend time corresponding to 688 million kroner in salaries on logging onto password-based authentication systems in the period 2009-2014.[1]

In addition to the time spend entering and reentering passwords for various systems, the users also have to wait for the systems to actually load a desktop environment after successful login. If a system runs the applications locally, the

user will have to wait for these applications to startup as well, even though the user often just wants to continue his work from were he left off on the previous system. As I have identified in this report, this inefficient workflow is a huge time-consumer in some Danish hospitals.

1.1 Problem statement

A potential solution to the password-based authentication problem, could be the use of multiple "weaker" authentication mechanisms that will require minimal interaction with the user. Such multifactor-authentication system will provide better usability, and together the authentication mechanisms will still provide a sufficient degree of security for a given application scenario. The authentication system should be practical in real-life use and it should require no to minimal interaction from the user, in order to make the system user-friendly and efficient.

To solve the problem with the inefficient workflow of the user, the authentication system would need to be integrated with a single sign-on solution and/or a centralized application platform with session migration.

It is the goal of this project to examine the possibility of solving both of these problems in a practical solution, which is suitable for large nomadic computing environments, like the ones found in the Danish hospitals. To identify the actual issues in such environment, a department on the hospital "Holbæk Sygehus" and their current IT infrastructure and workflow of their users, will be used as a case.

As a starting point, this project will identify and discuss existing authentication methods and their suitability in a multifactor authentication system. With two or more authentication methods identified, the next step is to evaluate existing single sign-on solutions and/or application platforms supporting session migration, which is suitable to integrate with the authentication system. As part of this evaluation, the actual IT infrastructure examined in the case should be taken into consideration, as it is unrealistic to make major changes to this environment, due to special requirements in terms of applications and hardware.

The result of this investigation forms as a basis for the system design and the following implementation of a prototype multi-factor authentication system.

To make sure the final solution and the prototype ends up meeting the expectations, I'll set a number of requirements now, which can be used throughout the project as guidance.

No.	Requirement	Description
01	Network based authentication	The authentication system should be network based and should store authentication data in a centralized server environment, rather than on the individual client systems
02	Minimal user interaction during log on	The authentication system should require zero or minimal interaction from the user when logging on
03	Minimal user interaction during log off	The authentication system should require zero or minimal interaction from the user when logging off
04	Integration with existing infrastructure	When the authentication system with single sign-on/session migration is designed, it should take the existing infrastructure from the case into consideration
05	Fallback to default authentication	If for some reason the authentication system is unable to authenticate the user, the system should allow fallback to the default (password-based) authentication system
06	Authentication system independent of session migration	The authentication system should be designed to work independently of the single sign-on/session migration system. This is required since it doesn't necessarily make sense to implement session migration on all types of clients - having special hardware connected directly to a dedicated client is one example where session migration can't be used on the client

Table 1.1: Project requirements

In addition to the functional requirements in table 1.1, the economical aspect of the final solution is important as well. While it shouldn't be difficult to identify a number of highly professional commercial authentication systems to form a multifactor system, the economy of such solution would be expected to be too expensive for use in a large nomadic computing environment, such as those found in the Danish hospitals. The final proposed solution should represent a realistic solution, which could be implemented at a Danish hospital.

Chapter 2

State Of The Art

This chapter will identify and discuss the current state of the areas within which this project will work.

First an overview will be given of authentication in general and of available authentication methods. Relevant authentication methods will be evaluated, in order to identify methods which could be suited for a multi-factor authentication system in a nomadic environment. As discussed in the case in the next chapter, Holbæk Sygehus has based its IT infrastructure on Microsoft Windows for both server and client systems. Hence a closer look will be taken at the current authentication technologies used in a Microsoft Windows environment. Finally, potential solutions for single sign-on and session migration will be identified and discussed.

2.1 Authentication

Authentication is a fundamental concept within computer security, which deals with the verification of a claim made by a subject, that it should be allowed to act on behalf of a given identity. In terms of user authentication, this is the process of verifying that the user in front of the computer is allowed to access the computer. Depending on the used authentication method, the identity

will either be given directly by the user - for example when the user enters his username or plugs a smartcard into the system - or indirectly, where the authentication method will try to identify the user as part of the authentication process. Biometrics is one example of this, where a user can be identified by comparing physical or behavioral traits with previously collected data about these traits.

It is common practice to divide authentication methods into three factors[3][4]:

- Knowledge - Something the user knows
- Ownership - Something the user has
- Inherence - Something the user is or does

Authentication methods within each of these factors will be identified and discussed in the following sections.

2.1.1 Knowledge-based authentication

Knowledge-based authentication, often referred to as KBA, relies on knowledge about personal information of the user.

If we use KBA as a general term, which includes all kinds of authentication methods based on knowledge, password authentication is one very common example of KBA. The password represent a secret which only the user should have knowledge of. However, the definition KBA is sometimes also used for referring to a specific group of methods, where a third relying party has to be part of the authentication process. Hastings and Dodson discusses three of such KBA models, in which the third relying party is part of the authentication process in different ways[2]. With the focus in this project on minimal interaction from the user, the approach of including a third party in the authentication process, is not likely. When references are made to KBA in this project, it will be references to the generic term - not the specific approach including a third party.

Passwords and PIN-codes (Personal Identification Number) are both typical implementations of KBA. While these are widespread, cheap to implement and commonly used, they don't necessarily represent a very secure or userfriendly authentication method. The user will need to select a password which is not easily guessable for others but at the same time easy enough to remember. Selecting a longer password from a larger key space is preferable, but again at cost of the usability.

Figure 2.1: Typical Windows 7 logon-promt requesting a password

G. Nielsen and M. Vedel[5] has proposed a method to improve this, with the use of longer and easier passphrases while accepting common typing mistakes from the user. While this can solve the issue of the user forgetting his password and at the same time give a higher level of security, it does increase the time it takes for the user to enter his password.

An alternative to the text-based authentication methods is image-based authentication methods. Instead of asking the user to remember a sequence of letters, numbers and special characters, the user is asked to remember a sequence or combination of pictures, which should be easier to remember[6][7]. While an image-based authentication method probably will improve the usability for the user, the problem with user interaction remains the same; it requires the user to spend time selecting the right images.

The time consumption in relation to authentication with methods based on KBA, seems to be the largest disadvantage with KBA in nomadic computing environments. The user will always need to spend time performing an action, in order to transfer his knowledge to the authentication system. This makes KBA inappropriate as a primary authentication factor in a nomadic computing environment with minimal or zero interaction from the user.

2.1.2 Ownership-based authentication

Ownership-based authentication relies on something which is in posession of the user. Throughout this report the abbreviation OBA will be used to indicate Ownership-based authentication. An example of OBA could be something as simple as a regular key, which can be used to unlock a door. The user who is

in posession of the key, is the user who is able to unlock and enter the through door.

Smartcards and RFID-tags are both examples of OBA methods. Smartcards are usually equipped with a chip, which requires direct contact with a reader in order to be read. The card contains the information needed to authenticate a user, eventually in combination with a PIN/password to decrypt the content of the card. RFID-tags including the so-called "Contactless smartcards"[12] can be read from a given distance, usually from a few centimeters up to several meters depending on the technology used. If an RFID-tag is used as an athentication method, the security level will be lower than regular smartcards, as the content of the tag can be read from a distance, without the owner of the RFID-tag noticing. On the other hand, this is an advantage in terms of usability, as the authentication can happen without any user interaction - if the RFID-tag or the contactless smartcard is within reach of the reader, the user doesn't have to perform any actions to confirm/initiate authentication.

Bispebjerg Hospital did in 2009 perform a pilot project, which solely used smartcards for authentication[8]. The solution was based on a Sun Desktop Solution combined with a Windows server environment, which should replace an existing Citrix Access Infrastructure[10][11]. According to DR News, the pilot project was considered a success and the solution was expected to get implemented at all IT systems at the hospital.[9]

Figure 2.2: A Sun Ray thin client similar to the ones used at Bispebjerg Hospital

The goals of the pilot project at Bispebjerg Hospital and the goals of this project are quite similar. The overall goal of both projects is to eliminate or minimize the time spend by the users when logging onto IT systems. However, it can be discussed if the solution implemented at Bispebjerg is insufficient in terms of security. Every user gets a smartcard, which they carry around while at work.

When they want to use a system they take the smartcard and plug it into the system, after which they get access to their desktop session. The smartcard is the only identification and authentication method used - so if a user puts his smartcard in his pocket on his jacket and leaves the jacket, suddently everyone who has access to the jacket, will be able to authenticate as the user. It's convenient and fast, but it results in a lower level of security than the use of regular password-authentication. When a password-authentication method is used, it will require a third party to somehow obtain the secret password of the user, but since the identity of the user is not automatically given, the third party will also need to obtain the username of the user.

To summarize, a major disadvantage with OBA is that it is easy to steal or copy the item, which is used for authentication, from the user. Hence, the security of OBA should be considered low in cases where OBA is the only authentication factor used. The largest advantage with OBA is that some of the authentication methods within OBA allows authentication with no interaction from the user, hence making it possible to improve the usability.

2.1.3 Inherence-based authentication

Inherence-based authentication is based upon something you are. "Something you are" refers to the unique biometrical physical and behavioral characteristics each human possess. If two humans are compared, their characteristics will differ: They don't look the same, they don't have the same weight, they don't share the same fingerprints, they don't walk the same way, etc. Authentication systems based on biometrics, use such characteristics to identify and authenticate a user.

In general, all biometric systems consists of a training scenario as well as a recognition scenario:

Training scenario

The training scenario is used to extract features from collected biometrical characteristics and connect these features to the authentication profile of the user. This is essentially the same as when the user selects a new password and the authentication system saves the password data for comparison in future authentication attempts.

The training scenario can be divided into the following steps:

1. *Collect data from one or more sensors.*
2. *Extract features from the collected data.*
3. *Assess the quality of the features.*
4. *Save the extracted features if the quality is acceptable.*

When collecting data from sensors, the input will often be raw unprocessed data - like a videostream from a webcam or a picture of a fingerprint from a fingerprint reader. Since the raw data usually contains unwanted information, which is unreleated to the features used for comparison, we want to extract the useful information and ignore the unwanted information. If the videostream is going to be used for face recognition, the face will need to be identified in the videostream and extract only the face and generate a set of features from it. If the quality of the features are too low, for example if the user turned his head away instead of looking at the camera for the face recognition, new data will have to be collected from the camera. Once a set of acceptable features have been extracted, the features are saved as a template for the user.

Just like in any other authentication method, the user has to be authenticated in one way or another before initiating a training session. The authentication can happen either through previously extracted features or some other authentication method, like KBA, with a valid combination of username and password. This is similar to password-based authentication, where a user is required to enter his old password, before he can select a new password.

It's worth noticing, that while it's possible to reset and select a new password in case a password gets compromised, then there's no such option when using biometrics. The user can't simple change his fingerprint or his face, in case the communication between a sensor and the authentication system is intercepted, and the authentication data is leaked. Therefore it's even more important to have focus on the environment handling the authentication data, when dealing with biometrics.

Recognition scenario

The purpose of the recognition scenario is to compare newly extracted features with previously saved features and evaluate if the features are similar.

The recognition scenario can be divided into the following steps:

1. *Collect data from one or more sensors.*
2. *Extract features from the collected data.*
3. *Assess the quality of the features.*
4. *Compare features with the stored features template.*
5. *Evaluate if the features are similar enough to accept the authentication attempt.*

While authentication based on knowledge or ownership immediately gives a binary authentication result (the password entered by the user must either be right or wrong, not something in between), then this is not the case with authentication based on biometrics, where a probabilistic approach is needed. The reason for this, is the use of sensors to collect information about the biometrical characteristics. The sensors will unlikely generate the same data during multiple measurements; for example when a user swipes his finger on a fingerprint reader, it will be very unlikely that the user will swipe his finger in exactly the same way the next time.

To turn the authentication result into a binary result, a threshold value for a match will have to be defined. If a comparison results in a value above the threshold, the authentication will succeed, while a value below the threshold will cause the authentication to fail. Hence, selecting a proper threshold value is critical for the security level of the authentication system - if the threshold is set too low, the system will incorrectly accept users who should not have been authenticated, and if the threshold is set too high, the system will deny users who should have been authenticated. The threshold value will have to be determined through intensive testing, to identify the best value in the given environment.

Biometric authentication methods

As briefly mentioned earlier, it makes sense to divide the biometric authentication methods into two classes; Authentication methods based on behavioral characteristics and authentication methods based on physiological traits. Physiological traits are biological/chemical traits that are inherent or grown to, while behavioral characteristics are mannerisms or traits that are learned or acquired.[13]

With the continuous improvements in technology and algorithms to extract biometric features, many authentication methods exists today. Physiological

traits include DNA, ear, face, fingerprint, hand (3D geometry, length of fingers, size of knuckles, vein pattern), iris, odor, palm print, retinal patterns, weight, etc. Behavioral characteristics include gait, hand writing, keystroke dynamics, voice, etc.

While some of these methods clearly doesn't fit into the objectives of this project (creating a solution for a nomadic environment, which at the same time is cost effective), several of these methods will potentially fit. Those which only requires compact and cheap hardware, are good candidates.

Figure 2.3: A low-cost Microsoft DG2-00002 Fingerprint Reader

Authentication methods based on recognition of ears, faces and hands can be accomplished with the use of 2D or 3D images[14][15][16], which can be generated by a regular camera or a stereo camera. Fingerprints are also interesting, as fingerprint readers are cheap and sometimes already integrated in laptops. This is also the case for voice recognition, which only requires a microphone. Recognition of hand writing and keystroke dynamics does also meet this goal, but since both of them requires significant interaction from the user, they don't meet the project goal of zero or minimal interaction. A weight sensor does not necessarily require any interaction from the user, if for example it is integrated in a large mat below the chair for a workstation, but it is questionable, if such a solution would be cheap/cost effective.

To summarize, the advantage of inherence-based authentication is the same as for OBA; several of the authentication methods allow authentication with no or minimal interaction from the user, which makes it possible to improve the usability. At the same time, several of these methods are relatively cheap to implement, making them feasible for use in this project.

2.2 Multi-factor authentication

As the name suggests, multi-factor authentication is about using multiple factors in order to perform authentication. A factor can be one of the three factors, which we are already familliar with: knowledge-based factors, ownership-based factors and inherence-based factors[4].

The general idea behind this, is to use distinct authentication factors, which each require significantly distinct attack vectors. In a multi-factor system, if the first factor is a knowledge-based factor - a password - and the second factor is a ownership-based factor - a smart card - then different attack vectors will be required to compromise each of the factors. If the user tries to logon to a system which has been remotely compromised with a keylogger, then the password (the knowledge-based factor) will be compromised. But since the smartcard is a physical item (a ownership-based factor), a completely different approach will be needed to compromise this factor. The same is valid in the opposite direction - even if the smartcard gets stolen, it will be useless without the password. Credit cards are an example of this, as they require both a ownership-based factor (the card) and a knowledge-based factor (the PIN).

Using two or more authentication methods within the same authentication factor, is generally not considered multi-factor authentication[4].

When designing a multi-factor authentication system, it is possible to take different approaches depending on the reliability of the authentication methods and the desired level of security. The simplest and the conceptually easiest system to understand, is the system in which each of the authentication methods is required to succeed, in order for the multi-factor authentication system to succeed. This approach is preferred when high security is a requirement.

In systems where many rather than few authentication methods exists, it might be preferred to take a probalistic approach instead. When many authentication methods are available, it might be reasonable not to expect all of the methods to be available or to succeed. If a user has several RFID-tags as ownership-based factors combined with a knowledge-based factor in the form of a username/password combination, we might want to accept the user, even though one or more RFID-tags are missing. The user will not get accepted only with the username/password combination and the user will not get accepted only with the RFID-tags. But the user will get accepted with the username/password combination together with *some* RFID-tags, as we hence assume that it most likely is the right user.

Another approach to take, is to consider authentication methods which them-

selves relies on threshold values to output a binary authentication result, and combine them together before producing a binary authentication result. As previously mentioned, most kinds of biometric authentication methods based on sensors, makes use of such threshold values. The idea behind this approach, is that by combining the methods together it should be possible to archive better authentication precision, compared to if the methods were being evaluated individually.[17]

2.3 Authentication in Microsoft Windows

Recent versions of the operating system Microsoft Windows - currently Windows Vista and Windows 7 - introduces a new interactive authentication model called Credential Providers. Credential Providers is the successor of GINA ("Graphical Identification and Authentication"), which both has the purpose of presenting an interactive authentication session as well as collecting credentials from the user and from external hardware. If someone wants to integrate their own authentication method(s) in the Windows authentication model, Credential Providers is the component to look at.

In order to understand how Credential Providers work, let's look at the Windows boot process and the components involved.

2.3.1 Windows authentication boot process

When Windows Vista or a later version of Windows boots, the LSA subsystem ("Local Security Authority") is started in the privileged session 0, in the form of LSASS ("Local Security Authority Subsystem Service"). The purpose of LSA is to authenticate and log users onto the local system. LSA also takes care of the Local Security Policy on the local system, which handles all aspects of local security[18]. WinLogon, which is responsible for the overall interactive authentication process is now started in another session with less privileges than session 0. WinLogon starts LogonUI, which is a new component in Windows Vista/Windows 7, that renders the graphical aspect of the logon and handles the Credential Providers. LogonUI now iterates each of the installed Credential Providers[19]. An overview of this structure can be seen in figure 2.4, with "CP" as an abbreviation for "Credential Provider".

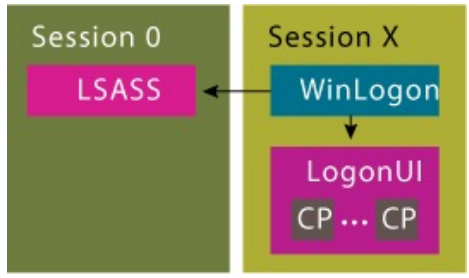

Figure 2.4: Windows Logon Architecture
Source: http://msdn.microsoft.com/en-us/magazine/cc163489.aspx

A Credential Provider is a package which implements a “provider” as well as one or more “credentials” for a logon scenario. A provider is a function which is capable of enumerating a list of users who should be able to log onto the system in a given state. A credential is the graphical representation of a user profile with text fields, buttons, and other UI-elements.

When no custom Credential Providers packages are installed, the default integrated credential provider will be used. Depending on the system and its configured users, the default credential provider will typically enumerate and display local users as illustrated in figure 2.5.

Figure 2.5: Two local users enumerated by the integrated credential provider

When a user clicks on one of the enumerated users and enters the requested credentials, LogonUI will pass the credentials to Winlogon and Winlogon will now pass on the credentials on to LSA[19]. LSA now tries to authenticate the user through the use of an “authentication package”[20]. For regular password-based authentication, the authentication package can be either one of the two integrated packages; “MSV1_0”, which supports local as well as domain-based authentication (domain-based authentication by using a passthrough approach through Netlogon) or “Kerberos SSP/AP” which only supports domain-based authentication. If one or more non-password-based authentication methods are used, a custom authentication package will have to be implemented. A custom authentication package extends the existing MSV1_0 package, since external code is not allowed to access existing credential data[22]. The available authentication packages can be seen in figure 2.6.

If the authentication succeeds, a “LSA Logon Session” will be created, which will

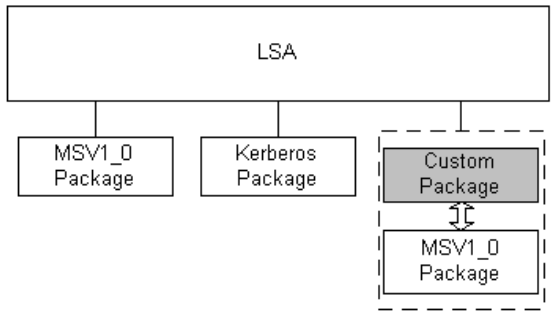

Figure 2.6: Authentication packages
Source: MSDN[22]

exist as long as the user remains authenticated[21]. The authentication package can optionally associate the credentials with the session, in order to allow subsequent authentication requests. The MSV1_0 package does this by default, by associating the username as well as the password hash to the session[23].

Once the session has been created, LoginUI will inform the user about the ongoing login process while the users' desktop environment will load.

2.3.2 Noninteractive authentication

In relation to the next chapter about single sign on and session migration, it's relevant to first look at the integrated technologies in Windows, for automatic authentication in userspace based on the initial logon authentication.

As mentioned in the last section, an authentication package is able to associate the authentication data of the user to the LSA logon session. This makes it possible for userspace applications to make use of the existing credentials to authenticate against network resources, without requesting the user to authenticate again. Through the use of SSPI ("Security Support Provider Interface") the application can request a noninteractive authentication. SSPI passes the request on to a SSP package ("Security Support Provider"), which checks the authentication with LSA and the authentication package, and finally returns the result to the application through SSPI. An illustration of this approach, can be seen in figure 2.7.[24]

Microsoft has implemented a SSP called CredSSP ("Credential Security Support Provider"), which lets an application delegate the user's credentials from the

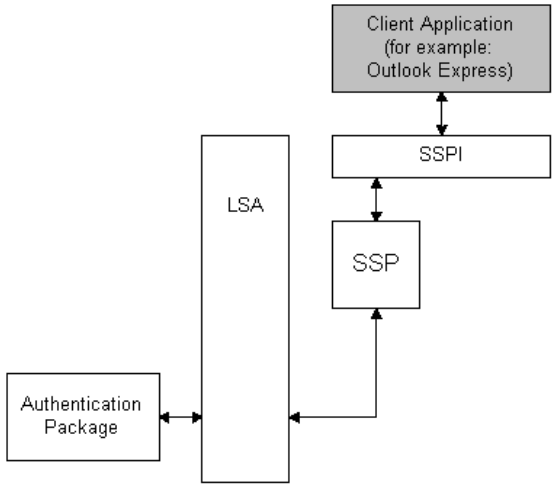

Figure 2.7: Noninteractive authentication
Source: MSDN[24]

client to a target server for remote authentication[25].

For an application to make use of CredSSP, a number of requirements have to be met:

- The authentication package used needs to associate credentials with the LSA logon session.
- The client application will have to implement CredSSP through the SSP interface, SSPI.
- The server application will have to implement CredSSP through the SSP interface, SSPI.
- The authentication data used in the authentication package will have to be compatible with the CredSSP protocol.

According to the CredSSP Protocol Specification[26], CredSSP only supports two credential types; password and smartcard. Hence it will not be possible to use CredSSP with a customized authentication package, which as previously mentioned, is required to integrate alternative authentication methods into the Windows boot process.

However, as we will discuss in chapter 4, it is possible to take another approach and avoid the customized authentication package, making it possible to utilize CredSSP with customized authentication methods.

2.4 Single sign-on and session migration

In this section I'll look closer at the two concepts "Single sign-on" and "Session migration" and why they are relevant for this project.

2.4.1 Single sign-on

The concept of single sign-on (SSO) is that multiple software solutions, applications and/or services, which require the same level of authentication, should only request authentication from the user once, instead of multiple times. This improves the usability for the user, reduce time spend on entering credentials and simplifies credential management by consolidating multiple authentication databases into one database.

Depending on the context, "single sign-on" can refer to local implementations on a single computer or in a LAN, but it can also refer to large Internet-based implementations. An example of the latter is "Windows Live ID" [1], a service by Microsoft to authenticate against many Microsoft websites and services, as well as third party websites/services which has implemented "Windows Live ID" through one of the public APIs[27]. When the terminology "single sign-on" in used throughout this report, it will solely be referring to local- or LAN-based implementations in a closed environment.

As most operating systems are configured to require an initial authentication in order to log on and get access to the applications, it makes sense to reuse this initial authentication for applications as well. On Microsoft Windows, this is possible through the use of the SSP interface, as described in chapter 2.3.2.

While single sign-on represent a major improvement in usability, it also increase the demand for strong authentication and security in general. If the authentication system gets compromised, the compromiser will gain access to multiple systems/applications instead of only a single system/application.

To build a single sign-on system, which allows multiple applications to share and automatically use the same authentication method, the applications themselves will have to implement the single sign-on system. This can prove to be an impossible task if the applications are developed by distinct software companies, who not necessarily can agree on a shared single sign-on system - or on implementing single sign-on at all.

[1] http://login.live.com/

2.4.2 Session migration

The concept of session migration, in relation to recent operating systems, is that the graphical working environment is not tied to a specific device/computer, but rather runs as a session, which can be seamlessly migrated between multiple devices/computers. In a nomadic computing environment, this can allow users to move around between computers, while keeping the same desktop environment, the same applications and the same data across all computers.

As briefly discussed in chapter 2.1.2, Bispebjerg Hospital did a pilot project in 2007 in which regular standalone computers were replaced with thin clients, as part of a Sun Desktop Solution with Citrix. This allowed users to move between the clients and once authenticated with their smartcard, they got access to their already running Windows desktop environment[9][11]. With the applications running on a central server, the session gets "migrated" instantly, since the thin client just displays the screen output from the server session as well as handle input from the keyboard or the mouse.

Citrix is another provider of commercial software within the area of desktop management and session migration on desktops. The "Citrix XenDesktop" solution[2] gives similar functionality to the solution from Sun, with applications running centrally on a server rather than on the clients. Unlike the solution from Sun, Citrix XenDesktop should support custom authentication methods, by utilizing the default authentication framework integrated in Windows[28].

Besides third party commercial providers, Microsoft also provides a similar solution with RDS ("Remote Desktop Services") in Windows Server 2008 R2[3] and TS ("Terminal Services") in Windows Server 2008 and earlier versions[4].

The Microsoft solution with TS/RDS is interesting, because Microsoft has implemented full support for CredSSP in both the server and the client software starting with the server software in "Windows Server 2008" and the client software in "Windows Vista"[29]. As explained previously in this chapter, in section 2.3.2, CredSSP lets an application delegate the user's credentials from the client to a target server for remote authentication. By utilizing this knowledge, it would be possible to implement a session migration solution with the use of recent Windows operating systems on both clients and servers.

[2] http://www.citrix.com/virtualization/desktop/xendesktop.html
[3] http://www.microsoft.com/windowsserver2008/en/us/rds-product-home.aspx
[4] http://www.microsoft.com/windowsserver2003/technologies/terminalservices/default.mspx

2.5 Chapter summary

The current state of areas relevant to this project, has during this chapter been identified and discussed.

Three kinds of authentication factors were identified, of which ownership- and inherence-based authentication factors were found to be the most appropriate for this project.

Different approaches to multifactor authentication were discussed and the current authentication framework in Microsoft Windows was explained.

Finally, the concepts of single sign-on and session migration were explained with a brief look at some of the available providers of session migration solutions.

Chapter 3

Case - Holbæk Sygehus

It's a known fact that the Danish hospitals have extensive problems with password and authentication handling, due to many heterogeneous systems and applications. As mentioned in the introduction, the Danish hospitals are expected to spend time corresponding to 688 millions kroner in salaries on logging onto password-based authentication systems in the period of 5 years[1].

In order to understand and identify the problems faced by employees at the Danish hospitals in their daily work, I've visited the Danish hospital "Holbæk Sygehus" as part of this project. This was done both to get a better understanding of the typical workflow of a doctor and a nurse, and to hear their views of the IT problems they are facing on a daily basis. It should of course be noted, that while this will give me some potential pointers to general problems, my findings will potentially only apply to the specific department at Holbæk Sygehus. This is not meant to be an exhaustive evaluation of all the authentication-related problems in the Danish hospitals, instead this should be considered an example from the real life, which can be used as a case for this project.

Niels Løkkegaard, Chief Physician at "Medicinsk afdeling, nefrologisk-endokrinologisk afsnit" at "Holbæk Sygehus" has kindly introduced me to his department and the IT infrastructure & applications used in the department. Furthermore he has given his view on the issues as he and his colleagues experiences them.

This chapter will briefly summarize and discuss these findings.

3.1 The IT-infrastructure

The department has 73 employees, of whom 8 are doctors and 65 are nurses and care assistants. The department has 10 wards, where some of these wards contains a desktop computer, on which the doctor or the nurse can look up journals or get access to other needed applications.

For those wards which doesn't contain a desktop computer, three trolleys, each with a mounted laptop computer, can be used. The laptop computers connect to the hospital network via a wireless connection, which can be reached from all of the wards. Every office contains one or more desktop computers, which is also the case for the common-rooms and the lunch-room. In total the department is estimated to have 25 clients of which 3 are laptops and 22 are desktops.

All of the computers were installed with Microsoft Windows XP Professional and were connected to a default Microsoft Active Directory domain. The computers were all configured to use password-based authentication against the domain, with no other authentication methods supported.

While the laptop computers were a couple of years old, all of the computers contained recent hardware, even though it wasn't brand new. The laptop computers, which had the lowest hardware specifications of the tested machines, was based upon a Intel Core 2 Duo 1.80GHz CPU with 2GB of RAM. Many of the desktop systems were based upon a faster Core 2 Duo 3,16GHz CPU with 4GB of RAM. Since all of the systems were running Windows XP Professional, which has very low system requirements compared to these systems[30], the hardware itself *should not* be the cause of any major performance problems experienced on the systems.

Some of the computers which were used in the wards, had external USB digital voice recorders connected to them. The recorders were used by the doctors to make verbal notes and afterwards transfer the notes to the computer. This is highly relevant for this project, as the use of external hardware can't be expected to work in a solution where the desktop session is run on a central server rather than on the local client. It will be needed to take this into consideration when designing the system.

All of the computers each contained around 10 special applications which were used by the doctors and the nurses in their daily workflow. Unlike expected,

none of these applications required any additional authentication after the initial Windows logon authentication. Løkkegaard informed that this was a fairly new change, introduced when "several of the applications were upgraded/replaced recently" - before this change, some of the applications required additional authentication in the form of username/password. Løkkegaard confirmed that these applications were still aware of his identity and demonstrated one application which clearly stated that he was the authenticated user and that he had the correct level of privileges in the application. This is also very relevant for this project, as it indicates that the department already has a functional single sign-on solution which integrates with Windows Active Directory authentication.

3.2 Identifying the issues

When Løkkegaard was presented to some of the issues faced by other Danish hospitals - like the issues identified on Bispebjerg Hospital [9][11] - he was able to recognize several of them from his own department. In particular the issues of slow system startup and users leaving computers in a logged on state, was something which they also were struggling with. When a computer would take several minutes to load the desktop environment and the required applications, nurses tend to share the session between each other, by leaving the computer logged on with the same user all day. With the introduction of the newer systems with single sign-on, this problem was minimized, as it now became impossible to switch the user profile within some of the applications, since the authenticated user was inherited from the Windows profile. While this was an positive improvement in terms of security, it had a negative impact on the usability, as the nurses now were forced to waste several minutes on waiting for the computer, each time they needed access to a given application with their own credentials. At the same time, the problem was not eliminated completely, as some of the computers wasn't used with such applications, giving no reason - besides security - for users not to share the same session on these systems.

As an illustration of the issue, Løkkegaard presented a desktop system in a lunch room, which were claimed to be the slowest system in the department, and hence almost never used by anyone. To get an idea about the severity of the boot issue, the boot time was measured with a stop watch: From a powered off state to the point at which the Windows credentials could be entered, was measured to approximately 5 minutes and 20 seconds. From the point of having entered the credentials, to the point of having a responsive desktop, was another 3 minutes and 20 seconds. This should be compared to another desktop system at Løkkegaards' office, which was measured to 30 seconds for the initial boot

phase and 15 seconds before getting access to a responsive Windows desktop.

It was not investigated why the boot process of the first computer was so slow, but the hardware specifications of the system was similar to the other systems, and by judging on the look of the hardware chassis and the number of applications and services running on the system, it seemed like it was an older system which has had many applications and services installed and uninstalled since the system originally was installed with Windows XP. Performing the same measurements on a number of the other computers, showed boot times between 15 and 40 seconds for the second phase - from login prompt to a responsive Windows desktop.

While the extraordinary slow computer doesn't represent a typical computer at the department, it does illustrate one very important point: In order to keep a Microsft Windows based workstation running efficiently in a nomadic computing environment, it has to be actively maintained with firm restrictions on the installation of third party software. Even though the hardware of all workstations were quite similar (with Core2Duo-based CPUs and at least 2GB RAM), major differences were seen in the measured boot time.

Besides the issue of slow boot times and the issue of users intentionally sharing logon sessions due to this, a third issue was also identified. When a user leaves a workstation, sometimes the user forgets, or simply doesn't have time, to log out of or lock the workstation. This can for example happen when a nurse or a doctor is called to a urgent task elsewhere, or when a doctor keeps the door to his office open all day for convenience, while entering and leaving the office - and the workstation in the office - throughout the day. When the workstation is left unattended, the workstation and its applications are vulnerabable to abuse. This is especially critical in a hospital environment, where confidential patient data and journals are managed on the workstations.

3.3 Chapter summary

In this chapter a department at the Danish hospital "Holbæk Sygehus" has been used as a case, in order to identify some of the issues related to authentication and session-handling in the Danish hospitals. This is by no means expected to cover all the authentication- and session-related issues in the Danish hospitals, instead this should be considered an example from the real life, which can be used as focus for this project.

The case has given some very valuable insight, which can be used in the design process:

- The department has based its IT-frastructure entirely on Microsoft Windows operating systems.
- Authentication on all workstations are password-based with Active Directory managing the credentials.
- Most workstation are stationary desktop computers, while a few laptop computers are used as well.
- Several workstations has external hardware connected, which likely will be incompatible with session-migration solutions.
- The applications which require authentication are already integrated with Active Directory, reusing the credentials from the logged-on user.
- A great amount of time is wasted by employees while waiting for workstations to present a usable desktop environment with applications.
- The waiting for workstations to load causes users to share desktop sessions for convenience.
- Sometimes the users forget to log out of systems due to busyness or forgetfulness.

Chapter 4

System design

In this chapter the system will be designed, based on the findings and conclusions in previous chapters. The chapter will be divided into two parts; The first part concerns a number of design considerations, and in the second part the final proposed system will be presented.

4.1 Design considerations

4.1.1 System platform

As we identified in the case, Microsoft Windows was the dominant, in fact the only, operating system on the clients at the department at Holbæk Sygehus. Furthermore all the computers which were tested, were installed with Microsoft Windows XP. There was no identication of any systems running older or newer versions of Windows.

This should make the decision of selecting a platform easy, but unfortunately it doesn't. Even though Windows XP currently seems to be the only operating system in use, it's unlikely that they'll continue to run Windows XP for more than a few years. The reason is that Microsoft stopped the mainstream support

of the OS back in 2009 and will definitely stop supporting Windows XP from the 8th of April 2014, when their "Extended support period" ends. After this date, no security updates will be issued from Microsoft, leaving systems based on Windows XP exposed to potential new vulnerabilities identified in the OS. On a related note, Microsoft stopped retail sales of Windows XP back in 2008 and sales through OEM channels in 2010[31]. The hospital will likely have a special license agreement with Microsoft, which allows them to obtain new licenses for Windows XP until 2014, even though it's not available through the ordinary sales channels.

That taken into consideration, and the fact that Microsoft has completely rewritten the security framework in Windows beginning from Windows Vista (as described in chapter 2.3), makes it unsuitable to develop an authentication system for Windows XP. If the final solution is going to make use of the CredSSP security package (as discussed in chapter 2.3.2) for remote authentication against servers, Windows XP would be a showstopper here as well.

Based on these considerations, the final proposed solution should be targeted for the latest versions of the Windows operating system - currently "Windows 7" for clients and "Windows 2008 R2" for servers.

4.1.2 Authentication methods

As discussed in the "State of the art" chapter, ownership- and inherence-based authentication factors were found to be the most appropriate for this project. The reason for this was the need for authentication factors which could authenticate users with minimal or no interaction from the user.

Due to time constraints in this project, a maximum of two authentication methods should be implemented in the final solution/prototype. As ownership- and inherence-based authentication already has been identified as the most appropriate factors, one authentication method from each of these factors should be implemented in the final solution, to fulfill the requirements of being a true multi-factor authentication system[4].

Two interesting ownership-based authentication methods identified in chapter 2.1.2, were RFID-tags and smartcards.

Smartcards are in generel considered very secure when combined with a PIN-code, as seen on credit cards. They are however, as discussed earlier, not very convenient, as they're easy to forget when moving around in a nomadic environment. If they at the same time aren't protected by a PIN-code, like on those

on Bispebjerg Hospital[9], they shouldn't be considered secure either.

On the other hand, the RFID-tags should't even be considered an authentication method, as a typical RFID-tag per design should be used for identification rather than authentication. A typical RFID-tag will share all of its content with everyone who requests it, making it easy to remotely read its data and afterwards write the data to a similar RFID-tag, essentially creating an unauthorized copy of the tag. Using a RFID-tag as a standalone authentication method would result in a very low security level. If however, the RFID-tags are used in combination with other more secure authentication methods, it can bring a big advantage to the overall system, in terms of usability. For example, if RFID-tags are integrated into something which the user can't leave behind or forget, the authentication system will now have a very good indicator of the availability of the user, making it possible for example to automatically log off the user when he leaves a workstation.

St. Olavs University Hospital in Trondheim, Norway, has implemented RFID-tags in their work garments in order to solve the logistic problem of handling the laundry[32]. With such a solution, every user would have a RFID-tag integrated into their trousers, which would be very hard to leave behind or forget, when moving around in the nomadic computing environment. An alternative solution, would be to simply integrate the RFID into the name badge of the users or into the users' ID-card, which they should carry with them most of the time.

Since usability is one of the central objects of this project, it makes sense to use RFID-tags as one of the "authentication methods" in a multifactor authentication system. Hence RFID-tags will be used as one of the authentication methods in the proposed solution.

The second authentication factor was the inherence factor - "something you are". Multiple potentially suitable biometric authentication methods were identified in chapter 2.1.3. Recognition of faces, fingerprints, voice and weight were some of the most relevant identified methods. Any of these methods would meet the goal of quick authentication with minimal interaction from the user and would be possible to integrate in a multi-factor authentication system. Which one of these methods are choosen, is not important for the prototype.

The face recognition method was chosen as the second authentication method for the prototype, due to the possibility of authentication with no interaction from the user, as well as being able to use the authentication method on laptop computers with a integrated webcam.

4.1.3 Session migration and single sign-on

As discussed in the case in chapter 3.1, the department at Holbæk Sygehus has already implemented a single sign-on solution for their applications. Hence this section will only focus on session migration.

In chapter 2.4.2 several commercial products were identified, which could provide session migration features in a Windows environment. When implementing a session migration solution, in which the session and the applications are run on a central server, rather than on the client, the need for powerful desktop hardware disappears. By replacing regular desktop systems with low-powered and energy-efficient thin Sun Ray clients, Bispebjerg Hospital claims to be saving at least 1 million danish kroner[10] in electricity expenses. This does however come with a cost, as arbitrary external hardware will likely be unsupported on a thin client with its session running on a central server. A solution like Sun Ray, does support passthrough of some devices, but the list of supported devices is quite limited[33].

During the case in chapter 3.1, we identified that the department at Holbæk Sygehus had several computers which were dependent on external hardware in the form of digital voice recorders. If these devices are not supported with a thin client solution, that will require the continued use of regular desktop computers in these situations. If the thin client solution and the regular desktop computers don't share the samt authentication system, this will introduce new usability issues for the user.

By designing and implementing a authentication system with support for session migration based on the integrated authentication framework and Remote Desktop Protocol capabilities in Windows, we should be able to design a flexible authtication process, which can be used both on thin clients running a remote session as well as regular desktop computers, running a local session.

4.1.4 Authentication storage

By introducing custom authentication methods, custom authentication data/credentials are introduced as well. For face recognition, a training face will have to be stored together with the user profile, in order to be able to recognize the face of the user. The RFID-tag will have to be registered as well. To resolve this, one of several approaches has to be taken.

As discussed in chapter 2.3.2, a custom authentication/security package (AP/SSP)

would make it possible to create custom credential types, hence allowing the use of for example face training data and RFID to authenticate a user. The implementation of a AP/SSP package is a complex task, but it should be the most powerful as it allows proper integration with Active Directory[34]. The creation of a custom AP/SSP package would be the most correct solution, in terms of the Windows authentication framework, but unfortunately it isn't ideal for the use case in this project.

As mentioned in chapter 2.3.2, the authentication data used in a AP/SSP package would have to be compatible with CredSSP, in order to allow delegation of user authentication from the client to a remote server through CredSSP. CredSSP in needed in this project to allow automatic authentication against remote sessions running on "Remote Desktop Services". The CredSSP Protocol Specification explicitly states that it needs domain, username and password as part of a TSPasswordsCreds structure on the client side[26] - in other words, it needs to know the user's password in cleartext on the client side to authenticate. The custom AP/SSP package will have no knowledge of the cleartext password, as it only gets access to the custom authentication methods - face recognition data and the data of the RFID-tag. In order to get access to the password, a workaround will have to be inplemented; the AP/SSP package will have to pull the password from Active Directory. With default domain settings in a Windows Server environment, Active Directory will store passwords encrypted. It is however possible to create a LSA-policy to enable the use of "reversible encryption" in Active Directory, which will make it possible to restore user passwords in cleartext from Active Directory[35]. Hence it should be possible to have a custom AP/SSP package to authenticate a user with custom authentication methods, pull the password in cleartext from Active Directory and pass it on to the local LSA session. When CredSSP will need the credentials, the LSA session will contain the data in cleartext and pass it on to CredSSP.

An alternative solution would be to use a customized database for credentials, outside of Active Directory, and get a customized Credential Provider to handle the customized authentication with the database as well as pull the cleartext password from there. This would greatly simplyfy the system, as a customized AP/SSP package would not be needed, since the Credential Provider already knows the cleartext password and hence can make use of the default authentication package, just like if the user had entered his password manually. Since the credentials will be stored outside of Active Directory, but Active Directory still will be performing the authentication, some sort of management tool would be needed to keep the databases in sync and ensure that changes are made in both databases.

By using a slightly different approach, the need of a management tool can be avoided, at least for a prototype; Instead of trying to keep the data of the two

databases in sync, in order to validate the correctness of the data entered by the user, no validation is performed on the external database. Eg. if a user registers himself in the authentication system as the user “martin” with the password “123456” and a given RFID-tag and face, then we assume it is corrent and save it in the external SQL database. When the user tries to authenticate with the use of face recognition and the RFID-tag, the previously entered credentials will be used for authentication against Active Directory - if the entered credentials were correct, the user is successfully authenticated, otherwise the user gets rejected, and will need to register again. The disadvantage of this approach is that users will be able to overwrite each others profiles, by entering a username of another user. When the password is changed, the user will need to register again.

Due to the greatly simplified design, an external MySQL database was chosen for storage of user credentials and other data needed by the authentication system. All connections between the SQL database and other systems should be encrypted with SSL.

4.2 The proposed solution

The proposed solution consists of three seperate applications, two which are run on the clients and one which is run on the SQL server:

- A ”RFID service” running on the SQL server. The RFID service keeps track of available RFID-tags and updates the SQL database accordingly.
- A customized Credential Provider running on the clients. This handles the registration/training process as well as authentication through the use of RFID-tags and face recognition.
- A ”client assistent” running on the clients. The ClientAssistent handles the remote & local sessions and logs off the user, when he leaves the workstation.

A brief overview of the prototype hardware setup can be seen in figure 4.1 - Server, clients and the RFID-reader are all connected together in a local TCP/IP network.

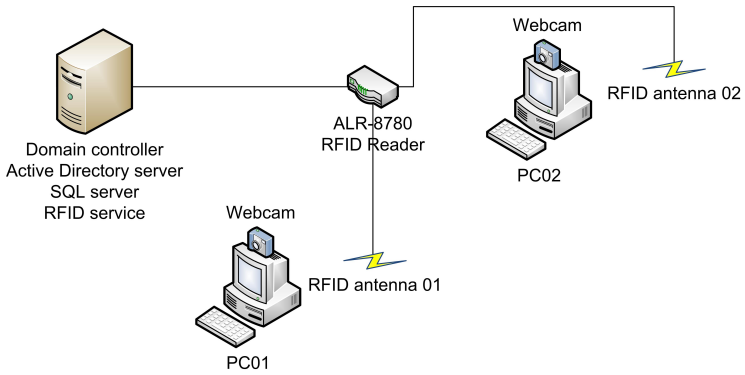

Figure 4.1: Overview of hardware setup

4.2.1 The RFID service and the RFID hardware

The purpose of the RFID service is to be an interface between one or many network-based RFID-readers and the SQL database. The RFID reader used for the prototype in this project is an Alien Technology ALR-8780[1] with 2 antennas for fixed mounting and with the possibility of using up to 4 antennas, which can be configured to be used independently of each other. The reader has an Ethernet connection as well as a serial RS232 connection. It has many advanced settings and can be configured to run in an autonomous mode, which for example allows it to trigger events on RFID-changes or send a continuously stream of updates to a specific IP-address over a TCP-connection. When not running in autonomous mode, current RFID-data can be fetched through Telnet or through a serial connection.

For this project, the RFID-reader has been configured to run in autonomous mode, sending an update-message to the implemented RFID service every few seconds. The RFID service receives the raw RFID-data through a TCP socket, interprets the data and updates the SQL database accordingly. The clients can then check the SQL database for the available RFID-tags on their associated antenna.

4.2.2 The Credential Provider

The hearth of a Credential Provider is the "Provider". The Provider is responsible for enumerating the users which should be able to logon. The Provider

[1] http://www.alientechnology.com/docs/ALR-8780_HR_DSA4F.pdf

is also responsible for re-enumeration of the available users, in the case of an external event - that could be the insertion of a smartcard, the availability of a RFID-tag or another external event.

The other part of the CredentialProvider is the "Credential", which represents an enumerated user and the graphical representation of the user in the CredentialProvider (including the picture of the profile, the username/password, the number of buttons/labels or other graphical elements which should be shown when the Credential is selected in the boot process.

In the designed system, the Provider will enumerate one user at bootup - an invalid user called "Unauthorized". This is the credential representing the implemented authentication system. Later in the process, when a user has been successfully authenticated, the unauthorized credential will get replaced with a real credential containing the username, password and domain name of the user.

Once the Provider has enumerated the invalid credential at boot, the Provider initiates the custom authentication system, which results in a pop-up window as illustrated in figure 4.2.

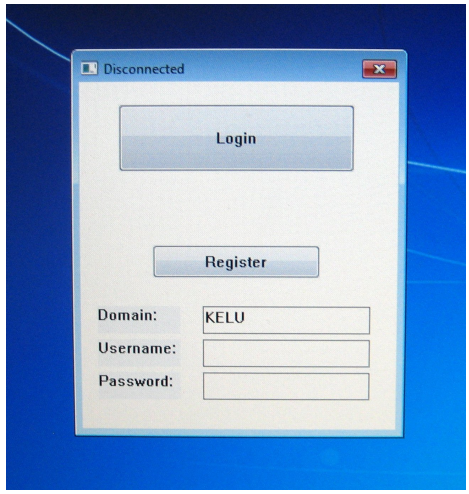

Figure 4.2: Initial pop-up window at boot

The system will now constantly be checking the availability of RFID-tags by checking the SQL-database. Hence, the workstation(s) are not physically connected to the RFID-hardware, which makes it obvious to put the RFID-readers on a seperate LAN, to minimize the risk of interception. The workstation is not completely free of hardware, as it does have a camera connected locally. For the development of this prototype, a Logitech C510 HD webcam was used.

Once a user sits down in front of the workstation with an RFID-tag, the authentication system activates the webcam, which is mounted on the screen of the workstation. A new window will now pop-up and show the video output from the webcam, as illustrated in figure 4.3. Inside the video output window, a rectangle will surround the face and track it in realtime - if a face is detected. In the lower right corner of the window, the face will be shown after postprocessing - this is the picture which is used by the system for recognition.

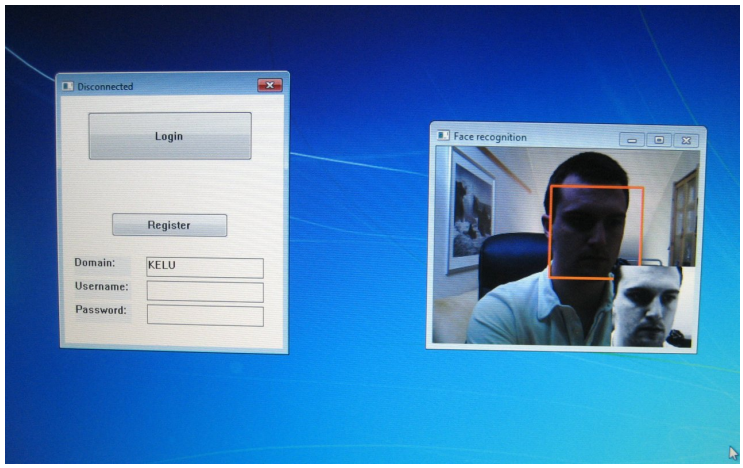

Figure 4.3: Video output pop-up with face detection

Everything related to the webcam, the videooutput, the face detection and the face recognition, has been implemented with the OpenCV library (Open Source Computer Vision)[2]. Face detection has been implemented with the use of a Haar Cascade classifier and face recognition has been implemented with the use of Eigenfaces.

At this point the system will be in a steady state, waiting for the user to perform an action.

If the user has not been using the system before, he has to register himself with his usual password-based credentials, after which the authentication system will initiate training of the different authentication methods, collect biometric data, extract and calculate a feature set and finally save everything in a user profile in the SQL database. Once registered, the authentication system should be able to authenticate the user, solely based on RFID and face recognition.

If the user wants to log on to the system, he just has to click on the "Login"

[2] http://opencv.willowgarage.com/wiki/

button, without entering anything into the text fields. If desired, it would be easy to change the implementation to skip this step entirely, as there's no reason why the user shouldn't be logged on automatically, if the user is available (identified by his RFID-tag) and if the system can authenticate him. The only reason why this isn't implemented currently, is that the current implementation makes it easier to maintain control of the prototype, avoiding unwanted login attempts while developing.

When the user clicks on the "Login" button, the authentication system will download the training data for the specific user from the server. The RFID-tag is used to identify the user. The system will now in real time - or close to real time depending on the hardware[3] - compare the frames received from the webcam with the training data. To illustrate this in the prototype, the username (fetched from the SQL database, based on the active RFID-tag) will be written in the lower left corner of the video output, together with a value representing the similarity of the training data and the current frame, with 1,00 being a perfect match. This can be seen in figure 4.4.

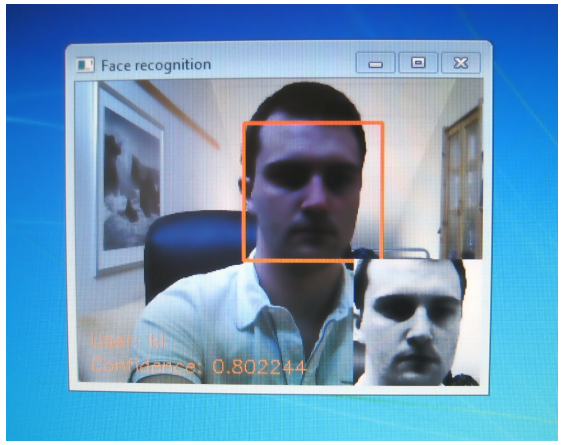

Figure 4.4: Face recognition comparing training data with current frame

If the confidence value exceeds a given threshold, the user will be authenticated. If the user isn't recognized within a given time frame, the system will abort and suggest the user to register again. The threshold depends on the environment (especially lighting conditions) and currently has to be determined through testing.

The Provider will now generate a new credential with the domain, username

[3]The authentication system has been performance-tested during development on low-performance hardware, making sure that the system will run on thin clients

and password with which the user registered, and replace the existing invalid credential with the new valid credential. The Provider will now initiate a local logon with the use of the credential and the default authentication package in Windows.

The Credential Provider doesn't perform any further actions at this point, as the ClientAssistent is taking over control once the desktop environment is loading.

4.2.3 The client assistent

Once the local login has been initiated, the client assistent starts up. The purpose of the client assistent is to keep track of the availability of the user and to initiate a Remote Desktop connection to a server, allowing the user to continue working on his existing desktop session, when moving between computers.

Like the Credential Provider, the client assistent is constantly checking the SQL database for changes in the availability of the RFID associated with the user. If the RFID-tag becomes unavailable for a given period of time, currently 5 seconds, it is expected that the user has left the workstation. When this happens, the client assistent initiates a log off on the local system, which leaves the session running on the server and brings the local workstation back to the Credential Provider.

As an alternative to the current restrictive detection of the availability of the users, another antenna could be set up near the door(s) used to enter/exit the room. As long as the user hasn't left the room, he is expected to remain authenticated and have his session running. This should help avoiding unattended log off actions.

4.3 Chapter summary

This chapter has looked into some of the considerations made before a prototype was designed. The conclusions can briefly be summarized as:

- The prototype should be targeted for recent version of Windows.
- RFID-tags and face recognition were selected as authentication methods, with RFID mostly as an "identification method" to improve usability rather than an "authentication method".

- The prototype should be based upon native Microsoft technologies included in Windows, in order to allow an authentication system which worked on thin clients running remote sessions as well as regular desktop systems running local sessions.
- A SQL database should be used for credential storage, to maintain compatibility with session migration and avoid unneeded complexity.

The final design of the prototype was afterwards presented, including the major hardware and software components.

Chapter 5

Implementation

In this chapter, a brief general overview will be given of the implementation for each of the different parts of the prototype. Furthermore, selected non-obvious code and implementations which deserves a more thorough explanation, than what is given through the in-line comments in the source code, will be described in this chapter.

5.1 The MySQL database

The MySQL database is used for storage of:

- User credentials for Windows.
- User trainingdata/registrationsdata.
- Mapping between netbios client hostnames and RFID antennas.
- Active RFID-tags and their associated antenna.

A diagram of the database can be seen in 5.1.

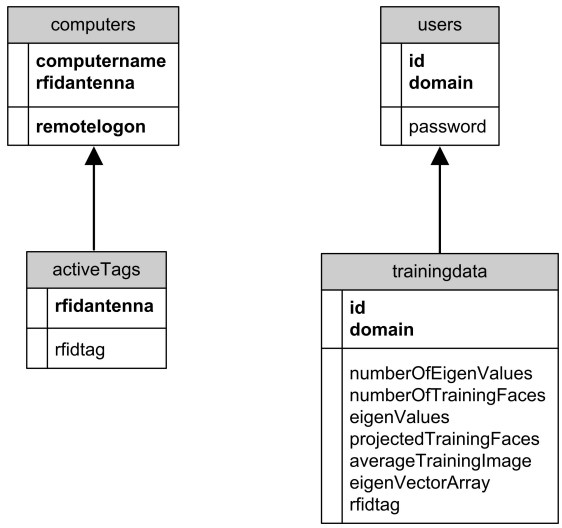

Figure 5.1: MySQL database diagram

In order to create the database structure, a SQL script has been developed. The script can be found in appendix C.

5.2 RFID service

The RFID service has been developed in C++ as a static library called "RFIDserviceClass". The static library can be used with two wrappers; "RFIDservice" or "RFIDconsole". The idea behind this, is that it can be hard to debug a Windows service when Windows is in control of it. Therefore the two wrappers are used to run the same code either directly in a console with debugging capabilities or as a Windows service, which needs to be installed in Windows. Once installed, it should hopefully look like figure 5.2

While it does work when it runs as a service, the implementation is not entirely completed, as the stop function hasn't been implemented properly in the current prototype.

Once running, either as a service or in the console, it will start a TCP socket on port 10000 and await incoming RFID-tags.

The Alien Technology ALR-8780 RFID-reader supports different output for-

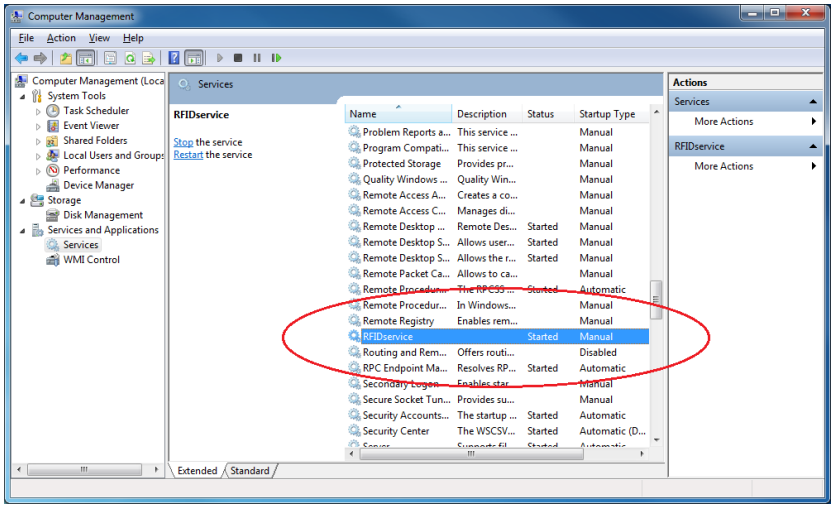

Figure 5.2: RFIDservice running as a service in Windows

mats, including XML. However with no native XML-parser in Visual C++, and with the need to manually parse the incoming data on the socket, it is preferred to keep the RFID-tag format simpel. Hence a custom RFID-tag format has been configured on the ALR-8780, through the use of the command "TagList-CustomFormat" as described in the ALR-8780 interface guide[36].

The ALR-8780 has been configured to send out the tags as cleartext data over a TCP raw socket connection, with TagListCustomFormat configured to let each line contain the antenna number, followed by a comma and the RFID-tag. This means that it will send out a string like:
`"x,yyyyyyyyyyyyyyyyyyyyyyyy[CR][LF][NUL]"`

where x is the antenna number and yyy.. is content of the 96bit RFID-tag.

The RFIDservice will receive the data, parse it and save it in the activeTags table i the SQL database.

5.3 Credential provider

The CredentialProvider is implemented in a combination of C++ and Visual C++, with the CredentialProvider being native Visual C++ and the MySQL Connector C++ and the OpenCV library being standard C++.

5.3.1 Attacking the Provider

The implementation of OpenCV in the CredentialProvider did cause some interesting behaviours/bugs, the most critical of them being the lack of video output, which unfortunately required a brutal "attack" at the Provider-code in order to work. This change also violates the way CredentialProviders are supposed to work, but it was decided that this tradeoff was okay, as it would result in working video output. Before explaining the problem, let's see how it is supposed to work.

As explained in the last chapter on system design, this prototype creates a credential called "Unauthorized", which is a graphical way of telling the user that the user has not been authenticated yet. A picture of the invalid "Unauthorized" credential, can be seen in figure 5.3.

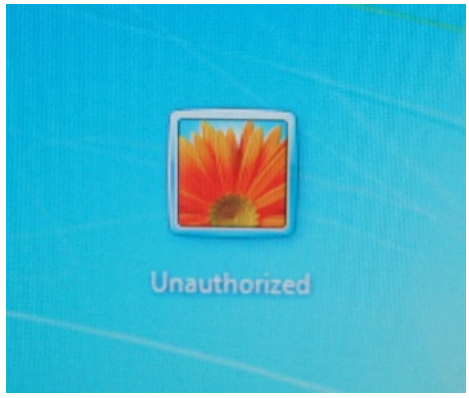

Figure 5.3: The "Unauthorized" credential before authentication

The way it should work is that once the user boots the computer, he'll see a number of credentials enumerated by the installed CredentialProviders. Hence the user will see his local users as credentials as well as the "Unauthorized" credential from figure 5.3 next to them. The user can't do anything with the "Unauthorized" credential, as it's not a real user, it's just an invalid credential to tell the user that he needs to authenticate. The idea is then to pop-up our custom login-window together with the window for video/webcam output and let the user authenticate himself. Once the user has been authenticated, the "Unauthorized" credential are removed and replaced with the real authenticated credential with the name of the authenticated user as seen in figure 5.4.

The above describes how it *should* work, but unfortunately, we're unable to display video output at this point in the Credential Provider. When we create

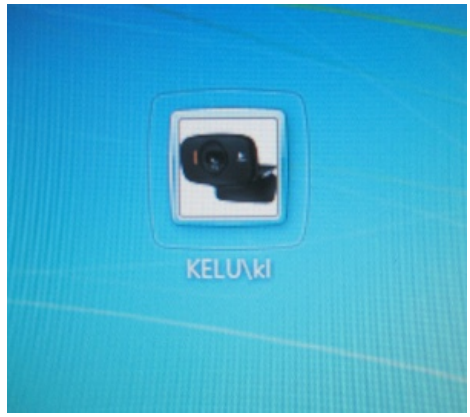

Figure 5.4: The new authenticated user-credential

the window for video output, the window remains grey with no error messages. This is apparently due to a half-hearted implementation of event-handling in OpenCV. In OpenCV a window created by the OpenCV HighGUI interface can only be updated by using a function called cvWaitKey() according to the OpenCV reference manual[37]. This seems to be the source of the problem, cvWaitKey() is in some way incompatible with the event handling in the CredentialProvider, which is not a surprise since cvWaitKey() is designed to catch key-events, when a user types on the keyboard. OpenCV has implemented another function called cvStartWindowThread(), which solves the issue, but this function is not implemented on Windows without using a Python wrapper around OpenCV[38].

The mentioned attack performed against the provider is then to interrupt the Provider before it has completed setting up the credentials, inject our own code here including the video output, and then maintain control of the CredentialProvider until we're done authenticating a user or if the user closes down the login/registration windows, to login using a local account.

If we look at the very beginning of the Credential Provider's boot process, it will look like the following[19]:

1. (The system boots)
2. (LogonUI.exe process is created)
3. (Credential provider DLLs are loaded)
4. Provider::CreateInstance

5. (User presses Ctrl+Alt+Del)
6. Provider::SetUsageScenario (CPUS_LOGON)
7. Credential::Initialize

At step 6 the Provider prepares enumeration of the initial credentials, including our "Unauthorized" credential. This is where we forcefully inject our own code, which we let cvWaitKey() be in control of the event-handling, since the CredentialProvider hasn't completed initialization of a GUI yet. This ensures that the video output works, but unfortunately it also means that if the user closes the custom authentication window, he will not be able to get it back again without rebooting or logging in and out again.

5.3.2 OpenCV and Face Recognition

As face recognition is a comprehensive study by itself, it is considered outside of the goals of this project to develop and/or implement a face recognition algorithm, as it in this project is considered as just another inherence-based authentication method. Therefore the face recognition method in this project has been implemented with the use of OpenCV, which provides a vast amount of libraries for computer vision. Many tutorials and books describes how to implement a simple face recognition method in OpenCV.

For the implementation in this project, the face recognition code has been inspired by a series of 5 articles from "Cognotics: Resources for Cognitive Robotics" called "Seeing with OpenCV"[39] about how to detect a face in a picture with OpenCV and how to recognize a face.

This code has then been expanded to support input from webcam, postprocessing of faces to improve handling of lighting, collecting training data, etc.

5.3.3 Storing training data

As the training data contains both binary OpenCV picture objects and extremely large OpenCV matrix objects, it's quite a challenge to design a database which will be able to store these datastructures in SQL. As a solution to this problem, an internal function in OpenCV has been used to serialize the binary OpenCV objects into XML-format to generate a dump of the binary data in

text format which we can then save in the SQL database as a very long text string.

This is by not means a beautiful solution, but it makes it possible to store the training data in the database, which is needed.

5.4 Client assistent

The client assistent has been developed as a C++ console application. It should have been converted into a service once the service-code in RFIDservice was done, but since service-code of RFIDservice never got there, the client application wasn't moved to a service.

The functionality of running an application as a service and as a console is the same, but when the console application is run, it will result in a pop-up console window rather than a hidden background process. This is a practical problem for the use case of the prototype, as the client assistent should be responsible for checking in the SQL database if the workstation host name should result in a local session (computer with special hardware connected) or a remote session.

While the client assistent is running as a console application, a request for a local session will result in a desktop session with the client assistent running in the middle of it.

Chapter 6

Evaluation

During this chapter, the developed prototype will be evaluated against the original goals, which were set in the beginning of the project.

Together with evaluation of the performance of the solution, this will determine if the goals have been achieved and if the developed prototype is a realistic solution to the identified problems.

6.1 Evaluating the achivement of the goals

At the beginning of the project, 6 requirements were identified for the project. These requirements can be seen in table 6.1. Each of these requirements will be evaluated and discussed below.

No.	Requirement	Description
01	Network based authentication	The authentication system should be network based and should store authentication data in a centralized server environment, rather than on the individual client systems
02	Minimal user interaction during log on	The authentication system should require zero or minimal interaction from the user when logging on
03	Minimal user interaction during log off	The authentication system should require zero or minimal interaction from the user when logging off
04	Integration with existing infrastructure	When the authentication system with single sign-on/session migration is designed, it should take the existing infrastructure from the case into consideration
05	Fallback to default authentication	If for some reason the authentication system is unable to authenticate the user, the system should allow fallback to the default (password-based) authentication system
06	Authentication system independent of session migration	The authentication system should be designed to work independently of the single sign-on/session migration system. This is required since it doesn't necessarily make sense to implement session migration on all types of clients - having special hardware connected directly to a dedicated client is one example where session migration can't be used on the client

Table 6.1: Project requirements

Requirement number 01, to store authentication data in a centralized manner, has been achived with the storage of credentials in a central SQL database. While the solution wasn't implemented using the native Windows authentication approach, the selected approach was the most reasonable choice, as the native solution was incompatible with one of the central goals of this project - to allow session migration together with minimal user interaction. Implementing a workaround-solution for the native Windows authentication approach, would make the final system overly complex.

Requirement number 02, minimal user interaction during log on, has been achived as well. The prototype in its current state allow an already registered user to log on with the click of a button. As discussed in the system design, it will be easy to remove this required step, and thereby allow users to log on with no interaction - other than looking at the camera and wait for a few seconds while the system authenticates.

Requirement number 03, minimal user interaction during log off, has mostly been achived - "mostly" because the requirement itself is completely fulfilled, but the solution seems to be way too aggressive logging off users. With the current implementation, the client assistent logs off the user, when the user's RFID-tag has been unavailable for 5 seconds. While this seems to be a very reasonable timeout when the user leaves the workstation, it's too aggressive when the user at times works near the workstation, rather than sitting in front of it. If the user needs to get something from a table a couple of meters away, he'll likely be logged off the system, before he returns. This is quite unfortunate, and makes the current implementation less practical for use in the real world. To improve this, a couple of suggestions will be given in the chapter on future improvements.

Requirement number 04, integration with existing infrastructure, has both been achived and not been achived. It has not been achived, because the prototype will not be compatible with the existing infrastructure. It has been achived, because the prototype will be compatible with the infrastructure, once the infrastructure will be upgraded. If the prototype was developed to the current infrastructure based on Windows XP clients, the prototype would be incompatible with the infrastructure once they upgrade, which as discussed, likely will happen within 3 years.

Requirement number 05, fallback to default authentication, has been fully achived. The prototype is designed to explicitly utilize existing password-based credentials. In case the user, for one reason or another, doesn't want to use the developed authentication system, he can just close the pop-up login window, after which the regular Windows password-based credentials will be shown.

Requirement number 06, authentication system should be separate from the session migration system, has been achived, but currently not usable in the prototype. The prototype has been designed to meet this goal, but as discusses earlier, the client assistent which handles the session of the user, currently runs in userspace in a window rather than as a background service, making it inappropriate to initiate local sessions. A solution will be summarized in the chapter on future work.

6.2 Performance evaluation - Authentication

The performance of the OpenCV Eigenfaces face recognition method used in this project has been evaluated, and the performance of it has been determined to be insufficient, in its current state, for use in a real authentication system. While it works quite well in a closed testing environment with the same lighting conditions, the change of lighting can drastically reduce the performance, rendering the method useless in it's current implementation.

The following tests, in which the confidence value is compared, illustrates the issue:

Two users (a woman and a man) register themselves in the authentication system. Afterwards they try to authenticate as themselves and each other, while the confidence values are logged. The following shows the ranges for the logged confidence values:

User1 as User1 = 0,84-0,89
User2 as User2 = 0,83-0,88

User1 as User2 = 0,70-0,75
User2 as User1 = 0,72-0,76

Even though the difference isn't great, the difference is large enough not to result in any wrong or missing authentications if the authentication threshold were set at 0,83. Even 0,88 would result in successful authentication for both User1 and User2 within a few seconds.

Later in the same day, in the evening, User1 tries to authenticate again with his training data from earlier:
User1 as User1 = 0,71-0,75

The confidence is now close to identical to earlier in the day, when the users

tried to exchange RFID-tags and authenticate as each other. The reason behind this was that the users had a lighting source behind them during daytime, due to a window. At night the sun had went down, and the lighting source was now a lamp on the right side of the user as well as the computer screen in front of the user.

It's without a doubt possible to optimize the performance, perhaps even with the current methods, but as it's not a goal of this project, little effort has been put into this. In the current implementation, quite a lot of the picture of the face contains data from the background - if this background data was removed, the results would likely be better. If it would be possible to improve the performance even a little, so that the lighting conditions don't make the results for one user look as bad as for another, the method could prove usable in a probalistic multi-factor system.

6.3 Performance evaluation - Session migration

The performance of the remote sessions has been measured as well. For the test, a Windows 2008 R2 Server was installed in a Vmware virtual server on a Core2Duo 2.0GHz laptop with 2GB RAM on top of a Windows 7 installation - in other words a setup which would be expected to be vastly slower than the average "real server" running Windows 2008 R2. The Windows server was configured with the following components: Domain controller, Active Directory server, SQL server, RFID service and of course Remote Desktop Services.

The clients used for the test, were both based on a 1.6GHz Intel Atom processor with 2GB RAM, which is expected to be quite similar in specifications to newly released thin clients.

Session migration between the two clients were tested and worked fine.

As expected, the performance of the two systems were identical, both systems had the following boot times measured in relation to the use of an external session / session migration:

First logon - Session not running on server

1. Fetching training data:
4 seconds

2. Load local desktop until Remote Desktop Client is triggered:
5 seconds

3. Access to usable session on remote server:
11 seconds

The above results are actually not that bad - 20 seconds in total when starting up a new session on the server. If the Login-button was removed, and face recognition was initiated automatically, as suggested earlier in the report, the fetching of training data would start already when the RFID was detected, which could bring down the logon time to 16 seconds.

Second logon - Session already running on server

1. Fetching training data:
4 seconds

2. Load local desktop until Remote Desktop Client is triggered:
5 seconds

3. Access to usable session on remote server:
2,5 seconds

A huge improvement...excluding the fetching of training data, the logon time is down to 7,5 seconds, which actually makes the system comparable to the Sun Ray Solution implemented at Bispebjerg Hospital. According to the doctor's presentation of the logon process in the DR1 News video[9], the logon time appears to be between 2 and 3 seconds. That is still 4,5-5,5 seconds faster than the prototype developed in this project, but the prototype in this project doesn't require the user to manually insert and remove a smartcard, hence making the two solutions comparable.

Chapter 7

Conclusion

In this thesis I've looked into some of the authentication and authentication-related problems seen in nomadic computing environments, like those found in the Danish hospitals. In order to clearly identify the issues and understand the IT-environment in which these issues exists, I've used a department at the Danish hospital, Holbæk Sygehus, as a case. The case revealed a number of issues related to authentication and performance, and gave some good indications of what the core problems were.

The most critical issue in the case was slow performance in the boot process of desktop systems, which by itself was a huge time consumer, but which also had a negative impact on the area of security, as impatient users shared their user-sessions and -credentials, to avoid logging off the systems.

Based on the input from the case, a solution was designed and implemented. The proposed solution was a multi-factor authentication system with optional integrated session migration capabilities. The authentication system was based upon two authentication factors; RFID-tags as the ownership-based factor and face detection as the inherence-based factor. The session migration capabilities were implemented with the use of Microsoft Remote Desktop Services, and the overall solution was designed explicitly for the latest Microsoft Windows operating systems.

The accuracy of the face recognition authentication method was evaluated and was found to be insufficient to work outside of a closed testing environment. It should be possible to improve the current implemention, but as it is based upon one of the simplest methods to perform face recognition, another approach/algorithm will likely be needed.

Using RFID-tags as part of an multi-factor authentication system, turned out to be a great idea. Not seeing RFID-tags as an authentication method, but rather seeing RFID-tags as an identification method. Using RFID-tags to identify users, made it possible to increase usability of the authentication system and hence indirectly increase the security level - for example by automatically logging off users when they leave a workstation.

The performance of the session migration were measured and found to be quite good. With minor modifications to the current implemention it would be possible to achive a logon time of down to 7,5 seconds on low power thin clients, including authentication through face recognition and RFID-tags - and without any interaction required by the user.

In summary, the designed multi-factor authentication does not provide a sufficient degree of security in its current implementation. If the face recognition method could be improved and one or two extra "no or minimal interaction" authentication methods could be added, it would likely be possible to create a probalistic multi-factor authentication system which is both user friendly, due to RFID, and secure, due to multiple authentication methods.

7.1 Future work

This section presents a list of directions for future work.

- Implement monitoring of inactivity (mouse, keyboard) in the ClientAssistent - If the user is continuously using the computer, he should not be logged off just because his RFID is out of reach for 10-20 seconds.

- Implement support for RFID-readers which can be used to unauthenticate users. If such a reader was placed at a door opening, the user passing it must either be entering or leaving the room and should therefore not be authenticated on any system. This could allow looser restrictions on the timeout used to determine when a user is leaving a workstation.

- Create a password checker - Currently the user will be requested by Windows to change his password when it expires, but his credentials will remain the same in the SQL database until he manually performs a new registration. Since Active Directory is also a LDAP-service and we already have the password in cleartext, we should be able to validate if the password is correct. If it isn't, show a pop-up to the user, requesting him to re-register.
- Give authenticated RFID-tags higher priority than non-authenticated RFID-tags: Currently the system can only handle one random RFID-tag per antenna. If another user joins the authenticated user, the authenticated user might get logged off, as the new RFID-tag becomes the active one.
- Improve the face recognition method and/or add addition authentication methods which require minimal interaction. Recognition of voice, fingerprint and weight all seems to require little interaction.
- Use contactless smartcards which is encrypted with a PIN - perhaps it's possible to receive the PIN from an authentication method not requireing interaction from the user.
- Remove "Login" button and replace it with a bit of logic which automatically logs on the user.
- Implement ClientAssistent as a service to allow local logon without a big console window.

Bibliography

[1] Version2: Password-kaos koster hospitalerne 688 millioner kroner http://www.version2.dk/artikel/14855-password-kaos-koster-hospitalerne-688-millioner-kroner

[2] Nelson E. Hastings & Donna F. Dodson, National Institute of Standards and Technology, *Quantifying Assurance of Knowledge Based Authentication*, Proceedings of the 3rd European Conference on Information Warfare and Security, pp. 109-116, 2004.

[3] Stephen F. Carlton, John W. Taylor & John L. Wyszynski, *Alternative authentication techniques*, 11th National Computer Security Conference, pp. 333-338, October 1988.

[4] The U.S. Federal Financial Institutions Examination Council (EFIEC): *Authentication in an Internet Banking Environment* http://www.ffiec.gov/pdf/authentication_guidance.pdf

[5] Glen Nielsen & Michael Vedel, Technical University of Denmark, *Improving Usability of Passphrase Authentication*, 2009.

[6] Search Security: Graphical User Authentication (GUA) http://searchsecurity.techtarget.com/definition/graphical-password

[7] Face in the Crowd - Passfaces authenticates users by having them recognize faces http://mcpmag.com/articles/2004/09/01/face-in-the-crowd.aspx

[8] C2IT: Hurtig login med SunRay Desktop løsning - Bispebjerg Hospital http://www.c2it.dk/nyheder/306-hurtig-login-med-sunrayvdi

[9] DR1 News: Sun Ray at Bispebjerg Hospital, Metropolitan Region, Denmark (TV AVISEN, 03/11/2009) http://www.youtube.com/watch?v=R497CzmKyVQ

[10] Bispebjerg Hospital tryller med login-tiden http://www.dagensmedicin.dk/nyheder/2009/04/17/bispebjerg-hospital-trylle/

[11] Citrix: Styr på it-udgifterne på Bispebjerg Hospital http://citrixcasestudy.com/da/case_detail.php?id=14

[12] Wikipedia: Contactless smartcard http://en.wikipedia.org/wiki/Contactless_smart_card

[13] M. Karnan, M. Akila, N. Krishnaraj, *Biometric personal authentication using keystroke dynamics: A review*, Applied Soft Computing Journal — 2011, Volume 11, Issue 2, pp. 1565-1573

[14] Banafshe Arbab-Zavar, Mark S. Nixon, *On guided model-based analysis for ear biometrics* Computer Vision and Image Understanding — 2011, Volume 115, Issue 4, pp. 487-502

[15] D. Voth, *Face recognition technology*, IEEE Intelligent Systems 18 (3), pp. 4-7, 2003

[16] Hand Geometry, National Science and Technology Council (NSTC) http://www.biometrics.gov/documents/handgeometry.pdf

[17] Lin Hong, Anil K. Jain, and Sharath Pankanti, *Can multibiometrics improve performance*, Technical Report MSU-CSE-99-39, Department of Computer Science, Michigan State University, East Lansing, Michigan, December 1999.

[18] MSDN library: LSA - Local Security Authority http://msdn.microsoft.com/en-us/library/ms721592(v=VS.85).aspx-#_security_local_security_authority_gly

[19] MSDN blog: Custom Login Experiences: Credential Providers in Windows Vista http://msdn.microsoft.com/en-us/magazine/cc163489.aspx

[20] MSDN library: Authentication package http://msdn.microsoft.com/en-us/library/ms721532%28v=VS.85%29.aspx#_security_authentication_package_gly

[21] MSDN library: LSA Logon Sessions http://msdn.microsoft.com/en-us/library/aa378338%28v=VS.85%29.aspx

[22] MSDN library: Creating Custom Authentication Packages http://msdn.microsoft.com/en-us/library/aa374784%28v=VS.85%29.aspx

[23] MSDN library: Authentication Packages http://msdn.microsoft.com/en-us/library/aa374733%28v=VS.85%29.aspx

[24] MSDN library: Noninteractive authentication
http://msdn.microsoft.com/en-us/library/aa378779%28v=VS.85%29.aspx

[25] MSDN library: Credential Security Support Provider
http://msdn.microsoft.com/en-us/library/bb931352%28v=vs.85%29.aspx

[26] CredSSP Protocol Specification http://download.microsoft.com/download-/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/%5BMS-CSSP%5D.pdf

[27] Windows Live ID SDK
http://msdn.microsoft.com/en-us/library/bb404787.aspx

[28] XenApp and XenDesktop Authentication
http://support.citrix.com/article/CTX126981

[29] CredSSP and SSO for Terminal Services Logon
http://technet.microsoft.com/en-us/library/cc749211%28WS.10%29.aspx

[30] System requirements for Windows XP operating systems
http://support.microsoft.com/kb/314865

[31] Windows lifecycle fact sheet
http://windows.microsoft.com/en-us/windows/products/lifecycle

[32] Case study: RFID cuts hospital laundry problems
http://www.usingrfid.com/news/read.asp?lc=x90157rx994zx

[33] Sun Ray Peripherals List http://www.sun.com/bigadmin/hcl/sunray/

[34] Dan Griffin's Blog: When is a credential provider not enough?
http://www.jwsecure.com/dan/2008/03/14/when-is-a-credential-provider-not-enough/

[35] Microsoft Technet: Store passwords using reversible encryption
http://technet.microsoft.com/en-us/library/cc784581%28WS.10%29.aspx

[36] Alien ALR-8780 Interface Guide
http://www.htz.com.tw/products/9800/manual/Reader%20Interface.pdf

[37] OpenCV reference: HighGUI
http://opencv.willowgarage.com/documentation/user_interface.html

[38] OpenCV-users discussion: Updating display in threaded applications
http://opencv-users.1802565.n2.nabble.com/Threading-amp-Display-under-Windows-td3391608.html

[39] http://www.cognotics.com/opencv/servo_2007_series/part_1/index.html

Appendix A

Source code

All source code for this project can be found on the attached CD-ROM. See also appendix B for information on the source code dependencies.

Appendix B

External source code dependencies

In order to compile and run the applications developed in this project, some specific external dependencies has to be met.

All the applications have been developed on "Microsoft Visual C++ 2008 Express" and the solution files are included with the source code.

All applications depends on..

1. MySQL Connector C++ 1.1.0 (older versions suffer from extremely bad performance when using "SQL prepared statements"). At runtime, the file mysqlcppconn.dll needs to be in the system path, for example in `C:\Windows\System32\`
   Download: http://www.mysql.com/downloads/connector/cpp/
2. The Boost C++ libraries are required by the MySQL Connector C++. Boost v1.44 has been used for this project.
   Download: http://www.boost.org/users/download/
3. MySQL SSL certificates
   At runtime, the applications will need to have access to a SSL client cer-

tificate, ca.cert.pem, in the same file as the executable. The client and server test certificates from the CD can be used for internal testing.

Additional dependencies in CredentialProvider

1. Windows SDK v7.1 is required in order to compile the Credential Provider.
2. OpenCV v2.1 is required both for compilation and at runtime.
   Download: http://opencv.willowgarage.com/wiki/
3. A OpenCV Haar Cascade file called haarcascade_frontalface_alt.xml is needed in `C:\Windows\System32\` for face detection. OpenCV includes a number of Haar Cascade files of which the mentioned file has been selected for use in this project - this file has been included with the source code, but can be replaced with another OpenCV compatible Haar Cascade file, if desired.
4. A BMP-image called tileimage.bmp with a resolution of 128x128 is needed for the credential. This image will be shown once the user has been authenticated (given auto-logon is disabled, so the credentials are actually shown). The image needs to be copied to `C:\Windows\System32\` - a sample file has been included with the source code.

Additional dependencies in ClientAssistent

1. The ClientAssistent will need access to a Remote Desktop profile called RDP_profile.rdp. This file must not contain any credentials, it should only contain the DNS-hostname (not the IP-address!) of the server and any additional configuration settings if desired. A test file has been included with the source code.

Additional dependencies in RFIDservice

1. When running as a service (not fully implemented in this prototype), the service needs to be registered in Windows. A Windows batch installation script and an uninstallation script has been included with the source code to perform the registration and the installation of executable and SSL-certificate. The files are called install.bat and uninstall.bat

Appendix C

MySQL database initialization

The following script can be used for creating the MySQL database structure:

```
DROP DATABASE IF EXISTS authsys_db;
CREATE DATABASE IF NOT EXISTS authsys_db;
ALTER DATABASE authsys_db \
   DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;

CONNECT authsys_db;

CREATE TABLE users (
        id VARCHAR(255),
        domain VARCHAR(255),
        password VARCHAR(255) NOT NULL,
        PRIMARY KEY (id, domain));

CREATE TABLE computers (
        computername VARCHAR(255),
        rfidantenna VARCHAR(255) NOT NULL,
        PRIMARY KEY (rfidantenna),
        UNIQUE (computername));
```

```
CREATE TABLE trainingdata (
        id VARCHAR(255),
        domain VARCHAR(255),
        numberOfEigenValues INTEGER NOT NULL,
        numberOfTrainingFaces INTEGER NOT NULL,
        eigenValues BLOB NOT NULL,
        projectedTrainingFaces BLOB NOT NULL,
        averageTrainingImage MEDIUMBLOB NOT NULL,
        eigenVectorArray LONGBLOB NOT NULL,
        rfidtag VARCHAR(255) NOT NULL,
        PRIMARY KEY (id, domain), UNIQUE (rfidtag));

CREATE TABLE activeTags (
        rfidantenna VARCHAR(255),
        rfidtag VARCHAR(255),
        PRIMARY KEY (rfidantenna));

INSERT INTO computers VALUES ('DESKTOP', 'antenna0');
INSERT INTO computers VALUES ('PC02', 'antenna1');

INSERT INTO activetags VALUES ('antenna0', NULL);
INSERT INTO activetags VALUES ('antenna1', NULL);

INSERT INTO users VALUES ('kl', 'KELU', 'aA123456');
```