Degree project

Differences in security between native applications and web based applications in the field of health care

Author: Andreas Dahl
Kristofer Nylander
Supervisor: Ola Flygt
Date: 2015-02-24
Course Code: 2DV00E, 15 credits
Level: Bachelor
Department of Computer Science
Abstract

Developing native applications for different platforms with different resolutions and screen sizes is both time consuming and costly. If developers were able to develop one web based application which can be used on multiple platforms, yet retain the same level of security as a native application, they would be able to reduce both development time and costs.

In this thesis we will investigate the possibilities of achieving a level of security in a web-based application that can equal that of a native application, as well as how to develop an application that uses the Mina Vårdkontakter (My Healthcare Contacts) framework.

Keywords: Mobile application security, Native and web applications, Healthcare IT security, Mina Vårdkontakter, My Healthcare Contacts
# Table of contents

1 Introduction .................................................................................................................. 1  
1.1 Background ................................................................................................................. 1  
1.2 The problem .................................................................................................................. 2  
1.3 Restrictions .................................................................................................................. 2  
1.4 Methods ....................................................................................................................... 2  
  1.4.1 Mobile devices and security .................................................................................. 3  
  1.4.2 Interviews .............................................................................................................. 3  
  1.4.3 Security models ..................................................................................................... 3  
  1.4.4 Implementation ...................................................................................................... 3  
1.5 Stakeholders ................................................................................................................ 4  
1.6 Structure of the report ................................................................................................ 4  

2 Mobile devices and security .......................................................................................... 5  
  2.1 Mobile devices .......................................................................................................... 5  
  2.2 Mobile threats and security ....................................................................................... 7  
    2.2.4 Sandboxing ......................................................................................................... 11  
  2.3 Mobile Applications .................................................................................................. 11  
    2.3.1 Native Applications ............................................................................................ 13  
    2.3.2 Web based applications ...................................................................................... 14  
    2.3.3 Hybrid Applications .......................................................................................... 14  
  2.4 Android ..................................................................................................................... 15  
    2.4.1 Android Security ............................................................................................... 15  
    2.4.2 Chrome for Android .......................................................................................... 17  
  2.5 Smart cards ................................................................................................................. 17  
  2.6 My Healthcare Contacts ............................................................................................ 17  
    2.6.1 My Healthcare Contacts SDK ......................................................................... 18  
    2.6.2 Security ............................................................................................................. 18  
    2.6.3 SITHS-Cards ...................................................................................................... 19  
    2.6.4 Net iD Access ..................................................................................................... 20  
    2.6.5 Laws and regulations ......................................................................................... 22
3 Interviews ....................................................................................................................... 23
  3.2 SecMaker AB ............................................................................................................. 23
  3.3 Landstinget Kronoberg ......................................................................................... 23
  3.4 Linnaeus University ............................................................................................... 24
  3.5 Softwerk AB .......................................................................................................... 24
  3.6 F-Secure ................................................................................................................. 25
  3.7 Summary and impressions ..................................................................................... 25

4 Security models ............................................................................................................. 27
  4.1 Security technologies ............................................................................................. 27
    4.1.1 Cryptographic protocols ................................................................................. 27
    4.1.2 Local data ...................................................................................................... 27
    4.1.3 Encryption ..................................................................................................... 27
    4.1.4 Operating systems/Platforms ...................................................................... 28
    4.1.5 Browsers ...................................................................................................... 28
    4.1.6 Smart cards .................................................................................................. 29
    4.1.7 Sandboxing .................................................................................................. 29
    4.1.8 Network Tunneling .................................................................................... 29
    4.1.9 Web Service Communication ................................................................... 29
  4.2. Proposed models .................................................................................................. 30
    4.2.1 Shared technologies between models ......................................................... 30
    4.2.2 Native app on a mobile device ..................................................................... 30
    4.2.3 Native app on a Windows 8.1 Pro tablet ................................................... 30
    4.2.4 Web app in Chrome browser on Android .................................................. 31
    4.2.5 Web app in web browser on Windows 8.1 Pro tablet ................................ 31
    4.2.6 Web or native app in application level sandboxing ................................ 31
    4.2.7 Hybrid application ...................................................................................... 31
  4.3 Our Restrictions .................................................................................................... 31

5 Implementation ............................................................................................................. 32
  5.1 My Healthcare Contacts ..................................................................................... 32
  5.2 Server and Web services ...................................................................................... 33
  5.3 Android application ............................................................................................. 34
  5.4 Web application .................................................................................................. 35
  5.5 Testing .................................................................................................................. 36
6 Result ....................................................................................................................................37
6.1 How does the security of web apps compare to that of native apps?.................................37
6.2 How does one develop an app that uses the My Healthcare Contacts framework?.........38

7 Discussion ...................................................................................................................................40
7.1 Future Work ...................................................................................................................................40

Bibliography ...................................................................................................................................41

Appendix ..............................................................................................................................................1
Appendix A. UML Diagrams for native application ...........................................................................1
Appendix B. UML Diagrams for web-application ...............................................................................4
1 Introduction

In this thesis we will investigate the possibility of achieving a level of security in a web-based application that can equal that of a native application, and how to develop an application that uses the Mina Vårdkontakter (My Healthcare Contacts) framework. In order to investigate this we will perform a literary study, conduct a number of interviews, construct a number of security models, and then implement two applications based on them, that will use the My Healthcare Contacts to set up a system for daily notes.

This chapter will give an overview of the background, the problem and our restrictions. It will also go over the methods and stakeholders for the thesis, and then the structure of the rest of the report.

1.1 Background

Sigma IT & Management Sweden AB (www.sigma.se) is a consulting firm in the IT-business with a focus on software engineering, management services and business systems.

One part of Sigma’s business is developing native apps for different platforms. But developing native apps for different platforms with different resolutions and screen sizes is both time consuming and costly. Sigma would instead like to be able to develop one web based application that can be used on multiple platforms, thus reducing development time and costs, yet retaining the same level of security as a native app.

Sigma wanted to investigate the possibilities of developing a system for handling daily notes (daganteckningar) and patient data in the healthcare field on mobile devices. The daily notes are used to document patients/residents activities as well as personnel efforts (e.g. administering medicine). The choice to do the thesis work with a connection to healthcare and patient data was proposed by Sigma in collaboration with Martin Östlund, who works at Linnaeus University as well as in the field of e-health.

To get access to patient data in health care, a national framework called Mina Vårdkontakter (hereafter referred to as My Healthcare Contacts) is being developed. My Healthcare Contacts was created and developed by Vårdguiden on behalf of the Stockholm County Council and is implemented using REST web-services to retrieve and store patient data. Because the framework deals with sensitive patient data, the demands on security is very high. In order to work with this service, there are a number of guidelines and rules in place that one is required to comply with.

We will perform a comparative study of the security for the native and web-based platforms using several methods and then use these as a basis for our study. We will perform a literary study as well as conducting a number of interviews with people within the field, and then come up with a number of security models. Based on those models we will also perform an implementation, in which we will develop two different applications, one web-based application and one native application for the Android platform, as well as two web services. The development of the applications itself is not our focus, but will be used to test the security on the different platforms, for example in the storage and transfer of data to and from the mobile devices.
The goal of the study is to deliver a recommendation on what we conclude to be the best solution for our stakeholders, in regards to the security perspective of native versus web based applications. The benchmark we will use for assessing security is based on requirements of the My Healthcare Contacts framework, as well as other security models that will be defined during the work, which we will use to assess the level of security.

1.2 The problem
The main problem that we wish to investigate is how security in web based applications compares to that of native applications, and if one can achieve the same level of security in web based applications that one can in native applications. We will evaluate the solution to this problem by comparing the security of our implementations with the level of security that is used in My Healthcare Contacts. We will also be formulating a number of security models in the thesis, by which we will also judge the level of security with.

There are several aspects one need to take into consideration when judging the level of security in this case. For one, we need to investigate the level of security on different technologies that can be used to see if they offer any weaknesses in the system.

Another problem that we will investigate is how one is able to access and develop applications for the My Healthcare Contacts framework, and if one can make a web application with the same level of security as the My Healthcare Contacts requires.

1.3 Restrictions
As we were tasked with comparing the security between native and web based applications, we were to develop an application for each of those platforms. Sigma asked us that the native app should be developed for the Android platform, and for the web based app to be developed using the ASP.NET framework.

As part of our task, Sigma also asked us to use the My Healthcare Contacts framework as the entry point for the authentication, and then use the level of security which that system uses to develop our own Daily Notes system locally. In order to contact the My Healthcare Contacts framework for authentication, a special type of smart cards called SITHS-cards are required.

As the system will deal with personal data, the level of security is required to be very high, and there are special considerations that need to be taken to ensure that it follows rules and regulations that deal with sensitive personal data. Our implementation did not deal with actual personal data, but we wanted it to follow the same guidelines that My Healthcare Contacts does. We will go over these rules and regulations in detail in the chapter dedicated to My Healthcare Contacts (Chapter 2.6).

1.4 Methods
In this paper we will conduct our study using a number of different methods, including performing a literary study, a number of interviews. With this we will also be able to perform a study of a few security models that we can utilize in our practical implementation part.
1.4.1 Mobile devices and security
A literary study will be conducted by searching for relevant books and articles using databases for scientific sources such as Google Scholar, DiVA (Digitala Vetenskapliga Arkivet), Science Citation Index Expanded etc, as well as search engines and online encyclopaedias. We will use these articles and reports to get information about the current subjects and give an overview of the current situation. We will also be looking for commonalities that we can use for creating questions for our interviews.

1.4.2 Interviews
We will perform interviews with a number of people who we feel could provide relevant information and perspective into the field which we will perform our research.

It will be conducted with people involved with the My Healthcare Contacts framework, app developers, and people involved with computer security.

While we conducted the interviews we made sure to ask the people we interviewed if we were allowed to identify them by their names in this thesis, which they all agreed to. Making sure of this was important, as it would be unethical otherwise to use the names of interviewees in a thesis without asking them first. We also made sure that there were no problems with including any information about our interviewees and their work. When conducting an interview it is important to be accurate and thorough, so we put a lot of effort to ensure that we did not misrepresent anything that was said during the interviews, and made sure that our translations and summarizations were true to their intention.

1.4.3 Security models
From the research that we got from our literary study and the interviews we conducted, we will formulate a few different security models that go over the different technologies one can use to achieve a secure application. We will use these to evaluate the security that we use in the implementation of our applications.

1.4.4 Implementation
We were tasked to create two simple applications, a native android application and a web-based application. These applications should be able to perform the same task.

As part of the requirement for the applications, the main functionality of the applications is to utilize the health care system My Healthcare Contacts. The parts of this system that we are to utilize are the authentication as well as some form of data retrieval.

The applications should then also have the added functionality of being able to create and read daily notes. These daily notes are not a part of the My Healthcare Contacts system, but a system that we will develop ourselves for a local server as per requested by Sigma, and ensure that the level of security followed that of My Healthcare Contacts.

These daily notes are used in health care to log daily events, which are not patient specific. The notes will be stored on our server, and not on the devices, and no data will be transferred to or from any public records.

This app will not be for stakeholders who are interested in base apps to develop from, but for those interested in the communication and security aspects of said apps.

As part of this chapter, we also performed tests to ensure that the level of security was up to match with the theory.
1.5 Stakeholders
Sigma is the main stakeholder for this thesis as they are the ones that tasked us with researching the problem that we are trying to solve.

Martin Östlund is also a stakeholder as he was the co-originator of the problem which was posed, in particular the My Healthcare Contacts part of it.

Other companies and developers of native apps who are considering switching to web based apps could be considered stakeholders in this work.

It may also be of use to developers who are interested in developing for the My Healthcare Contacts API.

1.6 Structure of the report
In this part we will go over how the rest of the report is structured.
In Chapter 2 we will go over the theory for the thesis work. We will first give a brief background and overview of mobile devices and mobile security, the current status and how the industry thinks it will develop. After that we will look into the background and current state of Mobile Applications and their security. We will after that go over the Android platform as well as the My Healthcare Contacts specifically.

Chapter 3 contains a write-up and overview of the interviews that we conducted during our work with various people connected to different fields that this thesis touches upon. Here we will talk about how we conducted the interviews, and we summarize the interviews.

In chapter 4 we will describe and analyze different security technologies that one can use in order to achieve a secure application in general. Then by taking this information we will devise a few different security models, which we consider to be the most plausible and realistic models for implementation.

Chapter 5 will describe our implementation, e.g. how we dealt with the My Healthcare Contacts framework, various security implementations and our different applications.

In Chapter 6 we will present the result of our work and summarize our findings. We use this chapter to present what we find to be the answer to the problem posed at the start of this thesis.

Chapter 7 will contain our discussion, in which we will discuss our work as well as our findings and future work.
2 Mobile devices and security

In this chapter we will go over the background and current status for the technologies that are relevant to our work with this thesis and the problem which we aim to answer, as well as an analysis of the current situation of these by conducting a literary study.

As this thesis uses the Android platform as a basis, we will only go over the current state of security on that platform.

2.1 Mobile devices

There are many definitions of what a mobile device is, but for the purpose of this thesis we will use the definition of mobile devices as smartphones and tablets.

A smartphone is a cell phone that includes additional software functions (such as e-mail or an Internet browser (Merriam-Webster, 2013). A tablet is a general purpose computer contained in a single panel and has a touchscreen as its primary input device (PCMag.com, 2013).

The first publicly available smartphone was the IBM Simon, released in 1994, and sold 50,000 units (Bloomberg Businessweek, 2012). You could use it for phone calls, it had a touch screen and was able to send messages and emails. It also had applications on it, such as calendar, calculator, address book and games, etc. More applications could be used by switching memory cards with applications on them.

The term smartphone had not yet been invented though, which happened in 1997 when Ericsson used it to describe its GS 88 "Penelope" concept phone.

As computing and the internet had grown extremely popular, people had increasing needs when it came to mobile computing, but because phones in this time were quite primitive by today's smartphone standards, additional devices like PDA’s (Personal Digital Assistant) were often used. In 2001, Palm, Inc. released the phone Kyocera 6035, the first smartphone in the United States. As smartphones emerged and grew more and more advanced, the need for PDA’s reduced as phones were now able to handle a lot of those features which PDA’s were used for before.

The first iPhone was released in 2007, and sold 6.1 million over its first five quarters (Apple, 2009), and has as of June 2012 shipped over 250 million units (BusinessWire, 2012). In 2007, there were 9 million smartphones users in the USA, and as of June 2013 there were over 110 million owners (comScore, 2012). Today, smartphones has grown more popular than ever, and has overtaken older feature phones in sales, with 225 million smartphones sold in the second quarter of 2013 alone (Gartner, 2013a), and is expected to continue expanding its market share, as shown in figure 2.1.
Figure 2.1 The growing market share of smartphones (Data from IDC, 2013b)

Tablet computers as a concept has been around in sci-fi novels and movies since the middle of the last century (TUAW, 2010). There has been many attempts at making tablet computers for decades, for example the Apple Newton, Microsoft Tablet PC and Nokia 770. But the first tablet to get mainstream popularity was the iPad, released on April 3, 2010, which Apple sold 15 million units of in its first year, compared to 3 million tablet sales combined for other tablet manufacturers (Business Insider, 2011). Since then, tablets has grown more popular, and it is estimated that 45.1 million tablets will be shipped in 2013, a growth of 59.6% from the year prior (IDC, 2013a).

The sales of personal computers have slowed down while the sales of smartphones and tablets only rises, and is expected to continue rising, as can be seen in figure 2.2. But smartphones have yet to reach the same level of penetration in developing countries as it has in the west. “Developing markets will drive smart phone market growth in 2013” says analyst firm Canalys (Canalys, 2013). Tablets are also considered to continue to rise in sales in the future, with total sales growing from 116 million tablets sold in 2012 and an estimated 197 million in 2013, to a projected amount in 2017 totaling 467 million sales (Gartner, 2013b).
Figure 2.2 Growth of the past few years, as well as the projected growth, of smartphone sales globally. (Data by BI Intelligence, 2012)

2.2 Mobile threats and security

There are many different things that threaten the security of smartphones and tablets and as these mobile devices have gained a great deal of popularity in recent years, they have become more and more of a target. This combined with the usage of them as devices for personal as well as business uses make the field of mobile security a very important one. And as mobile devices have gotten more and more popular, they will become more and more of a threat for attacks (Muttik, 2011), and preventing them is something that will need to be not only taken into consideration, but planned for.

2.2.1 Malware

Malwares are pieces of software that are designed to perform actions that pose a risk to the target’s system and data (F-Secure, 2013a). There are many different types of malware, but the major categories are backdoors, trojans, worms, spyware, trackware and adware.

- Backdoors are malware that allow for unauthorized access to a device.
- Trojans are software that are setup to gain access to system resources with which it can then perform malicious actions.
- Worms are pieces of harmful software that spreads by replicating itself.
- Spyware are software that sits in the background and collects data and information about the system and its user, and stores it locally or sends it to a remote destination.
- Trackwares are software which is essentially used to identify and track persons or devices to a third party through data-collection.
Adware is a malware which is used in to display advertisements to the user, often also gathering information about the user to target a specific types of advertisements to the user.

F-Secure uses the terms families and variants to keep track of different types of malware. These variants of malware are malware which have variations on them from each other, but not a large enough variation to be considered its own type of malware. The families are a collection of each of these variations (F-Secure, 2013b). These types of families and variations of malware for mobile devices has grown significantly in recent years. At the start of 2012, F-Secure categorized different 61 families and variations of families, and as of this year they have identified 149, as seen in figure 2.3.

![Figure 2.3 Rise in number of malware on Android (F-Secure, 2013a)](image)

It is not only the variations of malware for mobile devices that has grown a lot lately, but the sheer number of malware attacks as well. In 2011, malware for mobile devices grew 155%, and over 2012 that amount grew to 614% from 2011 (Juniper, 2013).

As figure 2.4 shows, the most common type of attack on these mobile devices were SMS trojans at 48% of all malwares. SMS Trojans is an attack in which the target mobile device is controlled and instructed by the malware to send out short SMS messages to premium text message services, which runs up costs for user. The second highest were fake installers at 29%. Fake installers are applications which impersonates a proper application, but once installed has access to resources and data which it can use for malicious actions. At 19% trojan spy malware was third. This is an attack in which the malware intercepts, stores and sends data and information about the user. The remaining 4% is made up out of various other types of attacks.
2.2.2 **Man-in-the-middle, eavesdropping and mobile threats**

There are also a number of other threats for mobile devices from the outside. As they are, in their nature, mobile and meant to be able to be carried with you and used wherever you are, the areas of which one is vulnerable for attacks are increased.

If the mobile device has wireless communication capabilities, it is mostly either through WiFi or a cellular network. Since wireless networks are inherently more difficult to secure than wired networks (Sanders, 2010), due to the fact that you are broadcasting the data using wireless technology that can be intercepted without having to physically tap into the network, there are a number of threats that can mobile devices can be exposed to when transmitting data wirelessly.

Man-in-the-middle and eavesdropping are attacks in which the attacker essentially positions himself or his tools between two parties which are trying to communicate. The attacker can then intercept messages in the case of eavesdropping, as well as perform specific types of this attack by impersonating one of the parties, modifying packages, creating own packages in the case of man-in-the-middle attacks (Wikipedia, n.d.a). For example, if one would utilize a man-in-the-middle attack within an online banking situation, the attacker could perform such actions as issuing fake bills and invoices or changing which accounts money is withdrawn from or deposited to.

One way to defend against this on a network level is to use strong encryption on the traffic. A solution that utilizes a secure network tunnel, such as virtual private network (VPN), is also a very good option. Secure network tunneling works by creating a virtual
point-to-point network connection between a device and a remote local network. It encrypts the data, and encapsulates it with a new header, which contains the information necessary that allows it to travel across a shared or public network and reach its endpoint securely, without the risk of being decrypted if intercepted. (Microsoft TechNet, 2001)

Another way for users to protect against this type of attack is to not connect and transmit personal or sensitive data over publicly available networks, or a network which one reasonably could consider to not be completely secure (Kaspersky, 2013).

2.2.3. Loss of device

Another potential threat for mobile devices and its data, is having it end up in someone else's possession. It can be from having the device being stolen, or just lost and then retrieved by someone else. This potentially gives full access to the mobile device and its data, thus making all data and information vulnerable.

One should also note the fact that businesses are using mobile devices a great deal these days (BusinessWire, 2013), if someone would get a hold of one of these devices, there is a chance that a great deal of potentially damaging data and information would be jeopardized, such as sensitive inside information, banking data, customer information and network access, as a few examples.

Preventing data loss like this is a very difficult area, and there are a lot of things one need to take into consideration, but there are ways which one can be more cautious and vigilant in preventing theft or losing the device, but if it has already happened, there is not much one can do unless one has taken precautions in advance (Cisco, 2008). For example, simple things like using a passcode or different authentication method before giving access to the device is advised for everyone. And if one is dealing with highly sensitive data on the mobile device, such as when it is used in business, there are additional steps one can take in order to help prevent data loss. There is encryption that one can utilize on the sensitive data, or even the entire mobile device. One option that businesses are sometimes implementing are Mobile Device Management software. These are essentially software which allows someone to get access control and monitoring abilities on one or more devices, from a remote location. This allows, for example, a security administrator in a company, in case of a lost device to access that device remotely and remove any sensitive data that may be on it, or even completely lock it down and make it unusable for anyone that may have access to it (techopedia, 2013). There are also ways one can enhance the authentication method, for example instead of only using a single pass-key, utilize a multi-factor authentication method. Multi-factor authentication is a way of handling authentication using two or more types of authentication. There are three major categories of authentication, which are something you know (a password for example), something you have (a smartcard for example), and something you are (a fingerprint for example), and combining these will allow you to achieve a much stronger level of authentication (Aladdin Knowledge Systems, 2008).
2.2.4 Sandboxing

Sandboxing, or containerization, is a security mechanism for separating running programs. It isolates code (and the impact that code can have in a runtime environment) and data, preventing other programs from touching it. From a technical standpoint sandboxing technologies do this by using an abstraction layer.

The source (Cocking, 2012) uses the analogy of a hotel. Each hotel guest has their own room and set of amenities, rather than sharing with every guest staying there.

The source puts sandboxing technologies into three categories.

1. “Bare metal” sandboxing
   This type of sandboxing uses a hypervisor on the firmware level. Above this hypervisor there are one or more virtualized operating systems that are completely sandboxed from each other. Usually considered the most secure type, but requires significant effort from device manufacturers, and has therefore not been used extensively in consumer devices.

2. “VMWare style” sandboxing
   This type of sandboxing works like VMWare running on top of Windows, i.e. a virtualization application running on top of an existing operation system. It requires less effort from the device manufacturers, usually only requiring some tweaks of the operating system load out and no modification of the firmware, but has the downside of using more of the host’s resources. This type of sandboxing has also not achieved much traction.

3. Application level sandboxing
   This type of sandboxing uses an ordinary application, which uses native OS environment and API’s, to provide a sandboxed environment for data or applications and introduces additional security mechanisms like encryption and advanced policy control. It uses three different types of wrappers to achieve this: Application wrappers, content wrappers and work space wrappers.
   Application wrappers provide an additional security layer by giving developers an SDK that handles a lot of the security for them.
   Content wrappers provide secure containers for documents and other data.
   Work space wrappers provide a full work space including applications for email, calendar and browsing.

Because no standards for the first and second types of sandboxing exist, the third type is currently very important for the mobile industry. This may change over time, but it may take a long time for new standards to appear.

2.3 Mobile Applications

Mobile applications, or apps for short, are software programs designed to be run on mobile devices, such as smartphones and tablets.

Simon, the first smartphone, had applications, and more could be added by plugging memory cards into the phone.

After the release of the iPhone in 2007 the popularity of smartphones greatly increased and with it, the popularity of apps has grown as well (Frommer, 2007). Not just for the iOS platform, but for other platforms as well, for example Android as shown in figure 2.5.
Figure 2.5 Shows the increase of popularity of applications on mobile devices. (Xyologic, 2011)

In July 2008 Apple released the App Store, which offers users a place to download new apps (Apple, 2008). As of June 2013 it offers over 900,000 applications, and has as of May 2013 reached over 50 billion downloaded applications (Lowensohn, 2013).

The Android version of the App Store is Google Play (formerly the Android Market). It has as of July 2013 over 1,000,000 applications, which has been downloaded over 50 billion times (Google, 2013a).

Microsoft’s Windows Phones has the Windows Phone Store (previously Windows Phone Marketplace) that distributes applications. It has as of February 2013 over 130,000 apps available for download, and 1,000,000 application downloads (Brix, 2013).

There are now also third party stores for distribution of applications, like the Amazon Appstore, which primarily serves for their own Amazon Kindle ecosystem, but is also available for the Android platform and is thus an alternative to Google’s own Google Play store (Amazon, 2013).

There are three major types of mobile applications: Native, web based and hybrid.
2.3.1 Native Applications
Native applications are apps that are built for a specific platform, e.g. Android, iOS, or Windows Phone, and cannot be used on other platforms (there are exceptions to this rule that we will not cover in this thesis). Consequently apps have to be ported which is time and effort consuming (Jalmlöv, 2012).
If one wishes to have an app for the three major mobile operating systems, i.e. iOS, Android and Windows Phone, one has to develop almost everything separately for those three platforms. iOS uses mostly Objective-C as its programming language (Apple, 2012), Android uses Java (Android, 2013), and Windows Phone uses C++, C# or Visual Basic (Microsoft, 2013). Another issue with developing native applications is the fragmentation of devices, sizes and manufacturers. This requires at least three different skill sets, and the time for development, testing and maintenance on all three.

Because the native apps are built for a specific platform, they can use all the security functions of that platform, e.g. an Android app can use the security functions of Android, which will be covered in chapter 2.4.1.

2.3.2 Web based applications
Web based applications, or web apps, are web sites built using standard web technologies such as HTML and CSS (Jalmlöv, 2012).

They are designed to be accessed over a network, e.g. the internet or a local network, using the mobile device’s browser. Because of this, a web app can be accessed from almost any mobile device with a web browser that has access to the network which the app resides on, be it the internet or a local network.

The use of standard web technologies means that the many web developers can make web apps. This also means that development time and effort which would otherwise had been spent on ensuring that the app would be able to be run on different platforms, devices and screen sizes are now offloaded to only one platform and device, and with the use of web design, all screen sizes as well. This makes this approach less time and effort consuming.

As the web app is accessed from a web browser the web app can then only be as secure as the web browser. More information can be found at 4.1.5.

2.3.3 Hybrid Applications
Hybrid apps are a combination of native and web app concepts. Like web apps, most of the code is written using web programming languages and technologies (e.g. ASP.NET, PHP, HTML5). This code is contained within a wrapper (e.g. WebView on Android), which means they can be installed, started and closed similarly to a native app. Hybrid apps can also contain native code in order to access native platform functions (e.g. GPS, accelerometer, camera).

Because the majority of the code written is used unchanged on different platforms, this approach is less time and effort consuming than native app development, but still more than web app development. There are hybrid app development frameworks that further lessens the time and effort of hybrid app development.

As the hybrid application utilizes sandboxing, and the web-view is also a part of the sandboxing, it will not be possible to access information from this web view (for example, its cache or cookies) from any other apps or web views (Android Developers, 2013).
2.4 Android

Android is an application execution environment for mobile devices that includes an operating system, application framework and core applications. The applications are written in Java using APIs provided in the Android Software Development Kit (SDK or ADK for short). Android is built around the Linux kernel and uses Linux for various things, e.g. device drivers, memory management, process management and networking (Shabtai et al, 2013).

2.4.1 Android Security

Android is a multi-process system, where each application and parts of system runs in its own process (Shabtai et al, 2013). It uses its Linux roots for enforcing the security between the applications and the system at the process level. It uses a permission system for controlling the different operations an application can perform. At installation an app has to ask the user for permission to do various operations, e.g. Internet access, dialing, sending SMS.

When an Android package file (APK) is installed on an Android device it gets its own unique Linux user ID. This means that two applications cannot run in the same process. This practically creates a sandbox around the applications, which means that an application cannot access other applications (without using Androids systems for this purpose) and vice versa, other applications cannot access it, as Google (2013b) put it: “A central design point of the Android security architecture is that no application, by default, has permission to perform any operations that would adversely impact other applications, the operating system, or the user.”

Because an application always runs in its own process, it can only perform operations that it has permissions for, no matter how the application was started. For example, even if
a contacts application starts a sms application, the contacts application cannot do the things that the sms application has permissions for and vice versa.

Files which are created and owned by applications can only be accessed by them, as they are part of the previously discussed Linux user that applications are, unless one specifically states during the development of that application that the information should be available to be read by other applications, or by saving it in a public location like the SD card, without a private flag.

All apps on Google Play require that they are signed with a certificate to ensure that its content has not been altered by a third party.

![Figure 2.9 An example of the permissions a user has to grant to install an app.](image)

All apps uploaded to Google Play Store, Google’s application store for Android, by developers undergo an anti-virus scan by an automated system called Google Bouncer that aims to reduce the amount of harmful apps to be available through Google Play. It was reported that it reduced the number of “potentially” malicious downloads between the first and second half of the first year it was implemented (Kaplan, 2012).

As of Android 4.2, the Android operating system introduced a built-in malware scanner that can help identify and neutralize malicious code from the mobile devices. This feature
was in July 2013 added to the Google Play Services, which is available on all Android devices which has Google Play Store and is running Android version 2.3 or later (Computerworld, 2013).

As of Android 4.3, there is an application permissions manager that you can use to alter applications individual permission’s after you’ve installed it.

One issue with Android is that security fixes are included in new versions of the operating system (TrendMicro, 2013). For various reasons most phones get these updates at a slow pace, thus leaving them vulnerable to the threats these security fixes are trying to fix.

Several security companies, i.e. Symantec, Kaspersky, TrendMicro, McAfee, Avast and AVG, have released security solutions for Android that are available on the Google Play Store.

2.4.2 Chrome for Android

Chrome for Android is a browser developed by Google and is the default browser on newer versions of Android (Beverloo, 2012). It was launched as a beta on February 7, 2012, and was released as stable on June 27 the same year. It is only available for Android version 4.0 (Ice Cream Sandwich) and newer. It supports most HTML5 features (Mobile HTML5, 2013).

2.5 Smart cards

Smart cards (also called chip cards or integrated circuit card) are pocket sized cards with embedded integrated circuits and are generally made of plastic. Smart cards can provide identification, authentication, data storage and application processing (Hendry, 2007).

Common examples of smart cards include credit cards, cards used in public transit as well as the SIM cards that phones uses.

There are three major types of smart-cards. There are contact cards, contactless cards and multi-component (or hybrid) cards. Contact cards have electrical contacts located on the surface of the card, as they require physical contact with the card-reader. Contactless cards are able to be communicate with the card reader without physical interaction using radio-frequency identification technology (RFID). Multi-component (also called hybrid) smart-cards have a dual-interface, in that they offer both contact and contactless components, and can have either shared or separate storage and processing abilities (CardLogix, 2010).

2.6 My Healthcare Contacts

My Healthcare Contacts is an e-service system that offers accessible and secure communication between patients, healthcare professionals, and people working in health care.

It is an ongoing project that was officially released to the public in June 2003. The authority responsible for My Healthcare Contacts is the Stockholm County Council, and is financed and maintained by Inera AB.

Inera AB is a company owned by all county councils (Landsting) of Sweden. Inera’s goal is to develop and maintain services within e-healthcare. Inera is one of the companies that take part of the national project National eHealth (Nationell eHälsa), which aims to offer new solutions and innovations within the field of healthcare.
My Healthcare Contacts can be used by patients and healthcare professionals to perform these tasks, amongst others:

- Make a doctor's appointment.
- Change or cancel an appointment.
- Renew a prescription.
- Renew a certificate of illness.
- Obtain an extract from a patient record.
- Obtain medical advice from nurses and psychologists/psychiatrists.
- Change a family doctor.

2.6.1 My Healthcare Contacts SDK
A part of the My Healthcare Contacts system is the My Healthcare Contacts SDK (Software Development Kit).

The SDK is financed by Vinnova, and is delivered within their program My Healthcare Flows. The goal of My Healthcare Flows is to give the general public a greater amount of services in their health care. This program is run by Stockholm County Council with Region Skåne and Region Västra Götaland (Mina Vårdkontakter, 2013b).

The SDK is a tool that allows the development of new services and solutions within the healthcare sector, by giving companies and developers, both within and outside the field of healthcare, access to data and resources (API’s) relating to My Healthcare Contacts and the healthcare field. These API’s (Application Programming Interface) provides access to open data, as well as patient-related data, after the secure approval from the actual patient. This patient data is obtained from medical record systems as well as other sources.

The SDK API architecture uses REST (Representational state transfer) technology. REST will be described in chapter 4.1.9.

In the case of My Healthcare Contacts SDK, the REST uses JSON (JavaScript Object Notation) to transfer its data. JSON is a text-based open standard that was created for structuring human-readable data to be transmitted over a network connection, often between a server and client.

2.6.2 Security
As My Healthcare Contacts is a framework that deals with a lot of highly sensitive information about caregivers and members of the public, the need for a high level of security is essential.

All of the information which it deals with is also considered to be health records, which is protected under the Swedish Health Records Act (Patientjournallagen) as well as the Swedish Secrecy Act (Sekretesslagen).

All traffic to and from the My Healthcare Contacts framework is encrypted to ensure that no one is able to intercept and read any of the information that it deals with. Also, none of the data which is sent to the user is stored on the computer or mobile device which was used to communicate with My Healthcare Contacts (My Healthcare Contacts, 2011).

The system is also designed with confidentiality and authorization to read information in mind. Users may only access, and perform tasks relating to, information and data which they have the right to read. Patients only have the right to see information and issues that relates to themselves, and the only other person which has the right to see it is the personnel which the issue is assigned to.
In order to first set up an account that can be used with My Healthcare Contacts, the user is required to be registered in the population registry for one of county councils/regions which the My Healthcare Contacts service is available. The new user is then checked and verified against this personal data registry. This check is then also performed again at every subsequent login to the system to ensure the continued validity of the user.

For caregivers and other personnel, they are also checked and verified against the county council's electronic catalogue which contains information about each medical center in Sweden and the personnel which works there.

In order for users to login to the My Healthcare Contacts online website, it is required that users either have a valid e-identification (e-ID) or go through a strong two-factor authentication (Mina Vårdkontakter, 2012a).

The e-identification is comparable to an ordinary ID document, such as an ID card or a driving license. In order to get an E-identification, you are required to have a Swedish personal identity number, be registered in the national civil registry and have reached 18 years of age. These e-identifications are issued either by banks or Telia (e-legitimation.se, 2013). As SITHS-cards contain an e-identification, it is possible to use these cards for logging in to the service as well.

The two-factor authentication consists of a user ID, password and SMS text message. The user ID is the person's Swedish personal identity number. You get the password when signing up by entering your id and phone number, and having it sent home to the address which is linked to the address which is connected to your personal identity number, from the national civil registry. You are then able to use your ID, password, and request a temporary SMS text message OTP (One-time password) whenever you wish to login.

For users to login to applications which utilize My Healthcare Contacts through API calls, the user is required to authenticate using SITHS-cards. These are smartcards which contains, amongst other things, an e-identity. More information about them will be presented in the next chapter.

In order to use the My Healthcare Contacts website, it is required to use SSL 3.0, with at least 128-bit encryption (Mina Vårdkontakter, 2009), whereas connections made to the My Healthcare Contacts SDK uses OAuth 2, and requires TLS 1.0 or 1.1 (with 1.2 not yet supported) (Mina Vårdkontakter, 2013a). The server is using RC4 to mitigate the risk of a BEAST attack (“Browser Exploit Against SSL/TLS”).

2.6.3 SITHS-Cards
SITHS is a smart-card based work identification for both physical and electronic identification, and can be used for various IT-services, such as logging into various personal or professional healthcare services.

The SITHS card has several areas of use and is adapted for all national services in e-healthcare. Other than being able to use it for physical and electronic identification, it can also be used for logging in to computers and systems if the departments has set up the system to allow that. It can also be used for electronic signing of documents, such as invoices, prescriptions, journals etc. It can also be utilized as a physical key for areas such as rooms and departments. (Inera, 2013)

A SITHS-card contains a personal Telia e-identification along with a work certificate. The card can also be complemented with a company specific certificate if one wishes to use it for further implementations (Inera, 2013).
2.6.4 Net iD Access

Net iD Access was created by SecMaker AB, which is a company that works with smart-card based security solutions for companies, government agencies, and organizations. They have created and maintain their Net iD product family, which offers solutions to secure information, systems, and data traffic. (SecMaker, 2013a)

Net iD Access is an application created to handle smart-card based authentication on mobile devices. In combination with Net iD Access, they also created Net iD Access Server in order to handle server side authentication.

It all works in such a way that the third-party native or web based client application sends its authentication request to the third party local server, which then sends the information to the Net iD Access Server. At the same time, the client app gets switched over to the Net iD Access application, in which the user authenticates him or herself using the smart-card and password. The Net iD Access application then sends this information to the Net iD Access Server, which makes sure the user information that is authentic. It then sends a message to the Net iD Access application, which then sends the user back to the client app. The Net iD Access Server also responds to the request that the server sent to it, which contains the authentication result of the login request. The server can then verify that the user is authenticated or not to the client app, which can then proceed to handle the successful or failed authentication. See figure 2.10 for an overview of the work flow.

All of the data communication between the server, Net iD Access and Net iD Access Server is encrypted to ensure security.

In order to use the SITHS-cards, a card-reader which is able to read these specific card types, is needed. The card-reader is connected to the computer or mobile device, and the SITHS-card is then used with the card reader. The website or software then prompts the user for the six-digit passcode, which was given out along with the SITHS-card, in order to help verify the user (Mina Vårdkontakter, n.d.).
Figure 2.10 The workflow of the Net iD Access application. (SecMaker, 2012)

Figure 2.11 Net iD Access iOS app. (SecMaker, 2013b)
2.6.5 Laws and regulations

My Healthcare Contacts follows a number of laws and regulations that My Healthcare Contacts follows, that one should also follow when dealing with the type of data that My Healthcare Contacts does. (Mina Vårdkontakter, 2012b)

These are as following:

- **The Health and Medical Services Act (Hälso- och sjukvårdslagen)**
  A framework law part of the Swedish constitution that applies to all health and medical services in Sweden. It lists what responsibilities the law enforces upon the county councils, the local authorities or whatever equivalent health care, provider has towards the patient.

- **The Patient Data Act (Patientdatalagen)**
  This law concerns patient safety and the protection of any sensitive information. The law contains provisions to deal with the handling of personal data from health care providers in the national health care system, and who has the responsibility to maintain the security of this data.

- **The Personal Data Act (Personuppgiftslagen, PUL)**
  This is a law that was created to ensure personal integrity of people and their data when personal information is handled. It also ensures that everyone has the right to know what information about you is stored in any healthcare registry.

- **Official Secrets Act (Sekretesslagen)**
  The law contains regulations that apply to health care personnel regarding their duty to uphold the confidentiality and security of patients and their data.

- **Act on professional activities in health and medical care (Lag om yrkesverksamhet på hälso- och sjukvårdens område)**
  A law which sets responsibilities and duties for health care personnel.

- **Freedom of the Press Ordinance and Archives Act (Tryckfrihetsförordningen and Arkivlagen)**
  Two laws which states how public authorities must handle information which is sent and created within an organization, and what type of regulations and guidelines that needs to be followed when archiving and storing this type of data.

- **The Swedish National Board of Health and Welfare's regulations and general advice regarding patient medical records**
  These regulations provide information regarding the handling of patient medical records.

- **The National Board of Health and Welfare's general advice regarding verification of identity, etc., concerning patients**
  Contains information and advice on how to obtain identity of a patient.

- **The Stockholm County Council Archives' regulations on long-term storage of electronic patient medical records**
  Contains information on how the Stockholm City Council archives the patient data.
3 Interviews

To improve our understanding of industry practices used in mobile security we conducted interviews with people connected to different fields that this thesis touches upon. We wanted to do these interviews for the thesis in order to get a wider perspective on the field from people who work with it in one way or another.

3.1 Interview process

We based our interview questions on the information gathered in the theoretical chapter, in order to get further information about the subjects we cover from sources that work in the business. We based many of our questions on the interviews in other reports in this field that we read as part of our theoretical study. We also asked each interviewee different questions, depending on what that person worked with or how we felt that person could contribute. If we felt that answers could be elaborated on or if answers inspired new questions we would ask follow-up questions.

The questions were conducted in person and by e-mail, in Swedish as well as English, throughout the duration of our work with the thesis.

3.2 SecMaker AB

Our first interview was with Jonas Öholm, Key Account Manager at SecMaker AB, the developer of the Net iD app mentioned in the last chapter. SecMaker works with Windows 8, Android and iOS, but have yet to work with Windows Phone. They consider mobile devices to be secure enough for their work, but suggests that developers consider Datainspektionen’s (a government agency that works to protect personal information) checklist for safeguarding personal information. Their customers use different kinds of web based apps, including HTML5 apps and hybrid apps. Jonas believes that an apps security depends on its implementation rather than if it is a web app or a native app.

As Jonas works with smart card readers he was a major source of information regarding that. In the beginning of our thesis he said that smart card readers for Android were not usable, this changed during our work, a couple of months later he said that a functioning reader now exists.

SecMaker only saves data from the smart cards that is public, and saves that data in the apps’ sandboxes.

3.3 Landstinget Kronoberg

The Application Manager at Landstinget Kronoberg is Helena Sjögren. She says that they are not currently working with mobile devices but will work with Windows 8.1 Pro tablets in the future. This is because they have standardized the use of Windows, and most of the programs and systems which are used in landsting in Sweden only works, or at least works better, on the Windows platform.

She considers Windows tablets to be as secure as Windows laptops or desktops. The requirements on mobile devices are the same as those on laptops and desktops, as are the policy controls. They are not currently using apps.

Landstinget Kronoberg is like most other county councils connected to My Healthcare Contacts and will use new national services as they are released. Today it is possible to for example book, cancel and reschedule appointments. This is done with a connection
between My Healthcare Contacts and their healthcare information system via the national service platform.

As Helena is involved with the higher level workings of their security she forwarded some of our questions to a developer. They were able to help by elaborating on a few of the things. Windows tablets are handled the same way as Windows laptops and desktops, as they are using the same operating system and software. Cell phones are not allowed in the IT infrastructure except to access e-mail, calendar or contacts through the use of a special software that administers and handles those items securely. Both My Healthcare Contacts and their healthcare information system uses a SITHS card for logging in.

3.4 Linnaeus University
We interviewed Linda Sohlberg, Web Coordinator and Project Leader at the Linnaeus University.

The university is developing a new student information app for iPhone and Android. The purpose of the app is to enable students to better keep track of important things in their studies, e.g. when to register for a course, when new results has come, when there is a book to pick up at the library etc. There is also a list of a student’s courses and messages from teachers. They are using open APIs to develop this.

Security is a big focus as they are connecting to systems containing sensitive personal information and data on students’ courses, especially when a student logs in and when data is pushed to the app.

The systems that are feeding data to the app has to follow the Swedish Secrecy Act which means that app has the same security requirements as for example Ladok (a Swedish system for documenting results from colleges and universities). They can only completely fake test accounts which has the consequence of making support difficult as they cannot see the same information that a student can.

They did not spend much time deciding whether to make a web app or a native app. The university’s web site will work on cell phones due to the work that has been done to make it responsive, in that it will adapt and work on many different screen sizes and resolutions. What is missing from a web app is the ability to use push notifications to alert students that something important has happened, e.g. room change, lecture cancellation, library pick-up or a exam registration. There are advantages with a web app but because of the lack of push notifications they went with native apps.

The distribution of mobile device operating systems that are going to the university web site is as follows:

- iOS 70 %
- Android 27 %
- Windows Phone 0.7%

3.5 Softwerk AB
To get the specifics of how the Linnaeus University handles security Linda got us in touch with Rüdiger Lincke, CEO at Softwerk AB, the company that develops the university app.

He told us that the app uses SSL secured connections and certificates which are automatically validated. It saves the username but not any passwords or secret data. Instead it saves a login-token as plain-text but “sandboxed” from any other applications. It is possible to extract this information if the phone is rooted. The apps follows the industry
standard.

He finds it in general difficult to guarantee the security of data on a mobile unit as it is possible to do anything if one gets access to the device. He suggests storing very important data on the server side, assuming the server and the network is secure. However, he considers mobile devices to be reasonably secure if you follow the standard procedures and use a non-rooted device. He adds that many applications contain no important data that needs protecting.

He finds the difference in security between native and web based apps depend on the app and the context. There can be similar or different security issues with both approaches. The examples of security issues he gives are spoofing network communication, comprising the server and the theft of devices. If you follow the common practices, both approaches are reasonably secure.

Softwerk has experience developing hybrid apps, as they have developed apps which uses HTML, CSS and JavaScript for certain parts, mainly for creating layouts for data. They usually use it for common but not device specific features.

He thinks that web based apps could probably be secure enough for use in healthcare if existing standards would be followed, but warns that it isn’t always possible to control how data is handled on the device by the web browser, e.g., caching of user data, cookies, etc.

He also said that he expects the security on mobile devices to be improved in the future.

3.6 F-Secure
Our final interview was with Johan Jarl, security specialist at computer security company F-Secure.

Jarl thinks that the security on mobile devices generally is low, with Android being the platform with most malware.

Currently most malware is contained in native apps, but he has seen examples of malware being spread by web. He thinks that web based apps will be the biggest security threat in the future.

He thinks that it is difficult to say whether a web based app is secure enough for use in healthcare. It depends on what information the app would handle and if the app would be hosted locally. For example, he would not consider sensitive information in a web based app hosted publicly on the internet to be safe enough.

For building a secure web app he suggests not storing or caching data client-side, authenticating users, encrypting traffic and hosting the app locally. Further, if the web server and the client are both secured he would consider it to be safer than a native app.

He thinks that in the future the security will be better on mobile devices, as companies and individuals will have to secure their devices. The threat will be large on these devices, the last 3 years the threat has greatly increased on the Android platform.

3.7 Summary and impressions
From the interviews that we conducted, we got a lot of good information that supported our theoretical findings, and was useful throughout our continued work with the thesis.

The interviews, specifically the interviews with Rüdiger Lincke, Jonas Öholm and Johan Jarl, gave us the impression that it is possible to create a web app that could be as secure as a native application. They also expressed that the security is more reliant on the implementation than whether the application is web or native based, i.e. the security
measures that a developer takes, and that there are certain implementation considerations that needs to be considered.

Although not all the interviewees who we asked about the subject definitively said that web applications are secure enough to be used in healthcare, we still got the impression from the ones that did answer, that it is possible to make an web secure enough for us in the field of healthcare, and that it once again relied more on the implementation itself rather than the platform.
4 Security models

By taking the information gathered in the theory section as well as through our interviews, we have devised a few different security models, which we define as packages of security technologies, which we consider to be the most plausible and realistic models for implementations.

4.1 Security technologies

There are a great number of technologies that one could use to set up a secure environment for mobile devices, and use in different security models. We will go over a number of the different options which each technology offers, and what is considered to be the best for certain circumstances.

4.1.1 Cryptographic protocols

Secure Sockets Layer (SSL) and its successor Transport Layer Security (TLS) are cryptographic protocols created to provide communication security over the internet. SSL exists in three versions, 1.0, 2.0 and 3.0, however the first version was never publicly released, and 2.0 came out soon after in 1995 with a number of security faults. Version 3.0 came out in 1996 after a complete redesign to enhance security. But SSL 3.0 is now old and vulnerable for attacks.

Further development for this was shifted to a new protocol, in TLS 1.0 which was released in 1999 (Allen & Dierks, 1999). It did not contain a great deal of fixes, but was meant as a stepping-stone from SSL to TLS and its future versions. In 2006 TLS 1.1 was released with a number of security improvements, such as Cipher block chaining (CBC) attacks and IANA parameter support (Dierks & Rescorla, 2006). TLS 1.2 was released in 2008 and is the most recent TLS protocol that is widely available to the public, and is generally considered the most secure with security upgrades such as support for SHA-256 (Dierks & Rescorla, 2008).

4.1.2 Local data

There are options for handling data locally, you can choose to not save it at all, save it as plain-text or save it encrypted. Not saving data locally is according to the theory and our interviews the safest, as there is then no data which can be accessed by any attacks. If that is not a possibility, then saving the data encrypted is the best option. Even better would be if you encrypted it using a salt, but more on that in the next part. The least secure way to store the data would be just storing it openly in plain-text, but this can be mitigated if you are using a sandbox environment, but we will go into more detail about that subject in a later part.

4.1.3 Encryption

There are several encryptions used for encrypting data. Some of the more common ones include DES, Triple DES, RC4 and AES.

DES is a block cipher that was published in 1977 as an official Federal Information Processing Standard (FIPS) for the United States. It inputs plaintext as a string and outputs a cipher text string with the same length as the plaintext. Its block size and key size is 64
bits. Although the key size is listed at 64 bits, 8 of those bits are parity bits used for error detection, this giving DES an effective key size of 56 bits. This is the primary reason that DES has since become to be considered insecure with the advancement of computing power, which in turn lead to the triple DES (Conrad, n.d.).

Triple DES (also known as the TDES) is a continuation of DES, and was published in 1998. Like DES it has a block size of 64 bits, but because it applies the DES cipher algorithm three times, its key size effectively becomes 168 bits (56 * 3). It is also a FIPS approved algorithm, although it has been set that it will no longer be approved after 2030 due to the fact that they instead wish to transition to the more secure AES (Conrad, n.d.).

RC4 is a stream cipher that was developed in 1987, but was not released until 1994. It has a variable key size between 40 and 2,048 bits. If one wishes to use this encryption, one should take into consideration that it has weaknesses that, if coupled with the wrong technology, can become exploitable. For example, if one use RC4 and WEP, there are a number of weaknesses that that makes is possible to exploit this and crack the encryption (Grier, n.d.).

In 1997, National Institute of Standards and Technology (NIST) set out to find a better encryption standard to replace the DES (and to some extent the TDES). The solution was the AES (Advanced Encryption Standard), which was created by two Belgian cryptographers, Joan Daemen and Vincent Rijmen. (TechTarget, 2011). AES is the newest of the previously mentioned encryption standards, being published in 1998. AES became the new FIPS-approved encryption in 2001 (Conrad, n.d.), and it is used by the U.S. government (Schwartz, 2000).

4.1.4 Operating systems/Platforms
There several operating systems/platforms you can choose to develop for, both for desktop and mobile devices. For example, popular platforms on mobile devices include Android, iOS, Windows and BlackBerry. Some popular platforms for a desktop environment are Windows, OS X and Linux.

As we detailed in Chapter 2 och 3, Android has more malware than the other mobile operating systems, which does not mean that it is impossible to develop secure applications on that platform, only that it is something that one should take into consideration when deciding upon a security model for the implementation on that platform.

4.1.5 Browsers
For some of the different operating systems/platforms there are different web browsers you can use for a web app.

On Android you can use the stock Android browser, Chrome, Firefox, Opera etc.

On iOS you can use Safari (the default browser), Chrome, Opera etc. Some browsers, such as Firefox, are not available on iOS due to platform restrictions (Firefox, 2013).

On Windows 8 you can use Internet Explorer (the default browser), Chrome, Firefox, Opera etc.

There are a lot more alternatives, which may or may not be secure, so it is recommended that if you count on dealing with sensitive information, using the more well-known or even default browsers might be a better option.
4.1.6 Smart cards
One way to handle the authentication in a safer way is to use certificates, and store them on smartcards. In order to utilize smartcards, you need a smartcard reader, and for the reader to be supported by the device you wish to use it on. For more background about smartcards, see chapter 2.5.

There are different options for implementing smart-cards for security. One can simply have the smart-cards themselves be the only authentication needed. This is not recommended however, as this simply gives proof of possession, not proof of identity. Instead, it would be better to utilize the smart-cards storage and processing capabilities to implement a two-factor authentication. For example, they can contain some simple user information securely encrypted on the device, such as username and password. This means that the user needs to both have the smart-card and know the password for the username. Another step further would be a smart-card such as the SITHS-cards, which are password protected and contains personal certificates.

4.1.7 Sandboxing
A more detailed description of sandboxing is available in 2.2.1 but to summarize; sandboxing, or containerization, is a security mechanism for separating running programs.

There are three different types of sandboxing; “Bare metal” sandboxing, “VMWare style” sandboxing and application level sandboxing. The bare metal sandboxing is the most intrusive method, whereas application level sandboxing is the least intrusive.

4.1.8 Network Tunneling
If one wishes to access resources from outside of a local network, one needs to connect to it over existing networks, such as the internet. This is of course not completely safe as anyone could potentially intercept your traffic. This is where network tunneling comes in, and VPN is the most popular one. VPN allows you to create a virtual point-to-point network connection between your device and the remote local network. It works by encrypting your data, and encapsulates it with a new header, which contains the necessary routing information to allow it to move across the shared or public network and reach its endpoint securely. If the data were to be intercepted, it would not be possible to decrypt the information without the encryption keys. (Microsoft TechNet, 2001)

4.1.9 Web Service Communication
The two most common technologies used in web services today are REST (Representational State Transfer) and SOAP (Simple Object Access protocol). Because My Health Care Contacts uses REST we will focus on and use it and not SOAP in our implementation.

REST is a software architecture for distributed systems that allows for clients to communicate with servers. In general, it works by having the client send a request to the server, which processes the request and returns appropriate data to the client through its response. REST typically communicates over HTTP and uses the common HTTP verbs like GET, POST, PUT and DELETE. (Wikipedia, n.d. b)
We will not focus on the security of REST as our work focuses on the differences between native and web based applications, and as we use REST in both implementations the security of REST does not impact that difference. Furthermore REST (or SOAP for that matter) is not inherently secure or insecure, the security lies within the implementation and the other technologies used to implement the communication.

4.2. Proposed models
These are the models we have devised using the security technologies available and are proposing as solutions to the problem of creating a secure environment for mobile applications.

4.2.1 Shared technologies between models
Here are the technologies that we feel should be utilized in all our following models.

The latest version of TLS, 1.2, should be used as a cryptographic protocol as it is the most recent, contains a number of bug fixes and should be more secure than the other alternatives. This is the best option if one is sure that all clients are able to use this technology, however, it is recommended that one also support the still somewhat secure TLS 1.0 and 1.1 to allow clients that does not support the newer TLS 1.2 version to still be able to use the service. One should take into consideration that if you want to allow TLS 1.0 for example, it does have weaknesses that can be exploited, so it is very important that one take precautions against the known weaknesses, such as the BEAST attack (AlFardan & Paterson, 2013).

Using certificates stored on smart cards that are read by a smart card reader gives a high level of security. If you are dealing with a lot of sensitive information, this is a very secure option, if available.

Depending on if you want a block or stream cipher, we consider AES and RC4 respectively to be the more secure encryption standards. Although, as we mentioned previously in the RC4 part, there are certain situations in which RC4 should not be used, which one should take into consideration when opting for a security model for a system.

The communication between the applications and the web service server will be using REST in all models.

4.2.2 Native app on a mobile device
Developing a native app is a good alternative as you get the security of that platform and can use more advanced APIs that uses the operating systems built in features.

It will require more time and effort during development, but as previously stated, it does offer more direct access to secure features.

If one needs to store data on the device, one should take into consideration to store it privately, and not with a publicly accessible flag, or directly on the SD card.

4.2.3 Native app on a Windows 8.1 Pro tablet
As mentioned in the interviews, developing a native app for a Windows 8 tablet is a valid option because of the long history of use in healthcare, and its existing standards for using Windows, coupled with the mobility of a tablet.

With a Windows 8.1 tablet you will get built-in security features of Windows 8.1, such as firewall and anti-virus.
4.2.4 Web app in Chrome browser on Android
If developing a web app for use on Android devices, the Chrome browser is a valid option as it updates regularly, and is available on most mobile devices.

If it is necessary to deal with sensitive data, there are considerations that should be taken, hosting the application locally is a good option. If it is not possible to host it locally, requiring the use of a network tunnel, for example through VPN software, is a good and secure alternative.

4.2.5 Web app in web browser on Windows 8.1 Pro tablet
It is possible to modify the previous tablet model to use a web app instead. There are several secure web browsers on Windows you can use to access a web app. For example, Internet Explorer is the default web browser in Windows 8.1, and it is possible to install other ones to use as well.

4.2.6 Web or native app in application level sandboxing
If one wants to use a web or native application, such as in the previous models, one can improve the level of security by implementing an application level sandbox for those applications. This means running the application inside a dedicated sandbox application that is either provided or implemented using an SDK.

4.2.7 Hybrid application
Hybrid applications can be an alternative if one wishes to utilize the advantages with both web applications and native applications.

Some of the advantages of web applications are that they take less time and effort to develop, as they essentially display the same content for all devices. But, unlike native applications, they are unable to use most of the mobile device’s API, such as certain security features. Hybrid applications are able to use these same features as native applications by encapsulating the web application with a native application.

4.3 Our Restrictions
The restrictions imposed by Sigma, ourselves, the My Healthcare Contacts system as well as other circumstances narrowed down the available technologies that we can use from the security models in our implementations.

As My Healthcare Contacts SDK does not support any SSL version, nor TLS 1.2, those are not an option for us. We will instead be using TLS 1.0 and 1.1.

We were instructed to use Android, so any native app model that uses any other operating system is not an option for us during the implementation of our applications.

To use the My Healthcare Contacts framework you need to use SITHS cards so the models that use certificate cards are to be used. As the Chrome browser is installed by default on newer versions of Android and should be considered as secure on as the desktop version, as it is a fork of Chromium, the open source project that Chrome uses as a base (Beverloo, 2012), we choose to use those models.
5 Implementation

In this chapter we will go over the various parts of the implementation.
We were tasked by Sigma to create two applications with the same functionality, one for
the web and one for Android. The task were to use the My Healthcare Contacts system.

5.1 My Healthcare Contacts
As described in the restrictions chapter, in order to access the My Healthcare Contacts
system, we were required to use SITHS-cards.

Because of this, we decided together with Sigma, Martin Östlund and our supervisor to
implement a server solution which would simulate the authentication by using the same
authentication calls, but to our own server, which would authenticate us. To do this we
developed a web service which would only accept and return the same authentication data
that My Healthcare Contacts would, with the same level of security on the connections, but
with our local certificates instead of using the SITHS-cards.

As we progressed with the work, a number of obstacles and challenges arose My
Healthcare Contacts.

One of the main obstacles we encountered during our work was that My Healthcare
Contacts uses SITHS-cards for its user authentication, which we detailed in chapter 2.6.3.
The problem we came upon with these are that smart-card readers that support the SITHS-
cards were very limited at the time. Plus, in order to use the SITHS-cards, a third party
application called Net iD Access is required. Net iD Access application facilitates the use
of smart card readers on Android that are compatible with SITHS-cards. Due to the fact that
smart card readers were severely lacking at the time, a Net iD Access application on the
Android platform had not been released. We will talk more about this application later in
the subchapter that is dedicated to it.

Another issue we came upon was that My Healthcare Contacts is still an ongoing project
that has not yet fully implemented all of its proposed features and functionalities. And as
certain parts of the framework are not yet finished, we would not be able to call certain
functions towards the My Healthcare Contacts framework. At the time of this writing,
October 15, 2013, 29 out of the 158 planned functions had been implemented, with that
number expected to reach 58 sometime during the next quarter.

Even if the functions that we wanted to access was available, in order to access them
using secure methods we would be required to fill a number of prerequisites, including
obtaining a EV Certificate (Extended Validation Certificate) from one of three approved
certificate authorities. These were Inera, VeriSign, and GlobalSign. Inera is the company
that is financing and maintaining the My Healthcare Contacts system, thus offering their
own version of the certificate. These special certificates was quite costly at all of the three
certificate authorities. The certificate offered by Inera is part of a subscription plan where
the server certificate costs about $529 (3500 Swedish kronor). At GlobalSign it costs $899
per year, and at VeriSign it costs $1,499 per year. This was obviously another problem for
us.

So there were a number of problems with connecting to the sharp My Healthcare
Contacts environment, but as one of the requirements from Sigma were that we develop our
own functionality to handle daily notes on our own server (daily notes not being a part of
My Healthcare Contacts, after discussing it with Sigma as well as our supervisor, we decided that we should instead develop simulations of these services that were part of the My Healthcare Contacts our local server. Then, with the help of these services, perform the same calls with the same level of security towards our own environment instead, as the goal of the thesis work was not specifically to investigate the security of the My Healthcare Contacts system, but that of mobile devices running native and web based applications, and the My Healthcare Contacts framework only being a tool with which we do that. This meant that we needed to ensure that we implemented those services with the same properties and level of security as the real ones.

5.2 Server and Web services

We wanted a set of web services that both the native and web app could use. In order to do this, we first needed to set up a secure server. We thusly set up a local test server running Windows Server 2012.

A database was set up on the server that were to host all of the data that we would be dealing with in the examples of running. The database that was running was a version of MySQL (5.6.11).

We created the database schema by taking into consideration what type of data we would be required to use throughout the applications. For an example of the authentication we created a table for the users, with fields for a user id, user name and user password. For the patient notes samples, we set up a table for notes, which contained fields for the note’s id, date, author (user), patient (patient id), and text. To tie the patient note to a patient, we used the patient id field in that table, and then created a table for patients, which contained patient id, person-id (global identifier), and name.

We then used its built in Web Server to create a place for the implementations for our authentication and web services.

The web server was, like My Healthcare Contacts, running a secure connection over HTTPS using TLS 1.0 and RC4-SHA, with 256-bit encryption (referred to as High). This ensures that the traffic we send between the devices and server is encrypted and secure. We chose to use the same security settings on the encryption on our server as the one which was being used on the My Healthcare Contacts servers, to ensure that we follow the same level of security and have a parity between them.

We set up two web services:

- **Authentication**
  
  As we previously discussed, authenticating ourselves against the real My Healthcare Contacts system was not possible at the time, so we instead set up our own authentication web service, which would be used to simulate an authentication request to the My Healthcare Contacts system through SecMaker.
  
  The web service was constructed to work in the same way as the My Healthcare Contacts system. It was built using SOAP (Simple Object Access Protocol), and it accepts SOAP posts to make the authentication requests. It can return positive and negative authentication results depending on what data is posted to it.

- **Daily Notes**

  We set up a web service to handle the functionality of daily notes in the system. This web service utilizes the REST (Representational state transfer) technology with the JSON text format to make requests.
The daily notes web service has a number of functionalities to allow working with the daily notes in a useful way. One can post data to the web service, which includes creating a new daily note, editing a daily note as well as removing a daily note. One can also retrieve all daily notes or a specific one using its ID.

We also set up additional services on the server to ease the workflow, like an FTP for getting and sending data, as well as a SVN server where we stored and handled all of our code.

5.3 Android application

We started by developing a test application for the Android platform that would check for a certificate that was installed on the system, then, if that certificate was valid, would give access to the application, which allowed the user, without any further authentication, to list some static information retrieved from the test-service on the My Healthcare Contacts server.

We initially tried to implement an Android app that was compatible with older versions of Android but we soon discovered that certain certificate handling requires Android 4.0+, so we began focusing on newer versions of Android instead. The certificate files used by our app are stored in the app’s APK file, instead of on SD-card or other external memories.

Because we are using a server with self-signed certificates we had to implement custom certificate handling. This proved to be very difficult, as there is very little information on how to do this. Our development device broke during the development, which made this even more difficult as it was not possible to test the app in an emulator. After a month we were able to borrow a device from the school which we were able to use to complete our development and testing.

The finished app consists of 4 Android activities (classes that represent a screen and inherits from the class Activity).

- One activity lets the user choose a certificate and enter the password.
- One activity lists all the notes.
- One activity lets the user view more information about the note.
- One activity lets the user add or edit notes.

The app also consists of several xml files that are used for layout and drawables, and image files. No data is stored in the app, all data is fetched from the server at runtime.

UML diagrams for the app can be found in the appendix under Appendix A, and a flow chart for using the application can be found in figure 5.1.
5.4 Web application
We also created a web-application using the .NET platform. More specifically, it was created using the ASP.NET web application framework with a MVC (Model-View-Controller) design pattern architecture.

It was a basic application which followed the same guidelines that we used to create the native application. It also uses styling with CSS to make it look similar to the native application.

Unlike the native application, custom certificate handling was not needed, and we instead utilized the browser’s built in certificate handling for the authentication of users, which automatically took care of the verification. Other than that, it contains the exact same features as the native application and uses the same security settings.
There are UML diagrams, that describe how the web based application is constructed, available in the appendices under Appendix B.

5.5 Testing
We performed a couple of simple tests to establish that the level of security on our implementations was working as intended.

We performed network analysis on the data being sent to and from the server using Wireshark (Wireshark, 2013). Wireshark is a packet analyzer tool that lets you troubleshoot, analyze and study network traffic on a packet level. It allows you to listen to network interfaces and capture the network traffic that is visible to that interface, and then analyze its packets.

On the server, we ran wireshark while using the application to capture packets to and from the server. By performing these tests we could analyze the packets and establish that the data which the two different applications (native Android application, and web based application) sent and received is the same. We did this by looking at each packet and the content of its headers as well as its payload, and compare it to the corresponding packet from the other application. By doing this we were able to see that the same type of packages was sent, with the same types of headers.
6 Result

In this chapter we will present our results.

The problem of this thesis was that Sigma wanted to explore the possibility of developing one web app instead of several native apps, and Martin Östlund wanted to explore the possibility of developing third party apps using the My Healthcare Contacts framework.

From this problem we devised two main questions:
1. How does the security of web apps compare to that of native apps?
2. How does one develop an app that uses the My Healthcare Contacts framework?

We have solved the problem by answering these questions using existing theory and literature, interviews with people in the business, constructing security models and implementing our own apps.

6.1 How does the security of web apps compare to that of native apps?

This question was the main question that we and our stakeholders were most interested in being answered in our report. As detailed in the first chapter, Sigma wanted to know if developing web applications is a viable option for them in the future. The reason they wanted to know this, was because they were interested in reducing development time and costs, as well as time spent on maintenance for multiple platforms. They therefore wanted to know whether or not the security in web applications could compare to that of native applications.

We have answered this question by:
1. In the first chapter we describe how we would perform a literary study on mobile security, Android, web apps, native applications, etc, by using sources from Google Scholar, DiVA (Digitala Vetenskapliga Arkivet), Science Citation Index Expanded etc, as well as search engines and online encyclopaedias. In the second chapter we detail the results of that literary study, and our findings. Our research found that it was possible to build web applications to be as secure as native applications.
2. In the first chapter we describe how we would perform interviews with a number of people we thought would be relevant to our research. We interviewed people like app developers, people involved in healthcare and its security, and security experts. We summarized those interviews in the third chapter, and through them we found that it was possible to build secure web applications.
3. In the first chapter we describe how we would develop a web application and a native application using all of the same security technologies as My Healthcare Contacts, such as the same type of authentication. When developing our applications we could use the same security technologies, which we regard as being able to achieve the same level of security on both applications.

We have found that our research, the interviews and the implementation of our apps have shown that it is possible for the security of web apps to be as good as that of native apps. In order to accomplish this it is imperative to take security measures, some of which we detailed in the chapter on Security Models (Chapter 4).
6.2 How does one develop an app that uses the My Healthcare Contacts framework?

Sigma and Martin Östlund was interested in this question, detailed in chapter 1, because they were interested in how the My Healthcare Contacts framework could be utilized. As a software development company, Sigmas interest was from a practical point of view, whereas Martin was more interested in it from a theoretical point of view.

We have answered this question by:

1. As part of our literary study we researched the documentation, frameworks and regulations of the My Healthcare Contacts. We used many different sources ranging from product documentation, API specifications, website information and other resources. These sources were primarily available on the My Healthcare Contacts website. We present the result of this research in chapter 2.6.

2. In the first chapter we describe how we would perform interviews with a number of people we thought would be relevant to our research. We summarized those interviews in the third chapter. Some of those interviews resulted in information that involved My Healthcare Contacts and gave us further insight into how to develop for it.

3. In the first chapter we describe how we would develop a web application and a native application using all of the same security technologies as My Healthcare Contacts, such as the same type of authentication. In chapter 5 we detail how we developed our applications with the My Healthcare Contacts framework and security as a reference. This can be useful for people who are interested in developing for My Healthcare Contacts.

We believe that our research, the interviews we have conducted and the implementation of our apps have illustrated how one can develop apps that uses the My Healthcare Contacts framework.

Another result of this thesis is our applications (seen in figure 6.1). These apps can be used as prototypes for further development.
Figure 6.1 The main view after logging in from our native application (left) and web based application (right)
7 Discussion

In this chapter we will discuss our work.

The information we have gathered and covered in the theory chapter has caused us to believe that more and more people will buy and use smart mobile devices, and less and less will buy and use traditional PCs. But there will still be a need to use different services that require high security, e.g. banking. We believe that the mobile devices will fill that need. This will lead to more and more security threats for those platforms. Secure mobile devices are therefore a necessity.

We believe that web and hybrid are more popular with developers than with users. For developers they lessen the effort and cost of developing apps for different platforms which is greatly beneficial, but as web and hybrid apps look and behave differently than native apps, users may find them unappealing. One way of combating this is for hybrid app frameworks to make hybrid apps look and behave like their native counterparts, but we do not believe that web apps will grow in popularity.

As this thesis focused on Android our results might differ from results gathered on the other mobile device operating systems such as iOS and Windows Phone 7 and 8, although the basics remain the same.

If we would have begun writing this thesis today the outcome would have been different. For example, when we started our work there were no smart card readers available for Android, nor was the My Healthcare Contacts system as developed as it is today. Some of these situations have changed, or are about to change, so it is possible our implementation would have included them.

7.1 Future Work

During our work, we encountered a number of areas that we feel could be the subject of future work.

One of these would be to incorporate the SITHS-cards and Net iD Access application as part of an implementation. We were unable to get to this due to the immaturity of the card readers available or being developed for Android.

Another aspect which one can work with in the future would be with the My Healthcare Contacts live system. This was not possible for us due to the immaturity of the frameworks.

As we focused our native implementation on the Android platform, some of the comparisons we drew between native and web based application may not be completely accurate on other platforms. Therefore, implementations of native apps for other platforms, such as iOS and Windows Phone could be beneficial to see if there is a difference to our findings on those platforms.

It would also be interesting to expand the investigation in hybrid applications further, to compare them and their security to that of native and web based applications. It would be interesting to investigate, as one could combine the security technologies of both the native and web platforms.

Another aspect of the thesis that we feel would be interesting to investigate is to further test the platforms’ security, and their differences. By doing this one could have received a more thorough security analysis.
Bibliography


Amazon, 2013. Getting Started with the Amazon Appstore for Android. Available at: <https://www.amazon.com/gp/feature.html?ie=UTF8&docId=1000626391&ref_=amb_link_363093842_2> [Accessed 2013-09-16]


BI Insider, 2012. ANALYST: Smartphone Sales Will Dwarf PC Sales This Year And Reach A Staggering 1.5 Billion Per Year By 2016. Available at: <http://www.businessinsider.com/smartphone-sales-forecast-2012-2> [Accessed 2013-09-16]

Bloomberg Businessweek, 2012. Before IPhone and Android Came Simon, the First


Firefox, 2013. *Is Firefox available for iPhone or iPad?* Available at: <https://support.mozilla.org/en-US/kb/is-firefox-available-iphone-or-ipad> [Accessed 2013-11-12]


Mina Vårdkontakter, 2009. *Hur kontrollerar jag min dators krypteringsnivå (SSL-
kryptering)? Available at: <http://www.minavardkontakter.se/C125755F00329208/p/KONT-7U6FTK?opendocument> [Accessed 2013-09-16]


SecMaker, 2013b. *Smarta kort med iPhone/iPad = Ja*. Available at: <https://service.secmaker.com/access/> [Accessed 2013-09-16]


Appendix
Appendix A. UML Diagrams for native application

Figure. UML diagram for Login activity

Figure. UML diagram for List activity
Figure. UML diagram for Note class
Appendix B. UML Diagrams for web-application

**UML Diagram for Index page**

- `ViewBag title : String`
- `ViewBag message : String`
- `header : Div`
- `index() : View`
- `- formattedUri(String) : HttpWebRequest`

**UML Diagram for Note page**

- `ViewBag title : String`
- `ViewBag message : String`
- `header : Div`
- `Note() : View`
- `- RemoveNote() : View`
- `- formattedUri(String) : HttpWebRequest`
- `- verifyRemove()

**UML Diagram for EditNote page**

- `ViewBag title : String`
- `ViewBag message : String`
- `header : Div`
- `EditNote() : View`
- `post()
- `- formattedUri(String) : HttpWebRequest`
Figure. UML diagram for AddNote page

<table>
<thead>
<tr>
<th>AddNote</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewBag title: String</td>
</tr>
<tr>
<td>ViewBag message: String</td>
</tr>
<tr>
<td>header: Div</td>
</tr>
<tr>
<td>form: Form</td>
</tr>
</tbody>
</table>

+ AddNote(): View
+ post()
- formattedUri(String): HttpRequest

Figure. UML diagram for Note class

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>-id: int</td>
</tr>
<tr>
<td>-date: String</td>
</tr>
<tr>
<td>-author: int</td>
</tr>
<tr>
<td>-patient: int</td>
</tr>
<tr>
<td>-text: String</td>
</tr>
</tbody>
</table>

+ getId(): int
+ setId(id: int)
+ getDate(): String
+ setDate(date: String)
+ getAuthor(): int
+ setAuthor(author: int)
+ getPatient(): int
+ setPatient(patient: int)
+ getText(): String
+ setText(text: String)