Självständigt arbete på grundnivå

Independent degree project - first cycle

Computer Science

Integration of BankID Services in a PhoneGap Based Mobile Application

Lars Eggestig
Mintesinot Wodajo
Integration of BankID Services in a PhoneGap Based Mobile Application
Lars Eggestig and Mintesinot Wodajo

MID SWEDEN UNIVERSITY
Informations- och kommunikationssystem (IKS)
Examiner: Ulf Jennehag, ulf.jenehhag@miun.se
Supervisor: Magnus Eriksson, magnus.ekksson@miun.se
Author: Lars Eggestig, laeg1100@student.miun.se and
Mintesinot Wodajo, miwo1102@student.miun.se
Degree programme: Computer engineering, 180 credits
Datateknik, 180 hp
Main field of study: Computer science
Date: 2014-05-29
Abstract

Security concerns became high with the rapid technology advancement and with the open nature of the internet. BankID is the leading electronic identification system in Sweden which is used by around 5 million people in a variety of public and private services. BankID allows users to securely authenticate themselves and digitally sign important documents and transactions over the internet. In 2011, BankID Security App was launched to be used in mobile smart phones and tablet computers. In this paper, different components of the Public Key Infrastructure (PKI) which is a cryptographic technique that enables users to safely communicate over the insecure internet has been studied in detail. Furthermore, a test BankID-integrated PhoneGap based app on the Android platform is implemented and a performance evaluation and security analysis were performed. The test implementation of the BankID-integrated app on the Android platform provides user authentication and digital signing functions. The implemented backend system consists of a server with digital certificate and a database. The performance test emphasizes on the measurement of the access time between the components of the system and usability of the application. Access time measurement includes a reasonable amount of time in which the user is able to perform different activities in the system. In usability assessment number of actions to perform a certain task and the ease of the user interface has been taken into consideration. The security analysis aims to identify potential security flaws in the system and discuss possible solutions. The potential security risks we identified during the implementation of the system are the man-in-the-middle-attack, the Heartbleed bug, losing the mobile device and physical access to the backend system. The potential security risks in the system were examined with regard to severity and probability of occurrence. Finally, the thesis project has been discussed in terms of the future work and system expansions. The result of the thesis will be used as a base in production development by Dewire, the company for which the thesis work has been conducted.

Keywords: Security, BankID, PhoneGap, PKI, Android
Acknowledgements

Thanks to...

**Dewire** for giving us this thesis opportunity.

**Johan Deckmar, Dewire** for help with paper structure and for discussing possibly implementation ideas.

**Tomas Andersson, Dewire** for providing support, ideas and feedback on our work.
# Table of Contents

Abstract .............................................................................................................. iii  
Acknowledgements ......................................................................................... iv  
Terminology ......................................................................................................... viii  

## 1 Introduction............................................................................................... 1  
1.1 Background and problem motivation ......................................................... 1  
1.2 Overall aim ................................................................................................. 2  
1.3 Scope ............................................................................................................ 3  
1.4 Concrete and verifiable goals ....................................................................... 3  
1.5 Outline ......................................................................................................... 4  
1.6 Contributions .............................................................................................. 4  

## 2 Public Key Infrastructure ( PKI ) ............................................................... 5  
2.1 PKI Components and Functions .................................................................. 5  
2.1.1 End Entity ............................................................................................... 5  
2.1.2 Digital Certificates .................................................................................. 6  
2.1.3 Certificate Authority .............................................................................. 7  
2.2 Secure Socket Layer ( SSL ) ....................................................................... 7  
2.2.1 SSL Architecture ................................................................................... 8  
2.2.2 HTTP Secure ( HTTPS ) ....................................................................... 11  
2.2.3 OpenSSL ............................................................................................... 12  

## 3 BankID ....................................................................................................... 13  
3.1 Relying Party ( RP ) ................................................................................... 13  
3.2 BankID Services .......................................................................................... 14  
3.2.1 Web Services ........................................................................................ 14  
3.2.2 BankID Error Management ................................................................. 17  
3.3 Mobile BankID Security Application ......................................................... 18  

## 4 PhoneGap ................................................................................................... 19  
4.1 Plugin Development .................................................................................... 19  

## 5 Analysis Background Information ........................................................... 21  
5.1 Sessions ...................................................................................................... 21  
5.2 Recommendations for Mobile Management ............................................. 21  
5.3 Database Management ............................................................................. 22  
5.4 Physical Server Security ............................................................................ 22  

## 6 Methodology ............................................................................................... 24  
6.1 Development Process ............................................................................... 24  
6.2 Hardware ..................................................................................................... 24  
6.3 Software ...................................................................................................... 25  
6.3.1 Development Tools ............................................................................... 25  
6.3.2 Graphic Implementation Tools ............................................................... 25  
6.3.3 Server Platform ..................................................................................... 25  
6.3.4 SoapUI ................................................................................................ 25
Integration of BankID Services in a PhoneGap Based Mobile Application
Lars Eggestig and Mintesinot Wodajo
2014-05-29

9.6.2 Binding Agreements and Action Clarifications.................................52

References........................................................................................................53
Terminology

**Android**  
Android is a Linux kernel based smartphone and tablet operating system.

**App**  
App is short for application and often used when talking about mobile applications.

**Backend**  
Backend is a term used to describe the service a user make use of, for example one or more server can build up a system providing users with various services, all servers building up this service is referred to as the backend. The backend is accessed by a interface (frontend).

**BankID**  
The term BankID is used to refer to the BankID system used for authentication and signing in a secure way. The BankID system contains a BankID server and a BankID client application. Together with a users BankID electronic legitimation a user can prove his or her identity and access service provided by the RP.

**CA**  
Certificate Authority is the main trusted component of PKI which used to issue, revoke, and manage digital certificates.

**CRL**  
Certificate revocation list contains list of serial numbers of Certificates that has been revoked and which are no longer be trusted.

**DB**  
DB is shortened form of database.

**Frontend**  
Frontend is the term used to describe the interface to a service (backend). In many cases the front end is a client application.

**Heartbleed**  
A recently discovered bug in OpenSSL cryptographic library which is capable of liking important user information.

**HTTP**  
Hypertext transfer protocol is used for stateless communication between server and clients.

**HTTPS**  
Hypertext transfer protocol secure is a securer version of HTTP.
IP Security  A suite of protocols for securing IP communications by authenticating and encrypting each IP packet in a data stream.

JSON  JSON (JavaScript Object Notation) is an open source, human-readable information exchange format used in all kinds of application to exchange information in a structured way. JSON provides a way to exchange Booleans, strings, numbers and arrays without breaking them down to simpler objects.

OpenSSL  A widely used open source implementation of SSL and TLS protocols.

Pfx  A archive format used to store private keys with its certificate in cryptography.

PhoneGap  A framework used to encapsulate HTML, CSS and JavaScript code to a multiplatform mobile app.

RP  Relying party, used to describe a system that provides services using BankID.

S/MIME  Secure/ Multipurpose Internet Mail Extensions is a standard for public-key encryption and signing of MIME data.

SOAP  SOAP refers to the XML-based communication protocol used for structured information exchange over the web and often used in web services.

SSL  Secure Sockets Layer a secure communication protocol used between a client and server.

Stateless  Stateless in computer science are used to describe communication protocols that treats every transaction as a new session or connection. Every time a transaction occurs, a new connection is established, when the transaction is complete the connection is terminated.

WS  The term WS is short for Web Service and a form of communication over the web between clients and server.
X.509  A standard which specifies the standard formats for public-key certificates.
1 Introduction

The thesis project is being conducted in cooperation with Dewire [1], a Swedish IT consultant company specialized in mobile solutions. The task given is to integrate a Mobile BankID security application with a PhoneGap based android application which will be used in the area of medicine and personal health care.

In this chapter we will give an introduction to the thesis project and corresponding research areas. First, we will give a brief background and problem motivation behind the thesis project, then the overall aim, the scope and the concrete goals of the thesis. Finally, the outline of this paper and the contribution of each individual during this thesis project will be presented.

1.1 Background and problem motivation

The Internet has changed many aspect of our lives in such a way we had never imagined since it came to existence. It has greatly influenced how we communicate with each other, how we study, how we do business and our daily lives in general. With the great increase of smart phones and tablets in recent years accessing internet evolved from using only larger size computers to devices with the size of our hands. Smart devices has emerged with powerful networking and computing capabilities, from checking emails to online banking. As a reason of this in the recent years the number of users using smart devices has increased rapidly. The first smart phone was the Nokia Communicator in 1996 and the Apple’s iPhone came after 11 years in 2007 [2]. The worldwide smart phone users passed the 1 billion mark in 2012 and are expected to be 1.75 billion at the end of 2014 [3].

A rocket increase of smart devices in the global market has raised the issue of security and privacy because smart devices make use of third-party applications from online app markets. These applications are vulnerable to sophisticated malicious attacks, especially smart device applications related with online banking and which share sensitive information. To prevent any potential attacks different vendors has taken different measures to tackle the issue of smart devices security. The most common ways used are application market and platform protection mechanisms for security [4].
BankID [5] is the leading electronic identification system in Sweden which is developed by a number of large banks in the country to be used by the public, authorities and companies. Today, there are around 5 million people who use BankID for a variety of public and private services.

In 2011 Mobile BankID [6] was launched for secure electronic identification in mobile phones and tablet computers, it allows mobile users to verify themselves by using their personal numbers and to digitally sign important documents and contents such as transactions. It has become an important tool in a variety of mobile payment services, mobile bank transactions and other services.

This paper describes the process to integrate BankID into a mobile application, a mobile application based on the open source PhoneGap framework [7]. This paper will also discuss some security and responsibility aspect of how sensitive information are managed.

1.2 Overall aim

As of 2011 most Swedish companies rely on Mobile BankID security application to give a secure and reliable services to their users on smartphones and tablet computers. It has been mainly used for electronic identification and digital signatures in a variety of services such as online banking.

The overall aim of the thesis project is to integrate BankIDs services into a PhoneGap based app to be used in the area of medicine and personal healthcare to provide security and privacy for its users. In additional to this a security analysis will be performed, taking a few security aspects into consideration. Lastly a performance test will be conducted to assure that the implemented systems user friendliness and performance is satisfying.

The implementation of BankID includes research on how Public Key Infrastructure (PKI) works and in addition, to asses different possible ways of integrating the Mobile BankID security application with the existing PhoneGap framework. The performance test consists of the evaluation of the speed, reliability and user friendliness of the system. During the security analysis we take the potential security flaws in the system into consideration and suggested possible alternative solutions to avoid them or justify our choice of implementation.
1.3 Scope

This paper’s theoretical part presents a detailed explanation of how PKI, BankID and PhoneGap works in addition to some fundamental knowledge necessary for the thesis security analysis. The named theoretical parts will lay the foundation to the systems implementation and security analysis.

During the thesis project the necessary implementations needed to provide BankID based authentication and signing possibilities for Android mobile devices will be implemented together with an appropriate test app for testing purposes. The focus will be the authentication and signing, not the actual app.

In the performance evaluation of the system we will evaluate the accessibility speed, user friendliness and the reliability of the services provided by our system.

Since security is a very broad subject to cover and due to time limitation, we will restrict the analysis to secure front- to back- end communication, the application session management and information access management. Other security aspects are not considered in this paper.

1.4 Concrete and verifiable goals

The major goals of the thesis project are:

1. Extensive research on Public-key infrastructure and BankID services
2. Implementation of secure front- to back- end communication
3. Mobile BankID Service configuration and implementation
4. Developing a PhoneGap based mobile application prototype for Android platform for testing purposes
5. Develop a PhoneGap plugin that launches the Mobile BankID app
6. Conduct a performance test where our system performance and accessibility will be evaluated
7. Conduct a security analysis on the implemented system
1.5 Outline

Chapter 2, 3, 4 and 5 are theory chapters, 2 covers BankID in great detail, chapter 3 and 4 discuss PKI and PhoneGap in some detail and chapter 5 cover the not already mentioned subjects the security analysis discuss.

Chapter 6 discuss the methods used during the thesis to design, analyze, implement and test the implemented system.

Chapter 7 covers the design and implementation of the front- and back-end parts of the system.

Chapter 8 presents the result of the security analysis and performance test discussed in chapter 6.

Chapter 9 discusses the result and explore future works.

1.6 Contributions

The PhoneGap based Android application, the backend and frontend structure of the system has been designed and implemented by the authors. Lars Eggestig was mainly responsible for setting up the server, the database and configuration of the certificates. Mintesinot Wodajo has been mainly working on the frontend implementation of the system, developing the PhoneGap based Android application and the BankID plugin.

The report was written in the collaboration of the authors. Chapters 1 and 2 have been written by Mintesinot Wodajo and chapters 3, 4, 5 and 7 by Lars Eggestig. The remaining chapters of the report were written in cooperation of the authors.
2 Public Key Infrastructure (PKI)

Public key cryptography (also called asymmetric cryptography) is a security mechanism used for confidentiality, integrity and authentication. Unlike symmetric-key cryptography which uses the same key for both encryption and decryption, public-key encryption uses the concept of key-pairs: one for encryption and the other for decryption, which means that a message encrypted with the public key can only be decrypted with its private key pair. Due to the advantage it has over the symmetric-key encryption, public-key encryption is widely used for key distribution and digital signature.

The security threat a public key cryptography faces is the user can be given a forged public key by a perpetrator who claims to be a legitimate provider of the key. To prevent this treat public keys are validated using digital certificates. A digital certificate is an electronic document that identifies the identity of a person, an organization or a server and binds the identity with a public key which is used to encrypt message and sign information digitally. It contains among other things, the public-key of the owner, the identity of the public-key owner, and the expiry date of the certificate. A certificate will be signed and provided by a trusted third party called a certificate authority (CA).

To properly use public keys and certificates there must be a good infrastructure to manage them effectively. As defined in RFC 4949 (Internet Security Glossary), Public-key infrastructure (PKI) is the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography [8]. The main focus of PKI is to provide trust to its users and used as a base for other security services to be built on.

2.1 PKI Components and Functions

2.1.1 End Entity

An end entity is a device or end user that uses or gives support to the PKI services. It can be an end user, a device such as a server or router, or anything that
makes use of the public key. Initially an end entity must obtain its own certificate following a certain procedure to be able to enroll in a PKI system.

### 2.1.2 Digital Certificates

As mentioned earlier, certificates are used to digitally identify an individual, an organization or a server. The use of digital certificates in PKI is vital to bind the individual’s or the certificate owner’s identity with the pair of keys (public and private) which will be used later to encrypt and sign information. Digital certificates are mainly used to verify the identity of clients and server on the web and to ensure secure communication between them. In a secured client-server communication, the server identifies it's identity by presenting its certificate to the client and similarly, the client identifies it's identity by presenting its certificate to the server. The secure certificate exchange between the client and server is performed over SSL or TLS, secure transmission protocols, which we will briefly discuss in the next section. The following information are included in any digital certificate:

- Owner’s public key
- Owner’s name
- Expiration date of the certificate
- Serial number of the certificate
- Name of the organization that issued the certificate
- Digital certificate of the organization that issued the certificate

X.509 standard has become widely accepted for formatting public key digital certificates. X.509 certificates are commonly used in different network security applications such as IP Security, SSL, and S/MIME. The structure or format of X.509 certificate has been revised and updated through time and is now in its third version as shown in the Figure 2.1.
2.1.3 Certificate Authority

A certificate authority is one of the main components and trusted body of PKI. Its main responsibility is to issue, revoke, and manage digital certificates. It could be a trusted third party by the user community such as a government agency or financial institute. A trusted certificate authority generates a certificate and electronically signs it with the CAs private key and the public key user will have at least one public key from a CA that the user trusts. A CA have a repository which is used to store keys, certificates and CRLs to make it easily accessible to the end entities.

2.2 Secure Socket Layer (SSL)

With the ever growing of applications which run on the internet, security is one of the major issues designing any web based applications should address. Today, with the evolving of SSL virtual e-commerce transactions and confidential information exchanged over the internet has become more secure and reliable then ever before. SSL is a TCP based protocol which provides reliable, secure end-to-end communication for web based applications. It ensures that the communication between web clients and servers is more secure by providing secure authentication, confidentiality and integrity of information [10].

Today, SSL and its predecessor TLS (Transport Layer Secure), are the most widely used secure standards to transfer data over an open network or the internet.
2.2.1 SSL Architecture

SSL has two layers of protocol stack. In the bottom layer we have SSL Record Protocol and on the higher layer stack, SSL Handshake Protocol, SSL Change Cipher Protocol, and the Alert protocol are included, as illustrated in Figure 2.2.

![Figure 2.2 SSL Protocol Stack [10]](image)

Record Layer Protocol

Record Layer Protocol is mainly responsible for providing message authentication and confidentiality to SSL connection using symmetric encryption and Message Authentication Code (MAC). The first operation by the record layer protocol will be to break down the application message received into smaller blocks and then compress the size by not more than 1024 bytes. It then calculates the MAC by using a shared secret key from the compressed block and then attach the MAC to the compressed block of data. Finally, it encrypts the block of data with the attached MAC and append SSL record header to it [10]. The SSL Record Protocol operation is illustrated in Figure 2.3.

![Figure 2.3 SSL Record Protocol Operation [10]](image)
Change Cipher Spec Protocol

It consists of a single byte message with the value 1 and used to cause the waiting state of the connection to be copied to the current state and updates the cipher suite (set of cryptographic algorithms) to be used during the connection [10].

Alert Protocol

The alert protocol is responsible for the connected parties to exchange SSL-related alerts. The protocol has two sections: value 1 which is a warning and value 2 which is fatal and will immediately terminate the session.

Handshake Protocol

The handshake protocol is the most complex and important part of the SSL architecture which allows the communicating clients and servers to authenticate each other and to agree on which cryptographic algorithms to be used when sending data over SSL. It is made of 4 consecutive phases as illustrated in Figure 2.4.
Phase 1: The client first initiates the secure communication by sending a `client_hello` message with some security parameter suggestions the client would like to use to the server. Then the server will respond with `server_hello` message to inform the client that it has agreed with the proposed security parameters (cipher suites) and wants to proceed with the SSL handshaking process [10].

Phase 2: The server proceeds immediately by sending `certificate` message which contains the server’s public key certificate to the client. Then the server may send `server_key_exchange` message and the `certificate_request` message to request a certificate from the client. At the end of the second phase the server will send `server_hello_done` message to the client to inform that the server has finished the initial negotiation [10].
Phase 3: At the beginning of this phase the client verifies whether the server provides a valid certificate or not. The client sends certificate to the server if it is requested to do so and also sends client_key_exchange message by encrypting the session key with the servers public key [10].

Phase 4: In the final stage of the handshake process, first the client sends a change_cipher_spec message to tell the server to activate the negotiated cipher suite and then the client send a finished message using the agreed upon new cryptographic algorithm and keys. The server will then respond with change_cipher_spec message to tell the client that the server will abide to the security parameters agreement and finally it sends its own finished message. The handshaking operation will be brought to end and from now on the client-server communication is well secured [10].

2.2.2 HTTP Secure (HTTPS )

Hypertext Transfer Protocol Secure (HTTPS) has become a widely used and reliable standard for the exchange of private and confidential information securely over the internet which enables applications that run on the web to encrypt data and authenticate the client or the server.

The client makes the initiation to a secure SSL communication to the server and SSL handshake takes place to negotiate an SSL connection and then the client and server start to exchange HTTP data over the secure SSL communication channel.

Man-in-the-middle-attack

MITM attacks occurs when an attacker intercepts the communication between two parties, the sender and receiver, without their knowledge and listens to the traffic or possibly modify it. The implementation of SSL/TLS in HTTPS is mostly used as a one-way trust relationship in which only the server owns a certificate. In one-way trust relationship of SSL/TLS, only the client or the user verifies the secure connection. Because of this, an attacker substitutes the server certificate with his own and could be able to read all the communication between the client and server [16].
2.2.3 OpenSSL

OpenSSL [13] is a widely used standard open source library for the implementation of SSL/TLS. It can be freely used for both commercial and non-commercial products. The OpenSSL library provides the following important toolkits for developing a secure communication over the web [14]:

- SSL implementation toolkits
- Algorithms for symmetric key and public key cryptography
- Hash algorithms and message digests
- Pseudo random number generator
- Support use of certificate formats

Heartbleed

Recently it has been discovered that the popular OpenSSL cryptographic library has a programming mistake called the Heartbleed bug [15]. It is caused due to the implementation error of the OpenSSL Heartbeat Extension, missing bounds check in the handling of the TLS heartbeat extension which can expose up to 64kb of memory to a connected client or server. The bug is identified as a buffer-over read since it allows more data to be read than should be allowed. An attacker will be able to disclose sensitive information from a client or server that may include private keys and passwords. The bug occurs by a very simple programming mistake in a very small piece of code and can be fixed with a very simple fix in the code. Even if the bug has been fixed after its discovery, it can have exposed a large amount of private keys and other secret information.
3 BankID

BankID is the Swedish leading electronic identification service used by banks, websites and all kinds of mobile and computer applications to authenticate customers identity and signing of legally binding activities such as transactions. BankID uses a users own personal number as identification and a password acquired by the uses e-identification provider. The e-identification provider is normally a Bank or a trusted body, this is a big factor to why BankID is recognized as such a secure service. Another reason to why BankID is recognized as a secure way of identification is because of the way the authentication and signing processes are implemented as we will be discussing in this chapter [17].

3.1 Relying Party (RP)

RP or relying party is the company or service provider that make use of BankIDs services in their own application or on their own Website to authenticate users and have them sign contracts, transactions and other binding agreements. The RP do not only include a Web Server or server but also applications the RP provides to its users to consume its services. To use BankIDs services a BankID certificate is needed, one can acquire from the banks providing the BankID service. It is important to understand that the BankID system provides secure authentication and signing and that the security do not come for free or without requirements. The acquired certificate needs to be installed and the RP will have to implement its own security between actors in its own system.

In Figure 3.1 an example of how a simple BankID system works is illustrated, here the BankID service provider is the one providing authentication and signing services. The RP in this case contains a server that verifies a user's right to access its services and an interface in the form of an application. The RP can regulates who got the right to access its services and to what degree. This might be necessary even with BankID, it is after all only a tool to authenticate users identity, not regulate their right to access RP content. In the figure we also see a BankID application which is part of the BankID system but located on the users computer or mobile device [18].
3.2 **BankID Services**

We now know that RP uses BankIDs services but what are those services? In this section we will discuss what services BankID provides, how they work and how they are used.

3.2.1 **Web Services**

The services provided by BankID come in the form of SOAP based Web Services (WS). There are four different WS in total BankID provides:

**Authentication**

Authentication or Auth, is a service that provides the RP with a way to ensure the user's identity when he tries to access some RP resource. The authentication process is initiated by the user when he wishes to access some resources the RP does not wish unauthorized persons to access. The authentication call takes one argument called an AuthenticationRequestType. The AuthenticationRequestType contains XML elements with necessary information to start the authentication process. The return value can be either a error message or an OrderResponseType as shown in Figure 3.2, the later contains an so called autoStartToken used to activate the BankID application (section 3.3) on the client side and a or-
derRef which the RP uses to verify the users authentication progress as shown in Figure 3.3 and is discussed in greater detail shortly [18].

![Figure 3.2 OrderResponseType](image)

**Signing and FileSigning**

Signing is an important feature BankID provides that allows RP to legally bind a user with some kind of agreement. The process works much like the Authentication process with the exception that the RP not only want to authenticate the user but also have a need to bind him to some kind of agreement, for example a transaction. The Sign call takes just as Auth one argument of the type SignRequestType and contains the exact same information with the exception of the string to sign and an optional field containing a user message. The return type on the other hand is identical and used in the same way as in the authentication process.

FileSigning works the same way as Signing with the exception of a file link instead of a text string [18].

**Collect**

The last WS is Collect and used by RP to verify the progress on Auth, Sign and FileSign processes. The Collect WS take the orderRef in a string format as an argument, the return value is an error or a CollectResponseType. A CollectResponseType contains different information depending on what kind of status a process has.

A list with relevant conditions and their description can be seen below.

- **OUTSTANDING_TRANSACTION**
  The authentication process have been started but the client have not yet been informed this.

- **NO_CLIENT**
  Same meaning as OUTSTANDING_TRANSACTION but in this case the clients BankID is not available.

- **STARTED**
  The BankID application have been started with the autoStartToken but no valid ID have been inputted.
Integration of BankID Services in a PhoneGap Based Mobile Application
Lars Eggestig and Mintesinot Wodajo 2014-05-29

- USER_SIGN
  The BankID application have been successfully started with a valid ID

- COMPLETE
  The user have successfully entered his personal code, at this point the authentication process is completed.

Upon COMPLETE status the return value contains personal information such as name and IP address of the authenticated user [18].

**Authentication Example**

In this section we discuss in detail how an authentication process is performed by referring to the diagram in Figure 3.3. In the text we will refer to the current step by denoting it with number (1-12). The authentication process starts when a client request to access some content, in the example the user will request to login on the RPs Website (1). The RP will receive the request and asks the BankID server for an autoStartToken using the BankIDs Auth call with the users personal number in the argument (2). When the autoStartToken is received the return message also contains an orderRef that serves as a unique identification for the started authentication process and will be used by the server to verify the authentication status (3). Once the autoStartToken is received the RP will try to start up the clients BankID application to allow input of the users personal code (4, 5). If the BankID application successfully starts the user will be able to input the code and submit it for verification (6, 9, 10). Throughout the process the server will actively keep track on the authentication process by calling the WS Collect (7, 8) using the orderRef as an argument, the return value will contain the status of the authentication process. Once the BankID server receive the users code (10) and can verify if the user is indeed who he claim to be the RP will be informed (11) of the result with the next Collect request (7, 8). The user can now continue to login to the RPs Website assuming he got the right to do so [18].
3.2.2 BankID Error Management

We have mentioned that WS calls can return either a successful value such as an OrderResponseType or an error message. In this section we will discuss error messages and how they are managed in some detail.

When RP communicating with the BankID server there are plenty of things that can go wrong, to inform the RP of such events they provide documentation containing error codes that describes the error. Their documentation also contain guidelines for recommended actions for each possible failed scenario, such as if the users BankID application is missing or if an authentication process are already in progress.

There are roughly two types of error messages, one that describes errors on the client side and one that describe errors on the RP side. The first is something the user or client normally is informed of, such as a missing BankID app, the later does not concern the user and is used solely by the RP side, an example of RP error is if the WS calls contain information in an invalid format which would indicate an implementation error in one of the WS calls [18].
3.3 Mobile BankID Security Application

Mobile BankID Security App is the mobile version of the BankID electronic identification system which has a vital role for authentication and signing purpose in mobile payment services, mobile banks and variety of public services. First, the user has to install the Mobile BankID Security Application on his mobile device. The user must then request for a Mobile BankID from the bank. During authentication and signing process the RP systematically launches the Mobile Security App from the same device [17][18].
4 PhoneGap

PhoneGap is an open source framework used for cross-platform mobile application development. PhoneGap encapsulates Web technologies such as HTML5, JavaScript and CSS3 into a working mobile app and is based on the open source Apache Cordova engine [19]. The PhoneGap based app development process (Figure 3.1) starts out by creating a web view and importing necessary JS libraries. All code are then encapsulated using PhoneGap and shipped as a mobile app that is able to target multiple platforms.

Figure 3.1 PhoneGap overview [20]

4.1 Plugin Development

To extend the possibilities and provide a numerous functions that normal HTML and JavaScript can’t provide, plugins (Figure 3.1) are used and can be created. A plugin is created by adding the same functionality in all targeted mobile platforms native language, this way the same app can be used on different...
platforms because the PhoneGap framework will use the plugin for the mobile platform it is running on. An example would be if the app is planned to run on iOS and Android, the plugin needs to contain code in both Objective C and Java that do the same task. To make use of a plugin the cordova.exec is used, the call is done with JavaScript and just a single line of code is enough. The plugin call take five parameters that describes what method and class to use, the values the method will use and how to act if an error occur. Beside the fact that plugins can be created there are also libraries of both plugins integrated in PhoneGap and third party plugins for download. The plugins are not included inside the PhoneGap framework but more like an extension of the application [21].
5 Analysis Background Information

In this chapter all relevant information for the security analysis not already mentioned in previously theory chapters will be presented.

5.1 Sessions

A session can be used to identify the same user over multiple requests without maintaining connection. For example a user that connects multiple times to a Web server for Web resources will not maintain the same connection through the requests, by using a session it is possible to identify this user over multiple conversations. A session is identified by a session ID and linked together with information the developer wish to be stored in between connections. To store session ID cookies are often used, for example if the user wish to login on a Web site the session can store the user login information for the next session and cookies can holds the session ID. Next time the user connects the cookies will contain the session ID used to access the same session as last time [22]. If the developer so wish a session can be given a lifetime, for example the session can be set to terminate after a set time if no activity using that session occurs before the lifetime runs out [23].

5.2 Recommendations for Mobile Management

When consuming services providing sensitive information on a mobile device the developer are required to implement a system that can provide the information in a secure way but not all responsibility lie on the developers. The user can:

◆ Set password on the device
◆ Lock the device after use instead of waiting for the device to lock itself
◆ Keep in mind where the device is stored when not used
◆ Keep in mind who can use the device
All security measure above are in most cases up to the user to use, it depends on what kind of information we are dealing with. If the information the service provides are information about the user in question then the service provided can only recommend the user to use the above steps, if on the other hand we are dealing with for example sensitive company information then there might be regulations for how usage are to be conducted from mobile devices and enforce the user to make use of the named security steps to minimize the risk of unauthorized information access [24].

5.3 Database Management

There are many questions to consider when administering databases, the database needs to provide reliable information to the right users and at the same time prevent non authorized users to access or change the database content. To control who can access the database user accounts can be created with different access privileges. The DB administrator can change the privileges based on the users need so that each user can access the necessary resources but nothing more. By doing so the risk of corrupting DB data or accidentally performing some harmful action are kept to a minimum. There are many ways to set up user privileges and even more ways to regulate table access. Below is a list of different actions that can be set for accounts:

- UPDATE, update a database value
- SELECT, create a query
- DELETE, delete entry
- INSERT, add entry

In addition the administrator and similar ranked accounts can change or add tables. By mixing the listed actions an user account can be customized for each users need [25].

5.4 Physical Server Security

Today it is almost a given to make communication secure and to store data in a secure way but if the physical security is bad all the work on security implementations can be for nothing. Take for example a Web server that access a DB containing sensitive information, to access the DB a username and password are necessary and in many cases this information are hard coded in the server so that on request the DB is accesses for requested resources. This approach works
as long as unauthorized person can’t access the physical server location. If the server can be accesses in this way then both the passwords and username risks being exposed and misused. There are many ways to solve this problem, some of them more extreme than others. If we assume the information is sensitive but that none extreme measures are enough for its protection then we can list a few good ways to protect it: [26]

**Server room access control** - By only granting access to people that got a reason to access the server such as administrators, it is possible to reduce the risk of physical sabotage and access to the server, it is good practice to limit the access a user have to relevant information and locations only.

**Disabling access ports** - By disabling all but the relevant access means to a server the risk of intrusion is reduced in the case of unauthorized access to the server location.

**Look the server** - By looking the server a password is needed to access it, this reduces the risk of unauthorized access same as **disabling access ports**.
6 Methodology

In this chapter we will talk about how we approached the thesis problem, what kinds of tools we used and how we verified the security and performance of the finished implementation.

6.1 Development Process

During the development of the RP server, client and PhoneGap plugin an agile like development approach where used. The reasons for the chosen approach was that all system part depends to some degree on each other for testing purposes. We first defined the main objectives, split up the implementation of named objectives and then tested the results. This process were repeated like in the example below throughout the project.

6.2 Hardware

During the thesis we used an Android phone when testing the client app. The phone used where HTC One with Android version 4.4.2. As for the server we used a Acer laptop Intel Core i5-3230M 2.6GHz with 12GB RAM.
6.3 Software

6.3.1 Development Tools

For client and PhoneGap plugin development Android SDK manager v. 22.3 where used, a version of eclipse used for Android development. The server where developed in Net Beans version 7.4. For editing XML configuration files Notepad++ were used.

6.3.2 Graphic Implementation Tools

In the development of the thesis various diagrams and pictures the online UML editor Creately were used located at http://creately.com/.

6.3.3 Server Platform

The server used during the thesis were running on the open source server platform Tomcat 7. For testing and comparison purposes the server platform GlassFish 4 were also used as to see if any advantages were found with either server platform. When no advantages were found in either one for our server Tomcat was chosen.

6.3.4 SoapUI

The open source cross-platform testing solution SoapUI were used to troubleshoot BankID WS communication at the start of the thesis.

6.3.5 External Libraries

In this section all external libraries used during the thesis will be presented.

JQuery mobile 1.3.2 is the library used for creating responsive user interfaces for the test application developed.
Java database connectivity (JDBC) is the Oracle supported database connectivity library used to access MySQL databases. During the project the backend server have been using JDBC to communicate with the database setup for testing. The version used were Connector/J 5.1.30.

PhoneGap is the framework used to encapsulate HTML, CSS and JS in the frontend client. During the project PhoneGap version 2.9.0 was used.

JSON Simple 1.1 was used backend side to create and read JSON objects used in the communication between the front- and back- end.

6.3.6 BankID Test Mobile Application

The BankID system require the user to have a BankID application to interact with the BankID server, the BankID test mobile application is the app BankID provided for developers for testing purposes to interact with BankIDs test server and used during the system development.

6.3.7 Certificate Managing Tools

The built-in Java program Keytool were used to generate and extract certificates during the HTTPS and BankID communication setup.

6.4 Performance Test

From a user perspective security and performance are major concerns when consuming services such as those RP provides. The system needs to provide beside security which we will discuss in 4.5, reliable and user friendly services. In the performance test we will determine if the requirements and goals have been fulfilled.

6.4.1 System Requirement Determination Approach

In this section we discuss the system requirements and the approach used to determine if the requirements are met. As mentioned before, speed, reliability and user friendliness are major factors to take into consideration when implementing a system. The most relevant and interesting aspects we consider are:
R1 - **Accessibility and reliability:** the user must be able to access the services RP provides within a reasonable amount of time and receive a response. We will measure the access time to RPs services and discuss different scenarios the user might find himself in by using RPs services.

R2 - **User friendly interface:** the user will be able to consume the RPs services through a responsive and user friendly interface. The focus of this requirement will be to discuss what a user friendly interface is and what approaches are possible to take.

### 6.4.2 System Performance Evaluation

During system performance evaluation we aim to measure access times between the components of the system and the user friendliness of the application. The following times will be measured during R1:

- The time it takes for a request to be sent from the app to our server and get response.
- The time between requests and responses between the BankID server and RP server.

And the following values will be measured and considered when testing R2:

- Number of actions to perform a certain task
- The response time when navigation between individual user interface elements of the application.

Each objective where speed are measured will be measured five times on two different occasions, the average time will be used to increase the accuracy of the test. The time actions takes will be measured using timestamps, in both the front- and back-end part of the system. One timestamps at the start of each action and one when the action is complete, the start timestamp will then be subtracted from the end timestamp to get the actions time.

### 6.5 Security Analysis of the Systems Components

The aim of the analysis is to identify security weaknesses, find improvements and discuss if the current implementation is the best possible or if there are other more feasible alternatives.
Integration of BankID Services in a PhoneGap Based Mobile Application
Lars Eggestig and Mintesinot Wodajo 2014-05-29

In the implemented system there are multiple places where security is of high importance, the most interesting points within the scope of this thesis are the following:

- Front- to back- end communication
- Management of sensitive personal information
- Backend access management

In additional to the discussions each security risk found will be given a severity and frequency score where the frequency indicates how often it is likely to occur and where severity indicates how sever the risk is. Both the severity and frequency will be measured on a 1-4 scale. The multiplicative of the two scores will indicate the overall risk [27].

6.5.1 Analysis Objectives and Investigation Approach

This section specifies the analysis objectives and approach to reach a result. Note that in each scenario only specified objectives will be discussed, all others are considered to be optimized for this scenario.

O1 - Secure front- to back- end communication is a requirement because of the sensitive information being transmitted to the frontend from the backend to be displayed on the users own pages in the Android app. The objective will be to inspect the secure means of communication, not the information being transmitted. This will be done by examining the protocol used by the backend server and frontend client to transfer information and compare it to alternative ways of communication.

O2 - Device and session management are one of the most important aspect in the implemented system, a user will have his personal information transmitted and displayed in his device, this must be conducted in the best possible way so that the risk of information leak is reduced to a minimum with the system still being user friendly. The analysis of this objective will be done by first briefly researching the current regulations and tips on how sensitive information are to be managed and considered how much the RP can do to limit the risk of leaking information. Finally, we consider the user friendliness against the information security and to determine where to draw a line before the security implementation affect the users user experience too much.

O3 - Information access management must be considered to limit the risk of unauthorized access to the database information. We will be discussing
DB management and server access. This will be done by exploring how implementation of access restriction can be done and discussing the needs of physical server security measures.
7 Design

In this chapter we will discuss how the systems different parts have been designed and implemented.

7.1 RP System and Actor Communication

In this section we will in detail discuss the system design and how communication is performed between the systems actors.

7.1.1 System Use-cases

The first step in the design and implementation of the BankID authentication and signing system was to identify all different use-cases. When taking the thesis goals into consideration a few predetermined requirements exist:

- There must be a PhoneGap based mobile app by which the user accesses the RP services.
- A BankID certificate must be used to consume the BankIDs services
- Even with a valid BankID the user shall only be able to access information the user got the right to access
- The user needs to authenticate to access personal information through the BankID system
- The PhoneGap app will provide a Web view that can display requested information

7.1.2 System Overview

After considering the problem at hand it was decided to implement the RP in two parts, a user interface and a user service provider (Figure 7.1). The alternative was to have the user interface interact directly with the BankID service provider which turned out to be both impossible, because of the need of a cen-
Integration of BankID Services in a PhoneGap Based Mobile Application
Lars Eggestig and Mintesinot Wodajo
2014-05-29

Central access managing authority and expensive as discussed in chapter 3. With this solution the user interaction and BankID communication are separated into a PhoneGap based Android app and a central server managing BankID interaction and information access control.

Figure 7.1 System Layout

With system design determined a more detailed system layout was designed as shown in Figure 7.2. In the more detailed layout the information management and BankID interaction are handled by a Java servlet. The access control is also managed by the servlet together with the local database. The servlet and DB are discussed in greater detail in section 7.2. The user interface as mentioned is implemented as a PhoneGap app and to make authentication and signing possible it is necessary to have a BankID app located in the same device as the app (chapter 3).

Figure 7.2 Detailed System Layout
7.1.3 Front- to Back-end Communication

One of the greatest concerns within the system is security, to authenticate a user and make sure personal information are only available to the user in question. Another aspect to consider is the type of communication that will take place between the front- and back-end. With that in mind HTTPS was used to communicate between the front- and back-end (chapter 2.2.2). The reason being that HTTPS is a protocol used for secure stateless communication and stateless communication is just right for a system where few requests are made and a stateful connection unnecessary. If the system required a more intense communication between the front- and back-end then a stateful protocol such as WebSocket might be of greater interest to not repeat any handshake procedures and connection setups between requests.

JSON Based Communication

During front- to back-end communication it is necessary to communicate so both parties involved understand the transmitted messages. A communication protocol is needed so that messages sent is interpreted the same way at both ends. After exploring different approaches it was decided to communicate with a JSON based protocol. The reason being that JSON is a widely supported information exchange format used over HTTP and also supports exchanges of data structure, another reason is that JSON is a favored exchange format by Dewire. During communication different keywords are used and mapped to a value, values like autoStartToken (chapter 3.2.1) or personal information.

7.2 Backend Structure and Implementation

The backend or server from Figure 7.2 is an access and information managing entity that provides its users with requested information and other kinds of services such as signing (chapter 3.2.1). In this section the backend implementation and BankID communication configuration will be discussed in greater detail.

7.2.1 Backend Use-cases and Responsibilities

After determining the need of a central server in between the client and BankID server the question still remains how much of the functionality will be placed
on the server. After examining all fact the following scenarios were found to be of importance for the servers implementation:

- A user request personal information from the server
- A user is requesting to perform an action that require the users signature
- The server needs to perform access control of users requesting its services

Based on the listed scenarios a set of use-cases were identified, each of which represent a possible scenarios on user requests.

**Personal information request** - A user can at any time request to access personal information, the server need to verify the requesting user identity before granting the user access to requested information.

**Signing of legally binding agreement** - A user can request to perform actions that require his or her agreement, the server will provide a way for the user to sign such agreements using BankIDs services.

**Error management** - If the user is the cause of some kind of error during communication that concern the user, then the server shall inform the user of this by responding with an error code.

**User access management** - When requests from users are received the server must beside authenticate the user, verify the users right to access requested content.

### 7.2.2 Service Implementation

The implementation discussed in this section is based on the use-cases from the last section. The server, Figure 7.2, contains a database that stores users with the right to consume the RPs services and a Java servlet. The servlet is configured to communicate over HTTPS and sends one response on each request given. There are in total two different processes the user can initiate, authentication and signing, both require two requests to the server. One to initiate the process to receive the autoStartToken and orderRef (chapter 3) if the user is within the DB and one to fetch the requested content such as medical lists or just permission to login. The second request contains the orderRef from the first request and are used to start the collect calls, once a complete status is returned the server responds with requested content. The reason for two calls and not just one is because HTTP is implemented to respond once for every request.
Instead of the named responses it is also possible that the server returns an error message.

In Figure 7.3 an activity diagram is shown for how the backend processes incoming requests and in what cases a request is responded with a valid response instead of an error.

Figure 7.3 Backend Activity Diagram
7.2.3 Java Servlet and Class Structure

The server consists of five main parts, the servlet, DB communication, ServerDAO, BankID communication and Session manager as shown in Figure 7.4. The servlet is the servers interface that receive requests and send responses back. The DAO is responsible to combine the various functionality provided by all other server classes such as BankID and DB communication. The BankID communication is managed by four different classes, one error management class that checks received values from the BankID server for errors, one Call class that run the Collect method, one BankIDRequests class that are used to store token and order values and one WebServiceCalls class where the WS calls are performed. For sessions two classes are used, one to create and edit the session information and one to store the session to file between requests.

Figure 7.4 Server Class Structure

7.2.4 BankID Certificate Configuration

As discussed in chapter 3.1 a BankID certificate is needed to even start communicating with the BankID server, this section cover the configuration of that certificate.
The certificate comes in a .pfx file where its private and public key also are located (chapter 2). Three steps are necessary to perform in order to establish a connection with the BankID server:

1. Create a keystore with the extracted .pfx content in
2. Create a truststore with the BankID servers certificate
3. Configure the server to use the stores from 1 and 2

Both step 1 and 2 were done by creating a keystore using the java Keytool program and simply importing the content into them, one store for each certificate.

Step 3 was done by using the Java “System.setProperty()” command and specify the location and password for each keystore. The truststore is the store containing the BankID servers certificate, in our case the test Certificate found in BankIDs documentation named in chapter 3. The keystore contains the certificate received from the RP BankID provider. The “System.setProperty()” was done before calling the WS within the server.

7.2.5 HTTPS configuration

To configure the server to communicate over HTTPS three steps was necessary.

1. Create a server certificate
2. Enable HTTPS in tomcats server.xml file
3. Enable cross origin communication in tomcats web.xml file

Creating a certificate was done by generating a certificate using the Java Keytool program and add it to a keystore (.keystore in Figure 7.5). To enable HTTPS communication using the generated certificate the lines in Figure 7.5 was added to the server.xml file, here the keystoreFile is the keystore containing the certificates location and the keystorePass its password.

```
        maxThreads="150" scheme="https" secure="true"
        clientAuth="false" sslProtocol="TLS"
        keystoreFile="C:\Users\Lars\Documents\NetBeansProjects\RP\CertsAndKeys\keystore"
        keystorePass="query123"
        />
```

Figure 7.5 (a) HTTPS Configuration

To enable HTTPS communication from other addresses than localhost, cross origin needs to be set in tomcats web.xml file as shown in Figure 7.6.
7.2.6 Sessions

When users connect to the server and successfully authenticate a session is created. The session information is saved to file and mapped to a unique UUID generated id and returned to the user. A session can be resumed by sending the session id back to the server in the JSON string. As long as the session is maintained the user do not need to authenticate again, signing will still be done every time no mater if the session is alive or not.

7.3 Frontend User Interface

In this section the structure and implementation of the frontend part of the system will be discussed.

7.3.1 Frontend Overview

The frontend is built up by a BankID security app, a PhoneGap based mobile app and a PhoneGap plugin that handles the BankID security app start-up as shown in Figure 7.7. Most of the frontend structure was already predetermined by Dewire, that is, a plugin that starts up the BankID security app and a PhoneGap app to test the plugin with.
7.3.2 **Frontend use-cases**

Based on the predetermined requirements the following use-cases were identified or already existing:

**Request information** - An authorized user must be able to request his or other users personal information if this person has the right to do so.

**Authentication** - The app needs to provide its user with a way to start up the BankID app for authentication.

**Signing** - The app needs to provide its user with signing possibilities for any given content that needs to be signed provided by the RP.

7.3.3 **Mobile App Implementation**

Because the main focus is to implement the functionality for signing and authentication the rest of the app serves as a test platform and are of little importance to the final frontend implementation. The requests for information or signing are both sent to the server, if an authorized user performed the request the server returns an orderRef and a autoStartToken (chapter 3) for launching the BankID app. Once a successful request is done the orderRef is sent back to the server as discussed in chapter 3. To start the BankID app the Java plugin is called using the `cordova.exec()` function (chapter 4). On a successful first call and after the BankID app is launched the user inserts his password for the process to successfully complete.

The test mobile application implemented shows some important functionality such as authentication, signing a text string, and checking whether the test string signed or not.

7.3.4 **Local storage**

Local storage is used on the device to store session id between requests. Local storage maintains the id in memory even if the app is killed and are used instead of cookies due to the fact that PhoneGap has problems with cookies.
8 Results

In this chapter the result from the performance test and security analysis will be presented.

8.1 Performance Test

8.1.1 Accessibility Speed, R1

In Figure 8.1 the two tests done are presented with a lowest, highest and average time in seconds. The table present the average time to communicate between the app, server and BankID server.

<table>
<thead>
<tr>
<th>Test 1</th>
<th>Fastest</th>
<th>Slowest</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>App - Server - App</td>
<td>0.36</td>
<td>0.50</td>
<td>0.40</td>
</tr>
<tr>
<td>Server - BankID - Server</td>
<td>0.28</td>
<td>0.34</td>
<td>0.30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test 2</th>
<th>Fastest</th>
<th>Slowest</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>App - Server - App</td>
<td>0.34</td>
<td>0.50</td>
<td>0.41</td>
</tr>
<tr>
<td>Server - BankID - Server</td>
<td>0.29</td>
<td>0.35</td>
<td>0.32</td>
</tr>
</tbody>
</table>

Figure 8.1 Access times

8.1.2 User Friendliness, R2

The user friendliness is defined by number of actions a user needs to perform to complete a process or navigate to a local page. Number of actions are counted from the moment a user starts the log in process, an action is either button clicks or user input such as personal number input in the authentication process. Figure 8.2 show number of actions performed to navigate the apps pages and
complete authentication and signing processes. The second column represent number of actions done when no session is active and the third column representative number of actions performed with an active session.

<table>
<thead>
<tr>
<th>Location</th>
<th>Number of Actions With Authentication</th>
<th>Number of Actions Without Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>HomePage</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Check</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>signed</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Action</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authentication</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Signing</td>
<td>8</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 8.2 Actions performed

8.2 Security Analysis

In this section each of the security analysis objective, O1-O3 from chapter 6.5, will be discussed. It is important to note that each objective, O1-O3, will only contain discussions of relevant scenarios and consider other as secure. 8.2.1 will cover communication between the front- and back- end, 8.2.2 will cover how transmitted information and sessions are managed and finally, 8.2.3 will cover how information and certificates are managed on the server side.

8.2.1 Secure Frontend to Backend Communication, O1

Due to the sensitive information transfers between the front- and back- end, securing the channel between them is an important task. For a secure front- to back- end communication i.e. the communication between the application running on the user device and the backend server, we have implemented using HTTPS which is the most popular secure communication protocol over the internet.

The security threats we have discovered in our front- to back- end communication are:
Integration of BankID Services in a PhoneGap Based Mobile Application
Lars Eggestig and Mintesinot Wodajo 2014-05-29

- the use of unsigned certificate on our server
- the Heartbleed bug

**Man-in-the-middle Attacks and Unsigned Certificates**

Although, HTTPS is the most trusted secure communication protocol over the internet, there are still some security risks by using it. The most common attack on HTTPS is the man in the middle attack, chapter 2.1.2. This attack can lead to a serious information leak if successful. To prevent such attacks the use of certificate signed by a trusted certificate authority is recommended.

**Vulnerable to Heartbleed Bug**

The most recent security flaw discovered over HTTPS is Heartbleed, a bug in the OpenSSL encryption library, chapter 2.2.3. This bug allows attackers to obtain secret encryption keys leaving no trace of intrusion. To prevent such attacks it is recommended to use the fixed version of OpenSSL or disable the part that causes the bug in the source code and recompile before use.

**8.2.2 Device and Session Management, O2**

When sensitive information has been transmitted to a remote device many complications can occur, especially when the device is a mobile phone:

- The Phone can be stolen or lost during session and still contain access to sensitive information
- The Phone can be left behind for whatever reason
- The owner of the phone might lend the device to another person

In all scenarios above the user lose physical control of the device, there are many options from an implementation aspect to minimize the risk of information leak and ways to limit the time the information can be accessed but also how the device are managed by the user.

**User Sided Security Actions**

There are multiple ways for a user to prevent sensitive information from being accessed by others if the phone in anyway are out of physical contact as dis-
cussed in chapter 5.2. The easiest way would be to always lock the phone when not in use with a strong pin code and set the device to lock itself if not in use after a short duration.

Session Security Measures

Beside the security measures the user can take, the server can also reduce the risk of unauthorized access of the information in the phone as discussed in chapter 5.1. If a user successfully login and have access to his or her personal information and do not need to authenticate himself again to access the next page, and if the phone is lost right after, anyone would be able to access the information. This is very hard to prevent but the server can limit the damage done by setting a session lifespan so the login duration only lasts a set time. That would mean the user have to authenticate himself on a regular basis to keep accessing any sensitive content. Another way to prevent others from accessing any sensitive information is to have the user authenticate himself each time he navigate to a page containing sensitive information. Beside the two mentioned approaches it is also possible to set a short session lifespan that renew itself on user interaction.

Usability Considerations

There exist many alternatives to make a mobile app that contains sensitive information more secure and ways for the user to reduce the risk of information leak as discussed above but how would the security measures affect the apps usability?

Session lifespan as mentioned would require the user to authenticate himself regularly or connect using the session ID before the session timeout. The question remains what session type and session time to use to not inconvenience the user with repeated authentication requests.

After discussing the options with Dewire it was decided to use the approach where the user are kept logged in for a set period of time and where the time are reset after each request using that session ID. The reason being for the app to be as user friendly as possible and not require the user to constantly log in but still log out the user if no activity are done within a set time. The decided time became 10 min after considering the risk against the user friendliness. One drawback with this approach is that if a logged in device is lost then anyone who finds the phone within 10 min can have limitless access to its content for as long as the session ID is not reset on server side or the device locked.
8.2.3 Information Access, O3

It is said that a chain is never stronger than its weakest link, even if all possible security measures in O1-O2 have been taken the information located on the server and in the database needs to be secured as well or the entire system risks leaking information to unauthorized persons or provide false or corrupted information to users using the services. Two interesting topics to discuss is:

- Database privileges, who can access what in the DB
- Server access control, who got access to the servers physical location

Database Management

To limit database access to authorized users only DB accounts can be created from the database admins account. Each account can if set contain a key used to access the DB content, with this key the user can access the entire DB or just a limited part based on the account settings set by the admin as discussed in chapter 5.3. One major question when discussing DB accounts is what type of privileges each account have. By setting different privileges based on the users needs will reduce the risk of corrupting DB data and still provide the necessary functionality to the user. In our case there are two different kind of DB access:

- Control if user is authorized to access sensitive information
- Retrieve and change his or her personal information

Based on chapter 5.3 two different accounts are needed:

- Account 1
  - SELECT for authorization control
- Account 2
  - SELECT for viewing personal information
  - DELETE for deleting of information
  - UPDATE for changing information
  - INSERT to add information such as signed strings

Because any user can try and access the server it would be good practice to reduce the privileges to that of account 1 on authorization and only allow full access for authenticated users.
Server Accessibility

There are a lot of security measures to prevent information from being accessed or changed by unauthorized users, so far only security implementations in the system have been discussed but physical protection is of great importance as well. In our system there are two weak links hard-coded in the server:

Database Communication - In the server code the DB password and username is hard-coded to access the DB on user request.

Key- and trust- store access - The access to the key- and trust- store are just as the DB access information hard coded in the server source code.

This can in some cases be implemented in another way where the user provide the hard coded passwords instead. This way is not possible in our system because of the fact that users do not have the right to access keys or certificates located in the key- and trust- stores. Instead of implementing security measures inside the server it is possible to restrict the physical access to the server and DB as discussed in chapter 5.4. In chapter 5.4 some ways to implement a more physically secure server is discussed briefly. In our case no such implementation are used or needed because of the fact that no real sensitive information are used but it is important to consider for a real system implementation.

8.2.4 Probability and severity

In this section each scenario from 8.3 are given a severity and probability score from which the risk is calculated. For a discussion on the scoring see the conclusion chapter.

Figure 8.3 Risk score
Integration of BankID Services in a PhoneGap Based Mobile Application
Lars Eggestig and Mintesinot Wodajo
2014-05-29

The risks in Figure 8.3 were found during the analysis and given a frequency and severity score.

**Man in the Middle** - Due to the fact that it is highly likely that traffic between the client and server can be monitored but highly unlikely that it can be decoded the risk is low even though it can occur often.

**Heartbleed** - The highest risk found in the analysis due to the fact that information such as private keys can be obtained.

**Lost Device** - A user who lose control of the device risk revealing personal information, because of the fact that information regarding that user is the only thing leaked the severity is low.

**Physical Access** - Due to the fact that server access normally require a key to the server room and a password it is highly unlikely any unauthorized can get access. If an access attempt is successful the damage done can be huge.
9 Conclusions

9.1 Thesis Aim and Goals

The aim for this thesis was to do a study of BankID, PhoneGap and PKI, implement authentication and signing possibility in a PhoneGap based app, construct a backend that provides data and regulate user access and to perform a performance test and do a security analysis of the implemented system.

When reflecting on the implemented system, studies done, test and analysis performed with our goals in mind (chapter 1.4) we came to the following conclusions:

1. An extensive research on both PKI and BankID have been performed where the results are presented in the form of theory chapters, PKI in chapter 2 and BankID in chapter 3.

2. To conduct secure communication between front- and back- end with HTTPS has been configured on the systems server, chapter 7.1.3. Due to the fact that the current server certificate is unsigned additional measures are needed to ensure the security required for a commercial product, still the current implementation is enough to fulfill the goal of secure front- to back- end communication.

3. The backend server consumes BankIDs WSs and have been configured to use BankIDs test certificates, both BankIDs own and the RP certificate provided for developers, chapter 7.2.

4. To test the implemented system and the plugin a test app was created based on the PhoneGap framework, chapter 7.3.

5. In additional to the app a Java plugin for the Android platform has been implemented, chapter 7.3. The plugin can launch the BankID test app used in BankIDs signing and authentication processes. To expand the usages to other none Android platforms additional plugins can be created as discussed in chapter 9.5.3.
6. A performance test has been performed and described in chapter 6.4. The result the test gave are presented in chapter 8.1 where both accessibility and performance been measured.

7. To evaluate the security of the system an analysis has been performed (chapter 6.5) where the result gave both suggestions for improvements and important fact to consider when launching the system for commercial usages, chapter 8.2.

With this we consider all our goal fulfilled.

9.2 System Performance

There are many factors to take into consideration when evaluating systems performance. Due to our thesis scope we limited the area of which we evaluated the system to access times and user friendliness.

R1 - Access Time

The access times are simply the time it takes to send a request until an answer is received, both between the app-server and server-BankID server. The result from the times measured were satisfactory due to the fact that responses were received in a relative short time.

R2 - User Friendliness

Due to our app being designed for testing purposes the user friendliness were measured in number of actions only, not how understandable the interface is. The number of actions performed were measured in two ways, one way take authentication into consideration, that is for each process authentication is necessary, for example if the user are logged in for longer then the session lifespan a new authentication needs to be done to perform any action. The second way assumes an active session exists. The result indicates that when a session exists just the bare minimum of actions are necessary which is what was aimed for from the start. It is possible to reduce the number of actions performed by keeping personal numbers in memory but which would also increase the security risk if the device is lost and therefor not implemented.
9.3 Security Analysis Summary

Since security has been one of the main focus in this thesis, we have proposed suggested solutions in chapter 5.2. The feasibility of the proposed solutions has been evaluated independently.

O1- Secure Front- to Back- end Communication

Due to the confidentiality of the information being carried from front- to back-end and vice versa, securing the communication channel between them was a crucial task. To achieve this great demand we have chosen to implement HTTPS communication between front- and back-end and examined the current potential threats.

A potential Security flaw that we found implementing a secure front- to back-end communication is the vulnerability of HTTPS protocol to the man-in-the-middle attack and the recently discovered Heartbleed bug. To prevent the man-in-the-middle attack we have recommended the use of a trusted certificate which is signed by a trusted certificate authority on the server side so that an intruder can not make a fake certificate. The recent discovery of the Heartbleed bug in the OpenSSL library which the HTTPS protocol are based on might allow attackers to retrieve private key or even collect users important information. The use of the new fixed versions of OpenSSL encryption libraries is important to prevent any attacks which make use of the bug to perform malicious attacks.

O2- Device and Session Management

The main concern of O2 was how to make the mobile app more secure without losing its user friendliness, see 6.2.2. The Usability and simplicity of the mobile application is equally important as its security. The use of session management is recommended as a solution. The duration of a users idle time using the app was set to a maximum of 10 minutes. But we believe that the given session time doesn’t guarantee the security we want to achieve but also can not be decreased due to the user unfriendliness this would bring.

O3- Information Access

The other security concern to look into was the security of the information stored at the backend server and database and the risk of unauthorized access. Since the database stores sensitive information about the user, it is important to
protect the database from unauthorized access by setting different privilege accounts to the users. The other important consideration will be the server physical security. In the server there are sensitive information stored about the database communication and key- and trust- store access. The physical security of the server will be very important to consider in the actual implementation of the system in a real world environment.

### 9.3.1 Justification of Frequency and Severity Scores

During the analysis each security risk found was given a frequency and severity score. The scores were decided on after discussing what the risk of a security flaw being used and how severe this risk is. The man-in-the-middle attack is highly likely to occur but due to the fact that HTTPS encrypts data sent, the harm done would be minimal if none at all, hence the risk is low and no additional security measure needs to be taken other than acquiring a signed certificate. Heartbleed, the biggest risk in the system is considered a very dangerous threat but the frequency of it being exploited are considered low, still it is of highly importance that any component in the system that suffers from Heartbleed is updated or corrected. A lost device, that is when the user loses physical control of the phone is considered to happen very often on a regular basis, the risk this brings is minimal due to the fact that the information at risk only concern the user, no other user will suffer if the device is lost. Physical access is just as Heartbleed considered a very sever security risk. The frequency is considered small due to the security implementations normally used and due to common sense. We think a employee know when he do not possess access rights, we also think he would not tamper with the server even if given the chance unless this employee got malicious intentions.

### 9.4 BankID From a Company Perspective

BankID is a good solution for authentication and signing for companies to use in their services. There are many aspects to consider during BankID integration as we have discussed throughout this paper.

- How the central authentication server is implemented
- How signed material are stored
- How content are returned on user request
- How BankID and HTTPS certificates are used
The only real cost integrating BankID is the purchase of a BankID certificate which make BankID attractive to use.

9.5 Future works

The limited scope of this thesis leaves a lot of room for continued development and implementation. When considering the thesis goals and BankID theory chapter, four extensions come to our mind:

- FileSign
- Commercial app using implemented functions
- Platform expansion
- Increase the scale of use

9.5.1 FileSign Implementation

As mentioned in the BankID theory chapter one additional WS are available for the RP, the FileSign function. Because this WS was unnecessary for our work, it was never implemented. When implementing FileSign there is one interesting aspect to consider, how to identify signed and unsigned files. When signing text strings as in the implemented system the text strings can be stored easily in a DB and a simple query can see if the text string is signed or not. When working with files the question of how a program can determine if a file is signed or not comes to mind, as well as how the signed files are stored and where. We consider those two questions to be most relevant for continued development of BankIDs services.

9.5.2 App Development

During the thesis the development of an app using implemented functions have always been for testing purposes only. It has been mentioned that the work done are the foundation to an app to come that will provide services for customers to manage their medical accounts, in addition to the mentioned app it is possible to use our work for other apps where authentication and signing are of interest. When integrating our work into a commercial system it is very important to note that the certificate used for BankID communication is a test version and
not for commercial use, the same goes for the certificate used for front- and back-end communication, this certificate is self generated and not trusted, as mentioned in the security analysis a certificate signed by a trusted authority is recommended to be used instead.

9.5.3 Mobile Platforms Expansion

In the current implementation the BankID security app can only be started from the Android platform which make the test app unusable on other platforms. If the app is planned for other platforms beside Android a new PhoneGap plugin must be implemented for each targeted platforms by using the platforms native programming language. One interesting aspect of expanding the app is the rendering of the apps content, if the content looks the same or if something differs. This might be of interest when the apps design is created.

9.5.4 Expanding the System Size

This thesis do not consider the impact on the system if implemented on larger scale, for example if hundreds of users access the systems services simultaneously. Interesting subjects to consider when expanding the system is if the current server implementation can handle the increased traffic, if any problems will occur with the BankID server communication and if the current data storage is the optimal solution for large scale storage.

9.6 Social and Ethical Impact

Due to the fact that the system will be used in the area of medicine and personal health care, it is important to consider the system from a social and ethical perspective. The system that will be implemented by Dewire using our work as a base allows elderly people to manage their medical accounts and gives them the option to share account access with family members and healthcare personnel. Due to the information our system can provide about the individual user this kind of account access sharing bring many ethical and social questions such as:

- How the sharing process are conducted
9.6.1 Access Control

In the planned system account sharing is a major feature due to many elders poor condition and limited knowledge on smartphones, some might not even own a mobile device, much less a smartphone. Due to the information each account got access to the big question is how access to this account are conducted without violating the owners personal integrity. There are multiple ways to approach this problem, one is to implement a sharing process using BankIDs signing function to sign invites sent to other and have them sign with their own electronic identification when accepting the invite to share access control. The original owner of the account should be able to decide how long time a invite is valid and be able to cancel active invites in case something happens with the invited person or their device. For security reasons anyone sharing someones account should not be able to send additional invites or cancel other invites. Alternatively have the owner agree to the cancellation or new invite. There are many reasons for the named precautions, for one, there is no telling if an invited person will mistreat the access privileges granted, in which case the damage can be reduced to a minimum by limiting the privileges granted through access sharing.

9.6.2 Binding Agreements and Action Clarifications

Another important aspect to consider is how binding agreements are conducted and clarifications on how actions will affect the user. It is important that there are clear instructions and clarifications for the user when for example sharing account access. Without clear guidelines it is possible the user will misunderstand what the consequences will be from some action performed.
References


Integration of BankID Services in a PhoneGap Based Mobile Application
Lars Eggestig and Mintesinot Wodajo

5-1/bql/configuration/developers/user/guide/bql/SSL.pdf


[14] P. Chandra, M. Massier, J. Viega, Network Security with OpenSSL, O’Reilly,
2002.


Authenticated SSL-sessions”, Available:

2014-04-17

http://www.bankid.com/Global/wwwbankidcom/RP/Mobile_BankID_Relying_
Party_Guidelines%201_0_2.pdf
Retrieved: 2014-04-17

Retrieved: 2014-04-17

phonegap
Retrieved: 2014-17-04

[21] PhoneGap, “PhoneGap plugin howto and documentation”, Available:
http://docs.phonegap.com/en/3.4.0/guide_hybrid_plugins_index.md.html#Plugin
Development%20Guide
Retrieved: 2014-17-04

[22] Session, “What is a session”, Available: http://javapapers.com/servlet/explain-
the-methods-used-for-session-tracking/ Retrieved 2014-04-25

54


