Mobile Social Network Platform

Lei Sun
Abstract

Mobile Social Network Platform

Lei Sun

The SWiN project is an abbreviation for Social Wireless Network Secure Identification project and it primarily focuses on the security issues of social networks on mobile platforms. This master thesis is a part of the SWiN project from SICS (Swedish Institute of Computer Science) in cooperation with Ericsson and Sony.

In this thesis project, we have designed and implemented a social networking prototype called FriendFinder. This prototype integrates different security solutions such as SAML and GBA to test the performance of them.
**Contents**

Acknowledgements ....................................................................................................................................... 7

List of Abbreviations ........................................................................................................................................ 8

Chapter 1 Introduction .................................................................................................................................. 9
  1.1 Introduction of SWiN Project .............................................................................................................. 9
  1.2 Motivation of My Thesis ................................................................................................................... 10
  1.3 Scope of Thesis ...................................................................................................................................... 10
  1.4 Development Process Model ............................................................................................................ 11
  1.5 Thesis Organization ............................................................................................................................ 12

Chapter 2 Background .................................................................................................................................. 13
  2.1 FriendFinder ........................................................................................................................................... 13
  2.2 Location-based Social Networks ......................................................................................................... 13
  2.3 GBA ........................................................................................................................................................ 14
    2.3.1 Introduction of GBA ....................................................................................................................... 14
    2.3.2 Elements of GBA ........................................................................................................................... 15
    2.3.3 Work Flow of GBA ......................................................................................................................... 15
  2.4 Certificate & SAML .............................................................................................................................. 16
    2.4.1 Introduction of SAML ................................................................................................................. 17
    2.4.2 SAML Assertion ........................................................................................................................... 17

Chapter 3 Methods and Techniques Required ............................................................................................ 19
  3.1 Android ................................................................................................................................................ 19
  3.2 SQLite .................................................................................................................................................. 19
  3.3 MySQL ................................................................................................................................................ 20
  3.4 Glassfish .............................................................................................................................................. 20
  3.5 REST .................................................................................................................................................... 21
  3.6 Jersey ................................................................................................................................................... 21
  3.7 Maven .................................................................................................................................................. 21
  3.8 SVN ..................................................................................................................................................... 22

Chapter 4 Implementation ............................................................................................................................. 23
  4.1 Social Networking Module ............................................................................................................... 24
    4.1.1 Requirement Specification ............................................................................................................. 25
4.1.2 UI Design

4.1.3 FriendFinderLibs

4.1.4 Local Database Design

4.2 FriendFinder Module

4.3 Authentication Module

4.3.1 Introduction of MWSB

4.3.2 MWSB Enabler Architecture

4.3.3 Procedures of MWSB

4.3.4 Implement MWSB on FriendFinder Application

4.5 REST Implementation

Chapter 5 Conclusion and Future Work

Reference
Acknowledgements

I would like to thank my supervisor Ludwig Seitz for guiding my thesis study. I want to thank Christian Gehrmann for the support. In addition, I want to say thanks to my thesis reviewer Olle Eriksson and Justin Pearson for their guidance. Thanks to Oscar Ohlsson from Ericsson for help with GBA. Finally, I would like to thank everyone who offered me help during my thesis project.
**List of Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAA</td>
<td>Generic Authentication Architecture</td>
</tr>
<tr>
<td>GBA</td>
<td>Generic Bootstrapping Architecture</td>
</tr>
<tr>
<td>B-TID</td>
<td>Bootstrapping Transaction Identifier</td>
</tr>
<tr>
<td>IMPU</td>
<td>IP Multimedia Public Identity</td>
</tr>
<tr>
<td>HLR</td>
<td>Home Location Register</td>
</tr>
<tr>
<td>HSS</td>
<td>Home Subscriber Server</td>
</tr>
<tr>
<td>MSNP</td>
<td>Mobile Social Networking Provider</td>
</tr>
<tr>
<td>MWSB</td>
<td>Mobile Web Security Bootstrap</td>
</tr>
<tr>
<td>NAF</td>
<td>Network Application Function</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
</tbody>
</table>
Chapter 1 Introduction

The chapter describes the current state of the SWiN project as well as outlines the motivation and scope of this thesis.

1.1 Introduction of SWiN Project

The SWiN project is an abbreviation for Social Wireless Network Secure Identification project. SWiN primarily focuses on the security issues of social networks on mobile platforms. This master thesis is a part of the SWiN project from SICS (Swedish Institute of Computer Science) in cooperation with Ericsson and Sony.

From the end of the 1990s, the rapid increase of social networking services has had a profound effect on communication. Virtual socializing has become a major trend to keep in touch with friends, family, and even strangers. No matter where people are they can access social networks via their mobile phones. When people turn on their computers, the first thing that they do might be browsing social networking websites. It is no surprise that social networks play a very important role in our daily life.

In recent years, the social network is moving into mobile domain due to the rapid development of smartphones. Social networking providers start to offer new functions by incorporating mobile features. For example, due to the availability of GPS in smart mobile phones, social networks based on geographic location are becoming popular. For example, users can check into place and share their location with other users. It is also possible to search for bars or restaurants nearby. These exciting features provide a new kind of user experience; they also result in a higher risk of leaking privacy. Most users would probably not like strangers to know where they are and what they are doing. Consequently, how to protect integrity and confidentiality of personal data from illegal data collection becomes an important factor to improve the quality of social networking services [14].

Social networks adapted to mobile platform increases the possibility of leaking users’ private data in mobile phones. On the other hand, it also provides an alternative to protect personal data by integrating security features from the mobile phone. The SWiN project addresses the question of how to improve existing and upcoming mobile security technologies such as USIM, ISIM, GBA in order to enhance the mobile user’s security experience [1]. Besides this, it also proposes effective security solutions that guarantee confidentiality of the communication between clients without social networking provider interaction. For example, when users create a group and communicate directly with each other over NFC or Bluetooth, the communication among group members should be confidential and protected from leakage to unauthorized users.
So far, the project mainly concentrates on three issues and puts forward three related solutions. The first one is how the server authenticates users which is an essential requirement for secure service. The second one is how to preserve the privacy and personal integrity of the user in a mobile social network. The last one is to improve the Android security which is originally only based on a simple access control model. But this thesis only pays attention to the former two issues. Regarding the last one refer to Qing Huang’s thesis [2].

1.2 Motivation of My Thesis

Theoretical security solutions for mobile social networks have been proposed in the previous research of the SWiN project [1]. There is now a need for a social networking application that can integrate and test the proposed security solutions. The application needs support for common social networking features.

In the first part of the thesis, we will implement a social networking application that fulfills these requirements. The application will be called FriendFinder. It will contain a service provider and a client. The client will be running on the Android operating system.

In the second part of the thesis, we will integrate and evaluate the security solutions proposed by the SWiN project.

This application is not intended for commercial use. Therefore, it aims to provide basic social networking functionalities before pretty user interface and advanced features. The application will still be highly extensible and well documented.

1.3 Scope of Thesis

The scope of the thesis primarily consists of:

- Study the SWiN project’s background and understand the security problems it addresses and the solutions it proposes.

- Design and implement a social networking client in Android with typical social networking features.

- Design and implement a social networking server.

- Integrate and validate the security mechanisms from the SWiN project in the social networking client/server.
1.4 Development Process Model

We adopted a sequential design process, the waterfall model, to develop this software. For each stage, we made a concrete plan with time schedule to ensure that we were able to complete the software with high quality on time. The details of the process are described below.

1. Requirements

The first essential stage was to set the requirements of the software. In order to draw up a reasonable and comprehensive requirement specification, background investigation was our main task in the first month. We concentrated on understanding the background of the SWiN project and figuring out the issues that the SWiN project has addressed and solutions it has introduced.

Because the system is used as a demonstrator for validating security solutions, it is designed to provide enough social networking scenarios that are needed for testing security strategies. In this stage, we have conceived all use cases and finished a strict requirement specification.

2. Design

The following month, we started designing the system architecture and created a system model in UML (Unified Modeling Language).

3. Implementation

The Implementation took us approximately 4 months. It was divided into two parts: the server and the client. The main task in this stage was to implement the code in each side and to follow the previous design. Moreover, both the client and server were shipped with the security solutions from SICS and Ericsson Lab.

4. Verification

During the implementation phase, we debugged the system in parallel. For each functional increment, our supervisor tested the system and gave us feedback. At this stage, the system was moved from development environment into the real environment to test if it satisfied the demands. We moved the finished software from emulator into mobile phone to test it in reality.
5. Maintenance

After implementation and verification, the last task was to write related documentation and fixed newly found defects. At the same time, we also started to write the report of thesis.

1.5 Thesis Organization

Chapter 1 is the introduction of this thesis which includes background and motivation. Chapter 2 covers more details of the SWiN project. Chapter 3 describes the methods and techniques which were used in implementation and the reasons why we decided to use them. Chapter 4 illustrates the details of the implementation. I was mostly responsible for the client part so my statement will focus on the client side. Chapter 5, the last chapter of this report, makes a conclusion of the whole thesis work and proposes future work.
Chapter 2 Background

This chapter introduces the background of FriendFinder which contains its design concept, initial intention and security mechanisms from the SWiN project.

2.1 FriendFinder

We implemented a social networking application called FriendFinder which is regarded as a test platform for security solutions. The main feature in FriendFinder is like the name suggests to find the location of your friends. This category of service is called location-based social networking service. It allows users to send messages and share their location by connecting with a server or direct communication such as WIFI, NFC or Bluetooth.

2.2 Location-based Social Networks

In the beginning, location-aware mobile applications allowed mobile phone users to view the current location of their friends. This kind of location-based service merely shared the location among friends. It didn’t provide any functionality for users to interact with the environment. An example of an application from this time is Google Latitude, where a user's cell phone location was displayed on Google Maps. Other users can see the location instantaneously. In order to preserve privacy, a user can specify a group of people that are allowed to see the location. It is also possible to turn off the service completely. Moreover, a user also can control the accuracy of what each of the other users are authorized to see [15].

The next generation of location-based social networking services expands this idea and allows users to interact with their environment through mobile devices. Foursquare is a location-based social networking website for mobile devices. Users "check in" at venues using a mobile website, text messaging or a device-specific application by selecting from a list of venues the application locates nearby [4]. Then they could recommend restaurants, pubs, and other places around “check in” venues or check the recommended places by other users. The second generation of location-based service discloses more sensitive user data, not only the location. It might also reveal a user’s habits, customs, characteristics, personalities and such kind of information. With bringing new fantastic functionalities, it also has increased security requirements.

The data could be manipulated to reconstruct a real person’s life easily. A mobile social
network user has to worry about being overseen or stalked by an adversary. Apart from this concern, an adversary can impersonate the user by falsifying or stealing private data from social networking providers. Another similar attack is “man-in-the-middle”. It is possible when the system lacks mutual authentication so that an attacker can impersonate each endpoint. The entire conversation is controlled by attacker and it is rather easy to gain the private information from victims.

The FriendFinder application supports direct communication which enables information exchange between clients by using short-range wireless technology such as Bluetooth or WIFI. For example, a user could establish a group and send invitation to friends by Bluetooth or WIFI. Then they can share locations among members in this group. In this scenario, the attacks mentioned before such as “man-in-the-middle” and spoofing are also existed. Additionally, eavesdropping should be considered as well. The eavesdropping issue may rise a reply attack. Besides this, there is another issue that has to be considered. A user in one social network will play a certain role with corresponding privilege such as group administrator, group owner or common member. It shall not be allowed that a common member pretend to be a group owner and proceed some activities which are only permitted by group owner. In direct communication by WIFI or Bluetooth scenario, how to distinguish the roles of each user and ensure a user is only allowed to perform authorized activities will be also addressed in this project.

This project will provide two authentication and identification strategies. One is used for interaction between social networking provider and users. Another one is used among social networking users themselves.

2.3 GBA

This technology uses the characteristic of mobile phones to implement authentication. It effectually prevents illegitimate users getting access to the system. The authentication mechanism will be launched when a user tries to log in.

2.3.1 Introduction of GBA

During the communication between a server and a client, it uses GBA [5] (Generic Bootstrapping Architecture) which is standardized at the 3GPP to authenticate a user.
The theory of GBA is based on a shared secret key which is integrated in the sim card and also in the HLR (Home Location Register)/HSS (Home Subscriber Server). Therefore, a prerequisite to launch GBA protocol in practice is that a user owns a valid identifier on HLR or HSS. GBA authenticates the sim card by making a network component challenge. The sim card will respond to the challenge and the response will be compared with the correct answer in HLR/HSS [5].

2.3.2 Elements of GBA

The actors in GBA are listed together with their main responsibilities:

**BSF:** Bootstrapping Server Function is used for mutual authentication between UEs and service providers [16].

**HSS:** Home Subscriber Server is a database that stores user profiles [17].

**SNP:** Social Networking Provider, it is a provider which offers relative social networking services.

**UE:** User Equipment, such as a mobile phone.

2.3.3 Work Flow of GBA

![Figure 2.3.3.1](image)
Figure 2.3.3.1 shows the workflow of GBA. Below are more detailed steps that explain how an untrusted UE is authenticated through GBA:

1. An untrusted UE tries to access a social network, social networking provider refers the UE to BSF.

2. UE and BSF are mutual authenticated with each other and bootstrap GBA.

3. BSF queries HSS to retrieve the UE’s profile including a secret session key.

4. The UE uses the secret session key to answer the challenge from social networking provider.

5. Social networking provider connects with BSF to verify if the response is correct.

6. Depending the result of the verification, the social networking provider decides to let the UE access social network or not.

As a result of successful bootstrapping, a secret key is shared between the UE and the social networking provider. The shared key is only valid for a certain time period.

2.4 Certificate & SAML

The problem with authentication between an online social networking provider and a UE has been solved by integrating GBA. However, the authentication problem still remains for the case where two social networking users communicate directly with each other over WIFI or Bluetooth. We assume Internet connection is not available in this case. Our concerns focus on the mutual authentication among users. How could we carry out a reliable authentication? Before figuring this question out, it is essential to first find out what should be authenticated for a social networking user.

1. Ensure that a user is a valid member in the group.
2. Ensure that the right of one user is in accordance with the user’s role.

The first concern was solved by importing a certificate issued by a trusted CA. Users in social networks should possess a valid certificate locally, for example storing certificates in mobile phones. Therefore, users can prove who they are through loading
certificates to the service provider. In this project, the certificate uses the X.509 certificate standard.

For resolving the second concern, standard SAML has been engaged.

2.4.1 Introduction of SAML

SAML (Security Assertion Markup Language) has been selected to assist users to identify each other. It is a product of the OASIS Security Services Technical Committee which is a XML-based open standard for exchanging authentication and authorization data between security domains, that is, between an identity provider (a producer of assertions) and a service provider (a consumer of assertions) [6]. For example, in our case, one user acts as a WIFI sponsor to provide others a private channel to join. This user is regarded as a service provider and the others who tries to join is identity provider. An identity provider has to supply identification to the service provider. The identification is made up by SAML.

2.4.2 SAML Assertion

SAML defines XML-based assertions and protocols, bindings, and profiles [6]. In this project, we only consider using SAML assertion. The assertion contains statements and the service provider will make access-control decisions based on it. There are three sorts of statements: [6]

1. Authentication statements
2. Attribute statements
3. Authorization decision statements

Authentication statements assert to the service provider that the principle did indeed authenticate with the identity provider at a particular time using a particular method of authentication [6].

An attribute statement asserts that a subject is associated with certain attributes which is simply a name-value pair in common. The service provider makes access-control decisions relying on attribute [6].

An authorization decision statement asserts that a subject is permitted to perform action A on resource R given evidence E. The expressiveness of authorization decision statements in SAML is intentionally limited [6].
Here is the attribute statement that satisfies our requirements:

<saml:Assertion A>
    <Issuer> B </Issuer>
    <Subject> C </Subject>
    <AttributeStatement>
        <Attribute Name= " D ">
            <AttributeValue> E </AttributeValue>
        </Attribute>
    </AttributeStatement>
</saml:Assertion>

The fragment of above simple SAML attribute assertion represents that:

1. Assertion A is issued by Issuer B regarding Subject C.
2. Subject C owns an attribute called D and its value equals E.
Chapter 3 Methods and Techniques Required

This chapter describes the techniques which have been involved in this thesis. It gives a brief explanation of each technique and the reason that we selected them.

3.1 Android

Recently, the development of Android operating system is incredibly quick. Especially during 2012-2014, the speed that Android operating system updates is unprecedentedly. As an open-source software stack for mobile devices, it is free and simple to start developing new applications. Android SDK provides tools and well-documented APIs. Comparably, the prerequisites for iOS application development are more complicated. A minimally sufficient development environment for iOS application is a computer with Mac operating system. The programming languages for iOS are Object-C and Swift which are not as widely used as Java [18]. So we decided to use Java based on Eclipse platform to develop our FriendFinder application on Android.

Before developing an application using Android, we have to figure out which API level we prefer and ensure that our test device is compatible with it. Depending on the debugging devices we have, we selected API level 10. Its corresponding Android platform version is 2.3.4 [20].

Apart from Android technique, we have to consider which database is suitable to manage data on client side. Through investigation and study, we considered SQLite.

3.2 SQLite

The reasons that we chose SQLite are:
1. SQLite was shipped into Android operation system, in other words, we could use an SQL query to create or update database without proceeding any configurations or set up. It is the most important reason that we selected SQLite as a database management system in our client side. Android provides a package named android.database.sqlite which exposes strict classes for an application to create databases, delete databases and perform other common database management tasks. Android ships with the sqlite3 database tool in the tools/ folder. It is feasible to use this tool to browse or run SQL commands on the device. Only run by typing sqlite3 in a shell window [11].
2. The size of SQLite is very small about 275Kb [19]. It requires little memory to run as well. That is a predominant benefit as we develop a mobile phone application. In spite of the capacity of memory on mobile phone is larger and larger. But it is still an important feature if an application requires rather little memory to run. The speed of the application will be impressively enhanced and the user’s experience will be considerably improved at the same time.

3. SQLite also has bindings for dozens of programming languages including Java. We decided to use Java to develop our system and SQLite is compatible.

On the server side, we also have to make a decision which database management system we prefer to use. And in addition to the database, we also need to decide which application server to use in the MSNP. In the end, we selected MySQL for the server database and Glassfish for the application server framework.

3.3 MySQL

We choose MySQL to establish and manage our server side’s database. For more details about the server database refers my partner’s thesis [7].

3.4 Glassfish

Glassfish is an open source application server for Java EE platform. In this project, we use it to set up a social networking server for FriendFinder Application. Even though there are various application servers available nowadays, but we selected Glassfish among its comparisons. The reasons are:

1. Well organized documentation of Glassfish includes official documentation, tutorials, and various FAQs.
2. There is a Glassfish plugin available for Eclipse.
3. Glassfish also offers a GUI which is used to deploy and un-deploy applications. For beginners, it is easy to use GUI comparing with the command line tool.
4. We plan to use HTTPS between client and server in the end. The modification from HTTP to HTTPS is rather easy by changing the port from 8080 to 4848 in Glassfish.

From the architectural view, our MSNP is a RESTful web service. The next part will briefly introduce the concepts of REST.
3.5 REST

Representational state transfer (REST) is an architecture style for designing networked applications. Rather than using complex mechanisms such as CORBA, RPC or SOAP to connect between machines, simple HTTP is used to make calls between machines [10].

RESTful architecture is a client-server paradigm. The client sends requests to ask the server for a specified resource. All the information in RESTful context is categorized as different resources. Every resource possesses a global unique identifier which is referenced to the location of corresponding resource on server. A client could get the resource by accessing its URI. However, in fact, the client only obtains the representation of the requested resource. A resource might have multiple representations such as plain text, JSON, or even image [21]. For example, we could describe a car by writing a paragraph text or directly draw a picture. Whatever text or picture, both of them represent a car which could be regarded a real resource. The text and picture are the different representations of this car. To the real application, the client needs to understand the format of resource. Besides the global identifier of resource client has to provide to server, the request action is also essential to implement client’s request. The request actions are a set of verbs such as GET, POST, PUT, and DELETE in HTTP protocol. The resource identifier tells server which resource the client wants to access and the verbs let server figure out how to manipulate the resource based the request from client.

3.6 Jersey

A standard and portable JAX-RS API has been designed to simplify development of RESTful Web services and their clients in Java. It is a toolkit to help developing RESTful Web services that seamlessly support exposing your data in a variety of representation media types and abstract away the low-level details of the client-server communication. Jersey RESTful Web Services framework is open source, production quality, framework for developing RESTful Web Services in Java that provides support for JAX-RS APIs and serves as a JAX-RS (JSR 311 & JSR 339) Reference Implementation [13].

The reason that we select Jersey is our application is built on Glassfish and Jersey is aligned with Glassfish. Jersey is supported from Glassfish version v3.1.1. It is very easy to install Jersey in Glassfish by using the Update Tool which is also shipped with Glassfish.

3.7 Maven
Maven is a tool which primarily targets to build and manage Java-based project. Maven uses an XML file to describe the software project being built, its dependencies on other external modules and components, the build order, directories, and required plug-ins. It comes with pre-defined targets for performing certain well-defined tasks such as compilation of code and packaging [12]. The original motivation we selected to build a Maven project is that managing dependencies of Jersey without building technologies like Maven is complicated. Comparably, it only needs to add several dependencies into the POM file of Maven project. In addition, as long as one person of our team configure the dependencies and build the project successfully, other members don’t have to repeat the setup one more time.

3.8 SVN

Except all listed technologies which have adopted in our server and client side, we also used to Subversion to maintain current and historical versions of source code and documentations. Team members work with this project and submit their work into server repository. The modification of source code is consistent and recorded.
Chapter 4 Implementation

Android Application as Client

We developed an Android application called FriendFinder as the client in our social network which provides basic social networking functionalities. Since our target is not to produce a ready-for-market product, we paid less attention to the user interface design and usability issues.

Integrating the secure solutions was the most important part. The final goal of this Android application is to provide a platform which is capable of demonstrating the security research results from the SWiN project.

Use Case

FriendFinder is an example application for mobile social network which is utilized to demonstrate security building blocks from Ericsson Lab and SICS. Exhibiting and sharing locations among friends is its essential feature. FriendFinder application supports peer to peer mode as well as server-client mode. The following scenario states the basic functionalities.

Alice, Bob, and Ceca are FriendFinder application users who have already registered on server. At this moment, Alice and Bob are online so Alice and Bob are able to connect with mobile social network provider (MSNP). Alice creates a new group called SICS and then she invites Bob via an online invitation. Bob checks his invitation and finds out there is a new invitation from Alice. Bob accepts this invitation and consequently becomes a member of SICS group. Later, Alice meets Ceca who is not connecting with Internet because she is worried about high roaming charges. But Alice still wants to invite Ceca as a member in group SICS. Therefore, Alice sends Ceca through Bluetooth two electronic documents: invite, which clarifies Alice invites Ceca; her membership assertion of SICS group, which is a proof that Alice is authenticated to invite others into group SICS. Ceca receives both documents. Hereafter Ceca becomes a temporary member in group SICS since there is no record about her membership in the group SICS on the server (MSNP) yet. Both the online user and offline user are able to share their locations.

By using FriendFinder, the group members could keep track of each other’s position and communicate with each other. It is needed to establish a broadcast channel. So Bob is initiative to build a group channel to play a role as WIFI sponsor. The rest of members in the group SICS could join in the broadcast channel. Since Ceca has been invited when she was offline, as a
result, server does not know she is a member of group SICS, neither Bob. If Ceca requests to join this channel, she has to show Bob her certificate and invite which are necessary to prove she has been indeed invited by Alice previously, and Alice’s (the inviter’s) membership assertion which states Alice has right to invite others into the group SICS. After Bob verifies those files are valid, Ceca is allowed to enter the broadcast channel.

Later, when Ceca connects with the server, she could apply a permanent membership by sending request to MSNP with Alice’s membership assertion and invite. If neither of those files do not expire, MSNP will supplement Ceca in the group SICS in server database and hand out a membership assertion to Ceca. Then Ceca becomes a permanent member in the group SICS.

We have designed and implemented the social networking framework, but the FriendFinder’s functionalities such as establishing group channel, joining group channel are not included in the scope of this thesis.

**Client Architecture**

In general, three fundamental components constitute client side:
1. Social networking module.
2. FriendFinder module.
3. Authentication module.

A social networking module is a platform to delivery functionalities of social networks such as creating group, inviting friend and so on. The FriendFinder module is a social network application. This module allows a user to share location with friends. The purpose of authentication module is to handle the communication between server and client or client and client is confidential and secure.

In above three modules, social networking module and authentication module can be reused later for supporting other social networking applications like FriendFinder. So we separated them for improving the expandability and re-usability of the whole system.

**4.1 Social Networking Module**

This module provides several common social networking functionalities. In this section, we divided the development into four parts:

2. User Interface design.
3. FriendFinderLibs library.
4. Local database design.

4.1.1 Requirement Specification

This section describes requirement specification from different perspectives including the roles and states of a user.

4.1.1.1 Roles

Three different roles in each group have been defined with corresponding privileges. They are administrator, owner, and member. Owner is the person who creates the group, so an owner is granted supreme authority comparing with other two roles. Administrator comes next, and member is endowed least competence.

4.1.1.2 States

As our application is not only designed to support client-server mode but also works in peer to peer mode, the state of a user is sorted into two categories. When Internet connection is available and the client could connect with the server, the user is in online state. Otherwise, the user is in offline state. Changing state by user is not allowed, as the state is automatically set based on the availability of Internet connection. Furthermore, a user also has different ability based on the state since several operations are impossible to be performed when the user is offline.

4.1.1.3 Functional Requirements

Like most social network, ours contains functionalities such as register a new account, create new groups, invite friends, and etc.

Register Account

For a first time using, a user has to register a new account by given a legal username. This name is globally unique.

Log in Application
A registered username is the only requirement for login. Password is not required because GBA protocol has been applied for authenticating a user.

Create Group

As long as a user creates a group successfully, this user is immediately entitled as the owner of this group. Group name is globally unique also.

Obtain membership assertion

Membership assertion is a time-limited certificate issued by MSNP to declare a certain user is a legitimate member in one group. A user could require membership assertion from MSNP at online state. The requested membership assertion will be stored in the SD card in mobile devices. It could be used when the user tries to proof the membership in certain group to someone else who doesn’t connect with MSNP.

Delete Group

Only an owner of a group is authorized to delete the group. All other members will be dismissed and all information of this group will be erased on MSNP.

Quit Group

Except the owner, other members have permission to quit a group. When a user quits a group, information of the group will not be received anymore.

Delete Member

An owner of a group has rights to delete members in the group. MSNP will stop sending information regarding the group if the member is removed. However, in the offline mode, the member still could participant the group as a regular member until the membership assertion expires.

Modify Role

An owner of a group is empowered with changing roles for a user is the group. An owner can grant one member to be an administrator as well as revoke an administrator to be a member.
Invite Friend

An administrator and owner are able to invite new friends into their group. Because a user can be online or offline, there are two options for inviting friends, online invitation, and offline invitation. The online invitation is handled by MSNP. Server has responsibility to store the details of an invitation. An invitee must connect server to deal with an invitation. On the contrary, the offline invitation is not handled via MSNP. It is send by an inviter via Bluetooth to an inviter. Consequently, the invitee does not have to connect MSNP to retrieve the invitation.

Check Invitation

An online user can only check the invitations which have been received when the user was online. An offline user can only check invitations which have been received when the user is offline.

Accept / Deny Invitation

A user will become a member in a group as long as the user accepts an invitation. He will be able to retrieve this group’s information and check the locations of members in this group. If a user denies an invitation, the user will not be added into the group.

As it mentioned above, there are two kinds of invitation, offline invitation, and online invitation. If a user accepts online invitation, that means the user receives this invitation from MSNP. Thereby, there is a record to show that the user is a member in a group on MSNP. We call this record permanent membership. If a user accepts offline invitation, that means MSNP does not know the user becomes a member of a group. So the user only has a temporary membership.

Obtain Permanent Membership from Temporary Membership

After a user accepts an offline invitation, the user will get a temporary membership. It is possible to connect MSNP and apply a permanent membership by submitting inviter’s membership assertion and invite file to MSNP. MSNP will check validity of those two files. If both of them are valid, MSNP will generate a permanent membership for the user.

Generate Invite

If a user wants to invite a friend by offline invitation, the first thing needs to be done is generating an invite. Invite is only used in offline invitation for announcing who invites whom
into which group.

The Figure 4.1.1.3.1 illustrates corresponding rights for different roles. The smiley face icon means the role is allowed to do relevant operations.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Role</th>
<th>Owner</th>
<th>Administrator</th>
<th>Member</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete Group</td>
<td></td>
<td>😊</td>
<td></td>
<td>😊</td>
</tr>
<tr>
<td>Quit Group</td>
<td></td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>Invite Friend</td>
<td></td>
<td>😊</td>
<td>😊</td>
<td></td>
</tr>
<tr>
<td>Delete Member</td>
<td></td>
<td>😊</td>
<td>😊</td>
<td></td>
</tr>
<tr>
<td>Modify Role</td>
<td></td>
<td>😊</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.1.1.3.1

4.1.2 UI Design

This section talks about the design details of user interfaces.

**Login Activity**

Login activity is the first interface of this application. User has to input correct username to log in. For the user who might not be good at remembering username, we designed “remember me” option. As long as a user inputs username once and selects remember me, the username will be remembered by the client. Next time the user only has to input no more than first two letters of the username, then the system will list all names which starts with the same two letters. Login activity is shown by Figure 4.1.2.1.

**Register Activity**

For the first time to use FriendFinder application, a user needs to register a new account with a legal username. A legal username should be made up by alphabet including uppercase letters and lowercase letters ‘A-z’, numbers ‘0-9’, and underscore ‘_’. Register activity is shown by Figure 4.1.2.2.
Group List Activity

After login successfully, the user is led to a new activity to display all the groups the current user has joined in and his/her roles in each group. The groups are categorized into two sorts: permanent group and temporary group. Permanent group indicates the current user has a permanent membership of the group, in other words, the user possesses membership assertion which is published by MSNP. The temporary group includes the groups which the current user has joined in offline state.

The invite and inviter’s membership assertion are the only reference to prove a user is a legal member in one group. However, an offline invitation is time limited. That is why the group list is called temporary group.

In this activity, it also presents state of current user. This UI is different according to Internet connection. When Internet connection is available, it displays as figure 4.1.2.3. Otherwise, it is figure 4.1.2.4. That is because several operations are not possible to perform when a user is offline. Comparing with figure 4.1.2.3, figure 4.1.2.4 lacks a create group button.
Create Group Activity

A user can create a group by submitting a group name to MSNP. MSNP will check whether the group name exists. If the group name does not exist, it will generate a group key for this new group and return it to the user. It also creates a group profile in the server database to reserve information of the group.
Member List Activity

By clicking one group’s name, it jumps into another activity which shows members and their roles in the selected group. Members are sorted into registered members and unregistered members. Registered members are the members which seize membership assertion from MSNP. Unregistered members are the members have been invited through offline invitations.

The privilege is rather different depending on the state of a user and the role, as a result, the components in this activity are variable. Six pictures below illustrate details.

An online owner of a group qualifies to delete the group. If the owner clicks delete group button, an alert dialog will pop up to double check if the owner wants to delete the group. With selecting yes, all the information of the group will be crossed out in the server database.

Another rights an online owner of a group has is to change a member’s role in the group.

In conclusion, an owner of a group with online state is admitted to change others’ roles, remove members and delete entire group. Quitting group is acknowledged by online member and an administrator.
Figure 4.1.2.7(Online mode) Figure 4.1.2.8(Offline mode)

Figure 4.1.2.9(Online mode) Figure 4.1.2.10(Online mode)
An online user could choose using online invitation or offline invitation, but an offline user can
only use offline invitation. Figure 4.1.2.15 and Figure 4.1.2.16 shows the different interfaces when a user is in different state. Clicking online invitation will connect with MSNP and MSNP will handle the rest of procedures. Clicking offline invitation will start Bluetooth device in mobile device and transfer membership assertion and invite. Figure 4.1.2.17 displays this situation.

Check Invite Activity

The process of checking offline invitations and online invitations is entirely different. As it mentioned before, the online invitation is sent from server. So a user must connect with MSNP to check online invitations. The offline invitation is sent via Bluetooth and it is stored in a local database in mobile devices.

A user is able to accept or deny online invitations. However, offline invitations can only be accepted. An offline invitation is comprised of inviter’s membership assertion and invite. The inviter sends these two files via Bluetooth to invitee as an offline invitation. If the invitee doesn’t want to join a group, these files should not be accepted. As long as a user accepts them, the user is authorized as a member of a group. That is why a user can only accept offline invitation. This will not be a problem in practice because the valid maximum distance of Bluetooth is about 10-20 meters in average, hence the inviter could ask invitee face to face if he/she wants to join in a new group in advance.

Figure 4.1.2.18, Figure 4.1.2.19 and Figure 4.1.2.20 are the user interfaces for checking invitations.
4.1.3 FriendFinderLibs

FriendFinderLibs is an extra Java library developed by SICS which manipulates certificates, membership assertion and invite for our application. It performs the following functions:

1. **Generate a PKCS#10 certificate request**

   In order to apply for certificate, a user has to generate a request in PKCS#10 format first and send it to MSNP for asking certificate of the user’s public key.

2. **Create an X.509 certificate from a PKCS#10 certificate request**

   The certificate structure follows the X.509 certificate standard for authentication and signatures [8]. The distinguished name in X.509 certificate content is the username adopted in FriendFinder application.

   However, due to the limitation of time, MSNP is provisional regarded as CA and its private key is taken to sign users’ certificates. In future work, CA could be divided from MSNP to keep the design module.

3. **Generate a self-signed certificate**

   To MSNP itself, it generates a self-signed certificate as a root certificate.

4. **Generates a new RSA key pair**
A user generates a new key pair in RSA algorithm involved public key and private key which will be used in generating certificate request and certificate.

5. Generate membership assertion

A membership assertion is necessary in offline mode to proof a user’s membership. The assertion is linked to the authentication certificate that the user obtained when registering at MSNP, and is signed by MSNP [8]. Thus a user holds a membership assertion and a certificate which can prove the user is a valid member of the group which specified in the assertion.

Figure 4.1.3.1 shows the SAML format of an example membership assertion.

```xml
<Assertion ID="117bffc93737ff6"
IssueInstant="2011-08-15T10:41:18Z">
   <Issuer>MSNP</Issuer>
   <Signature>...(by MSNP)...</Signature>
   <Subject>
      <NameID>Alice</NameID>
   </Subject>
   <Conditions NotOnOrAfter="2011-08-22T10:41:18Z"/>
   <AttributeStatement>
      <Attribute Name="group:Swin">
         <AttributeValue>administrator</AttributeValue>
      </Attribute>
   </AttributeStatement>
</Assertion>
```

Figure 4.1.3.1

The value of an issuer attribute is the organization who issues membership assertion. In this case, MSNP is responsible for publishing membership assertion to users. Certainly, it contains MSNP’s signature. The name of subject is the user’s name who requests membership assertion. It should keep consistently with the distinguished name in certificate. Notably, it represents membership and group-role as one attribute. The attribute name starts with keyword group and is appended with group name. The value of attribute is the role user plays in a group. Moreover, the period of validity is finite. We assume one week in our project. The above membership assertion example asserts that Alice is an administrator in group Swin issued by MSNP until 10:41:1 o’clock on 22nd, August 2011.
6. Generate an invite

Invite is a SAML assertion signed by inviter and it is generated by SICS SAML Attribute Assertion library. It displays who invites whom into which group. An Inviter’s user name is filled in the issuer filled of the SAML assertion and invitee’s user name is used to in the subject field [8]. It represents membership and group-role in same attribute.

Figure 4.1.3.2 shows the SAML format of an example invite:

```
<Assertion ID="567ffSec57318"
IssueInstant="2011-08-15T12:15:10Z">
<Issuer>Alice</saml:Issuer>
<Signature>...(by Alice)...</Signature>
<Subject>
  <NameID>Bob</NameID>
</Subject>
<Conditions NotOnOrAfter="2011-08-16T12:15:10Z"/>
<AttributeStatement>
  <Attribute Name="group:Swin">
   <AttributeValue>member</AttributeValue>
  </Attribute>
</AttributeStatement>
</Assertion>
```

Figure 4.1.3.2

We propose that an invite is valid for one day. During this day, Bob has right to participant group Swin like a regular member. The above invite announces that Alice invites Bob into group Swin and this invite will expire after 12:15:10 o’clock on 16th, August 2011.

7. Verify membership assertion & invite

In offline mode, membership assertion and invites are indispensable. When a user applies permanent membership from temporary ones, the user must deliver inviter’s membership assertion, invite, and own certificate to MSNP.

Additionally, when a user represents membership to others, the user also must show the invite, membership assertion which are received from inviter and own certificate.
FriendFinderLibs library is not only integrated in client side, but also in the server side to complement application functionalities. Among above eight performances, there are five of them used in client side containing item 2, 4, 6, 7.

4.1.4 Local Database Design

We designed one local database as well because Friend Finder application is supposed to be used even without Internet connection. When the Internet connection is not available, the local database has responsibility to provide data. When the Internet connection is available, the server database is in charge of data supplement, even though all data is actually retrieved from local database, however, the local database keeps synchronous with the server database all the time.

Structure of Local Database user_info

Table user_info is used to store a user’s information included the following contents:

<table>
<thead>
<tr>
<th>Column ID</th>
<th>Name</th>
<th>Type</th>
<th>Not Null</th>
<th>Default Value</th>
<th>Primary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>user_name</td>
<td>varchar</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>user_impu</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>user_cert</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>user_privatekey</td>
<td>varchar</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.1.4.1

0. user_name: username used by registration. It is globally unique. It is the same one with distinguished name in user’s certificate.

1. user_impu: user name on operator.

2. user_cert: path of user’s certificate document. When a user registers on MSNP, MSNP generates a user’s certificate with MSNP’s signature. The user can retrieve the certificate from MSNP and stores it on a mobile phone.

3. user_privatekey: user’s private key is written into a document and stored in SD card. This column is the path of a user’s private key document.

perm_group_info & temp_group_info
The perm_group_info and temp_group_info tables are actually in same structure except for the different names. Both of them are used to store the information of groups.

<table>
<thead>
<tr>
<th>Column ID</th>
<th>Name</th>
<th>Type</th>
<th>Not Null</th>
<th>Default Value</th>
<th>Primary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>group_name</td>
<td>varchar(255)</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>group_key</td>
<td>varchar(255)</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>update_interval_for_groupkey</td>
<td>String</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>next_schedule_update_groupkey</td>
<td>Date</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>update_interval_for_sessionkey</td>
<td>String</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>first_update_for_session_key</td>
<td>Date</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.1.4.2

0. **group_name**: the name is used when the owner creates the group.

1. **group_key**: the secret key of this group. It is generated by MSNP when the group is created. After a user joining in a group, the relevant group key is sent from MSNP to the user. This key helps members to find out a secure group channel.

2. **update_interval_for_group_key**: the time interval for updating group key. In initial plan, the group key varies over time to make it harder to track a group. But for now, we assume the time interval for updating group key is infinite. In other words, it won’t be updated.

3. **next_schedule_update_groupkey**: next time to update group key. This value is also to be set as infinite. Because in our case, we won’t update the group key.

4. **update_interval_for_sessionkey**: the time interval for updating session key. This session key is used to encrypt and unencrypted the messages which are transferred among members in one group. The session key is generated from group key by using RC4 algorithm.

5. **first_update_for_session_key**: the first time when the session key is updated. This is used to calculate the current using session key. More detailed information could be found chapter 5.7.1 in [8].

In our case, a user might be invited into a group when he/she is offline. He is not invited via MSNP, so MSNP does not have any record of this pending invitation. Even though he/she accepts received invitation later and joins into a new group. MSNP still cannot generate a membership assertion for this user in his/her new group due to lack of Internet connection. That means this user does not register a permanent membership assertion on MSNP for this group. So this group information should be stored into temp_group_info rather than perm_group_info.
**received_invite & send_out_invite**

The received_invite and send_out_invite tables are only applicable for offline invitations. The received_invite table is used by invitee to display the details of a received offline invitations. On the contrary, the send_out_invite table helps inviter to record a send-out offline invitation.

![Figure 4.1.4](image)

0. inviter: the username of inviter which is used in registration.

1. invitee: the username of the user who is invited.

2. group_name: the name of a group. This column shows which group a user is invited in.

3. invite: the path of invite document in SD card. Like user’s certificate and private key, we store the path of documents instead of substance.

4. membership_assertion: the path of membership assertion document.

5. certificate: the path of inviter’s certificate.

6. flag: flag displays whether an invitation is handled or not. When flag equals to “done” means the invitation is handled. Otherwise, it is not.

The reason that we do not delete information of an invitation after dealing with it is because this information could still be useful later. For example, when a user wants to turn from a temporary membership to a permanent one, the user must show invite and membership assertion, in this case, the information helps the user to find out the paths of these two documents.

**re_user_role_for_group & un_user_role_for_group**
The structure of these two tables are totally same and they display what kind of role a given user in certain group.

If a user joins a group by accepting offline invitation, this users’ information will be stored in un_user_role_for_group. On the other hand, re_user_for_group stores the user’s information whom joins the group through online invitation.

<table>
<thead>
<tr>
<th>Column ID</th>
<th>Name</th>
<th>Type</th>
<th>Not Null</th>
<th>Default Value</th>
<th>Primary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>group_name</td>
<td>varchar(255)</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>member_name</td>
<td>varchar(255)</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>role</td>
<td>varchar(20)</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.1.4.4

0. group_name: the name of a group used in creation.

1. member_name: the name of a user.

2. role: the user's role in a group. There are three roles: member, administrator, and owner.

4.2 FriendFinder Module

The target of FriendFinder application is to assist users to detect locations of their friends. This module is out of scope for this thesis. But in user interface design, we kept a map button for FriendFinder implementation in future.

4.3 Authentication Module

The Android client has been equipped with an authentication module named MWSB which has been developed by Ericsson to assure the security of communication between client and server.

4.3.1 Introduction of MWSB

The MWSB has been implemented following the 3rd Generation Partnership Project (3GPP) standard TS 33.220 "Generic Bootstrapping Architecture" (GBA) [9]. In our application, both the client and server side use the Mobile Web Security Bootstrap module. The basic feature of this module is to generate a secret key between the server and client to ensure confidentiality and integrity of the data sent between them.
4.3.2 MWSB Enabler Architecture

![Diagram of MWSB architecture](image)

Figure 4.3.2.1 [9]

The picture above shows the general architecture of MWSB. It is made up by four fundamental parts: Bootstrapping Server Function (BSF), Home Subscriber Server (HSS), GBA client and GBA NAF. BSF and HSS are servers which are hosted by Ericsson. GBA client and NAF run separately in client application and server application, but they are still regarded as a part of the MWSB enabler.

4.3.3 Procedures of MWSB

1. Mutual Authentication between BSF and User

Generally speaking, in this step, the user sets up the bootstrapping context to obtain a long term key (Ks) and a Bootstrapping Transaction IDentifier (B-TID).
Figure 4.3.3.1 elaborates the procedures. First, the Client Application requests the GBA Client to bootstrap. Then GBA client sends an HTTP request to BSF including a private IP Multimedia Private Identity (IMPI). BSF forwards an HTTP 401 message to demand GBA Client to authenticate itself. After that the SIM card verifies that the challenge comes from an authorized network and it returns a long term key Ks and a response RES. Next, GBA client sends another HTTP request, containing the Digest AKA response (calculated using RES) to BSF. BSF acridities the GBA client by verifying the Digest AKA response; keeps Ks and generates and returns an HTTP 200 OK message containing the Bootstrapping Transaction IDentifier (B-TID). At last, the GBA Client returns a B-TID to the Client Application [9]. For more detailed information, please see Ericsson lab page regarding GBA.

2. Build Secure Connection with Shared Session Key

When this step is accomplished, the server and client share a secret session key Ks_NAF which is used to authenticate a user in the application
The client side fetches Ks_NAF by sending a request to the GBA Client with B-TID and Ks_NAF. The GBA Client then generates Ks_NAF based on the B-TID and NAF_ID.

Meanwhile, the Client Application delivers a B-TID to the Server Application. The Server application requests GBA NAF to retrieve the application specific key. GBA NAF forwards the B-TID and its NAF_ID in an HTTP request to BSF. Afterward, BSF searches the bootstrapping context associated with the B-TID and returns Ks_NAF. GBA NAF returns Ks_NAF to the Server Application [9]. For more detailed information, please see Ericsson lab page regarding GBA. In the final result, both the client and the server sides share the same secret session key Ks_NAF.

The next chapter describes how the MWSB operates and how our application uses the shared secret key.

4.3.4 Implement MWSB on FriendFinder Application
Figure 4.3.4.1 displays the details of the MWSB implementation and how the shared secret key is used to verify the authenticity of the UE by digest access authentication.

Below is the detailed explanation of the 8 steps which are shown in above picture:

1. UE sets up the bootstrapping context and retrieves a B-TID and a Ks from BSF. Ks serves as a long-term key to obtain the session key Ks_NAF through communicating with the GBA client.

2. After bootstrapping, the UE derives the session key Ks_NAF based on the long term key Ks.

3. The UE sends a request to access one of the protected resources on the server with a B-TID which is used for MSNP to look up corresponding Ks_NAF in BSF.

4. MSNP fetches the session key Ks_NAF from BSF according to the B-TID which is provided by the UE. In fact, BSF not only returns a Ks_NAF to MSNP but also an IMPU value. From now on, MSNP and the UE share the same secret key Ks_NAF. In the following step, MSNP authenticates the UE by HTTP digest authentication using Ks_NAF.
5. The server replies the UE’s request with a 401 “Authentication Required” status.

6. The UE hashes the received none, Ks_NAF, and B-TID and replies MNSP with the hashed result.

7. At the same time, MSNP also runs the hash function and then compares with the result from the UE. If the two results are equal, it proves that the Ks_NAF in the UE is in the same that the MSNP holds. Consequently, the UE passes the validation and is able to access the sensitive resources on the server.

The following code performs GBA protocol on the client side to fetch the B-TID and the Ks_NAF for the current user based on the SIM card. Then it uses the B-TID and the Ks_NAF to proceed with the HTTP digest authentication.

Precondition

In this case, we use a software SIM card to replace the real one. It is necessary to offer IMPI and secret key values to simulate a SIM card. So we have to register for an HSS user account on Ericsson Lab web page to get three parameters which are IMPU, IMPI, and K [9].

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPI</td>
<td>IP Multimedia Private Identity</td>
</tr>
<tr>
<td>IMPU</td>
<td>IP Multimedia Public Identity</td>
</tr>
<tr>
<td>K</td>
<td>Permanent shared secret key</td>
</tr>
</tbody>
</table>

Source Code

// The address of BSF server
String bsfUrl = "http://bsf.labs.ericsson.net:8080/bsfv2/bsf";

// Since we establish the MSNP server locally so far, the NAF_Fqdn is set as localhost
String nafFqdn = "localhost";

// Parameters received at HSS user registration.
String impi = "SunLei@labs.ericsson.net";
String secretKey = "e067a8b52100fc149b913d4458f4b9a5";
// Simulate a new software SIM card
Sim sim = new Sim(impI, secretKey);

// Initialize a GBA client
GbaClient gbaClient = new GbaClient(bsfUrl, sim);

// Start bootstrapping to obtain B-TID and Ks_NAF
BootstrappingContext bc = gbaClient.bootstrap();
byte[] KsNaf = bc.getKsNaf(nafFqdn);
String password = Util.encodeBase64(KsNaf); // Ks_NAF
String username = bc.getBtid(); // B-TID

// HTTP digest authentication

// Create a credential with username and password
// The username is B-TID, password is Ks_NAF
UsernamePasswordCredentials upc = new UsernamePasswordCredentials(username, password);

// Set the scope of authentication
AuthScope as = new AuthScope(null, -1);

// Generate an authentication provider based on the scope of authentication and
// credential
BasicCredentialsProvider bcp = new BasicCredentialsProvider();
bcp.setCredentials(as, upc);

// Initialize a new HTTP client and set credential provider for this HTTP
client DefaultHttpclient client = new DefaultHttpClient();
client.setCredentialsProvider(bcp);

If we try to access resources by the browser directly, it also demands to proceed with digest
access authentication. The extra benefit to apply GBA is to prohibit a malicious attacker from
obtaining secret resources by a web browser. The following picture illustrates how it looks like
when we try to access a protected resource by a browser. For example, we try to retrieve
Alice’s information on the server. The URL of this resource is
http://ServerIP/SwinServer/user/Alice. Once we input this URL and try to access the content of
the resource, a dialog pops up to require username and password.
For offering social networking services, a RESTful web service as MSNP has been established. As introduced in the previous section, data on the server side is regarded as different types of resources. Every resource possesses a global identifier which is a named URL to indicate the content of the resource. In our case, we conceived eight types of resources and correlated URLs. Each URL has implemented a portion of common HTTP methods contained PUT, POST, DELETE, and GET.

The POST method is used to request that the origin server accepts the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line [3]. For example, a user could create a new subordinate of the resource on the server through HTTP POST method. There is a resource named “user information” which is used to store the detailed information for different FriendFinder users and the link to this resource is http://ServerIP/SwinServer/user. Through POST method, user Alice, we assume, has the capability to create her own information under the root user resource. Her information will be saved in http://ServerIP/SwinServer/user/Alice.

The PUT method requests that the enclosed entity be stored under the supplied Request-URI. If the Request-URI refers to an already existing resource, the enclosed entity should be considered as a modified version of the one residing on the origin server. If the Request-URI does not point to an existing resource, and that the URI is capable of being defined as a new resource by the requesting user agent, the origin server can create the resource with that URI [3]. More precisely, the PUT method is only used to modify an existing resource in our case. For instance, if a user of the FriendFinder application wants to change his/her role from
administrator to regular member, he/she can use the PUT method to access the URL of his/her role resource. If the resource exists, the PUT method will modify the content of the resource, in other words, changing the user’s role on the server.

The DELETE method requests that the origin server deletes the resource identified by the Request-URI [3]. In the FriendFinder application scenario, if a group is deleted by its owner, the resource of this group on the server also should be removed.

The GET method means retrieval of whatever information (in the form of an entity) is identified by the Request-URI [3]. It is the most common method in our application which helps the user to obtain the requested resource from the server.

In conclusion, a user could delete, modify, get or create a resource by sending HTTP request with suitable methods to MSNP. Furthermore, MSNP connects the server database, so the data in server database will be updated based on the user’s request.

Resource List:

1. Certificate;
   Root URL: http://ServerIP/SwinServer/cert
   This is the root URL of resources concerning users’ certificates. A user is able to require the certificate issued by MSNP through accessing this URL.

2. Group List;
   Root URL: http://ServerIP/SwinServer/grouplist
   The resource under this URL is the list of groups which a user has joined in. By accessing this URL, a user is able to check which groups the user joins in. MSNP will check server database to acquire all the groups this given username has joined in and return with an HTTP response.

3. Group Info;
   Root URL: http://ServerIP/SwinServer/group
   This URL shows the details of a group containing group name, group key, group session key and other valuable information. For fetching certain group's information, the user has to provide a group name in an HTTP request. In principle, the user who is not a member of a certain group is not allowed to obtain group information. MSNP will check the server database to ensure this. If the user has membership in this requested group, MSNP will reply with the group information. Otherwise, MSNP will reject the request.
4. Member List;
Root URL: http://ServerIP/SwinServer/memberlist
This URL will return a list of all the members in a group. The group name is given as an argument.

5. Role info;
Root URL: http://ServerIP/SwinServer/role
This URL will return the role of a user in a certain group. For instance, Alice can get her role in a group by providing her username and group name.

6. Membership Assertion;
Root URL: http://ServerIP/SwinServer/assertion
A user can obtain his/her membership assertion of a group by sending a request to this URL with username and group as arguments.

7. Invite info;
Root URL: http://ServerIP/SwinServer/invites
This is the repository for online pending invitations. Offline invitations are not handled via MSNP. For example, a user can acquire detailed information of an invitation by accessing this URL with the inviter’s name, invitee’s name and affected group’s name.

8. User info;
Root URL: http://ServerIP/SwinServer/user
This URL hosts the profile of users.

4.5 REST Implementation

The following part enumerates several fundamental operations and explains the implementation from the view of the server side and client side.

Registration
1. The UE sends an HTTP request with POST method to register a user in the MSNP.

Source code:

RestClient is a self-generated class which encapsulates HTTP methods including PUT, POST, GET, DELETE. Each time when UE tries to send an HTTP request, a new RestClient object will be initialized based on the URL UE asks to access.

```java
// Initialize a RestClient with the URL of the resource
RestClient postclient = new RestClient(rootURL + "user");

// Provide the username which UE requests to create on server
postclient.addParam("userName", userName);
postclient.execute(RequestMethod.POST);
```
2. Both UE and MSNP implement GBA protocol which has been explained in later section. As a result, UE retrieves the secret session key $K_{s_{NAF}}$ and B-TID; server fetches user’s IMPU and shared secret key $K_{s_{NAF}}$ from BSF.

3. MSNP authenticates UE using HTTP Digest authentication in order to determine that they share a same secret key $K_{s_{NAF}}$.

4. After mutual authentication is completed successfully, MSNP connects with server database to prepare to insert an information involved username and IMPU of new user.

5. Before inserting a new user in the server database, MSNP checks if given username has been registered or not. If the username exists in the server database, it will return “post new user failed”. Otherwise, MSNP will return “new user created successfully” message to UE. Of course, before sending registration request, the system will locally check the validity of given username. If the selected username contains illegal characters, the client will request the user to try again until a valid username is given.

6. The UE proceeds to generate a new key pair in RSA algorithm which is used to generate the certificate request.

7. The UE stores his/her private key in local SD card under “KeyDir” folder as an electronic document.

8. The UE generates a certificate request in PKS#10 format and sends it to MSNP for requiring signed certificate.

Source Code:
```java
RestClient postClient = new RestClient(rootURL + "cert");
postClient.addParam("certReq", CertGen.getString(certReq));
postClient.execute(RequestMethod.POST);
```

9. MSNP obtains the certificate request from the UE and fetches its private key from the server database. It uses the private key to sign the user’s certificate.

10. MSNP sends a reply to the UE with the signed certificate.

11. Finally, the client stores the certificate on the local SD card under the “Certificates” folder. The registration process is completed. Storing sensitive data such as certificates unencrypted in
the SD card is not secure. Due to lack of time, we save the documents of users’ certificates, membership assertion and Invites in the SD card. This could be improved in the future work by encrypting the sensitive data.

**Login**

No password is required to login due to the implementation of GBA. The login procedure is fully automatic. The figure below shows the details of GBA login procedure.

1. After the user inputs the username and tries to log in, the UE requests to access the user resource on the server with HTTP GET method. We suppose only the user has right to get or alter own information on the server. If the given username exists in the repository of user resources, it will return an HTTP response with the profile of given username. Otherwise, it will return a null value.

**Source Code:**

```java
// UE sends request to get given user’s information under user’s resources on server
RestClient getUser = new RestClient(rootURL + "user/" + userName);
getUser.execute(RequestMethod.GET);
```
2. As same as usual, the next step is carrying on GBA processes. It is notable here, MSNP does not only fetches Ks_NAF from BSF but also relative IMPU(1) according to B-TID. IMPU plays a profound role in login procedure.

3. MSNP accredits UE using HTTP digest authentication in order to determine that they share a same secret key Ks_NAF. Normally, if the mutual authentication between MSNP and UE succeed, UE will be allowed to derive requested resources. It is not enough to protect the confidentiality of user’s account because MSNP fetches relative Ks_NAF from BSF based on the B-TID which is sent from UE. Additionally, for UE, it is easy to get a valid B-TID and correctly relative Ks_NAF as long as possessing a SIM card which has been registered on BSF. In consequent, for example, Alice owns SIM card 1 and implements GBA protocol locally, she is able to get B-TID(Alice) and corresponding Ks_NAF(Alice). The same situation with Bob who has B-TID(Bob) and Ks_NAF(Bob). Once Alice sends her B-TID(Alice) to MSNP, MSNP could fetch relative Ks_NAF(Alice) from BSF since SIM card 1 has been registered on BSF. As Alice and server share with the same secret key Ks_NAF(Alice), Alice will be authenticated whatever which user resource she requested. In other words, Alice could enter resource of Bob’s information, so does Bob. That is a serious vulnerability of system. Binding SIM card and username can address this problem.

There is an assumption as a precondition: a SIM card is bound with a person. That means every person only can use his own SIM card in FriendFinder application and his/her SIM will not give to others. Every SIM card are belonging to a globally unique IMPU. So there comes a point that we could use IMPU to distinguish a user. Thereby, it could check if a user is trying to access the resource of his/her own information. As long as we remember the IMPU of a user when he/she registers in server database. Then we only have to compare IMPU values to ensure whether the same user who request resources with the one who has registered before. That is the reason that we insert username and his/her IMPU as well into server database when user registers.

4. Now back to login function, MSNP connects with server database to find out the IMPU (2) depended on the username.

5. MSNP compares IMPU (1) and IMPU (2) to check if they are same one or not. If they are same, it states the user who requests login in is the user whom he/she clarifies. Log in succeed. Otherwise, log in rejected.

Every time when MSNP gets the request from UE, both of them proceed bootstrapping first, then the authentication will be performed. Only when UE passed the accreditation, MNSP will
deals with UE’s initial request. Hence the explanation of bootstrapping and authenticating UE will be skipped in following operations. It mainly focuses on how MSNP handles UE’s request.

Create group:

1. The UE asks to create a new group on server by sending a POST to the group resource.

Source Code:

```java
RestClient postclient = new RestClient(rootURL + "group");
postclient.addParam("groupName", groupName);
postclient.addParam("groupOwner", userName);
postclient.execute(RequestMethod.POST);
```

2. MSNP receives this request and checks whether the given group name exists in the server database. If it doesn’t exist, MNSP will generate a group key for the new group. If the group name already exists, MSNP will request the UE to provide another group name.

3. MSNP inserts the newly created group information into the server database and promotes the UE to become the owner of the group.

4. MSNP sends back the information of the created group to the UE including the group key.
5. The UE obtains the group information and stores it in the local database.

**Obtain membership assertion**

When the group list is shown in the client GUI membership assertions will be downloaded from MSNP and stored in the SD card. If a membership assertion already exists in the SD card it will be updated. The membership assertions are saved under the “Membership” folder in the SD card.

Source Code:

```java
RestClient getClient = new RestClient(rootURL + "assertion/" + userName + "/" + groupName);
ggetClient.execute(RequestMethod.GET);
```

**Online invitation/Offline invitation**

Below is a simple scenario to explain the usage of an online invitation and an offline invitation. Alice invites Bob into her group SICS and Bob checks the invitation from Alice, then he accepts the invitation.

Scenario: Alice invites Bob by an online invitation.
1. Alice invites Bob by an online invitation. For creating a new pending invitation, Alice needs to send an HTTP POST request to the invitation resource on the server.

Source Code:
```java
RestClient postclient = new RestClient(rootURL + "invites");
postclient.addParam("inviteeName", friendName);
postclient.addParam("inviterName", userName);
postclient.addParam("groupName", groupName);
postclient.execute(RequestMethod.POST);
```

2. MSNP receives the request from Alice and checks if Bob is a registered user in MSNP. If he is not, Bob cannot be invited. If Bob is a registered user, MSNP inserts a new pending invitation record in the server database to indicate that Alice has invited Bob into the group SICS.

3. Then MSNP notifies Alice that the invitation was successfully created.

Source Code:

RestClient getClient = new RestClient(rootURL + "invites/list/" + userName);
getClient.execute(RequestMethod.GET);

5. MSNP do a lookup in the server database and finds there is a pending invitation for Bob from Alice.

6. Next MSNP returns Bob the details of the pending invitation.

7. Bob receives the pending invitation and he can select to accept or deny it. Whatever he selects, it will trigger MSNP to delete the pending invitation. Therefore, he accesses the URL of the invitation resource with the HTTP DELETE method.

Source Code:

RestClient deleteClient = new RestClient(rootURL + "invites/" + userName + "/" + inviterName + "/" + groupName);
deleteClient.execute(RequestMethod.DELETE);

8. MSNP gets Bob’s response and it erases the pending invitation from the server database. If Bob selects to accept this invitation, an HTTP POST request will be triggered and MSNP will add a new membership for Bob in the group SICS. Otherwise, MSNP will do nothing.

9. MSNP sends an HTTP response to Bob to tell him that invitation has been handled.

Scenarios: Alice invites Bob by an offline invitation.

Similar with an online invitation, only registered users could be invited.
1. Alice creates an Invite for Bob with her own signature. The Invite is a SAML assertion.

2. Apart from the Invite, Alice also send Bob her membership assertion and certificate which are stored in her phones local SD card.

3. The system merges all three documents into one to deliver it to Bobs phone via Bluetooth. When Bob receives the merged offline document, it will be saved in his SD card under “Bluetooth” folder as default. If he clicks “accept invite” button, the document will be parsed and divided back to three files: Alice’s certificate, Invite and membership assertion. The three files will be stored separately under three different folders “Certificates”, “Invites” and “Membership”. Additionally, the system will insert a pending invitation into the local database. The absolute path of those files will be written into the pending invitation table. Moreover, as the limitation of Android Bluetooth API, we merge the files into one as jpg format since it only supports to transfer files in mp3 and jpg format via Bluetooth. But after Bob receives them, the format of the files will be recovered.

4. Alice opens her Bluetooth device and searches for Bob’s Bluetooth device. An assumption
here is that Bob is nearby Alice because the effective distance of Bluetooth in average is about 10-20 meters.

5. After Alice’s and Bob’s Bluetooth devices are paired, Alice begins to transfer the file to Bob.

6. Bob accepts the file and stores it into SD card under “Bluetooth” folder. This is the default path for storing files which are received by Bluetooth.

7. Bob accepts the invitation. The merged document is divided into three documents and saved in the SD card under Membership, Certificates, Invites folders separately.

8. Bobs client will insert the received invitation in the local database. Since the invitation was accepted its status will be set to “done”. Meanwhile, the system also inserts a new membership in the local database to indicate that Bob is a member of the group SICS.

**Temporary membership turns to permanent membership**

![Diagram of the process](image)

**Figure 4.5.6**

1. The UE sends a request to join in a group as a permanent member. The user wants to replace the temporary membership for a permanent one. An HTTP request has to be sent to MSNP with the user’s certificate, Invite, inviter’s membership assertion and certificate.

2. MSNP verify the membership assertion and invite.
3. If the UE passes the verification, MSNP will insert a new membership for UE in the server database to show that the user is a permanent member of the requested group.

4. Then MSNP tells user that the application succeeded.
Chapter 5 Conclusion and Future Work

In this project, we have designed and implemented a mobile social network platform named FriendFinder with security enhancement by applying theoretical security solutions for mobile social networks such as GBA (mutual authentication in online mode) and SAML (mutual authentication in offline mode). The system consists of two components: a MSNP server and an Android application. The system supports the traditional online mode communication as well as the peer-to-peer mode communication. The peer-to-peer mode communication means that the FriendFinder application supports direct communication between the clients by using Bluetooth other than client-server paradigm.

Because the system is currently intended for experimental use, it only features basic social network functions such as group creation, invitation of friends, etc. The major aim of the project is to integrate and use GBA and SAML into the system.

GBA is used to authenticate the user when internet connection is available. Instead of using traditional ways of authentication with username and password, the GBA protocol makes it possible to authenticate a user by a SIM card. A shared secret key is integrated in both SIM card and Home Local Register or Home Subscriber Server. The FriendFinder server challenges an un-authenticated user and verifies that the given answer is same as the one which is calculated by the Home Local Register or Home Subscriber Server. One of the benefits of using GBA is that the user does not need to remember complicated usernames and passwords anymore. Especially nowadays a smartphone user has to remember more and more usernames and passwords for different applications. It is very common that people forget their passwords or just create very simple passwords that they can remember.

For the peer-to-peer mode, we have used a SAML assertion to help users to identify each other. An identity provider can transfer his/her SAML assertion to a service provider to identify them self. A service provider can make an access-control decision based on the content of the assertion. The offline invitation functionality uses a SAML assertion as well. A user can generate a SAML assertion for another user to allow them to join a group in some period of time.

The major purpose of this thesis is to develop an Android application and integrate with the security solutions. The FriendFinder application could be extended further in the future. An ad-hoc network could be added to provide communication when Internet is not available. Another feature that could be added is the map functionality that could let users in the same group share each other’s location on the map.
Besides the future in the functional area, there are several potential improvements in the security domain. Currently, the membership assertion, invite and user’s certificate is stored in unencrypted files in the SD card of the mobile phone. That is not a very secure solution. A future improvement could be to encrypt these files.

Another improvement that could be done is related to the invites, membership assertion and user’s certificate. All those documents are valid only during a specified time period. This solution causes one problem. For instance, if Alice was a member of the group SICS and she stored her membership assertion in her mobile phone. She could prove that she is a member of the group SICS by using this membership assertion before it expires. No matter whether the owner of this group removes her from this group or not. Unfortunately, this issue is inevitable since the assertions are stored in local mobile devices right now. This should be addressed in the future work.

In this project, we tried to discover and revolutionize a new possibility of authentication. We used the telecom techniques to achieve a new way of social networking authentication. In this experiment, we also found limitations and problems regarding the new approach. No matter if this new authentication strategy will be used in commercial products or not in the future, it was a very interesting and valuable investigation.
Reference


[8]: System Architecture & Secure Identity Management Design & Social Network Application Design (D2, D3, D5) SWiN Project, Ludwig Seitz SICS, Oscar Ohlsson Ericsson AB, Elena Kozhemyak SICS/KTH, October 6, 2011


Accesses [2016-11-20].
[15]: See where your friends are with Google Latitude, February 4, 2009, Official Google Blog
Accesses [2017-03-20].
[16]: BSF Bootstrapping Server Function. 3GPP Specification # 33.220.
[17]: HSS Home Subscriber Server. 3GPP Specification # 29.336.
[18]: Apple Developer API Reference. 
Accesses [2017-03-20].
[19]: About SQLite, SQLite.org.
Accesses [2017-03-20].
Accesses [2017-03-20].