A secure mobile social network

Wu Yucheng
Abstract

A secure mobile social network

Wu Yucheng

Mobile social networks are a new trend to improve the quality of our life, so they should not bring any extra burden to users. The users should not anymore be worried how to set up a secure complex password and memorize it to prove their identities. The Generic Bootstrapping Architecture is used in this thesis as a novel way of authentication. The GBA protocol allows to generate and share cryptographic keys between a mobile phone and third party service providers using the operator as intermediary. These keys are generated transparently without any user interaction. In a traditional mobile social network, an Internet connection is compulsory but we believe this should not be a barrier in the future. An ad-hoc network may be used to communicate in a private group of friends. Thus, users need to be capable of verifying membership assertions and generating offline invitations for new friends without a connection to the social network server. Our implementation uses SAML assertions to certify group memberships and offline invitations. Beside these security solutions, a real mobile social network has been implemented, using the REST framework for communication and databases for storing information both on the client and the social network server.
Acknowledgments

This thesis has involved many people's hard working, so it would not have succeeded without the entire team work. Firstly, the author would like to express her gratitude to the thesis supervisor Dr. Ludwig Seitz who provided invaluable assistance during the entire thesis implementation work. The entire thesis work has been accomplished as a peer work so the deepest gratitude also goes to the project team member Lei Sun for her excellent collaboration and humble attitude. Deepest gratitude also gives to Mr. Oscar Ohlsson from the security department of Ericsson, Dr. Christian Gehrmann from SICS, and thesis reviewer Dr. Olle Eriksson from Uppsala University. The author also likes to convey thanks to the SICS and Sony Ericsson for providing the financial means and the laboratory facilities. Beyond this, the author wishes to express her abundant gratitude and love to her families and friends. Only on families' and friends' continuous love and support, the author can concentrate on her study without any worries.
List of Figures & Tables:

Figure 3-2 The main actors in the architecture
Figure 4 Architecture of SWiN project
Figure 5-1 indicates the GBA protocol architecture.
Figure 5-2 MWSB enabler architecture
Figure 5-2-1 Setup Bootstrapping
Figure 5-2-2 Establish a secure communication channel
Figure 6-2 User registration
Figure 6-4 Group list interface
Figure 6-5 Member list interface
Table 8-2 Six resources and XML structure
Figure 8-1 ER diagram
List of abbreviations

MSNP ......................................................... Mobile Social Network Provider
IMPU ........................................................... IP Multimedia Public Identity
CA ............................................................. Certificate Authority
BSF .............................................................. Bootstrapping Server Function
HSS ............................................................. Home Subscriber Server
SAML ........................................................... Security Assertion Markup Language
GBA ............................................................. Generic Bootstrap Architecture
UE .............................................................. User Equipment
SSC ............................................................. Support for Subscriber Certificates
MWSB ........................................................ Mobile Web Security Bootstrap
IMPI ............................................................. IP Multimedia Private Identity
FQDN ............................................................ Fully Qualified Domain Name
REST .......................................................... Representational state transfer
URI ............................................................. Uniform resource identifier
Content

1. *Introduction*.........................................................................................................................9
   1.1 The SWiN project description..............................................................................................9
   1.2 The scope of this project.......................................................................................................10
   1.3 Methodology........................................................................................................................10
   1.4 Research area.......................................................................................................................10
   1.5 Audience.............................................................................................................................11
   1.6 Limitation............................................................................................................................11

2. *Background*............................................................................................................................12
   2.1 Social network......................................................................................................................12
   2.2 Information that a user shares.............................................................................................13
   2.3 The trend of social networks...............................................................................................13
   2.4 Major privacy threats in the social networks.......................................................................13
   2.5 Security requirements for mobile social networks.............................................................14
   2.6 Blueprint of the mobile social network................................................................................14

3. *Requirement Specification*....................................................................................................16
   3.1 Project description................................................................................................................16
   3.2 Components Description........................................................................................................16
   3.3 Contribution..........................................................................................................................17
   3.4 Scenario Description.............................................................................................................18
   3.5 Functional Specification Analysis.........................................................................................19
   3.6 Project goal..........................................................................................................................20
      3.6.1 Implement functionality..................................................................................................21
      3.6.2 Integrate the components.............................................................................................21
      3.6.3 REST concept.................................................................................................................21

4. *Architecture Design*...............................................................................................................22
   4.1 Client Architecture..............................................................................................................22
      4.1.1 Authentication module..................................................................................................22
      4.1.2 Social network client.....................................................................................................23
      4.1.3 FriendFinder application...............................................................................................24
   4.2 MSNP Architecture................................................................................................................24
      4.2.1 Authentication module..................................................................................................25
      4.2.2 Assertion server.............................................................................................................25
      4.2.3 Social network application............................................................................................25
   4.3 Certificate Authority Architecture.......................................................................................25
   4.4 Operator Architecture...........................................................................................................25

5. *Authentication protocols*.......................................................................................................27
   5.1 GBA protocol description.......................................................................................................27
   5.2 MWSB Enabler Architecture...............................................................................................27
   5.3 Bootstrapping Steps.............................................................................................................28
      5.3.1 Step 1: Set up bootstrapping.........................................................................................28
      5.3.2 Step 2: Establish a secure channel...............................................................................29
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4 Android Limitation</td>
<td>30</td>
</tr>
<tr>
<td>6. Function Description</td>
<td>31</td>
</tr>
<tr>
<td>6.1 Installation of FriendFinder</td>
<td>31</td>
</tr>
<tr>
<td>6.2 Account Registration of FriendFinder</td>
<td>31</td>
</tr>
<tr>
<td>6.3 Authenticate a registered user</td>
<td>33</td>
</tr>
<tr>
<td>6.4 View group lists</td>
<td>34</td>
</tr>
<tr>
<td>6.5 View a member list</td>
<td>34</td>
</tr>
<tr>
<td>6.5.1 View an online member list</td>
<td>35</td>
</tr>
<tr>
<td>6.5.2 View an offline member list</td>
<td>35</td>
</tr>
<tr>
<td>6.6 Group creation</td>
<td>35</td>
</tr>
<tr>
<td>6.7 Membership assertion</td>
<td>36</td>
</tr>
<tr>
<td>6.8 Invite a friend</td>
<td>37</td>
</tr>
<tr>
<td>6.8.1 An online invitation</td>
<td>38</td>
</tr>
<tr>
<td>6.8.2 An offline invitation</td>
<td>38</td>
</tr>
<tr>
<td>6.9 Manage Group</td>
<td>40</td>
</tr>
<tr>
<td>6.9.1 Delete a group member</td>
<td>40</td>
</tr>
<tr>
<td>6.9.2 Grant a member to admin</td>
<td>42</td>
</tr>
<tr>
<td>6.9.3 Revoke admin to member</td>
<td>42</td>
</tr>
<tr>
<td>6.9.4 Delete a group</td>
<td>43</td>
</tr>
<tr>
<td>7. Restful web architecture</td>
<td>44</td>
</tr>
<tr>
<td>7.1 The basic concept of REST</td>
<td>44</td>
</tr>
<tr>
<td>7.2 REST constraints</td>
<td>45</td>
</tr>
<tr>
<td>7.2.1 Client-Server</td>
<td>45</td>
</tr>
<tr>
<td>7.2.2 Stateless</td>
<td>46</td>
</tr>
<tr>
<td>7.2.3 Cacheable</td>
<td>46</td>
</tr>
<tr>
<td>7.2.4 Uniform interface</td>
<td>46</td>
</tr>
<tr>
<td>7.2.6 Layered System</td>
<td>46</td>
</tr>
<tr>
<td>7.2.7 Code on demand</td>
<td>47</td>
</tr>
<tr>
<td>7.3 Build a restful social network</td>
<td>47</td>
</tr>
<tr>
<td>7.4 Security issue</td>
<td>51</td>
</tr>
<tr>
<td>8. Server Implementation</td>
<td>53</td>
</tr>
<tr>
<td>8.1 Framework &amp; Environment</td>
<td>53</td>
</tr>
<tr>
<td>8.2 Project Structure</td>
<td>53</td>
</tr>
<tr>
<td>8.3 Server configuration</td>
<td>56</td>
</tr>
<tr>
<td>8.3.1 Configure web.xml file</td>
<td>56</td>
</tr>
<tr>
<td>8.3.2 Configure pom.xml file</td>
<td>58</td>
</tr>
<tr>
<td>9. Database design</td>
<td>59</td>
</tr>
<tr>
<td>9.1 Entities and entities relations</td>
<td>59</td>
</tr>
<tr>
<td>9.2 Stored procedure</td>
<td>60</td>
</tr>
<tr>
<td>9.2.1 Advantage of stored procedures</td>
<td>60</td>
</tr>
<tr>
<td>9.2.2 Access control</td>
<td>62</td>
</tr>
<tr>
<td>9.3 Trigger</td>
<td>64</td>
</tr>
<tr>
<td>9.4 Database implementation</td>
<td>65</td>
</tr>
<tr>
<td>10. Conclusion</td>
<td>66</td>
</tr>
</tbody>
</table>
10.1 Implementation ........................................................................................................... 66
10.2 Future work ................................................................................................................. 66
1. Introduction

This chapter gives a brief background knowledge of the Social Wireless Network Secure Identification[1] project with an introduction of the project. The scope of the master's thesis within the SWiN project is also presented. Furthermore, the limitation of the research is discussed.

1.1 The SWiN project description

In recent years, social networks have become a daily communication tool for many people, sharing information, photos and thoughts with their global friends through social network. Social networks are not only used to keep in touch with friends. They are also applied in the industry. For example, companies utilize social networks to provide efficient communication between employees. With technology developing, social networks do not anymore require users to sit in front of a PC to share information. Instead this can be done through a mobile device. Mobile social networks provide a more convenient platform to share users' geolocation, short messages, multimedia messages etc in real time. To help more social network users to move to mobile platforms, several critical issues have to be taken into consideration. Security and privacy are the most common keywords connected with a social network. In the digital world, it is hard to combine security and user-friendly authentication, so a lot of attack opportunities are left to malicious users. Yet, users of social networks continue to expect that their private information is only available to their friends when they switch to mobile social networks.

When a user signs into a mobile social network, a proof of the user's identity is required, so that the mobile social network can authenticate the user. The traditional way of authentication is to use a user name and a password to verify the identity of the user. Many users choose weak passwords, therefore, the traditional techniques cannot satisfy both security and user-friendliness requirements. A new secure authentication is required for the mobile social network. Additionally, users may separate their friends into various groups and share different information with the different groups, so proof of group membership may be needed. Finally, the mobile application should still be available for users without an Internet connection, so it is necessary to design a secure offline invitation generator that allows to disconnect users to participate in the group's activities.

The SWiN [1] project of SICS (Swedish Institute of Computer Science), Ericsson Research and Sony Mobile Communications has delivered some theoretical security solutions for mobile social network to solve above issues.

- Secure and user-friendly authentication of mobile users.
• Offline proof of group membership.
• Offline invitation to mobile social network groups.

1.2 The scope of this project

The objective of this project is to build up a mobile social network with the SWiN project security solutions. The social network should contain two parts which are the mobile social network server and the mobile social network client. The mobile social network server is based on the services from Ericsson labs platform, and the mobile social network client is developed on the Android OS.

This thesis is going to focus on the mobile social network server, and the client part is discussed in Sun Lei’s master thesis [2]. In order to achieve the goal, we break the project into several small tasks, as listed below.

• Study the requirements of the SWiN project for social networking functionality.
• Design an extensible mobile social network.
• Implementation and documentation of the designed mobile social network.
• Functional tests.

1.3 Methodology

The objective of the thesis is to validate the early phase research results, so the thesis is done as an applied research. Sun Lei [2] and the author have to study the theoretical research problem and results to implement a demonstrator in a research environment.

Steps of research:
1. Understanding the theoretical research problem and results.
2. Developing objectives for the demonstrator.
3. Examining applicable technologies.
4. Designing the architecture and communication patterns of the demonstrator.
5. Programming the designed solution.
6. Testing and documenting the designed solution.
7. Preparation of a report on the results of the work.

1.4 Research area

As a platform for solving the security issues, we have to build an extensible mobile social network server, which requires a good architecture design based on the requirement specification. Before the project design, it is necessary to analyze the requirements and all the components of the security solutions that have to be involved in the design, so that they can be smoothly integrated in the mobile social network.
Furthermore, we have to increase the usability of the mobile social network, so the offline mobile social network users are capable of sharing the information with others when they are in proximity to each other. Most social networks require the user to connect to a central service provider first, in order to retrieve his/her friends’ information through a server database. Obviously, the advantage of this infrastructure is keeping data consistency, however, it limits the scope of usability. If an area does not have strong 3G signal or even is not covered by a 3G operator, a mobile user is not able to use the mobile social network application anymore. Furthermore, the charge of the 3G service is often based on how much the data is downloaded or uploaded, so many mobile customers do not keep the service on when there is no free Wifi connection. Without a network connection, an application user can not connect to the server to use the application. In this situation of mobile social network does not provide real time communication or sharing. In the design phase, the offline situation has to be taken in consideration, so that the requirements can be fulfilled. Moreover, many frameworks can be chosen to build the server, so we have to find the most suitable one to employ. During the implementation phase, we have to constantly pay attention to security issues, which may not have been addressed by the previous provided security solutions.

1.5 Audience
The audience of this research can be a mobile application developer, since it contains some security solutions for mobile applications. Mobile service providers might be interested in the thesis, especially if they plan to extend their business to the mobile social networking field. Finally, web server developers are introduced to a new architecture style to build up an extensible server in a simple way.

1.6 Limitation
The current research is still a prototype demonstrator, so the provided solution is based on a number of assumptions. For example, our authentication solution currently only works with a virtual software SIM card. Moreover, the final product is focused on research demonstrations, so that performance of the product has not been examined thoroughly, especially in the case of the increasing number of mobile social network users.
2. Background

In this chapter, the background of the project is given, so that a reader would have a comprehensive picture of the motivation of the project and the process of the project development.

2.1 Social network

A social network is a good platform to link people and extend friendships into the virtual world. With the increase of social network users, social network providers design a variety of social network services to attract users. In general, social network applications can be categorized in five major types [3].

- **Personal friendship networks**
  In this kind of networks, users can create their own online profiles and share personal information with friends. Facebook, MySpace are examples of this kind of platform.

- **Text update networks**
  These social networks allow users to upload their short messages or posts, so all the friends of the user may see the status at once. Examples of these social networks are Twitter and WeiBo\(^1\). The users can broadcast their information publicly in a very short time.

- **Geography-sharing networks**
  GPS is widely used in mobile device, so mobile users can share their locations with friends in real-time by using geography-sharing networks. Normally, only authorized friends are able to view the user’s location. Google Latitude is an example of such a social network.

- **Content-sharing Networks**
  The content-sharing networks are platform services that can be used to upload data from mobile devices or computers to the networks. The data can be presented as video, picture or music, etc. Youtube, Picasa and Thesixtyone are the examples of these networks.

- **Gathering-interest networks**
  Some of the social networks help users to find friends who have common interests. The networks may group users by their education background, similar hobbies, religious views and etc. Some examples of these networks are Goodreads and Linkedin.

---

\(^1\) It is one of the most popular sites in China and overs 30% of Internet users with similar market share as Twitter in the USA. It has more than 300 million registered users as Feb of 2012.
2.2 Information that a user shares

Based on the social networks listed above, we summarize the type of data that users often upload on these networks. The list below shows the summarized data types:

- Name, age, contacts and gender
- Biographical information (education background, work place, hometown and etc.)
- Posts
- Location
- Interests

Many users consider at least parts of this information as private, so a good protection of the data is necessary. Many of the social networks offer options so that a user may share the information only with his/her friends instead of the entire public networks. However, the most commonly used mechanism of authentication, username and password, is very weak, which means that many times an unauthorized user may access private data. This is illustrated for example by the following statement by one of the most popular social networks:

“We cannot guarantee that only authorized persons will view your information. We cannot ensure that information you share on Facebook will not become publicly available. We are not responsible for third-party circumvention of any privacy settings or security measures on Facebook”[4]

2.3 The trend of social networks

With the increasing number of the smart phone users, the number of mobile social network users will grow. eMarketer predicts that the number of mobile social network users in the U.S. will be more than double in the next five years, from 38.9 million in 2010 to an estimated 79.1 million in 2016 [5]. Based on this analysis, we believe that mobile social network will be a new future trend of social networks.

2.4 Major privacy threats in the social networks

In this section, we introduce some common threats in social networking, so the developer can address these issues to find the solutions. These threats can affect both, the mobile social network users as well as the social network itself.

- Social Engineering
  By applying various social engineering methods, in context of security, the malicious attacker may trick a social network user to obtain his or her personal information such
as password, credit card number and etc. For example, an attacker may act as a social network provider to ask for user personal information and gain the user’s password.

- **Malware**
The malicious software is installed on the mobile device and it steals the information from the users’ mobile devices. Sometimes, the malware broadcasts the link, which contains malicious software to all the user’s contacts.

- **Man-in-the-middle**
This attack method as the name implied implies that the malicious user will interrupt the communication between the user and the server to steal the information that transferring in between.

2.5 *Security requirements for mobile social networks*
To protect the user privacy, we summarize some requirements that the mobile social network provider has to keep in mind.

1. New users of the mobile social network have to be registered in a way that they can later authenticate to both the social network and other users of the network.
2. The registration process has to be conducted over a secure channel
3. The registered user must be unique in the network’s database and the database must be protected against malicious modifications.
4. The mobile social network should be able to detect and prevent automated registration of users.
5. The mobile social network server should perform mutual authentication with its users before each session.
6. Users must be able to mutually authenticate each other, even offline.
7. Anonymity and unlinkability techniques should embed into the mobile social network, so the activity of the user cannot be tracked.

In order to build a secure mobile social network, we have defined a requirement specification with the project supervisor Ludwig Seitz, and the requirement specification is discussed in chapter 3.

2.6 *Blueprint of the mobile social network*
To build up a mobile social network, is like constructing a building. Before the construction work starts, a blueprint is required. The software architecture design is the blueprint that the developer has to follow during the entire project development. We have found the following definition of software architecture:
“The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [6].

Based on the definition of the software architecture, we have to address the system decomposition into various structural elements. In the chapter 4, the mobile social network system has been broken down into several pieces. Each of the structural element should satisfy the principle of decoupling, which can be accomplished by keeping a simple interface and encapsulating the element. After the system has been divided into small elements, several questions can be asked when examining the system integrity.

Question 1: Do we have all the elements that the mobile social network needs?
To answer the first question, we have to make sure that all the required functions of the project have been fulfilled by the designed elements. In this document, the requirement description will be introduced first in the chapter 3, and then the architecture design is discussed in chapter 4.

Question 2: Do all the elements fit together?
In the mobile social network project, many components are involved and delivered by several developers from different teams, so the compatibility of each components are important. In chapter 6 of this document, we have listed all the functions that may be needed in order to verify the interaction of all the components.
3. Requirement Specification

Analysis of a project requirement is a significant step for both customers and programmers to reach an agreement on the implementation work. This chapter introduces the project specification based on the SWiN project requirement. The requirement states four major involved components of a mobile social network and describes the relations between each components. Moreover, a scenario is provided to show how the mobile social network is applied to the real world. The last section analyzes the scenario and lists all the required functions for the implementation work.

3.1 Project description

This thesis project is going to implement a web server and a mobile application called FriendFinder to allow a user to build and manage their own small social network through a mobile device. In the social network, users could share their confidential data with their trusted friends. The research department of Ericsson and the security department of SICS provide some security solutions to verify the user's identity to make sure that only authenticated users are allowed to read the confidential data. The thesis project integrates those security solutions to develop the social network security.

3.2 Components Description

This section is a high level description of the four major components of this mobile social network. The components are the MSNP (mobile social network provider), client, phone operator, and CA (certificate authority). The MSNP includes a database to store the user information, and a server is to transfer data between the user and the database. The client is a user that holds a smart phone with an installed mobile social network application called FriendFinder. The operator is used to authenticate the client using the GBA protocol discussed in chapter 5. The CA is an optional components of the architecture, it signs a certificate for the client, so the client identity can be authenticated through the signed certificate. In the project implementation, the MSNP performed the functions that CA does. The figure 3-2 displays the architecture of those components.
3.3 Contribution
As section 3.2 described, there are various components that are involved in the project. During the research phase, Ericsson lab has provided both theory and technical support to implement the operators functionality. The main function operator provides the GBA protocol to authenticate the client. This means that Sun Lei [2] and the author have to integrate the GBA authentication for both the MSNP and the client sides in order to connect with an operator as is shown in figure 3-2. After the GBA authentication, each client will retrieve a X.509 certificate from a third party and the certificate is an identity of the client. SICS had provided the code to generate a X.509 certificate and we have added the X.509 certificate generator on the MSNP side. When a client registers an account from the MSNP, a X.509 certificate will be generated and sent to the client. The client stores the X.509 certificate in the mobile device and exposes it to other clients or the MSNP when needed. If a client is creating a new group, becoming a new group member, or changing a role in the old group, a SAML assertion is going to be generated by the MSNP. The SAML assertion has been used in the project to prove group membership. SICS has provided us the code for generating a corresponding SAML assertion format, which could clearly display the client's id, client’s role and the group to which the client belongs. Moreover, SICS provides us the SAML library for Android so that we could easily use it for developing the client side. Take an example to demonstrate how a SAML Android library is used to the client side, an offline client named Alice has been invited to a group called Uppsala University by the owner of the group Bob. In this case, Bob’s mobile device has to generate a membership assertion to Alice so that the SAML library for Android is needed.
Based on the project architecture design, we have separated the project into two parts, which are the MSNP and the client. The MSNP has been mostly implemented by the author and the client side has been mainly done by Sun Lei [2]. However, it is hard to draw a line between the MSNP and the client to separate the development work since components of the project are connected. Therefore, Sun Lei [2] and author have worked together to develop both sides. The purpose of dividing the project is to provide a better focus on the development and documentation for developers.

During the theory study and the implementation, the author focuses on study the GBA authentication and the MSNP implementation. The MSNP implementation includes both web server development and server database implementation. Moreover, the author integrated both authentication methods into the MSNP to demonstrate how the authentication method works in a real case. In the documentation, the author focuses on describing the GBA authentication method and the MSNP development.

### 3.4 Scenario Description

In order to design a mobile social network that covers all the requirements, a scenario is invented to describe how the mobile social network is used in the real word. The scenario tries to cover all the user cases.

Alice and Bob are both employed in a big electronic enterprise called Electro in the US. They have been asked to participate in a trade fair to search good products and partners. Due to the limited time, Alice and Bob have to work separately. To keep track of each other, they decide to use a mobile application called FriendFinder. Users may build their own social network through FriendFinder and share their location information with the friend who belongs to the social network. After FriendFinder has been installed in both Alice and Bob’s smart phone, Alice and Bob registered accounts for FriendFinder. Later, Alice creates a group called Electro and sends an online invitation to Bob. Once Bob accepts Alice’s online invitation, Bob becomes a member of Electro. As a group member of Electro, Bob shares his location information with all other group members and all the members send their location to Bob as well. In this example, Bob only shares his location with Alice and Alice is the only one displayed on Bob’s map. Both of Alice and Bob’s phones have very good Internet connection, so they could share their information immediately when they moved to a new location. This big trade show is not only attracting Electro, but also the competitor of Electro. The location of Alice or Bob may reveal Electro’s future business strategy to their competitor, so FriendFinder should be designed to prevent information leakage.
In the middle of the trade fair, Alice unexpectedly meets her colleague Charlie, so Alice recommends Charlie to use the application FriendFinder. Charlie should first register an account in Friendfinder and gives her FriendFinder account to Alice, so Alice could invite Charlie to the Electro group. Unfortunately, Charlie’s phone does not have a stable Internet connection, so Alice has to send an offline invite to Charlie. Charlie signs into FriendFinder without network connection, and accepts the offline invite. Charlie becomes a temporary member of Electro and starts to share Electro group information with other group member through an Electro group’s Ad-hoc network signal\(^2\). With the offline invite, Charlie could be registered as a permanent member of Electro when Charlie’s phone connects to the Internet. However, the offline invite has a short time of validity, so Charlie has to register herself before the offline invite expires.

3.5 Functional Specification Analysis

Before the mobile social network implementation, it is necessary to analyze the main functions of the mobile social network. In the descriptive section of the scenario, it narrates that the FriendFinder application should supply services for both online and offline state users, so it is necessary to know if all the functions can be performed in both states. The interface design would follow the same pattern if the two states could perform exactly the same functions. However, based on the scenario description, some of the functions are only provided under the online state, and some of them can only be performed under the offline state. Normally, the functions only presented in online state requires a network connection to the web server, so the client can manipulate data from the server database. For instance, a user registers an account from FriendFinder, it requires that the chosen account name is sent to the MSNP and the MSNP verifies whether the name is available or not from the server database. The process of registration requires a communication channel between the client and the server, so the network connection must be available to fulfill the function. While some functions that can be only performed when the user is disconnected to network, for instance, the actions of receiving and accepting offline invitation could only take place in an offline state. To list the specific functions of each state, helps the interface design and implementation work, so the two lists below have been summarized.

**Online State Functions:**
- Client logs in to the FriendFinder application
- Client registers an account from the FriendFinder.
- Client generates its own certificate.
- The MSNP signs the certificate for the user.

\(^2\) Ad-hoc network has not been implemented in this project, but we are planning to set up a ad-hoc network for the group member to share their location without Internet.
- Client stores the resigned certificate.
- Client creates a group at the MSNP and the client is assigned as the owner of the group.
- Client views the group list.
- Client views the member lists of each belonging group.
- Owner of a group can grant the user of the group as Admin.
- Owner can revoke the admin of a group as a member.
- Owner or Admin of a group sends an online invitation to the friends.
- Client checks the received online invites.
- Client accepts/denies the received online invites.
- Owner can delete the created group.
- Admin or member can quit the belonging group.
- Client registers himself/herself as a permanent group member from a temporary offline group member.

**Offline State functions:**

- Client logs in to the FriendFinder application.
- Client views the group list.
- Client views the member lists of each belonging group.
- Owner or Admin of a group generates an offline invitation to the friends.
- Owner or Admin of a group sends an offline invitation to the friends via Bluetooth[20].
- Client checks the received offline invites.
- Client can receive an offline invite via Bluetooth.
- Client accepts/denies offline invite.

A user can act in three different roles, and the role presents what a kind of permissions the user has. These three roles are distributed in a hierarchical structure. The owner is the one who is on the top of the structure and the one that created the group, and he/she owns the entire permissions of the group management. The admin has less permissions than the owner. But compared with member, the admin is allowed to invite a friend to join the group, which the member is not allowed to do. A member resides in the bottom of the role structure. He/she is only allowed to view the group information and quit the group. While the network connection would also affect to the group management permission, it inhibits owner or admin to modify the roles when the connection state shows disconnected.

### 3.6 Project goal

Based on the requirements, there are three major problems the author has to face, one is to fully implement the SWiN project functionality, another one is to integrate all
the components of the project together and the last one is to set up a web server by utilizing a new architecture style called REST.

3.6.1 Implement functionality
Based on the section 3.5, the two groups of functions in the SWiN project could be divided into online and offline state functions. Online state functions require interactions between the client and the MSNP, that offline state functions do not require. In this thesis, the author focusses on the implementation of online state functions.

To implement online state functions, the author has to analyze the entire process of each function. For instance, the user login function requires several steps to be implemented. Firstly, a user has to build up a secure channel with the MSNP and prove his/her identity to the MSNP. The MSNP has to invoke an authentication mechanism to authenticate users identity. In the end, a message has to be sent from the MSNP to inform if the user can log in or not. In order to fulfill the requirement, the author has analyzed the process of each function and this is described in chapter 6.

3.6.2 Integrate the components
A project is the same as a human body that contains different parts and the cooperation condition of those parts may affect the quality of project. Normally, a middle size project requires a group of people to work together, so the project has to be divided into several parts. Mostly, the project, the team members only focus on their own work, not on others, but to accomplish the project team members have to understand the entire structure of the project. The structure of a project is a map to guide the team member to connect their work with others, so a good architecture design is needed in a project. In chapter 4 the author will introduce the architecture design of the entire project, the necessary components of the project are displayed and the connection has been made between each components so that the author can have a global view to implement the project.

3.6.3 REST concept
The REST is a new concept of architecture that we are to apply in our web architecture. In order to understand the REST concept, author has to understand the basic theory of the REST, chapter 7 has described the REST concept. Based on the theory, author has designed the MSNP and the databases that will be mentioned in chapter 8 and 9.
4. Architecture Design

Based on the requirement specification, the architecture of the application can be designed. This chapter splits the application into four major components, which contain the client, the MSNP, CA and the operator. Each of these components is constituted by various small elements and some of the elements are linked. The interaction of the elements contributes to the connection of components. Figure 4 is an overview of the entire architecture design, and the dotted line of the figure 4 indicates the connection of the elements. The architecture design has been developed by SWiN team[18].

4.1 Client Architecture

The client component is the user that holds an android OS phone with an installed mobile social network application. This component contains three major elements:

- An Authentication module
- A social network client
- A FriendFinder application

The social network application delivers a secure platform for the clients. On top of the client component is the authentication module and it is connected with the component CA and the MSNP to authenticate a client’s identity as Figure 3 shows. Beside the authentication module, a social network platform is required to be implemented and it connects to the MSNP. The connection of the MSNP does not only allow the client to retrieve a corresponding membership assertion and the own certificate from the MSNP but also to manipulate the data from the server databases. A client membership assertion proves a membership in a group and in the later section we will introduce more details about the membership assertion. Appendix 1 shows what a membership looks like. The figure 3 indicates that the FriendFinder application stays at the bottom of the client component. The reason to separate FriendFinder as an individual element is because FriendFinder is an independent function that the social network provides. Removing the FriendFinder function will not affect executing mobile social network application.

4.1.1 Authentication module

In the introduction section, a security question has been raised. How could we authenticate the client identity in the mobile social network? The authentication module utilizes several protocols to achieve the goal of the client authentication in the client components. Firstly, the GBA protocol has been applied, the client has to
implement the UE part of the GBA. The GBA protocol uses the secret key on the SIM card to authenticate the client to a third party via the operator. The details of the GBA protocol will be discussed in the chapter 5. The SSC-UE extension implements SSC protocol to create and register a subscriber certificate. This process requires that the SSC-UE connects to the CA as Figure 4 shows. Additionally, the authentication module contains a X.509 library to generate a certificate request.

![Diagram of the architecture of SWiN project]

Figure 4. The architecture of SWiN project. [7]

4.1.2 Social network client
The social network platform implements the basic features of the social network, which includes user registration, group creation, group management, and generating an invitation. As the chapter of the requirement specification mentioned, some of the
functions require that the client has a network connection to establish a communication channel between the MSNP and the server database to manipulate data. However, the social network should still be available when the client disconnects with the MSNP. For example, the client could still browse group lists and member list without retrieving information from the MSNP and the server database. That means that a client needs to have a data storage to store the user information and the group information for offline state performance. The information has been saved in the memory of the mobile device and the programmer utilizes SQLite to build tables to store the data in the local database. When the client is connected to the server database, the client’s local database will be updated according to the server database.

Inviting friends to join a group is one way to extend the created social network. An invitation has to be sent out to inform friends that they are welcome to join the group. However, the online invitation and offline invitation are generated in different ways. An online invitation is generated by the MSNP, but an offline invitation is created by the user. The invitation generator displayed in the figure 4 is an element to generate an offline invitation directly from user’s mobile device. An offline invitation contains various information\(^3\) by following SAML standard. SICS SAML is a software library that implements parts of the SAML standard for Android OS mobile device. In section 6.8.1, it is described how an online invitation is generated and in the section 6.8.2 we describe the procedure for an offline invitation.

Beside the local database and the invitation generator, the user has to store his/her own certificate, inviter’s certificate, group membership assertion, offline invitation and client private keys of certificate into the corresponding folders of the mobile device’s SD card. The above mentioned information requires more space so it is stored as a file instead of stored as data in SQLite database. The section 6.6.2 explains why the user needs this information.

4.1.3 FriendFinder application
FriendFinder is the application that the user could use to check his/her friend’s location. This is done by establishing a group channel, and group members broadcast and receiving each other’s location information. Based on the received location information, the application will display the friend’s location on a map. Due to the time limitation, this part has not been implemented in this thesis.

4.2 MSNP Architecture
There are three elements are involved in the MSNP architecture:

---

\(^3\) Appendix 2: Offline invitation
● Authentication module
● Assertion server
● Social network application

4.2.1 Authentication module
Basically, the authentication module provides two functions. The first one is to implement the NAF part of the GBA to connect with the BSF server. Moreover, a X.509 library is needed to perform the subscriber certificates so the MSNP should own CA’s certificate to do so. In the real implementation, the MSNP substitutes the CA to sign the client certificates and performs the function of the CA.

4.2.2 Assertion server
The assertion server is developed by SICS as an identity management tool. This tool issues a SAML assertion with various attributes and the attributes can be edited based on the requirement. The assertion server is designed as a lightweight library, so it will be integrated in the MSNP during the implementation work. The MSNP could issue a membership assertion to the client after the assertion server that is integrated in the MNSP. Section 6.6 describes how a membership assertion could be issued.

4.2.3 Social network application
The social network application should deliver the information that the client needs from the database. A client could also modify their own data through the social network application. The social network application has been designed to present various resources from the database. Chapter 7 describes a concept called REST, and the web application has been built based on the concept.

The database is used to store and verify the client information. There are four tables designed to store all the data, that include the user_info, the group_info, the user_role_for_group and the pending_invite. In the chapter 9, we will describe the attributes of each table, tables’ relations and some stored procedures.

4.3 Certificate Authority Architecture
CA is a third party to authenticate a client identity. According to the Support for Subscriber Certificate (SSC), the CA should implement the NAF. The CA has to connect both with the UE and the operator to authenticate each client’s identity and to sign the certificate for the client. In the project, the CA has been merged with the MSNP.

4.4 Operator Architecture
The operator is a server that plays the role as HSS and BSF of the GBA protocol and communicates with the MSNP to identify the client as a third party.
5. Authentication protocols

In a traditional way, the social network authenticates a user through an account name and a password. However, the account name and password can be stolen in many ways, even sometimes the users may forget their passwords. If the qualification of authentication is not based on an account name and a password, the authentication may improve. This chapter is going to deliver an improvement of the mobile social network authentication, which has been developed by Ericsson lab[9].

5.1 GBA protocol description

A mobile phone SIM card cannot only be used for cellular access but also applied to identify the client. The GBA is the protocol that used the SIM card authentication and it is standardized by 3GPP [8]. The concept of the GBA protocol is to generate a shared key between the client and the mobile social network. The client generates the secret key through the SIM card, and the mobile social network cooperates with a HSS and a BSF to gain the secret key. Figure 5-1 indicates the GBA protocol architecture.

![Figure 5-1 GBA protocol architecture. [9]](image)

If the secret key that has been generated on both sides matches, then it certifies the identity of the client. Moreover, a secure communication channel can be established between the mobile client and the mobile application server.

5.2 MWSB Enabler Architecture

A component called the mobile web security bootstrap transfers information between the mobile client and the mobile application server as figure 5-1 shows. This
component contains BSF and HSS. The figure 5-2 displays in detail how the BSF and the HSS connect and construct a mobile web social bootstrap enabler.

The HSS holds the client’s information and connects with the BSF that delivers security connection between the GBA client and the GBA NAF. The GBA client and the GBA NAF are a part of the MWSB enabler, which are implemented by Ericsson. As figure 5-2 shows, the GBA client is integrated within the mobile application and the GBA NAF is part of the application server. Currently, Android OS does not provide an application level API for SIM access, so Ericsson lab simulates a software SIM card to achieve SIM card based authentication.

![Diagram](image)

**Figure 5-2 MWSB enabler architecture [9]**

### 5.3 Bootstrapping Steps

There are two major steps involves to implement the GBA protocols. The following subsections describe these steps and how the GBA protocol could provide a secure communication channel between the client and the application server.

#### 5.3.1 Step 1: Set up bootstrapping

The bootstrapping starts from the client, the client application calls the client GBA to set up the bootstrap. In the SWIN project, FriendFinder is the client application that initializes the bootstrapping. After the client GBA receives the call from the client, it sends an HTTP request to the BSF with the client IMPI value. The BSF requests the authentication vector from HSS, and the authentication vector contains RAND, AUTN, XRES and Ks. Among those parameters of the authentication vector, the BSF forwards RAND and AUTN to the GBA client in an HTTP 401 message and the message requires the client to authenticate himself/herself. The client SIM card retrieves the challenge AUTN to verify if the request is from an authorised network.
If the verification succeeds, a long term key Ks and response RES are returned from the SIM. A digest AKA response is calculated using RES, and sent in an HTTP request from the GBA client to the BSF. The Digest AKA response is used to verify the GBA client authentication by BSF. BSF keeps the Ks received from HSS and returns an HTTP 200 with a Bootstrapping Transaction IDentifier (B-TID) to the GBA client. The GBA client returns the B-TID to a client application FriendFinder. Figure 5-3-1 shows the entire process of bootstrapping setup.

![Figure 5-3-1 Setup Bootstrapping][9]

5.3.2 Step 2: Establish a secure channel

The client and the application server establish a secure communication channel by sharing an application specific key Ks_NAF.

![Figure 5-3-2 Establish a secure communication channel][10]

The Figure 5-3-2 displays how the key has been generated on both sides. After the first step, the client application holds a B-TID and the GBA client owns a Ks. In the second step, the client application sends B-TID and NAF_ID to the GBA client. The
NAF_ID is equal to NAF FQDN (Fully Qualified Domain Name) in the project. Later, the GBA client hashes NAF_ID and Ks to generate the Ks_NAF for the client application. After the client application owns the Ks_NAF, then it is time to generate Ks_NAF for the application server. B-TID is sent by the client application to the application server, and the application server forwards the B-TID to the GBA NAF. The GBA NAF delivers an HTTP request with B-TID and its NAF_ID to BSF, and BSF searches the bootstrapping context, which corresponds to B-TID and returns Ks_NAF by hashing Ks and NAF_ID. Ks_NAF and IMPU are sent to the application server through the GBA NAF. If the Ks_NAFs of client and application server matches, a secure communication channel is built. The secure communication channel can be setup because the client Ks_NAF is generated based on the B-TID, which is generated from the SIM. Only the client that holds the SIM card could own the corresponding Ks_NAF. The application server retrieves the Ks_NAF from the third party BSF based on the client B-TID, but not directly from the client. It avoids a malicious user steals Ks_NAF from direct share channel between the client and the application server. Moreover, the IMPU value is bound with each client account name in the database, the application server uses the IMPU as a password of each user to verify the client. However, the IMPU is not directly declared by the client but from the third party BSF based on the B-TID, which was generated from BSF by passing SIM card authentication. The IMPU could prevent the forged user to pretend other users to communicate with the application server. The conclusion of authentication is that the identification of the client is based on the physical SIM card rather than a self declaration. Additionally, the shared secret key and the client password IMPU value are not directly transferred between the client and the application server but from the neutral third party BSF.

5.4 Android Limitation

In the implementation, we have used Android OS as our mobile application platform, however, Android does not have any application level API for SIM card access. In other words, the operator has a very strict requirement on the SIM card access, and the access control model of Android does not have a very good solution to match the requirement of the operator. Therefore, we have designed a new approach called Domain-Based access control to extend the access control of Android, and this approach has been discussed in the master thesis of Qing Huang [11]. Since this approach is still in the research phase, a virtual SIM card has been used instead of the real one to implement the GBA protocol in this project.
6. Function Description

In chapter 3, we list all the use cases according to the requirement. This chapter provides all the possible functions of the FriendFinder application based on the use cases. Executing these functions normally requires a pre-condition, so it is necessary to state clearly requirements of running certain functions. Moreover, we have described the process of each function. The description of the functions gives a picture of how the architecture components interact with each other.

6.1 Installation of FriendFinder

Requirement:
- To install the FriendFinder mobile application, it is required that the user has a smartphone with the Android operating system.

The FriendFinder mobile application can be installed in both online and offline state. Users may download the application from the Android market through Internet, or receive the application with Android package file (APK) from friends via Bluetooth. The application can be installed by an Android file manager.

6.2 Account Registration of FriendFinder

Requirement:
- An installed FriendFinder mobile application.
- The FriendFinder mobile application should be configured with X.509 certificates of the MSNP.
- The MSNP has an agreement to use the GBA protocol with the client’s phone operator.
- The mobile device has a stable network connection.

Steps of registration:
1. Client accesses to the registration activity from the mobile device to enter a chosen name.
2. The mobile application verifies if the chosen name follows a legal pattern, which only contains 26 case-sensitive letters and underscore.
3. The client connects with the MSNP through the FriendFinder mobile application, and the MSNP responds with its certificate to start one way TLS authentication.
4. Once the secure communication has been established, the client sends a registration request to the MSNP.
5. The MSNP sends an HTTP digital request for a user name (B-TID) and a password (Ks_NAF) to verify the client identity.
6. The GBA protocol starts. The client setups bootstrapping to retrieve B-TID from the GBA client, then B-TID and Ks_NAF are sent as user name and password to the MSNP.

**Figure 6-2 User registration**

7. The MSNP delivers the B-TID to the GBA NAF and the GBA NAF sends B-TID and the NAF_ID to the BSF. The BSF returns Ks_NAF and the client IMPU value to the MSNP through the GBA NAF. The MSNP compares the values of Ks_NAF, that
are received from the client and the BSF. If the values match, the identity of the client is confirmed. A secure channel is established between the client and the MSNP. In the later step, when the client sends a request to the MSNP the step 6 to step 7 will be repeated.

8. The MSNP checks if the chosen account name has been registered or not.
9. The MSNP returns the result to the client. If the name is still available, the MSNP will be bound the chosen account name and IMPU value in databases.
10. The client creates a certificate request and a private key as well. The certificate request looks like a self-signed certificate.
11. The client generates two certificate requests (CSR) in a PKCS#10\(^4\) format: one for user authentication certificate and other one for a signing certificate.
12. The client submits the PKC#10 request for an authentication certificate to the MSNP and the step 6 to step 7 will be repeated so the identity of client can be confirmed.
13. The MSNP processes the PKC#10 request, and creates a self digitally signed certificate to the client.
14. The MSNP replies the client a new generated certificate as an HTTP response [12].
15. The client stores the certificates in the mobile device.
16. The MSNP returns a message to inform the client that registration is done.

6.3 Authenticate a registered user

**Requirement:**
- The client has a good network connection.
- The client has a registered account.
- The MSNP has an agreement to use the GBA protocol with the operator.

**Steps of authentication:**
1. Client enters an account name.
2. The account name will be passed to a local program, which could verify that the entered name follows a correct pattern. The correct pattern should only contain 26 case sensitive letters and underscore.
3. The client connects to the MSNP.
4. The MSNP sends an HTTP digest request to the client.
5. The GBA protocol starts, the process follows the registration step 6 and 7.
6. The MSNP checks if the entered account name and the retrieved IMPU from the BSF are matched with the account name and the IMPU from the server database.
7. The MSNP responds a result to inform the user if the authentication succeeded. A secure communication channel is established between the client and the MSNP.

---

6.4 View group lists

The users of FriendFinder may view a group list to see which groups they belong to and which role they play in the group. In the following section, we will describe the process of viewing a group list.

**Requirement:**
- The client has a registered account.
- The MSNP authenticates the client successfully as the section 6.3 described.

**Steps of viewing a group list:**
1. The client sends a request to show the group lists to the MSNP.
2. The MSNP sends the client name to databases to collect all the groups that a client belongs to, and the MSNP asks which role the client plays in the group from the server database.
3. The MSNP sends the result of the group list, which have group names and the roles the client plays.
4. The client will remove the old group lists from a local database and stores the new one. Each time the client sends a request to view a group list, the local database will be updated as well as all the membership assertions.

![Figure 6-4 A group list interface](image1)

![Figure 6-5 A member list interface](image2)

6.5 View a member list

**Requirement:**
- The client has a registered account.
• The client should belong to the group. The group member list is a list that shows all the group member’s name and the role group member plays in this group. No matter if the client is online or offline, she/he could view the group member list. However, the connection states would determine if the client is going to update the newest group member list or read the group member list without updating from local database. Figure 6-5 shows an online member list interface, and the user is the owner of group SICS, so he or she has the right to delete the group.

6.5.1 View an online member list

Requirement:
• The client has a good network connection

Steps to view online member list:
1. The client sends a request to request a group member list from the MSNP. The request includes the client name and the group name.
2. The MSNP verifies if the client belongs to the group.
3. If the request has passed the MSNP verification, then the MSNP sends the group member list to the client.
4. The client removes the old group member list from the local database, and stores the new group members list information.

6.5.2 View an offline member list

Requirement:
• FriendFinder did not find the network connection.

Steps of viewing offline member list:
1. The client sends a request for a group member list to the local database. The request contains the client name and the group name.
2. The local database returns the group member list information to the client. The group member list information is updated when the client requests from the MSNP, so the information may not be so accurate as the server database. It might be a new group member joins to the group or old group member quits the group, so the old group member list gives a general vision for the offline user.

6.6 Group creation

Requirement:
• The client has a registered account.
• The MSNP authenticates the client successfully as the section 6.3 described.

Steps of group creation
1. The client selects a group name.
2. A local program verifies that the group name follows the rule, which demands that the name can only include 26 case-sensitive letters and underscore.
3. The client submits the group name follows the correct pattern to the MSNP.
4. The MSNP sends an HTTP digest request to the client, and the GBA protocol starts and the process of the GBA follows the step 6 and 7 of registration.
5. The MSNP checks if the group name has been registered before. If not, the MSNP will generate a group key and the database stores the group key and the group name. The group key is used to calculate the group ID when the user is offline, so a user could find group members that is nearby through group ID.
6. The MSNP confirms that the group creation succeeds and returns the group key to the client.
7. The client sends a viewing group list request to the MSNP.

Notice: In the step 6, the MSNP checks all the registered group names even if some of the registered groups have been deleted by the owner, the group names will still remain in the database as a registered group name. This is because the members that belong to the deleted group may still have valid group membership assertions after the owner deleted the group. Each membership assertion has a valid period so if a group member has received a new membership assertion before the groups are deleted, the deleted group member could prove himself or herself as a group member to other members for a short time. If a group has been registered to have a same name as a deleted group, then those members who belong to the deleted group may obtain the newly registered group information with a valid membership assertion. Membership assertions are introduced in the section 6.7. Those members hold the valid deleted membership assertions could obtain new membership assertions if a new group occurs to have the same name as the deleted one. Either the information of the member who belongs the deleted group or the private data of the member belongs to the new group will leak out in this case. To avoid this issue new groups cannot have the same name as deleted old groups.

6.7 Membership assertion

A membership assertion is a proof of membership in a group and can be generated by an integrated assertion server library. Appendix 1 illustrates how a membership assertion looks like. A membership assertion contains an assertion ID, an issue date, an issuer name, a digital signature of the issuer, the member name as subject, the time of assertion expired, and a group of attributes which are the group name, and the role of the member plays in the group. In the implementation work, the issuer of the assertion is the MSNP, and it has a long expiration period. Membership assertion can be updated easily, the client only needs to send a membership assertion request to the MSNP for a current membership assertion. The FriendFinder application is
implemented in a transparent way to collect current membership assertions for clients without their notice when a client connects to the MSNP. The client can prove his/her identity to other friends by showing a valid membership assertion and its own certificate. This identity certification is used when the client is disconnected with the MSNP.

**Requirement:**
- The client has a registered account from the MSNP
- The client authentication is completed as the section 6.3 described.
- The client is a registered member of a group or has a valid invitation from the group.

**Steps of retrieving membership assertion:**
1. The client sends a request to the MSNP for a group membership assertion.
2. The MSNP verifies that the client belongs to the group that the client inquire for a group of membership assertion.
3. If the step 2 succeeds, the MSNP creates the membership assertion\(^5\) and forwards it to the client.

**Membership assertion verification:**
When a user receives a membership assertion, he/she will follow the steps below to verify the membership assertion.
1. The receiver verifies the assertion issuer signature.
2. Compares the current time with the expired time of the assertion to see if it is still valid.
3. Check if the subject name is the sender’s name.
4. Check that the membership assertion applies to the right group.

**6. 8 Invite a friend**

**Requirement:**
- The inviting and invited users are both registered from the MSNP.
- The inviting user plays a role as an administrator or owner in the group.

To describe clearly how to add a new friend to the group, we will use Alice as an inviting user and Bob as an invited friend. Alice is owner of group SICS, and she intends to invite Bob to join the group SICS. Actually, there are two possible ways that Bob can be invited, one is to send an online invitation and the other is an offline invitation. These two invitations work differently, so we will discuss them separately.

\(^5\) Appendix 1 shows the structure of membership assertion
The above requirement should be fulfilled to generate a valid online or offline invitation.

6.8.1 An online invitation

Preconditions:
Alice should connect with the MSNP to send an online invitation to Bob through the MSNP, and Bob has to be online to make decision of accepting the invitation.

Steps of using online invitation:
1. Alice sends a request to the MSNP to generate an online invitation. The invitation contains Alice account name as the inviting user name, the group name SICS, the role of Alice playing in the group SICS owner, and the invited user name Bob.
2. The MSNP verifies the information of the invitation.
3. If step 2 succeeds, the MSNP will generate a pending invite for Bob.
4. When Bob connects to the MSNP, Bob requests all the pending invites from the MSNP.
5. Bob could deny or accept the invitation. If an invitation has been accepted, Bob is added to group SICS as a normal member in the server database, so Bob could request a membership assertion to prove that he is a member of group SICS and the group key of SICS is sent to Bob as well. However, if Bob denies the invitation, the invitation will be removed from pending invite table in the server database.

6.8.2 An offline invitation

Preconditions:
Alice should have a valid membership assertion and a certificate that are signed by the MSNP. The offline invitation is sent through Alice’s mobile device by Bluetooth [20] so both of Alice and Bob’s mobile devices should support Bluetooth. Bluetooth is a short-range wireless transmission technology so Alice has to be located very close to Bob to transfer the offline invitation.

Steps of sending an offline invitation:
1. Turn on Bluetooth from both mobile devices.
2. Alice and Bob have an agreement to establish a communication channel through Bluetooth.
3. Alice input Bob’s account name to generate an offline invitation. The offline invitation is generated by a SICS Android SAML library, which is integrated in the FriendFinder mobile application. The offline invitation contains an assertion ID, the issue date, the name of issuer, the digital signature of the issuer, the invited user name, the expired date, and a group of attributes that are group name and the role of invited user assigned. Appendix 2 shows how an offline invitation structured. The offline invitation is generated by SICS SAML library and the structure of offline invitation is
almost same as the membership assertion. The only difference between offline invitation and membership assertion is the issuer signature, offline invitation is signed by the inviter and the MSNP is the issuer of the membership assertion. Furthermore, offline invitation has shorter expiration time than a membership assertion.

4. Alice extracts her own membership assertion of SICS and group information of SICS from mobile device and integrates with the already generated offline invitation as an invitation file.

5. Alice sends her the invitation file to Bob.

6. Bob receives an invitation request from Alice, and he can make a decision to accept or deny the invitation. If Bob accepts the request, the invitation file will be transferred to and stored in Bob’s mobile device, otherwise Bob breaks the communication channel with Alice so the invitation file can not be transmitted.

7. Alice adds Bob as a temporary group member in the local mobile databases if Bob received the invitation file successfully.

The offline invitation is only valid for one day, so Bob has to register himself as a permanent group member from the MSNP in one day after received the invitation file if he wants to stay in group SICS for a longer time. However, Bob could shares his information with other friends in the group SICS when he has a valid offline invitation. In the next two sections, we are going to describe how to verify Bob’s offline invitation and how Bob can register himself as a permanent group member from the MSNP.

**An offline invite verification:**

Assume that Bob has received an offline invite file from a group owner Alice to join the group SICS. To verify that Bob is a temporary group member of SICS, Bob has to present his certificate with the MSNP’s signature and the offline invitation file.

**Steps of verification:**

1. Verify Bob’s certificate to see if the certificate is valid or not. Additionally, the issuer’s signature should be signed by the MSNP in our case.

2. Verify Bob’s offline invitation. Firstly, it checks if the offline invitation is still valid by comparing the current date and expiry date. Secondly, it checks if Bob’s account name is matched with the subject name of offline invitation. Thirdly, the group name has to be checked for the right group name. Moreover, it has to check the issuer’s signature, which is Alice. In the end, the issuer of the offline invitation should match with the subject name of the membership assertion of Alice.

3. Verify the membership assertion of Alice. To make sure that Alice is the owner or the admin of the group SICS and that the assertion of Alice is still valid. Moreover, the issuer signature should be signed by the MSNP.
Join a group by using offline invite:
As the previous part mention, Bob has to present three files to pass the verification, and the files include Bob’s certificate, Alice’s membership assertion, and the offline invite. These three files have to be checked before Bob becoming a permanent group member. After the verification has been done, the MSNP would add Bob to the group of SICS as a member. Once Alice or other group members of SICS update the group member list of SICS, Bob will be moved from the temporary member list to the permanent group list. When Bob updates his permanent group list, group SICS will be added to the group list and removed from the temporary group list. Updating a permanent group list requires the client to send a view group list request to the MSNP. The request of view a group list has been discussed in section 6.3.

6.9 Manage Group
Each group has a creator to manage the group, and the creator is the owner of the group. The owner has a responsibility to take care of the group so the owner has more rights to administrate the group than other members. As the requirement specification chapter mentioned, there are three different roles a user can play in a group, and the roles are member, admin and owner. Admin has only one extra right than normal member that is to invite friends to join the group, so in fact the owner is the only one that has the right to manage the group members. The following list states the rights owner has:
- Delete a group member
- Grant a member as admin
- Revoke the admin as member
- Delete a group
The later sections will discuss how the above actions would be processed separately and the precondition of the actions.

6.9.1 Delete a group member

Requirement:
- There is more than one permanent member in the group
- The group owner has a good network connection on the mobile device.
- The owner has to log in to FriendFinder account successfully.

Steps of delete group member
1. The owner selects a group member to be deleted, the owner should be excluded.
2. The owner sends a delete member request to the MSNP with the deleted members name, owner’s name and the group name.
3. The MSNP checks the deleted member to make sure that the owner is not deleted and verifies the request sender. The sender should be the owner of the group.
4. If the previous steps have been completed, then the MSNP removes the selected member name from the group member list in the server databases.
5. When the group members send a request for the group member lists from the MSNP, the deleted member will not anymore appear in the member list. When the deleted group member sends a request for a group list, then the group is not appearing in the group list.

The deleted member may still have a valid membership assertion after the owner kicks him/her out from the group. The valid membership assertion is stored in the SD card of the mobile device, and the user may certify himself or herself as a valid group member to other group members when he/she is under the offline state. However, the membership assertion has only one week validity in our project, and the deleted member could not require any new membership assertion after the assertion expired. Furthermore, in the FriendFinder mobile application interface, there is no such group appearing in the deleted member’s group list, so basically the mobile application does not support the deleted client to view the old group member list⁶. That means that the mobile application should not allow the removed group member to send a group member list request for viewing the old group information. We will take an example to explain this case in a clear way. Alice is the owner of group SICS. Bob and Cici are the members of the group SICS and Alice removed Bob from the group SICS, so Bob will not be anymore a group member of the SICS. The problem is that Bob may still have a valid membership assertion of the group SICS that stored in his mobile device for the offline situation. To solve the problem, we have designed the mobile application in a way that when Bob turns to be online and he sends a group list request to the MSNP, the MSNP database sends a group list to Bob for showing the group Bob belonging group and the group SICS should be not anymore displayed in Bob’s belonging group list. In this case, Bob cannot launch a group member list for viewing the group SICS’s information through the mobile application. However the potential problem of this design is that if Bob could launch an offline group member list request to all the group members for viewing the group member information of the group SICS, and all the group members of the group SICS would trust Bob’s valid membership assertion and responded their information to Bob. It is hard to solve this problem but we assume that the owner should trust all his/her invited friends, so the short time valid membership assertion would not take any harmful result for other group members.

---

⁶ The old member list means the owner of the group has kicked a member out, and the kicked out member does not allow to view the group information anymore and the group member list is called the old member list.
6.9.2 Grant a member to Admin

**Requirement:**
- There is more than one permanent friend in the group and the friend should be a normal member role.
- The group owner has a good network connection on the mobile device.
- The owner is logging to the FriendFinder account successfully.

**Steps of granting a member as Admin:**
1. The owner selects the friend who plays the member role in the group. We assume the select friend name is David.
2. The owner sends a request to the MSNP to change the role of David to Admin. The request should include the group owner name, group name and that the selected friend name.
3. The MSNP verifies that the one that sent the request is the owner of the group, and the selected friend David belongs to the group.
4. If all the above verifications succeeded, the MSNP changes the selected friend from a normal member to the admin of the group in the database.
5. When group members request the group member list, the MSNP will send the new group member list that contains Davids name and his new role. When David requests for the group membership assertion, the MSNP sends the one with the attribute of group role as admin and the old group membership assertion will be removed from David mobile device.

6.9.3 Revoke admin to member

**Requirement:**
- There is more than one permanent friend in the group and it should contain at least one admin. We assume that the admin of the group is Lily.
- The group owner has a good network connection on the mobile device.
- The owner is logging to FriendFinder account successfully.

**Steps of revoking admin to member:**
1. The owner selects Lily as the admin of the group.
2. The owner sends a request to the MSNP for revoking Lily’s admin role to member. This request should contain the owner name, the group name and Lily’s name and the new role for Lily.
3. The MSNP verifies that the one that sent the request is the owner of the group and checks if Lily is a member of the group.
4. If the request has passed the step 3, the MSNP changes the role of Lily to a member from the admin.
5. When group members request for a group list from the MSNP, a new group list with Lily’s new role will be sent out. When Lily requests a membership assertion, the
new membership assertion states that Lily is a normal member of the group will send out. The new membership assertion replaces the old one.

6.9.4 Delete a group

**Requirement:**
- A group exists in the user’s group list and the user is owner of the group.
- The owner should have a good network connection.
- The owner log in is successfully.

**Steps to delete a group:**
1. The user selects the group to delete.
2. The user sends the delete request to the MSNP. The MSNP verifies the request to see if this is sent by the owner of the group.
3. After the request passed the verification, the MSNP removes all the members from the group, but keeps the group name in the database. The reason is to avoid that any newly created group has the same name as the deleted group. The duplicated name may cause some security problem and it has been discussed in the section of 6.4.
4. After a group has been deleted, the group member requests a group lists from the MSNP, the deleted group will be removed from the group lists. No group member can retrieve their membership assertions from the MSNP through FriendFinder application anymore.

The members of the deleted group may still have valid membership assertion, so they may use it to prove that they are belonging to the group when they are in offline communication state. However, the membership assertion is only valid for maximum one week so after that no communication channel can be established by the deleted group members after the group has been deleted for one week.
7. Restful web architecture

REST is a high-level architecture style that involves many different technologies and abstract properties, so we will start with the basic concepts of REST to provide a simple example how a REST concept can be used in a normal web shop. Later, six constraints of REST are described to state the benefit REST brings to us at a high level. Based on the REST concept, section 7.3 describes how to build a restful social network, and section 7.4 explains the security issues of the implementation.

7.1 The basic concept of REST

Representational state transfer (REST) is a software architecture style introduced by Roy Fielding’s dissertation in 2000 [13]. REST is a concept that focuses on the resource and it has been widely utilized recently in many popular websites, such as Twitter, Yahoo’s web services, Amazon and Ebay. The resource is the heart of REST, but it is not a new concept only appearing in REST. The resource is the information that is exposed to the client. In the design phase, the programmer has to consider data entities and domain objects as a resource. Each resource has a unique identity to distinguish from each other, normally a web-based system uses a URI[14] to differentiate resource. Take a web shop for instance, each product can be viewed as a resource and the product ID can be used in the URI to separate the products, which may look like www.webshop.com/product/productID. The productID has been highlighted in red color, that indicates the productID. The productID is not a fixed term but is structured with different unique numbers according to the product ID. When a web server receives a GET request for a URI that is www.webshop.com/product/12346, web server will search the resource that has a product ID equals 12346 from the database and display the resource information to the client. The resource can be formatted in various ways, HTML, XML and JSON are the most commonly ways but a picture or plain text can be represented resource as well. A client could decide in which way the resource should be formatted.

We have mentioned a “Get request” in the previous paragraph and there are three more action requests. All of these actions are based on the HTTP methods, this is discussed in this paragraph. The actions of REST could help a client to view and manipulate the resource, and the following example continues to take the web shop as application and displays how a resource can be presented and modified.

1. A client desires to see a camera’s information, and the camera’s product ID is 1234.
The client calls a *Get Request* to retrieve a URI that matched with productId equals 1234, and the URI may look like `www.webshop.com/product/12346`.

2. *Admin of the web shop wants to change a product name and price.*
The client calls *Put Request* to modify an existing resource information. The following URI is called to change the original product name and price and the pink color of the URI has highlighted the information the client has to transfer as parameters.
`www.webshop.com/productInfo/{NewName/NewPrice}`
In fact, the client will only call URI `www.webshop.com/productInfo/` with the parameters of a new name and new price to update the new information. The first URI of this section with the pink color is used to instruct the user which parameters have to send.

3. *Admin of the web shop wants to add a new product to the web shop.*
The client calls *Post Request* to add a new product to the product list. The productId is the parameter that has to be sent with the URI of `www.webshop.com/product/`.

4. *Admin of the web shop removes an old fashioned product from web shop.*
The client calls *Delete Request* to remove an old product from the product list. For instance, a product 123 is the one needs to remove, so the client could directly call the URI of `www.webshop.com/product/123`.

Many textbooks suggest that the client should view the resource as a noun and the four actions of REST as verb to manipulate the resource. The resource and four actions are the core technique of REST. In the later section, we will discuss the constraints of REST to discuss the reasons to utilize REST.

7.2 *REST constraints*
Six constraints have been applied to the REST architecture, each of these constraints can be implemented independently. The following subsections elaborate the six constraints and the benefit of these constraints are brought out.

7.2.1 *Client-Server*
The client-server constraint allows the client not concern about data storage, all the data are stored on the server side. Moreover, the server and the client can be developed separately, the server does not need to care very much about the platform the client is using as long as a uniform interface is built between the client and the server. Obviously, the good thing to follow this constraint is the portability of the client code is improved and the development of server is simpler.
7.2.2 Stateless
The second constraint takes place in the interaction between client and server. During the communication, each request from the client has to contain all the information the server needed, so that server can fully understand the request. Every time the server answers it has no knowledge of the history, so the previous request error will not influence in the current response. The advantage of the stateless constraint is to arise a reliable response from partial failures. Moreover, the constraint requires the server to be stateful. As we mentioned in the previous section, all the resource can be addressed by URI’s at the server-side, and each of these resources is presented a state of the server. In this case, there is no need to store the state between requests, so the server component could release the resource quickly, so the scalability is increased.

7.2.3 Cacheable
A lot of redundant information may be resent between the client and server to insure that each request includes all needed information for the server. This redundant information can take plenty of resources, and the cache constraint eliminates the interaction of the client and server. Each response can be labeled as cacheable or non-cacheable to a client, the client will store those cacheable responses in order to reuse the data of the response instead of sending equivalent requests. The good thing of adding the cacheable constraint is the communication efficiency is improved and also the scalability, however, it may decrease the reliability when an old cached data is significantly different from the data that directly gained from server.

7.2.4 Uniform interface
The uniform interface is one of the most important features to distinguish REST from other web-based architecture styles. The uniform interface is used to connect with different components. Instead of delivering specific information to an application, the information will be sent in a standardized form. To implement a uniform interface, we use the basic REST principles, which have been already mentioned in the section of 7.2; the resource has to be identified and manipulated through four actions. The advantage of this constraint is that all the provided services are decoupled, which means the services could be executed independently. Moreover, visibility of interactions is improved as well.

7.2.6 Layered System
The other constraints are called the layered system which is primarily used to satisfy the requirement of Internet-scale. A layered system dispatches different components
to a hierarchical architecture. In this architecture, the client may connect to an end-server indirectly. The intermediary server may increase the web performance by enabling load balancing and by delivering shared caches. A security solution can be added into the intermediary server to provide identity authentication or protect data transferring.

7.2.7 Code on demand

The code-on-demand is the only optional constraint among the other six. The server may provide optional functionality to client by transferring executable code. A client could request the most important features to be pre-implemented in the beginning, the additional features can be launched later according to the users’ demand. Therefore users can extend new features more flexibly. However, many new features become invisible to the users, so users may lose the chance to experience it.

Based on the six constraints, we could make a brief conclusion that REST provides good performance and scalability. Additionally, REST demands a well designed URI, and this may help the programmer to have a clear picture of data architecture.

7.3 Build a restful social network.

Designing the resource is the core work to design a restful website, so it is necessary to consider all the necessary resources that the website might need. In this section, we will consider what kind of the resources the client may need in a social network, and with which URI the resource could be represented in a clear and understandable way.

As the basic concept of REST mentioned, a resource can be considered as a core design work in the REST architecture style. A resource is the data that the server exposes, so it can be considered as a noun and the action of manipulated resource can be implemented as one of the four REST actions. In order to cover all the resources, we could list all the data that is transferred between the client and the server. Moreover, the corresponding URI is going to be designed. In the below URI, the pink color represents the variable attribute.

1. The client registers a new user

Clients send the account name with a Get request to the MSNP, and a statement returned to inform the client if the chosen account name is available or not.

http://ServerIPAdd:8080/SwinServer/user/exist/userName (Get request)

For instance, the client chooses an account name Alice, the client sends a Get request by calling below URI:

http://ServerIPAdd:8080/SwinServer/user/exist/Alice (Get request)
If Alice has been registered as an account name by some other user, then the statement will return “True”, otherwise, a “False” is displayed.

The user sends the available **account name** to the MSNP, and the BSF sends the **IMPU value** to the MSNP.

http://ServerIPAdd:8080/SwinServer/user/ (Post request)
Take Alice as an example, Alice chooses her name as the account name and the name is still available, so Alice sends a post request with the chosen name through the URI below to add herself to the database, and BSF will transfer Alice’s IMPU value to the MSNP as well.
http://ServerIPAdd:8080/SwinServer/user/ (Post request with the parameters of Alice account name and the IMPU of Alice)
Alice retrieves a MSNP **signed certificate** through below URI.
http://ServerIPAdd:8080/SwinServer/cert (Post request)

2. The client views the group list to see which groups the client has involved.
The client sends the **account name** with a group list Get Request to the MSNP.
The MSNP returns **group list** to the client.
http://ServerIPAdd:8080/SwinServer/grouplist/.userName (Get request)
For example, the client name is Alice, then the client will call the below URI:
http://ServerIPAdd:8080/SwinServer/grouplist/Alice
A group list will be sent to Alice, and it includes all the groups that Alice participates.

3. The client obtains membership assertions from different groups.
The client sends the **account name** and the **group name that client belongs** to the MSNP
The MSNP sends **a group membership assertion** to the client.
http://ServerIPAdd:8080/SwinServer/assertion/userName/groupName (Get request)
For example, the client name is Alice and she belongs to the group SICS. To retrieve Alice’s SICS membership assertion, the following URI can be called through a get request.
http://ServerIPAdd:8080/SwinServer/assertion/Alice/SICS (Get request)
Alice could obtain her membership assertion of the group SICS.

4. The client views all the group members information from a chosen group.
The clients send the chosen **group name** to the MSNP.
The MSNP returns **a group member list** to the client.
http://ServerIPAdd:8080/SwinServer/memberlist/groupName/ (Get request)

5. Owner changes group member’s role.

---

*View GBA protocol section 4.3.2 & Registration section 6.2.*
The owner sends the **group name**, the **selected member's name**, the **new role** assigned to the new member, and the **owner's account name** into a Put Request to the MSNP by following below URI.

http://ServerIPAdd:8080/SwinServer/role/ (Put request)

For example, Alice is the owner of group SICS and she wants to change the group SICS member Bob’s role to admin. Alice has to send four parameters which are group name SICS, chosen member name Bob, new role for chosen member Admin, and the owner name Alice with a put request through the URI below.

http://ServerIPAdd:8080/SwinServer/role (Put request)

6. Owner deletes a selected member from the group.
The owner has to send the **group name**, the **deleted member's name** and the **owner name** to delete a selected member.

http://ServerIPAdd:8080/SwinServer/role/groupName/ownerName/ownerName/ (Delete request)

For example, Alice is the owner of the group SICS and she deletes a SICS member Bob by sending a delete request to the MSNP with the URI below.

http://ServerIPAdd:8080/SwinServer/role/SICS/Bob/Alice (Delete request)

7. Owner deletes a group.
The owner has to send the **group name** and the **owners name** of the group to the MSNP by calling the URI below with a delete request.

http://ServerIPAdd:8080/SwinServer/group/groupName/userName/ (Delete request)

8. A normal group member or Admin quits a group.
The client has to deliver the **group name**, the **client account**, and the **name of the group owner** through a Delete Request according to the URI below.

http://ServerIPAdd:8080/SwinServer/role/groupName/userName/ownerName/ (Delete request)

9. Before the client creates a group, she/he sends a request to the MSNP to verify if the group name is still available.
Client has to send the **chosen group name** to the MSNP.

http://ServerIPAdd:8080/SwinServer/group/groupName (Get request)

This page only returns "true" or "false" to inform if the group name is available or not. True means group name is available, and false is the opposite.

If the chosen group name is available, client will send a post request with the **chosen group name** and the **name of the group creator** through the URI below to the MSNP.

http://ServerIPAdd:8080/SwinServer/group (Post Request with the parameters of the chosen group name and the name of the group creator.)

10. Admin/owner generates invites through the MSNP.
Client has to send his/her name, the friend's name and the group name to the MSNP for generating a new invitation. The three attributes are sent when the URI below is called by the client.

http://ServerIPAdd:8080/SwinServer/invites (Post request with the parameters of the client name, friend’s name and the group name)

For example, Alice is the owner of the group SICS, and she wants to invite the friend Bob. Alice has to send a post request with her name, group name and Bob’s name as parameters to the MSNP through the URI below.

http://ServerIPAdd:8080/SwinServer/invites

11. Client view all the pending invitations.
Client sends his/her name to the MSNP for all the online pending invitation.

http://ServerIPAdd:8080/SwinServer/invites/list/username (Get request)
The MSNP responds all the pending invites of the client through the URI above.

12. Client removes a pending invitation.
Client sends own name, the inviter's name and the group name to the MSNP for the deleting the pending invitation. The MSNP will remove the chosen invitation from database.

http://ServerIPAdd:8080/SwinServer/invites/inviteeName/inviterName/groupName (Delete request)

13. The MSNP adds a temporary group member to the permanent group list. The temporary group member holds an offline invitation, which allows the user to communicate with the group member for only a short time. However, the temporary group member could call the URI below to add himself or herself as a permanent group member.

http://ServerIPAdd:8080/SwinServer/verify (Post request with the parameters of the client name, group name and offline invite.)

In order to help the administrator of the social network to manage all users, we have designed the URI below.
1. Admin views the entire created groups information from the database through the URI below.

http://ServerIPAdd:8080/SwinServer/group/list (Get request)

2. The admin views all registered users from the database by calling the URI below.

http://ServerIPAdd:8080/SwinServer/user/list (Get request)

3. The admin views all the pending invites through the URI below.

http://ServerIPAdd:8080/SwinServer/invites/list (Get request)
The three resources are directly presented in the URI above, which brings convenience to the administrator of the social network, however, it initiates a security issue that every registered client who passed the GBA authentication would view these three resource without challenge. In the section 7.4, it discusses how we can protect our resource.

7.4 Security issues

Each resource has been signed to a corresponding URI, so the user could simply call the URI with some request method to view or manipulate the content of the resource. Consequently, everyone could manipulate the content of the resource through a web browser. In the chapter 4, we have discussed the GBA authentication, and it has been implemented into HTTP digest authentication so that each time user calls URI through a browser, an HTTP digest authentication window will pop up for the user name and password. When the user and the BSF respond with a correct account name and the IMPU, the resource can be manipulated. In our application, the GBA authentication is not enough, since it could only filter the registered from unregistered users to manipulate the data, but not control the access of registered user. For example, Alice is a registered user of the social network and she is the owner of the group SICS, but she does not belong to group UU, so Alice should not have the right to view group UU’s member information. Alice could easily pass the GBA authentication to prove that she is a registered user, but the GBA authentication does not prevent group UU’s information to be exposed to user like Alice who is not belonging to UU. To protect the confidential resource, we have added store procedures to control users’ access. Each time, a user needs to manipulate or view resource, a store procedure is executed to verify that the user has the right to access the resource. The store procedures will be discussed in chapter 9. Based on the user information and the group information, different stored procedures can be designed to authenticate users and determine who has access to the resource. Section 9.2.2 provides the details of this solution.

To authenticate the user and the relationship between the user and group, store procedure might be one of the simplest ways to solve the problem, but some vulnerable factors may still exist for malicious users to discover. Dan Forsberg from Nokia research center had a comprehensive research on the RESTful security to discuss the security issue of a restful web application and clarify the solutions[15]. REST security issue is not only discussed in a theory level but also in an implementation level, OAuth [16] is one of them, which may be considered to be used in the later implementation. On our stage of the project, we did not focus on the REST security issue.
8. Server Implementation

This chapter describes the implementation work of the project, which includes the project framework, work environment, the structure of the project and some code examples. The project implementation description only focuses on the REST server side. The client side implementation is described in the Sun Lei thesis [2].

8.1 Framework & Environment

There are many frameworks that support the restful concept, and we have chosen Jersey as our project framework, which involves the concept of REST. Jersey is the open source JAX-RS (JSR 311) Reference Implementation that is developed by Sun. We have chosen Glassfish3 [19] as an application server, Eclipse Indigo as a development environment and Maven as the software project management tool.

The REST relevance of classes have been defined using annotations by JAX-RS. We will shortly introduce the most common annotations, especially the annotations that is used in the project.

The common annotation list:

@PATH(your_path): Sets the resource URI. The base URI is the application name, which can be configured from the web.xml configuration file.

@POST, @GET, @PUT and @DELETE: Indicate that the methods will answer to an HTTP POST, GET, PUT and DELETE requests.

@Produces: It indicates which MIME type is used by a method annotated with @GET. The media type can be set as XML, JSON or others.

@PathParam: Used to extract resource values from the URL.

8.2 Project Structure

In the server side implementation, we have separated the project into several parts. The first part is used to communicate with the BSF to execute the GBA protocol and implement the HTTP Digest. The second part is used to define the resources and the resource structure.

In the second part, the JAXB annotation of the XmlRootElement has been used to serialise the resource structure so the output of the resource can be presented in a
XML format. Take user info as an example, we could simply define a UserInfo class, and a userName, impu are the attributes. The code 8-2 illustrates the class of UserInfo, and the table 8-2 shows the output of the XML structure of different resources.

```java
@XmlElement
class UserInfo {
    private String userName;
    private String impu;
    UserInfo(String _userName, String _impu) {userName=_userName; impu=_impu;}
}
```

**Code 8-2 UserInfo Example**

Six resources have been defined in the second part, they are userInfo, groupInfo, pendingInviteInfo, groupRoleInfo, memberGroupRoleInfo and memberRoleInfo. UserInfo, groupInfo and pendingInviteInfo deliver user information, group information, online pending invitation information. Table 8-2 simply shows the six resources and the XML structure of these resources.

<table>
<thead>
<tr>
<th>ResourceName</th>
<th>XML Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UserInfo:</strong></td>
<td>&lt;userInfo&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;userName&gt;Alice&lt;/userName&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;impu&gt;<a href="mailto:Alice@Ericsson.lab.com">Alice@Ericsson.lab.com</a>&lt;/impu&gt;</td>
</tr>
<tr>
<td><strong>GroupInfo:</strong></td>
<td>&lt;groupInfo&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;groupName&gt;SICS&lt;/groupName&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;groupKey&gt;KEY&lt;/groupKey&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;nextSchedualUpdateGroupKey&gt;KEY&lt;/nextSchedualUpdateGroupKey&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;firstUpdateSessionKey&gt;KEY&lt;/firstUpdateSessionKey&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;updateIntervalForGroupKey&gt;KEY&lt;/updateIntervalForGroupKey&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;updateIntervalForSessionKey&gt;KEY&lt;/updateIntervalForSessionKey&gt;</td>
</tr>
<tr>
<td><strong>PendingInfo:</strong></td>
<td>&lt;pendingInviteInfo&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;groupName&gt;SICS&lt;/groupName&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;inviteeName&gt;Bob&lt;/inviteeName&gt;</td>
</tr>
<tr>
<td><strong>GroupRoleInfo:</strong></td>
<td>&lt;GroupRoleInfo&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;groupName&gt;SICS&lt;/groupName&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;role&gt;Admin&lt;/role&gt;</td>
</tr>
<tr>
<td><strong>MemberGroupRoleInfo:</strong></td>
<td>&lt;MemberGroupRoleInfo&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;role&gt;Admin&lt;/role&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;groupName&gt;SICS&lt;/groupName&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;userName&gt;Alice&lt;/userName&gt;</td>
</tr>
<tr>
<td><strong>MemberRoleInfo:</strong></td>
<td>&lt;MemberRoleInfo&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;memberName&gt;Alice&lt;/memberName&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;memberRole&gt;Admin&lt;/memberRole&gt;</td>
</tr>
</tbody>
</table>

**Table 8-2 Six resources and XML structure**
The third part of the project is a connector to establish a bridge between the web server and the database. When a client sends a request to view or manipulate the resource, this part of the project is invoked. For example, a server receives a delete request from a client that is the owner of a group and he/she wants to delete the group. Consequently, all the group members have to be removed from the group. The function below belongs to one of the connector classes called GroupService, and the class will be instantiated in the serverRest part that is the fourth part of the project and the function below will be invoked so that all the members can be removed from the group. In the example code 8-6-2-A, “delete_group_by_owner” is a stored procedure that checks the membership of the owner and that removes all the members from the group, so that the members cannot update their membership assertion from the MSNP after the group has been removed.

```java
public void deleteGroupByOwner(String groupName, String ownerName)
    throws SQLException {
    procedure = connection.prepareCall("{CALL delete_group_by_owner(?,?)}");
    procedure.setString(1, groupName);
    procedure.setString(2, ownerName);
    procedure.execute();
    procedure.close();
}
```

**Example code 8-6-2-A: A connector between the server and the database.**

The fourth part is delivering the REST web service, so the server could respond according to the client request. JAX-RS defines annotations that we have used in this part of the project. The code example 8-6-2-B explains how the REST service works. Every user is allowed to obtain his/her own information. When a REST server receives a request from a client to retrieve the user's own information, the server will check whether the user has the right to send such a request. If yes, it will return the user's information. Otherwise, it returns empty information. The function of the below example code is used to implement a response for user Get request and verifies if the resource is going to be presented or not. The first annotation @Path("/user") shows the resource URI is [http://localhost:8080/SwinServer/user](http://localhost:8080/SwinServer/user) and UserService has been instantiated so that REST web service could connect to the database. The second @Path shows the parameter that follows [http://localhost:8080/SwinServer/user](http://localhost:8080/SwinServer/user) URI. @Produces annotation indicates the resource will present in XML and JSON format. The code after @Produces is used to set the user IMPU value based on what the BSF responded. The user name sent to the REST server and the IMPU value sent by the BSF would match the user name and the IMPU stored in the database, then the user information could be displayed to the client.
import javax.ws.rs.FormParam;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
@Path("/user")
public class UserRest {
    private static UserService userService = new UserService();
    @GET
    @Path("/{username}")
    @Produces({ MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON })
    public UserInfo getUser(@PathParam("username") String userName) throws SQLException
    {
        String userIIMPU = AuthenticationFilter.USER_IIMPU;
        return userService.checkUser(userName, userIIMPU);
    }
}

Example code 8-6-2-B : REST service of user resource.

The last part of the project contains all the different tools that may be utilized in the previous parts. The tools can be different XML parsers that disassembles the different resources from the XML structures documents to the normal text that can be displayed according to the mobile application interface design, so several XML resource handlers have been implemented. There is another tool that is used to connect the web server to the database, so the resource can be viewed and manipulated.

8.3 Server configuration
In order to develop the REST service, we have set up the environment by configuring two files, and in this section, we focus on describing the configuration of these two files.

8.3.1 Configure web.xml file
To manipulate or display the resource, we have to direct all the REST requests to the Jersey container and we could configure the application’s web.xml file to define a servlet dispatcher. File configuration 8-3-1-A shows how the file has been edited. Moreover, the second initialization parameters have been defined to indicate where the resources have been placed, so that Jersey could easily look for the resource. In our project, the resource has been assigned in the Java package of se.sics.FriendFinder.
<servlet>
  <servlet-name>friendfinder</servlet-name>
  <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
  <init-param>
    <param-name>com.sun.jersey.config.property.resourceConfigClass</param-name>
    <param-value>com.sun.jersey.api.corePackagesResourceConfig</param-value>
  </init-param>
  <init-param>
    <param-name>com.sun.jersey.config.property.packages</param-name>
    <param-value>se.sics.FriendFinder</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>friendfinder</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>

**File configuration 8-3-1-A. Define servlet and indicates the place of resource.**

In section 8.3, it was mentioned that the server requires an HTTP digest authentication when a client request has been sent. To build up the HTTP digest authentication, a filter needs to be set up in the web.xml file. File configuration 8-3-1-B demonstrates how a filter can be added. The first parameter of the filter is used to set the name of the filter, and the filter-class indicates Java package name of the authentication filter. Three initialization parameters have been declared. The first two initialization parameters are used to point out the URL address of the GBA NAF and the BSF. The NAF is actually running the mobile social network server side, so the URL of the NAF should point to the server, which is localhost. BSF server is operated by Ericsson lab, and the URL of the BSF server is indicated. The Jersey filter will retrieve the IMPU values from the URL below: [http://bsf.labs.ericsson.net:8080/bsfv2/requestBootstrappingInfo](http://bsf.labs.ericsson.net:8080/bsfv2/requestBootstrappingInfo).

The last initialization parameter is a api-key [17] that can be obtained from Ericsson lab and it is used to build up the GBA NAF in the server side.
<filter>
    <filter-name>AuthenticationFilter</filter-name>
    <filter-class>com.ericsson.research.gba.naf.AuthenticationFilter</filter-class>
    <init-param>
        <param-name>naf-fqdn</param-name>
        <param-value>localhost</param-value>
    </init-param>
    <init-param>
        <param-name>bsf-url</param-name>
        <param-value>http://bsf.labs.ericsson.net:8080/bsfv2/requestBootstrappingInfo</param-value>
    </init-param>
    <init-param>
        <param-name>api-key</param-name>
        <param-value>VkiPE6FcBv8uE4drHq3pj1nUkUpeIo6BFa6JhPKT</param-value>
    </init-param>
</filter>

File configuration 8-3-1-B: Define a filter to implement digest HTTP authentication.

8.3.2 Configure pom.xml file

To implement the mobile social network server, we have to import some libraries from outside, and these libraries have to be added as dependencies. File configuration 8-3-2 demonstrates an example of adding a dependency. Developers can manually add dependency by using below command:

mvn install:install-file -DgroupId=com.sun.jersey-DartifactId=jersey-core -Dversion=1.3 -Dpackaging=jar -Dfile=/path/to/jarfile

Developer could directly edit the pom.xml file to add dependency as File configuration 8-3-2 shows.

<dependency>
    <groupId>com.sun.jersey</groupId>
    <artifactId>jersey-core</artifactId>
    <version>1.3</version>
</dependency>

File configuration 8-3-2: Added dependency
9. Database design

In chapter 8, we have discussed the concept of REST, and a good database design would give a programmer a guideline to design resources. This chapter is going to show different entities and their relations in the server side. The client side database has been focused in Sun Lei’s thesis [2]. In the server database, the stored procedures have been implemented to achieve faster search and user authentication. In the end of the chapter, the database project is discussed together with the convenience programs provided to the users.

9.1 Entities and entities relations

In the database design, we have created five tables to match the server needs. The five entities are msnp_info, user_info, group_info, user_role_for_group and pending_invite. The names of these entities have been chosen to self-explain what data the entities would store.

The entity of msnp_info only stores a digital certificate of the MSNP and a private key of the MSNP. A third party signs a digital certificate for the MSNP and the certificate is the identity proof of the MSNP. The private key of the MSNP is the key that the MSNP could use to generate a digital signature to authenticate the registered user. User_info contains the user name and the IMPU value. User name is the primary key of the table to identify each column of the table and IMPU is IP Multimedia public identity that is used to verify the user's identity through the GBA protocol. Group_info is a table that is used to store a group information such as group_id, group key and etc. Group_id is the identification of the group so it is designed as a primary key of the group_info table. Additionally, group_info table includes some confidential information as well such as different group keys: interval_groupkey_time, next_update_group_key_time, interval_sessstionkey_time, first_update_sesstionkey_time. The reason to have various keys is to achieve a function that a user can keep a secure communication with group members when they are in the offline state. In the documentation of social application design, by Ludwig Seitz[18], chapter 6.7.1 describes how these keys are used. In order to protect the group information, a user_role_for_group entity has been created to connect user_info and group_info and verify which user has the right to access group_info. User_id, group_id and user_role are the attributes involved in user_role_for_group table. The table of user_role_for_group indicates which group a user belongs to and which role the user plays, and when the user requests an access or manipulation of data from group_info table, the user_role_for_group table will be invoked first and check if the user is belonging to the group and has the right to modify the current data. The last table of pending_invite is used to store unresponded invite, so it is necessary to include who
has sent the invite and to whom the invite is sent, which group the invitee has been invited. The following entity relation figure 9-1 shows how the tables connect with each other.

To make sure that every user and group are registered in both user_role_for_group and pending_invite, the constraints have been established. In the user_role_for_group, the user_id is connected to the user_id of user_info table as a foreign key and the group_id of user_role_for_group is a foreign key as well that refers to group_info table. The entity of pending_invite is connecting with user_info and group_info too. The invitee_id and inviter_id of pending_invite are both foreign keys of user_id of user_info. The group_id of pending_invite is referred to the group_id of group_info.

**Figure 9-1 Entities relations**

### 9.2 Stored procedure

In this section, we have listed the reasons why stored procedures have been implemented and explain several representative stored procedures.

#### 9.2.1 Advantage of stored procedures

A stored procedure normally contains multiple SQL statements, and it allows procedural code of the applications to be stored in the databases. In MySQL database system, a stored procedure can simply be executed by using the CALL statement. There are several reasons that make us to use a stored procedure and the following list provide a summary of the advantage of stored procedures.
1. **Atomic statements**:
There might be several SQL statements involved in a stored procedure. Once the stored procedure has been compiled in the SQL server, either all the statements of stored procedure will be executed or none. This makes the transaction atomic. In our project, we have a stored procedure called `delete_group_by_owner`, the procedure will first validate the role of user who is executing this stored procedure and then decides if the user has the right to remove all the group member from `user_role_for_group` table and delete all the group pending invites. Without using stored procedure, it may easily miss one of these SQL statements and the data structure may be inconsistent.

2. **Faster execution**:
The SQL statements of stored procedure can be compiled and optimized from a database server in a better way than standard SQL statements. If a stored procedure has been invoked many times, it will save even more time by following the execution plan of the database. Furthermore, the entire SQL statements of stored procedure will be directly sent to the database server to execute, so the network traffic is decreased and no need to pass chunks of SQL code over a network.

3. **Move the burden with database**:
The database system is famous for searching and organizing the data. Instead of running all the code in the web server, we could move some to the database server. Stored procedure supports condition statement IF ELSE, logical operators and For, WHILE expression. In our project, we often use IF ELSE statement to verify user role. For instance, we have a stored procedure called `change_user_role`, and this stored procedure is used to help the owner of the group to change a selected group member’s role. When a user executes this stored procedure, the user is required to input his/her name, the chosen member name, the new role for the member and the group name. If the user ID is matched with the owner of the group, then the update statement can be executed to change the role for the selected group member. The store procedure 9-2 shows how `change_user_role` stored procedure looks like.

```sql
CREATE PROCEDURE change_user_role( owner_id VARCHAR(266), ck_user_id VARCHAR(266), ck_group_id VARCHAR(266), change_user_role ENUM("Admin", "Member"))
BEGIN
DECLARE oname VARCHAR(266);
SET oname = (SELECT ruser_id FROM user_role_for_group WHERE ruser_role = "Owner" AND rgroup_id = ck_group_id);
IF owner_id = oname THEN
   UPDATE user_role_for_group SET ruser_role= change_user_role
   WHERE ruser_id = ck_user_id and rgroup_id = ck_group_id;
END IF;
```
Store procedure 9-2: change_user_role

4. Security
Stored procedures can also separate the user from the data layer and the business logic layer. For example, we could allow user access to a stored procedure that generates only one group member’s information, and the user is never allowed to see all the group information directly. Since group_info is the table contains confidential data, so we implement security solutions on it to prevent a user to have direct access. A view_group_information stored procedure has been added as a business logic layer that users can access. In this way, we force the user to go through the business logic layer instead of the data layer.

In addition, stored procedure can protect against injection attacks. Take the stored procedure of change_user_role for example, Alice can input the command “CALL change_user_role(“Alice”, “Bob”, “SICS”, “Admin”):” to change Bob’s role as admin. Although the attacker tries to insert SQL statement into a stored procedure, stored procedure will treat it as a parameter, which is not much harmful to database.

5. Modularization
As mentioned in the security section, stored procedures can work as the business logic layer. When developing a project, the project team developers could deliver the business logic layer to the database and the stored procedure can be designed according to the customer needs.

9.2.2 Access control
In section 7.4, we have mentioned one of the REST security issues, in this section we will discuss some of our stored procedures to see how to use the stored procedures to avoid this issue. Take new_pending_invite for instance, this stored procedure can be used by the owner or admin of the group to send an online invitation to other friends. The user has to input three parameters before invoking this stored procedure, one is the user name, and the last two are invitee name and the group name. When client sends a POST request to add a new pending invitation to the database, these three parameters have to be included in the request as well. In the mobile application interface, clients only need to enter their friends names and press the online invite button, which has been designed as a trigger to send a POST request with the friends’ name and the other two parameters to a web server. The new_pending_invite stored procedure will be executed by the web server from the database to see if a new invite can be created or not.
CREATE PROCEDURE new_pending_invite(
IN ninvitee_id VARCHAR(266),
IN inviter_id VARCHAR(266),
IN nigroup_id VARCHAR(266)
)
BEGIN
DECLARE inviter_role_correctness INT;
SET inviter_role_correctness = (SELECT COUNT(*) FROM user_role_for_group WHERE group_id = nigroup_id AND ruser_id= ninvitee_id AND (ruser_role=1 or ruser_role=2));
IF inviter_role_correctness=1 THEN
  INSERT INTO pending_invite (invitee_id, inviter_id, igroup_id)
  VALUES (ninvitee_id, inviter_id, nigroup_id);
END IF;
END

**Stored procedure 9-2-2-A create a new pending invite**

In the stored procedure 9-2-2-A shows the *new_pending_invite* stored procedure. There are three parameters that with keyword “IN” in front are the parameters the users are required to input when the stored procedure is invoked. The keyword “BEGIN” indicates the start point of the function. *inviter_role_correctness* is a variable that verifies if the inviter exists in the group as owner or admin. Furthermore, the *inviter_role_correctness* has been defined as INT type. The result of the SELECT statement is assigned to *inviter_role_correctness*, and the statement searches whether a user exists that matches with the WHERE condition. In the WHERE condition, it searches the data that *ruser_role* matched with number 1 or 2 this is because the attribute of *ruser_role* is designed as an enum type so that number 1 means owner and 2 means admin. If the result of SELECT statement equals 1 means the inviter is the owner or admin of the group. The second part of the stored procedure starts with IF, ELSE and END IF conditional statement to determine if the *new_pending_invite* can be created or not. The keyword END shows the stored procedure code has been completed. In the functional requirements of the project, a mobile web server has to guarantee that the invited friend is registered user before inserting a new online invitation into the database. The stored procedure of *new_pending_invite* does not check if the invitee is a registered user or not, that is because several constraints have been set up between *user_role_for_group* and *user_info*. Both of invitee and inviter attributes have to exist in the *user_id* of *user_info* so that we do not need an extra conditional statement to determine if the invitee is a registered user or not.

Assume that Alice is an owner at group SICS and Bob is a registered user in the mobile social network. Alice enters Bob’s account name to send him an online invitation by pressing online invite button through mobile application interface. The application sends a POST request to the web server with Alice’s account name, the
name of the group SICS and invitee name Bob’s account. The web server transfers the three parameters to new_pending_invite stored procedure. The stored procedure verifies that a user that has a user_id as Alice’s account name in group SICS and with group role as owner or admin is existed. In this case, Alice is matched with all the requirements so a new online invite has been created for Bob.

The stored procedure of 9-2-2-A is one example that would control user access to insert a new online invitation. In the project, there are nearly 20 stored procedures that has been designed to control user access. View_group_information is another stored procedure that judges who is allowed to obtain a group information from group_info. The information of group_info contains a lot of confidential data, which should not be exposed to the public, so a view_group_information stored procedure has been created as the stored procedure 9-2-2-B shows. The 9-2-2-B stored procedure first verifies if the user exists in the group or not, if the user exists then only the chosen group information will send back to answer the GET request from user.

CREATE PROCEDURE view_group_information(  
    IN vgroup_id VARCHAR(266),  
    IN vuser_id VARCHAR(266))  
BEGIN  
DEQUEUE user_exit INTEGER;  
SET user_exit = (SELECT COUNT(*) FROM user_role_for_group WHERE user_id = vuser_id AND rgroup_id=vgroup_id);  
IF user_exit > 0 THEN  
SELECT * FROM group_info WHERE group_id = vgroup_id;  
END IF;  
END

**Stored procedure 9-2-2-B: view a group information**

### 9.3 Trigger

A trigger has quite a similar technique to a stored procedure, it is a small program developed by a database programmer and it is embedded in the database to control the association of the tables. Once the defined condition of the trigger has been matched the trigger will be executed. In our project, we have taken advantage of the trigger as well and the trigger we have been displayed in the trigger 9-3. This trigger is used to generate an attribute first_update_sessionkey_time value when a group is created in the group_info table. The first_update_sessionkey_time is automatically generated as the time type and it records the time when the group has been created and plus the interval_sessionkey_time. The trigger has saved a lot of time for developer to generate an accurate value for first_update_sessionkey_time.
CREATE TRIGGER insert_groupID
BEFORE INSERT ON group_info
FOR EACH ROW
BEGIN
SET NEW.first_update_sessionkey_time= adddate(localtimestamp, interval new.interval_sessionkey_time hour_second);
END

Trigger 9-3 Generate first_update_sessionkey_time

9.4 Database implementation

In the project, we have chosen MySQL version 7.0 as our database system. To deliver a user friendly program, we have built up a Java program that creates the database, tables, stored procedures and triggers. Users may only simply input their database password and run the Java program, the database and all other entities will be created. This Java program includes several parts which are MySQL statements, a database connector and a database builder. The MySQL statement class includes all the MySQL statements that create the database, tables, stored procedures and triggers. The database connector records the database root user name and password. The last part, the database builder is the class that the user could run to set up a social network database.
10. Conclusion

In this section, we are going to review what functions have been implemented in this project and what work has to be done in the future.

10.1 Implementation

In this project, a mobile social network has been implemented and two major security solutions are integrated into the social network. These two security solutions are the GBA authentication and proof of membership. The GBA is a protocol that is used to authenticate a mobile user’s identity through a SIM card. The mobile social network provider does not anymore require a user in a traditional way to enter user name and password to prove his or her identity but communicates directly to a third party for a cryptographic key. The user may retrieve the same cryptographic key from the local mobile device SIM card. Based on this cryptographic key, authentication can be achieved without putting an additional burden on the user. Another security solution is designed so the users that can prove their identity to their group members without network connectivity. SAML assertion has been used to certify group memberships and offline invitation, and it has been embedded in the mobile phone so that the mobile user can generate and verify an offline invite through the mobile application without relying on any third party support.

Moreover, we have employed a modern framework Jersey to build up our mobile social network server, and this thesis has declared the reasons of choosing the REST architecture. The REST concept is simple so that the developer can easily understand and it provides good performance and scalability too. During the implementation, the server database design is another important task that has been discussed in the paper. The design work is not only focused on creating the needed entities but also stored procedures. Therefore, some of the REST authentication issue has been solved. The performance of the web server has been improved as well after introduction of stored procedures.

10.2 Future work

This project can be developed in many different aspects in the future, and the work is going to be continued by the security team of SICS.

From the functional point of view, the project has to build up an ad-hoc network between the group members, so that each group member could communicate with others when they are in the offline state. Moreover, the map function can be added, so the user can easily view their friends’ locations through a map. Before the product
can benefit to other commercial products, the real SIM card has to be used instead of software SIM card, so the BSF could authenticate the user's real SIM card and identity. In the GBA protocol implementation, we directly use user’s IMPU value as a password and stored in the mobile social network database, and the IMPU is a unique number acquire from the BSF. The unique IMPU value may expose the user’s identity, so that the pseudonyms should be generated through the IMPU from the BSF in the later implementations to protect user’s privacy.

There is more research work can be done in the future too. In the implementation work, we have used a very simple way to deal with user authentication to build a simple access control of the resource. However, there is a lot to do in developing REST security. The REST security issue is not going to be less important because we do not focus on it, this framework has been widely used recently and the REST concept is a new trend for many websites. When we work on the REST security issue, we found many research papers [15], [16] that have been done in this field, but there is no accurate standard for user authentication in the REST concept yet. In the future, it would be valuable to standardize the way of access control. In the project implementation, we have built a small communication channel for the offline user to transfer an offline invitation to friends through Bluetooth. However, malicious user may pretend to be user’s the friend to defraud the offline invitation. In the later development phase, we should improve the security of the offline invitation transportation. Furthermore, the security of transferring confidential data in ad-hoc network should be a field to take consideration in the next phase development.

Last but not least, testing is the work should be always continuing, so the functionality of the product would match the customer requirement in a better way and the fault of the program can be detected and fixed.
Appendix 1:

1. Membership Assertion:

00 <Assertion ID="117bff93737ff6"
01 IssueInstant="2011-08-16T10:41:18Z">
02 <Issuer>MSNP</Issuer>
03 <Signature>...(by MSNP)...</Signature>
04 <Subject>
05 <NameID>Alice</NameID>
06 </Subject>
07 <Conditions NotOnOrAfter="2011-08-22T10:41:18Z"/>
08 <AttributeStatement>
09 <Attribute Name="group:Swin">
10 <AttributeValue>administrator</AttributeValue>
11 </Attribute>
12 </AttributeStatement>
13 </Assertion>
Appendix 2:

2. Offline Invitation

00 <Assertion ID="667ff6ec67318"
01 IssueInstant="2011-08-16T12:16:10Z">
02 <Issuer>Alice</Issuer>
03 <Signature>...(by Alice)...</Signature>
04 <Subject>
05 <NameID>Bob</NameID>
06 </Subject>
07 <Conditions NotOnOrAfter="2011-08-16T12:16:10Z"/>
08 <AttributeStatement>
09 <Attribute Name="group:Swin">
10 <AttributeValue>member</AttributeValue>
11 </Attribute>
12 </AttributeStatement>
13 </Assertion>
Reference:


Available at [2012-06-16].


http://glassfish.java.net/
Available at [2012-12-21].

http://www.bluetooth.com/Pages/Bluetooth-Home.aspx
Available at [2012-12-21].