Implementation of an Android Application to Enhance Security in Mobile Social Networks

GRADUATE PROJECT REPORT

Submitted to the Faculty of
The School of Engineering & Computing Sciences
Texas A&M University-Corpus Christi
Corpus Christi, TX

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Pavithra Roscilllin Bandela
Spring 2014

Committee Members

Dr. Mario Garcia
Committee Chairperson

Dr. Longzhuang Li
Committee Member

Dr. David Thomas
Committee Member
ABSTRACT

In this research, privacy preservation with user profile matching in mobile social networks (MSN’s) is studied by introducing a family of novel profile matching protocols. An explicit comparison based profile matching protocol (ecpm) is been proposed first which runs between two parties an initiator and a responder. The ecpm prevents the attribute values from disclosure by enabling the initiator to obtain the comparison based matching result about a specified attribute in their profiles [3]. The second protocol is the comparison-based profile matching protocol which works by allowing the initiator to directly obtain some messages instead of comparing result from the responder. The above two protocols achieve the confidentiality of user profiles based on the anonymity analysis. Offensive decoy technology is proposed which is used for securing monitoring and detecting abnormal data access patterns. This technology protects against the misuse of real data by launching a disinformation attack returning large amounts of decoy information to the attacker. Experiments conducted in a local file setting provide evidence that this approach may provide unprecedented levels of user data security in a cloud environment.
## TABLE OF CONTENTS

Abstract .......................................................................................................................... ii

Table of Contents .......................................................................................................... iii

List of Figures ................................................................................................................ vi

List of Tables ................................................................................................................ viii

List of Code ................................................................................................................... ix

1. Background and Rationale ...................................................................................... 1
   1.1 Introduction ............................................................................................................ 1
   1.2 Attacks on Social Networking Sites ................................................................. 4
       1.2.1 Attacks on Skype’s social media .............................................................. 4
       1.2.2 Attacks on stealing customer’s credentials ............................................. 5
   1.3 Securing data with fog ....................................................................................... 5
       1.3.1 User Behavior Profiling ............................................................................ 6
       1.3.2 Decoys ....................................................................................................... 6
   1.4 Combining User Behavior and Decoy technology for Masquerade Detection . 7
   1.5 Android Operating System .............................................................................. 8
       1.5.1 Eclipse Integrated Development Environment ...................................... 8
       1.5.2 Android Development tools .................................................................... 8

2. Narrative ................................................................................................................... 10
   2.1 Problem Statement .............................................................................................. 10
   2.2 Motivation ............................................................................................................ 10
   2.3 Project Objective ................................................................................................ 10
   2.4 Functionalities of the project ............................................................................ 10
3. Proposed System Design .............................................................................................................. 12
   3.1 System Design and Architecture .......................................................................................... 12
   3.2 Use Case Diagram ............................................................................................................... 14
   3.3 Class Diagram ..................................................................................................................... 15
   3.4 Sequence Diagram .............................................................................................................. 18
   3.5 User Interface ...................................................................................................................... 19
   3.6 Environment ........................................................................................................................ 25
       3.6.1 Eclipse Kepler ............................................................................................................... 25
       3.6.2 Oracle WebLogic Server 12c ....................................................................................... 25
       3.6.3 MySQL Lite 5.1 ........................................................................................................... 26
       3.6.4 EJB 3.0 ........................................................................................................................ 26
       3.6.5 Servlets ........................................................................................................................ 26
       3.6.6 JSP’s ............................................................................................................................ 27
   3.7 Software Modules .................................................................................................................. 27
4. Implementation of Modules ....................................................................................................... 29
   4.1 Authentication Module ........................................................................................................ 29
   4.2 Fog Module .......................................................................................................................... 30
   4.3 Ecmp and icmp Module ....................................................................................................... 32
5. Testing and Evaluation ............................................................................................................... 35
   5.1 Testing of Authentication Module ...................................................................................... 35
   5.2 Testing of Friend Request Accept procedure under Friends button .................................. 38
   5.3 Testing of Fog Module ......................................................................................................... 41
6. Conclusion and Future Work .................................................. 43
7. Bibliography and References .................................................................. 44
8. Appendix : Code Snippets ........................................................................... 46
LIST OF FIGURES

Figure 1.1 Smartphone OS Market Share (Nielsen.com, 2012) ......................................................... 3
Figure 1.2 Example of Syrian attack on Skype ..........................................................4
Figure 3.1 Layers of System Architecture ........................................................................12
Figure 3.2 System Architecture .............................................................................14
Figure 3.3 Main use case diagram ...........................................................................15
Figure 3.4 Authentication Class Diagram ....................................................................16
Figure 3.4.1 Friends Request Class Diagram ..............................................................17
Figure 3.5 Authentication Sequence Diagram ................................................................18
Figure 3.5.1 New Friend Request Sequence Diagram ..................................................19
Figure 3.6 Login Page .........................................................................................20
Figure 3.7 Registration Page ..................................................................................21
Figure 3.8 User Profile Page ..................................................................................22
Figure 4.1 Security Code .......................................................................................30
Figure 4.2 Fog Data ..........................................................................................31
Figure 5.1 User 1 registration page ........................................................................35
Figure 5.2 User 2 registration page ........................................................................36
Figure 5.3 Login Page .........................................................................................37
Figure 5.4 Profile Page ........................................................................................37
Figure 5.5 Testing of Update button ........................................................................37
Figure 5.6 Testing of Post button ...........................................................................38
Figure 5.7 Sending Friend Request ........................................................................39
Figure 5.8 Accepting Friend Request .....................................................................39
Figure 5.9 Testing of Blog Create Button.................................................................40
Figure 5.10 Testing on Blog related Parameters.......................................................41
Figure 5.11 Fog Related Data..................................................................................42
LIST OF TABLES:

Table 1. Fog User Count 31
Table 2. Friend Request Table 32
Table 3. Icmp related queries 34
LIST OF CODE

5.1.1 Code Snippet Blog Implementation 46
5.1.2 Code Snippet Send Friend Request 46
5.1.3 Code Snippet Getting and Accepting Friends Request 46
5.1.4 Code Snippet Comment Bean Code 47
5.1.5 Code Snippet Post Bean Code 47
1. BACKGROUND AND RATIONALE

1.1 Introduction

Social Networking makes digital communication technologies sharpening tools for extending the social circle of people. It has already become an important integral part of our daily lives, enabling us to contact our friends and families on time. The mobile social networks (MSN’s) users are not only able to surf the internet, but also communicate with their peers in close vicinity using short-range wireless communications. Due to its geographical nature, the mobile social networks (MSN’s) support many promising and novel applications. Privacy preservation is a significant research issue in social networking. Since more personalized information is shared with the public, violating the privacy of a target user becomes much easier. When the social networking platforms are extended into the mobile environment, users require more extensive privacy-preservation because they are unfamiliar with their neighbor’s in close vicinity, who may eavesdrop, store and correlate their personal information at different time periods and locations [2]. Once the personal information is correlated to the location information the behavior of users will be disclosed to the public.

Applications which are content-sharing, provide no control mechanisms or feedback to users, this may lead to identity information and inappropriate location disclosure. To overcome the privacy violation in mobile social networks (MSN’s), many privacy enhancing techniques have been adopted into these applications. There is lot of disadvantages for content-sharing applications. Firstly, there is possibility of hackers to launch spam, virus attacks and to commit fraud. Secondly, online scams that seem
genuine for the people, results in data or identity theft attacks. Thirdly, employees may receive negative comments about the company, if they use the sites to view illicit or offensive material.

There are two security issues concerning cloud computing. First one is the security issues faced by the cloud providers, and second one is the security issues faced by their customers. Cloud providers provide software, platform and infrastructure services via cloud. In this project, infrastructure-as-a-service (IaaS) has been used. Infrastructure-as-a-service (IaaS) helps in using infrastructure of any one provider, get services from the provider and pay only for the resources they use. The main security problem associated with Infrastructure-as-a-service (IaaS) is the trustworthiness of data which is stored in provider’s hardware.

There has been a great improvement in the technology of smart phones in the recent years. Smartphone is a device which provides numerous services such as both communicational and computational services, which are used in our day-to-day life. Development of Smartphone applications has increased, with the increase in usage of the Smartphone’s. Millions of applications have been developed for various platforms such as Android, iOS, Symbian and Windows etc. Out of all these platforms Android occupies over 52% of the Smartphone market as shown in below Figure 1.1.
A different approach for securing data in the cloud using offensive decoy technology has been proposed. Data access pattern in the cloud is monitored and abnormal data access patterns may be detected [2]. Therefore, a complete different approach of securing the cloud using decoy information technology is been proposed which is also called as fog computing. This technology is used to launch disinformation attacks against malicious insiders preventing them from distinguishing the real sensitive user’s data from fake worthless data [6]. Two ways of using fog computing to prevent attacks such as the Twitter and Face book attacks is proposed. It can be done by deploying decoy information within the cloud by the cloud service customer and within personal online social networking profiles by individual users.

Existing data protection mechanisms such as encryption, failed in preventing data theft attacks such as, perpetrating by an insider to the cloud provider. Many mechanisms have not been able to prevent data compromise, such as sophisticated access control and encryption mechanisms. These mechanisms were unable to prevent unauthorized and
illegitimate access to the data. Cryptography alone can’t enforce the privacy in cloud computing, such as fully homomorphic encryption (FHE) technique.

1.2 Attacks on Social Networking Sites:

1.2.1 Attacks on Skype’s social media:

Face book, Twitter and the official blog of Skype has been hacked by a Hacker group known as the Syrian Electronic Army. This group posted messages to public profiles in Skype, alleging that Skype and Microsoft spy on their users. This group posted two messages across Skype’s accounts which are shown in Figure 1.2. The first, directed at Skype itself, which read simply, “Stop spying on people!” The other message asserted that Microsoft monitors users emails through hotmail accounts and outlook and is “selling the data to the governments [12].”

Figure 1.2 Example of Syrian Attack on Skype
1.2.2 Attacks on stealing customer’s credentials

Trust wave’s Chicago-based Spider Labs researchers discovered the credentials of popular social media websites which includes Face book, Google, Twitter and Yahoo. These criminals were investigating in Netherlands on a server, which cyber criminals used to control a huge network of hacked computers known as the ‘Pony botnet [13].’ Researchers have found more than 90,000 internet service providers and websites, whose customer’s credentials were on the server. The data includes 326,000 Face book Inc accounts, 60,000 Google Inc accounts; more than 59,000 Yahoo Inc accounts and nearly 22,000 Twitter Inc accounts, according to the Spider Labs [13].

1.3 Securing Data with Fog:

Cloud-based services can be accessed whenever user may connect to the internet. It also provides various methods to store files, documents and media in a remote service. The core security problem to date is the problem of providing security of confidential information in a manner where only user and no one else can gain access to that data. It has not provided the levels of assurance most people desire.

Encryption techniques, standard access controls and many proposals have been made which are used to secure data remotely in the cloud. For a variety of reasons, all standard approaches have been demonstrated to fail from time to time, which includes mis-configured services, insider attacks, buggy code and faulty implementations. These were not able to avoid sophisticated attacks.

It is not enough to build a trustworthy cloud computing environment because accidents still happen and when they occur information gets lost. One needs to be prepared for such accidents and there is no way to get it back. Damage of stolen data can
be limited if we decrease the value of that stolen data to the attacker and this can be achieved by a preventive disinformation attack. The following are the two additional cloud services which have been implemented to secure information in the cloud.

1.3.1 User Behavior Profiling:

Accessing user’s information in the cloud exhibits a normal means of access which is quite expected. To this kind of scenario user profiling technique is applied to model how, when and how much users access their information in the cloud [6]. In order to determine whether abnormal access to a user’s information is occurring or not, normal user behavior can be continuously checked. This method of behavior-based security is commonly applied in fraud detection applications. Such profiles would naturally include volumetric information and how often documents are typically read. Abnormal cloud access can be detected by simple user specific features which are partially based upon the scale and scope of the data transferred.

1.3.2 Decoys:

Decoy documents, honey files, honeypots and other various-bogus information are called as decoy information. It can be served as a means of detecting unauthorized access to the information and to poison the attacker’s ex-filtrated information. Decoys will confuse and confound the adversary into believing they have ex-filtrated the useful information, while they have not [6]. User behavior profiling technology can be integrated with decoy technology for securing information in the cloud. Decoy information may be returned by the cloud whenever abnormal access to the cloud service is been noticed and delivered in such a way to appear completely legitimate and normal.
Whenever decoy information is returned by the cloud, the true user would readily identify and through a variety of means, could alter the cloud’s responses. True user inaccurately detects the unauthorized access and informs the cloud security system. If the access is correctly identified as an unauthorized access, unbounded amounts of bogus information is been delivered by the cloud security system. Thus, this decoy technology secures the user’s data from unauthorized disclosure. The two main important purposes of decoys are

1) Whenever abnormal information access is detected, the decoys are used to validate and authorize the data.

2) The decoys are used to confuse the attacker with bogus information.

Combination of these above two mentioned security features will provide unprecedented levels of security to the cloud [9]. Currently, no other cloud security mechanism is available that provides this level of security. Combining both techniques can yield better detection results and this approach can work effectively in a cloud environment.

1.4 Combining User Behavior Profiling and Decoy Technology for Masquerade Detection:

In this approach, user search behavior is been profiled and user models are developed. These user models are trained with one-class modeling technique, which is otherwise called as one-class support vector machines [10]. The importance of using one-class modeling technique helps to build a classifier without sharing the data from different users. Thus the privacy of the user and their data is therefore preserved.

Using this approach, abnormal search behaviors that exhibit the deviations from the user baseline are detected. These deviations signal potential masquerade attack.
Previous experiments have demonstrated and validated this approach, by detecting all simulated masquerade attacks with a very low false positive rate of 1.12%.

1.5 Android Operating System:

Mobile platforms provide easy and efficient use of technologies for day to day use. Android having open SDK’s and APIs provides to programmers easy and abundant opportunity for development. The Android mobile platform being the most popular mobile operating system today has risen from its inception in October 2008. Though Android fails to support advanced scripts such as AppleScript, it does provide numerous third party libraries to do the job for free. In addition it is very easy to test Android application. Unlike Apple, it is easy to install unpublished Android application. Also, according to Forbes, Android devices held 51% and currently hold 70% of market share in Smartphone industry [14]. Considering this growth, the project is implemented on the Android platform, to reach as many users as possible.

1.5.1 Eclipse Integrated Development Environment:

Eclipse is used to develop various programming languages like C, C++, Java, COBAL, PHP etc. It is a Multi-Language Integration Development Environment (IDE) that comprises workspace and a plug-in system. It is widely used to write Java applications. It includes various development tools that help the developer. It is free and open source software.

1.5.2 Android Development tools:

The Android Development Tools (ADT) for Eclipse enables us to use the Android software development kit tools in the Eclipse Integrated Development Environment to develop applications in android platform. Android development tools help the developers
to add packages based on android framework, to develop the projects quickly and debug
the code etc. The usage of Eclipse with Android development tools is one of the fastest
ways to develop the applications in android platform.
2. NARRATIVE

2.1 Problem Statement:

The main problem in online social networking is some unknown person reading or observing common user profiles without proper permission. The unknown user who has not listed in his/her friends list can comment on his/her posted photos or any other personal data. That means they can trace maximum all the data from social network. To avoid this, we are providing some new concept to flood unknown user with fake data.

2.2 Motivation:

Previously many techniques and permissions based on data security purpose have been proposed. Various data encryption methods have been applied on these techniques but hackers are tracing this type of secured data. Due to these reasons, decoy technologies have been implemented on social networks. Here, user related data is stored in decoy folder with fake information. When correct user needs to read his/her information, at this time decoy will show him/her the original data, but when the unknown user wants to read any unknown user’s data who is not listed in his/her friend’s list that time decoy will show him fake information [8].

2.3 Project Objective:

The Main Scope is to protect user related information from unknown users and unknown groups by enhancing security.

2.4 Functionalities of the project:

Millions of apps are developed and many more are being developed for smartphone users. These are developed for various platforms like Android, iOS, Symbian and Windows etc. Each application has different purpose, provides specific services and
reduces user effort in its own way. This project aims at providing a convenient way of securing user profiles. This application helps to access user profiles, if he/she is present in his/her friends list. It blocks unknown user profiles, if he/she is not in his/her friends list and is trying to post text, videos and comments on his/her profile. This project is developed using Android SDK version 4.4 (KitKat). Below are the tools which are used to develop this Android Application.

- Eclipse Integrated Development Environment (IDE)
- Android Software Development Kit (SDK) 4.4 (KitKat)
- Android Development plug-in Tools (ADT) for Eclipse
- Java SE 7 Development Kit (JDK)
3. PROPOSED SYSTEM DESIGN

3.1 System Design and Architecture: There are three layers in our proposed project architecture. In this application, fog related data comes from secured folder. Here fog stores the dummy data in a secured folder. It works by counting unknown users hit list, by observing their profiles from ejb module. In this project, the hit list taken is 0-5 times, which means number of times unknown user is trying to view his/her profile. It is not necessary to have for 0-5 times.

3.1 Layers of System Architecture

1. Entity Bean Service Layer: This layer has been used for security and reusable purpose. Within this layer we have six user related modules they are user register, user login, user posts, user comments, user profile and unknown user profile as shown in Figure 3.1. It is compatible to add or delete any other modules in this layer. Authentication, profile matching and social network user update are the three services provided in this layer. Here in case if we need to add any module or need to update any existing module no need to disturb all the modules. This layer provides high security.
with encryption and decryption phases being a common part. This layer uses Advanced Encryption Standard (AES) algorithm for securing sensitive data

2. **Web Service Layer:** Here, in this layer, two platform services have been developed one is website and other is mobile app (Android). Here JSP’s and servlets are been used for website purpose and Android SDK is been used for mobile application. These two services are receiving data from the user and sending to EJB service and again receiving data from EJB service as shown in Figure 3.1.

3. **Fog Service Layer:** This layer observes both the EJB and web services. Incase if any unknown person is watching any unknown user images, Fog will store his/her profile into its memory, after that fog will show dummy information to the unknown user. This layer should be generated separately. Depending on the url, fog will observe the information which is coming in and out from EJB services. Fog will hide the original information of the particular user. This entire scenario is called as a decoy procedure which provides the file system security.

The system architecture is shown in Figure 3.2. In this system if he/she is not a registered user, first registers. Then the user will login into his/her account. In the user profile page, user can view home, settings, friends and blog buttons. User can add posts, add likes and add comments in his/her home page. In friend’s page, user can find a friend based on the common attribute, add a friend or send friend request to him/her. In Blog page, user can create or share blogs by giving permissions. Under Profile page, user can view his/her friends profile or his profile. User’s related data is stored securely in cloud. Decoy service which is in below Figure 3.2, is used to provide dummy information to users who are viewing or posting anything offensive in others profile.
3.2 Use Case Diagram: In the use case diagram which is shown in Figure 3.3, an actor is a class of a person, device, organization or an external software component that interacts with the system. Examples of actors in this case are User and Actor1. Use cases are used to represent the actions performed by one or more actors. In the use case diagram shown in Figure 3.3, the actions done by the users are Register, Login, Find Friend by email, Find Friend by Col or Ofc, Check Friends Profile, Check Profile. In the ecmp module the users can Add Blog or Add Post. The posts are blocked by ecmp only to that particular user. In the icmp module, the users can Check Post or Check Blog.
3.3 **Class Diagram:** Class diagrams consist of the interfaces, the methods, variables, relationships and the databases which are shown in above Figure 3.4: Authentication Class Diagram. This authentication class diagram consists of an interface which is called as Authentication Remote; it has the corresponding methods such as register, login, getProfile and updateProfile. It also has a class called authentication which has the same methods as of Authentication Remote interface and databases such as db and userProfiles.
Another class diagram called the Friend Request Class diagram is shown in Figure 3.4.1. This class diagram consists of an interface called FriendRemote; friendRequest, getFriends, getFriendsRequest, friendRequestAccept, findFriendsFromColorOfc,
findFriendsFromEmail, getFriendProfile, getAllFriends are the methods of this interface. This diagram consists of a class called Friend which has the same methods such as the Friend Remote interface as shown in Figure 3.4.1. It consists of the database named as db.

Figure 3.4.1: Friends Request Class Diagram
3.4 **Sequence Diagram:** The sequence diagram models the collaboration of objects based on a time sequence. Authentication Sequence Diagram is shown in Figure 3.5. It checks if the user is authenticated with the database, else it will create a new user profile.

![Authentication Sequence Diagram](image)

**Figure 3.5: Authentication Sequence Diagram**

New Friend Request Sequence Diagram is shown in above Figure 3.5.1. This diagram depicts the generation of New Friend Request accept or reject process.
Figure 3.5.1: New Friend Request Sequence Diagram

3.5 User Interface: Whenever user registers, he/she can login into his or her profile successfully which is shown in Figure 3.6.
If the user is not a registered user, then he/she needs to register with the following attributes such as Username, Password, RetypePassword, Fullname, Email id, ofc or College, Address, Date of Birth, Privacy and Gender as shown in Figure 3.7.
User can post anything on his/her profile which is shown in Figure 3.8. User can perform the activities such as sending add request to a friend, creating blog, giving permissions in the blog, commenting on user data and posting.
3.5.1 Classes:

*Authentication Remote.java*

It is a Remote Interface between web client and server side business logic. All the methods in this interface override in Authentication class.

*Authentication.java:*

It is a stateless session bean class which is implemented with AuthenticationRemote interface methods. This class overrides all authentication module methods like login, register and user Profile methods.

*BlogImpleRemote.java*
It is a Remote Interface between web client and server side business logic. All methods in this interface override in BlogImple class.

**BlogImple.java**

It is a Stateless session bean class to maintain application blog related methods; it has been overridden from remote interface.

**DatabaseConnection.java**

Connection class for java to Database application. This class uses only type4 drivers, for type 4 driver purpose jar classes path in been arranged.

For example, the below lines shows the functionality of this class,

- jdbc:mysql://localhost:3306/socialnetwork
- Database: socialnetwork
- Username: root
- Password: admin
- Driver Type: 4
- Driver Class: com.mysql.jdbc.Driver

Below is the supporting jar files mentioned for his class,

- For Database :mysql-connector-java-3.1.1.4-bin.jar
- For EJB Client :wlthin3client.jar
- For WEB Server : weblogic.jar
- For Android APP: android-support-v4.jar

**Database.java**
This database class is used to pass insert, update and delete SQL queries from Java to Database.

Security.java

Privacy preserving module is been seen in this class. Total encryption and decryption procedure, which passes the data from client to server and from server to database, appears in this class. Below are the security details shown for this class.

- Encryption Algorithms: PBE with MD5 and DES
- Decryption Algorithms: PBE with MD5 and DES
- Security Key: “PAVITHRA.”
- Cipher Salt: (8bit) { (byte) 0xA9, (byte) 0x9B, (byte) 0xC8, (byte) 0x32, (byte) 0x56, (byte) 0x35, (byte) 0xE3, (byte) 0x03 }

FogImplementor.java

FogImplementor is an interface which is used to interact with server side business layer. This interface is also used to drop unknown users.

FogImplementor.java

It is a bean class to implement with FogImplementor Interface.

ServerMethodConnector.java

This class is implemented for calling all the methods from Main business server to client side. Here all the instances are called and are overridden in Interfaces.

ServerConnector.java

It is a connector class for remote interface. In this class we mention all JNDI properties. For remote server purpose we call the url’s and classes. For example, Class: weblogiv.jndi.WLInitialContextFactory, URL:t3://localhost:7001.
**MainActivity.java**

MainActivity is the android class, which is used to execute our website into Android mobile phone class. We call web view class to import service from social website.

### 3.6 Environment:

#### 3.6.1 Eclipse Kepler:

Eclipse is an integrated development environment in computer programming. In order to customize the environment, it contains an extensible plug-in system and a base workspace. It is written in Java, and it can also be used to develop applications in other programming languages by means of various plug-ins [15]. Eclipse can also be used to develop packages for software Mathematics. It uses plug-ins to provide all the functionality within and top of the runtime system. It is a platform for report designers, web engineers and Linux developers. This Eclipse Kepler has added support to MongoDB and Apache Cassandra which are NoSQL data sources.

#### 3.6.2 Oracle WebLogic Server 12c:

The Oracle Database is an object-relational database management system which is produced and marketed by Oracle Corporation. Oracle database system has been identified by an alphanumeric system identifier or SID, which has at least one instance of the application. SQL functions have been built into Oracle database which are used in various SQL Statements. This Oracle database simplifies the consolidating process of the databases, without making any changes to the existing applications [16]. It increases server utilization and scalability by providing efficiently database provisioning, upgrading and patching. Oracle WebLogic Server 12c is the best application server in the
industry [16]. It is used to build and deploy Java EE applications and supports many features, such as, improving performance, lowering cost of operations, enhancing scalability and supporting the Oracle applications portfolio.

3.6.3 MySQL Lite 5.1:

MySQL offered MySQL lite in two different variants. The first one is the open source MySQL Community Server and the second is the commercial Enterprise Server. It includes following features such as triggers, cursors, Information schema, updatable views etc. [17].

3.6.4 EJB 3.0:

Enterprise JavaBeans (EJB) technology enables simplified and rapid development of transactional, distributed, secure and portable applications based on Java technology. EJB 3.0 specification targeted at ease of development, defining the new simplified EJB API [18]. This EJB 3.0 includes the new Java Persistence API for the management of persistence and object/relational mapping with Java EE and Java SE. Applications developed using the Enterprise JavaBeans architecture are transactional, scalable and multi-user secure. The introduction to the specification of EJB 3.0 is a giant step forward and will go a long way attracting developers back to EJB’s.

3.6.5 Servlets:

Servlets are basically Java Program that executes in a web server or an application server. It acts as a middle layer between clients and database on the HTTP server. By using servlets, users can obtain information from web forms and display records from a database. It is a Java class which is used for compiling the Java Servlet API. For the
communication of the Servlet and the Servlet container Javax.servlet.http is been used as a package specifying HTTP specific sub classes [19]. These servlets are easy to use and are more powerful as compared to Traditional CGI scripts.

3.6.6 JSP’s:

These JSP’s are the server-side programming technology, enabling the creation of platform-independent, dynamic method for building Web-based applications. To the entire family of Java API’s these JSP’s has access, including the JDBC API to access enterprise databases [20]. A JSP web page display certain parameters which are content based. It is actually a powerful scripting language which is executed on the server side but not on the client side. These are integrated in a web page in HTML using special tags notifying the web server. These acts as a 3-tier architecture which acts as a mediator between the client browser and the database.

3.7 Software Modules: There are three software modules in this project which are shown below.

1. **Authentication Module:** Authentication module is the first module in this project. It is used for user, to register, login and update his/her profiles.

2. **Fog Module:** This module observes both the EJB and web services. Incase if any unknown person is watching any unknown user profile, Fog will store his/her profile into its memory, after that fog will show dummy information to the unknown user. This module should be generated separately. Depending on the url, fog will observe the information which is coming in and out from EJB services. Fog will hide the original
information of the particular user. This entire scenario is called as a decoy procedure which provides the file system security.

3. **Blog creation and Friend request accept procedure Module:** This module covers the creation of blog functionality and maintains privacy within these blogs. Privacy within these blogs comes from icmp internal module, which tries to block the offensive data. It also focuses on accepting friends based on common attributes. In this scenario, ecmp internal module helps in accepting friends based on the common attribute.
4. Implementation of Modules

This project mainly consists of three modules which is explained below

- Authentication Module
  - Security Module
- Fog Module
- Ecmp and Icmp Module

4.1 Authentication Module:

The Authentication module consists of AuthenticationRemote, Authentication, Register, Login, PasswordChanger and UserProfileBean classes. Under Authentication module there is one more module called the Security module. The Security module has Security class.

The Authentication.java class is been implemented by authenticating with the database, below is the code snippet for Authentication.java class. This Authentication class uses the database connection for storing new user profiles.

```java
public Authentication() {
    db = new Database();
    userProfiles = new UserProfileBean();
}
```

The Registration.java class is implemented by code snippet that follows. Here the registered users will be stored into databases.

```java
@Override
public boolean register(UserProfileBean userProfile) {
    return db.userRegister(userProfile);
}
```
The UserProfileBean.java class is developed by matching the users based on college or office by using getCollORofc() method. This is shown in the below code snippet.

```java
@Override
public UserProfileBean getProfile(String username) {
    UserProfileBean obj = db.getProfile(username);
    System.out.println("College or Office: " + obj.getCollORofc());
    return obj;
}
```

The Security.java class is been implemented by using the DES algorithm and is shown in Figure 4.1. The 8-bit cipher-salt is been stored in an array and is generated by the following code snippet.

```java
public class Security {
    static Cipher ecipher;
    static Cipher dcipher;

    static byte[] salt = {
        (byte) 0xA9, (byte) 0x9B, (byte) 0xC8, (byte) 0x32,(byte) 0x32
    }
    static int iterationCount = 19;
    static String key="PAVITHRA.");
}
```

**Figure 4.1 Security Code**

### 4.2 Fog Module:

This module is been implemented by using the following classes such as FogImplementor.java and FogImple.java. The FogImplementor.java is an interface which is developed by interacting with ejb web server.

FogImple. Java is an important class which is used to retrieve dummy values onto the fields of users registered data as shown in Figure 4.2, whenever unknown user is trying to view his/her profile. The following class is been implemented by the below code snippet.
@Override
public UserProfileBean getFriendsProfile(String username, String friend) {
    try {
        if (db.blockuserCount(username, friend)) {
            obj = db.getFriendsProfile(username, friend);
        } else {
            obj.setUsername(username);
            obj.setFullname("Not Acceptable");
            obj.setPhone(999999);
            obj.setEmail("NotAcceptable@match.com");
            obj.setCity("Not Acceptable");
            obj.setColorOfOffice("Not Acceptable");
            obj.setDob("Not Acceptable");
            obj.setGender("Not Acceptable");
        }
    }
}

Figure 4.2: Fog Data

Here in the Figure 4.2, we can see the Username, Fullname, Phone, Email, City, ColorOfOffice, Dob, Gender fields are filled with corresponding required dummy values such as “username”, “Not Acceptable”, “999999”, NotAcceptable@match.com, “Not Acceptable”, “Not Acceptable”, “Not Acceptable”, “Not Acceptable”, “Not Accepted”. These values are hardcoded. Whenever unknown user tries to view his/her profile up to 5 times, at this time the count gets incremented and dummy values are shown automatically to the user by Fog. Table 1 Fog User Count shows, no of times unknown user trying to view his/her profile.

<table>
<thead>
<tr>
<th>Fog User Count</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>select * from fog information where username=&quot;&quot;+username+&quot;&quot; and hacker=&quot;&quot;+friend+&quot;&quot;</td>
<td></td>
</tr>
<tr>
<td>Insert into fog_information value(?,?,?)</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Fog User Count
4.3 **Ecmp and Icmp Module:**

4.3.1 **Ecmp Module:** The Ecpm module works from initiator to the responder. It tries to know the comparison result between initiator and responder based on the common attribute. In this project the common attribute is finding friends, who are in the same office or college. The ecpm module consists of BlogImpleRemote, FriendRequest.FriendRemote, PosterImplRemote, RemoveFriendFromGroup, PosterImpl and BlogImpl classes.

BlogImple.java is been implemented to create a blog with attributes ‘username’ and ‘blogname’. Here the parameters are the username and blogname with String data type.

The Friend.java class is used to send friend request and is also used to accept friend’s request.

FriendRequest is been queried as shown in Table 2. Friend Request Table, by using mysql database query.

<table>
<thead>
<tr>
<th>Friend Request</th>
<th>SQL Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>select * from friends_table where username=&quot;&quot;+username=&quot;&quot; and friends=&quot;&quot;+friend=&quot;&quot;&quot;&quot;; insert into friends_table values(?,?,?)&quot;</td>
<td></td>
</tr>
<tr>
<td>Get Friend Request</td>
<td>select * from friends_table where friends=&quot;&quot;+username=&quot;&quot; and request=&quot;request&quot;</td>
</tr>
<tr>
<td>Get Friends</td>
<td>select * from friends_table where username=&quot;&quot;+username=&quot;&quot; and request=&quot;friend&quot; select * from friends_table where friends=&quot;&quot;+username=&quot;&quot; and request=&quot;friend&quot;</td>
</tr>
<tr>
<td>Accept Request</td>
<td>update friends_table set request=&quot;&quot;+request=&quot;&quot; where username=&quot;&quot;+friend=&quot;&quot; and friends=&quot;&quot;+username=&quot;&quot;</td>
</tr>
</tbody>
</table>

Table 2. Friend Request Table
4.3.2 Icmp Module:

Icmp is the user related behavior module. Icmp modules come into picture whenever user groups are created in our profile, messages will be exchanged within that particular group. Icmp blocks the data, whenever a person is trying to add offensive comment in user related groups or blogs. Icmp module consists of the following classes such as FriendRequest.Friend, AddForGroup, BlogDropper, CreateBlogPageServlet, LikesServletPage, CommentBean, FriendList, FriendsBean, LikesBean, PostBean and UserPost.

CommentBean.java class has two string variables ‘who comment’ and ‘comment’ with their corresponding values “” and “”. This class is used to comment on user posts and checks to see who commented on user posts.

In this insert query is used to insert comment values onto comment_table database which is given by the following query, insert into comment_table values (?, ?, ?).

PostBean.java class has a string variables username and post. It consists of array elements which are comment, comment friends, likes and likes friends. This class mainly focuses on commenting and liking the posts, posted by users in their profile.

The following Table 3. Icmp related queries shows the corresponding queries related to the following methods in icmp module
<table>
<thead>
<tr>
<th>Operation</th>
<th>SQL Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Post</td>
<td><code>insert into post_table values(?, ?, ?, ?)</code></td>
</tr>
<tr>
<td>Add Likes</td>
<td><code>insert into likes_table values(?, ?)</code></td>
</tr>
<tr>
<td>Retrieve Post</td>
<td><code>select * from socialnetwork.post_table where username=&quot;&quot;+username+&quot;&quot;</code></td>
</tr>
<tr>
<td>Retrieve Comment</td>
<td><code>select * from comment_table where id=&quot;&quot;+id&quot;&quot;</code></td>
</tr>
<tr>
<td>Retrieve Likes</td>
<td><code>select * from likes_table where id=&quot;&quot;+id&quot;&quot;</code></td>
</tr>
<tr>
<td>Create Blog</td>
<td><code>insert into blog_table values(?, ?)</code></td>
</tr>
<tr>
<td>Drop Blog</td>
<td><code>delete from blog_table where author=&quot;&quot;+username+&quot;&quot; and blogname=&quot;&quot;+blogname+&quot;&quot;</code></td>
</tr>
<tr>
<td>Blog Friend</td>
<td><code>insert into blog_group value(?, ?, ?)</code></td>
</tr>
</tbody>
</table>

Table 3. Icmp related queries
5. Testing and Evaluation

In this phase the functionality of the developed application is tested and evaluated. The testing of the application is performed on Motorola g, which is an Android device. In this application, two users communicate for testing; by copying mobile .apk file into multiple phones and connecting to the server via wifi. Each of the components in the application is tested and the results are provided.

5.1 Testing of Authentication Module: In Figure 5.1 shown below, username ‘Pavithra’ registers into ‘Profile Matching App’

![User1 registration page](image1)
![User1 registration page](image2)

Figure 5.1: User1 registration page
One more user with username “Sangeetha” is been registered into Profile Matching app. This project is been tested with two users, who tries to perform their actions in android app.

![User 2 Registration Page](image)

**Figure 5.2: User 2 Registration Page**

After the two user’s registers into profile matching app, the second user with username “Sangeetha” logs in, which is shown in Figure 5.3. Profile page is shown in Figure 5.4. It has the following buttons such as Home, Settings, Friends and Blog. User profile page is shown in home button. If there are any updates which are to be made to the user profile, it is shown in settings page. In order to accept any friend’s requests, it is shown in friend’s page. In order to create a blog or posts any blog related data, it is shown in blog button page.
5.1.1 **Testing of Update button under Settings page:** The user with username “Sangeetha” changes her college name, which is shown in Figure 5.5 and is been tested.

![Figure 5.3 Login Page](image1)
![Figure 5.4 Profile Page](image2)

![Figure 5.5 Testing of Update button](image3)
5.1.2 Testing of Post button under Home page: In the Figure 5.6, both the users with usernames “Pavithra” and “Sangeetha” are posting some information in their respective Home pages.

![Figure 5.6 Testing of Post Button](image)

5.2 Testing of Friend Request Accept procedure under Friends button: In this project, the two registered users have the same common attribute ‘college name’. Now the user with username ‘sangeetha’, is sending friend request to other user with username ‘pavithra’ and is shown in Figure 5.7. User with username “pavithra” accepts “sangeethas” friend’s request after logging into her profile as shown in Figure 5.6. Under the user page with username “pavithra”, accept and done accept buttons, confirm button, send request and profile button is been shown in Figure 5.8.
5.2.1 Testing of Blog create button: In the below Figure 5.9, the user with username “Sangeetha” is creating the blog. In this scenario the create buttons have been tested. Person who has permissions to add in his/her blog list can view his/her blog related information.
5.2.2 Testing on blog related permissions: The user with username “sangeetha” has created her blog. She needs to give blog related permissions to other users, who want to be in her blog list and who want to comment on her blog as shown in Figure 5.10. Now in this scenario, the user with username “pavithra” has got blog related permissions, and is been added in the user with username “sangeetha” blog list which is shown in Figure 5.10.
5.3 Testing of Fog Module:

This module comes into picture whenever unknown user is trying to match his/her profile. In this project, if the profile is been viewed for 5 times, the count will be incremented in the data base up to 5 and the fog module is going to show dummy data to that specific user. This module has been tested between two users “sangeetha” and “pavithra”. The user with username “pavithra” continuously watches user with username “sangeetha” profile, In this scenario, if “pavithra” views for 5 times , then automatically fog is going to provide dummy data to user “pavithra”, which is shown in Figure 5.11.

![Figure 5.10 Testing on blog related permissions]
Figure 5.11  Fog Related Data.
6. Conclusion and Future Work

Basically all social networks will prefer secure encryption and decryption standards based on SSL. The main thing in this project is the fog service, which has been used to create illusion concept to unknown users or attackers by using decoy technology procedure. This project presents the importance of android app by enhancing security, which is been used in mobile social networking platform. It helps in preserving privacy among the user profiles, by using decoy technology. Decoy technology is been used to present dummy data to the users who are trying to match other user profiles.

The future work of the application involves adding some more smartphone functionalities to it. This project concept can be broadly used in social networking sites such as face book, twitter and linkedIn by preserving privacy. The user interface of the application can be improved in an attractive way and user friendly by using more Android Widgets.
7. BIBLIOGRAPHY AND REFERENCES

8. APPENDIX: CODE SNIPPETS

```java
@Override
public void createBlog(String username, String blogname) {
    db.createBlog(username, blogname);
}
```

Code Snippet 5.1.1: Blog Implementation

```java
@Override
public void friendRequest(FriendsBean bean) {
    try {
        db.sendFriendRequest(bean);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Code Snippet 5.1.2: Send Friend Request

```java
@Override
public ArrayList<FriendsBean> getFriendRequests(String username) {
    return db.getFriendRequests(username);
}
```

```java
@Override
public void friendRequestAccept(FriendsBean fb) {
    db.friendRequestAccept(fb);
}
```

Code Snippet 5.1.3: Getting and Accepting Friends Request
public class CommentBean implements Serializable
{
    public int id=0;
    public String whocomment="";
    public String comment="";

    public void setid(int id)
    {
        this.id=id;
    }
    public int getid()
    {
        return this.id;
    }

    public void setwhocomment(String whocomment)
    {
        this.whocomment=whocomment;
    }
}

Code Snippet 5.1.4: Comment Bean Code

public class PostBean implements Serializable
{
    int id=0;

    String username="";
    // String friendname="";
    String post="";
    String[] comment=new String[2];
    String[] commendFriends=new String[2];
    Integer[] likes=new Integer[2];
    String[] likesFriends=new String[2];
}

Code 5.1.5: Snippet Post Bean Code