Location Privacy in MobiShare

Student Name: Sonia Soubam

IIIT-D-MTech-CS-IS-11-011
April 30, 2013

Indraprastha Institute of Information Technology
New Delhi

Thesis Committee
Vinayak Naik (Chair)
Ponnurangam Kumaraguru
Dipanjan Chakraborty

Submitted in partial fulfillment of the requirements
for the Degree of M.Tech. in Computer Science,
with specialization in Information Security

Keywords: Location Privacy, MobiShare, Mobility Profile, Privacy of non-users

Certificate

This is to certify that the thesis titled **“Location Privacy in MobiShare"** submitted by **Sonia Soubam** for the partial fulfillment of the requirements for the degree of *Master of Technology* in *Computer Science & Engineering* is a record of the bonafide work carried out by her under my guidance and supervision in the Security and Privacy group at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

Professor Vinayak Naik
Indraprastha Institute of Information Technology, New Delhi

Abstract

This work focuses on providing privacy in MobiShare, which is a location-based service that assists users in searching and sharing files. A user's mobility profile is summarization of its raw mobility data in terms of GPS coordinates and timestamps. Mobility profile is denoted in terms of places visited by the user, with their arrival and departure times for every day. MobiShare takes users' raw mobility data to compute mobility profiles. It stores phone number of the users, phone number of their social contacts, and the meta data of the files to be shared. In the current architecture of MobiShare, the cloud learns about locations, social contacts, and files of the users. For preserving privacy of the users, we redefine architecture of MobiShare. The new architecture neither takes users' raw mobility data nor their mobility profiles. All the locations are stored on the phone itself. This is equivalent to personal data vault, fully in control of the user. The sharing of mobility profiles among the users is done in a privacy preserving way via the cloud. In other words, the cloud never learns about users' locations.

We propose two models: *Partial Privacy Preserving Model* and *Full Privacy Preserving Model*. In the first model, contacts learn about actual location information of the user. In the second model, the user share his/her mobility profile without even disclosing it to the contact. We borrow Paillier homomorphic encryption technique [8, 13] so that a user can share the mobility profile for the purpose of finding rendezvous point without disclosing the actual profile. This encryption is done once for a user and there is no need to repeat this for different contacts.

We report the overhead of introducing these Privacy Preserving Models on Android phone. The cloud never learns the phone number of the users, who are not participating in using MobiShare. This provide social privacy. In future work, we will work on preserving privacy of what files users are sharing.

Acknowledgments

First and foremost I offer my sincerest gratitude to my supervisor, Dr Vinayak Naik, who has supported me throughout my thesis with his patience and knowledge whilst allowing me the room to work in my own way.

I dedicate this thesis to my parents who have always supported me and believed that I could do it.

I would like to thank Kuldeep Yadav for his advice and many insightful discussions and suggestions.

Last but not the least, I would like to thank my friends who were always willing to help and give their best suggestions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Location-based services (LBS) - applications that provide information to users based on their location- provides very interesting services to users. However, a dark side of these services is location privacy of users. Location information collected by LBS can reveal much more than just latitude and longitude, such as visit to hospital, participation in a rally, home and office address, usual hangout place, etc. Many LBS collect location information even when users are not actively using the application. Users have very little control over the information once it is uploaded and have no choice than to trust the service providers. Some LBS collects location information indefinitely unless the user manually deletes his data, and some don't even provide this option. Location information collected over a long period of time can expose a lot details about the user's life and habits. This work focuses on providing location privacy for MobiShare, a location-based service that assists users in searching and local sharing of files.

The idea of MobiShare is that people in developing nations like India have multimedia phones but are limited to low speed network. Such users don't have the luxury of downloading files (such as songs, videos) using the internet connection in their phone. Instead they get it from friends, either via Bluetooth or by swapping flash cards. To get a file from a friend, user first finds out whom among his friends have the file and get the file transferred from the friend which may involve fixing a time and place for meeting if they are not nearby at that time. MobiShare simplifies this whole process by allowing users to search a file using MobiShare mobile application and returning list of friends who have the searched file along approximated time of delivery. It notifies users when such friends are in proximity and users can then exchange files using short-range wireless technologies such as Bluetooth or can swap memory cards. There are two main components of MobiShare: mobile application running on phone and cloud service. Through the mobile application, cloud

learns about location, social contacts and shared files of users. MobiShare generates *mobility profile* denoted in terms of places visited by the user, with their arrival and departure times for every day and are used to predict rendezvous of users. MobiShare is described in details in Chapter 2.

Privacy concerns in MobiShare:

1. **Location Privacy:** Location data of users in cloud can be used to track users and can be misused in various ways.
2. **Privacy of non-users:** In order to find out user's friends, MobiShare requires user to upload their phonebook and optionally their Facebook friends list. This reveals contact information of friends who are not using the MobiShare service.

1.2 Contribution

We redefine the architecture of MobiShare and make the following contributions:

1. The new architecture neither takes users' raw mobility data nor their mobility profiles. All the locations are stored on the phone itself. This is equivalent to personal data vault, fully in control of the user.
2. The sharing of mobility profiles among the users is done in a privacy preserving way via the cloud. In other words, the cloud never learns about users' locations.
3. User can choose to share his/her mobility profile without even disclosing it to the contact. We borrow Privacy Preserving distance comparison using Paillier homomorphic encryption [8, 13] so that a user can share the mobility profile for the purpose of finding rendezvous point without disclosing the actual profile.
4. User can also choose a simpler and less data consuming approach in which his location informations are encrypted using symmetric key encryption. Friends learn about location information of the user but not the cloud.
5. These encryptions are done once for a user and there is no need to repeat this for different contacts. We report the time taken by the encryption and decryption on the Android phone.
6. The cloud never learns the phone number of individuals who are not using MobiShare. This provide social privacy.

Chapter 2

MobiShare

MobiShare is a mobile-cloud based system designed to support searching and local sharing of content in user's social network and enable scalable content search with real-time computation of expected encounter time between content source and requester. It is a novel system especially for the users of developing nations who are constrained by limited bandwidth connection and rely on local communication technologies such as manually swapping flash cards or Bluetooth for sharing large content.

The main assumption of MobiShare is, there are some frequently occurring places such as "Home" and "Workplace" in a person's mobility profile and it is likely that a person will encounter mostly the same people at these frequently visited places across different days. This service cannot be used for people with very dynamic mobility profile such as taxi drivers or delivery boys.

2.1 Architecture

MobiShare is based on a hybrid architecture that uses a central entity i.e. the cloud for storing, aggregating and performing analysis on the information which is uploaded by frontend mobile application. Information flow in MobiShare is classified as control and data. Control information consists of users' location, meta-data of the files to be shared, users' search queries and content request intent and a notification to share the files. Data information consists of actual multimedia content to be transferred. Small payload size of control information (utmost few KBs) as compared to actual data, results in quick and low-energy transmission using 2G connection. Actual data transfer happen using small range wireless technologies such as WiFi or Bluetooth.

MobiShare forms its own social network for its users, using a combination of phone contacts and Facebook friend list. It provides a trusted environment by allowing users to

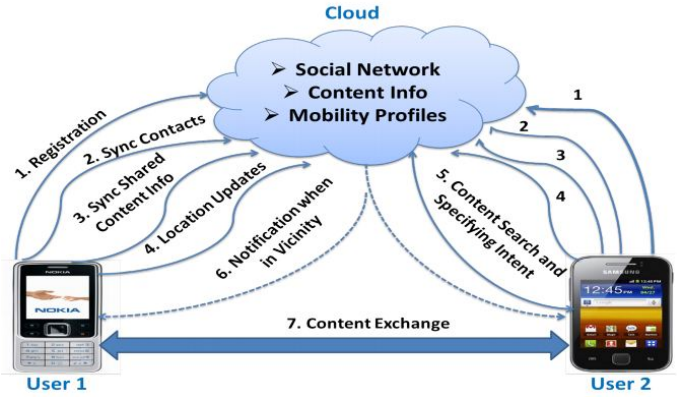

Figure 2.1: Mobile-cloud Architecture of MobiShare

search and share content within their social network only. Its mobile application provides user an option to select the content that he wants to share. Accordingly, attributes of all the selected files such as name, size, type, genre, album name, artist name are synced to the cloud.

The cloud aggregates global information of available data, across all users, which can be searched in real-time using mobile application. After searching for the content, the cloud also computes the expected delivery time using the mobility profile of users within the social network of the user who is requesting the content (client). All the users are ranked based on computed delivery time to give an estimate of the time by when the client can expect the content to be delivered. Since content exchange can only happen when two users are in close proximity, users' location updates and historical mobility profiles to detect physical proximity of users and correspondingly initiates a neighbor device discovery using Bluetooth or WiFi to exchange the desired content. If the desired content source is not found in the vicinity, communication interface is switched off, until a new rendezvous opportunity arrives.

2.1.1 Usage Scenario

Let Alice, Bob, and Carol be three unique MobiShare users, who are also socially connected with each other. Suppose, Alice wants to have a video "V" but is unaware of who among her friends has this video. The steps that Alice will follow to get the video "V", using MobiShare are :

(1) Alice opens the MobiShare mobile application and searches the video "V" by specifying a set of key-words, similar to a Google search query. (Refer Snap-shot 2 in 2.2))

(2) The application requests the cloud to search video "V" in the social network of Alice and finds matches which are lexically similar to the search query.

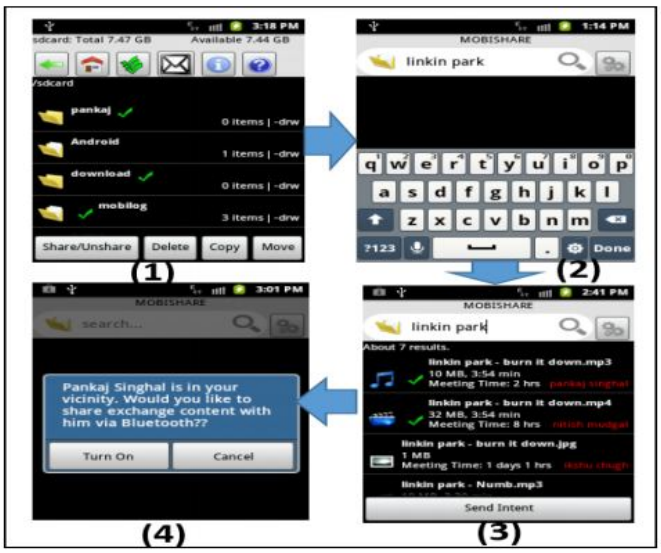

Figure 2.2: MobiShare Mobile Application Snapshot

(3) Assuming that the video “V” is available with both Bob and Carol, the cloud computes the expected delivery time of Alice with Bob, and Carol by looking at mobility profiles of all the three users, ranks the search results based on delivery time, and returns the sorted results to the mobile application.

(4) From the returned results, Alice selects the user (s), say Bob from whom she wants to fetch the video (Refer Snapshot 3 in Figure 2.2)).

(5) The mobile application informs the cloud about the intent that Alice wants to receive the “V” from Bob.

(6) The cloud thereafter tracks locations of Alice and Bob to infer the time when they come in physical proximity and then send notifications to both users (Refer Snapshot 4 in 2.2)

(7) After reception of the notifications, both the devices starts neighbor discovery to find each other and ex-change the video “V” if the discovery is successful. If the discovery is unsuccessful, the short range communication is switched off and then periodically turned back on to recheck and save on overall energy consumption.

2.2 Tables in MobiShare

These section describes structure of the important tables used in MobiShare.

2.2.1 User Information

Table name UserInfo

Description Contains information about the user

Attribute	Data type	Can be null?	Description
id	int	No	Primary key
UserID	int	No	ID of the user
Name	int	No	Name of the user
CellNumber	varchar	No	Phone number of user
EmailID	varchar	Yes	Email address of user
FbID	vachar	Yes	Facebook ID of user

2.2.2 GPS Location data

Table name Location

Description This table consists of GPS location data of user

Attribute	Data Type	Can be null?	Description
id	int	No	id of the entry
latitude	double	No	GPS latitude
longitude	double	No	GPS longitude
Timestamp	Timestamp	No	Date and time

2.2.3 Phone Contacts

Table name PhoneContacts

Description Contains user's phone contacts

Attribute	Data type	Can be null?	Description
id	int	No	Primary key
UserID	int	No	ID of the user
Name	int	No	Name of the user
CellNumber	varchar	No	Phone number of friend
MessageExchanged	int	Yes	Number of messages exchanged
CallsExchanged	int	Yes	Number of calls exchanged

2.2.4 Facebook Friends

Table name FacebookFriends

Description Contains details of Facebook friends of user

Attribute	Data type	Can be null?	Description
id	int	No	Primary key
UserID	int	No	ID of the user
FbID	varchar	Yes	Facebook ID of friend
MessageExchanged	int	Yes	Number of messages exchanged

2.2.5 Search File

Table name FileSearch

Description This table contain details of file search

Attribute	Data Type	Can be null?	Description
id	int	No	id of the entry
UserID	int	No	User ID
query	varchar	No	User query
type	varchar	No	Type of file searched
timestamp	Timestamp	No	Time when the entry was made

2.2.6 Shared File

Table name SharedFiles
Description This table consists of details about the shared files

Attribute	Data Type	Can be null?	Description
id	int	No	id of the entry
UserID	int	No	User ID
Album	varchar	Yes	Name of the Album
Artists	varchar	Yes	Artist name
Duration	Time	Yes	Duration of the file
FileName	varchar	No	Name of the file
FileSize	varchar	No	Size of the file
FileType	varchar	No	File Type
Genre	varchar	Yes	Genre of the music
Path	varchar	No	Where the file is located in user's device
Title	varchar	Yes	Title of the music

2.2.7 Selected User

Table name Selected
Description This table contain details which user selected whom to get the queried file

Attribute	Data Type	Can be null?	Description
id	int	No	id of the entry
RequestorUID	int	No	User ID of requestor
SelectedUID	int	No	User ID of the selected user
timestamp	Timestamp	No	Time when the entry was made

2.3 MobiShare Processes

This section describes the messages exchanged during the various steps of MobiShare

1. **User Registration**: Registration is done by sending a registration request consisting of user name and phone number. Email Id and Facebook ID is optional. For any new

username-mobile number pair, a unique UserID is returned, which is used to uniquely identify the user in the whole system. If the pair was previously registered, then its previously assigned UserID is returned.

2. **Contact and Facebook Sync**: After registration, MobiShare syncs user phone contacts and Facebook friends to find MobiShare friends. These information are stored in Tables 2.2.3 and 2.2.4 respectively.

3. **Shared File Details**: User selects the list of files she wants to share through Mobishare. Details of the selected files are stored in the Table 2.2.6. This table is referred whenever there is a file query.

4. **Search Query**: A search query consists of the parameters shown in Table 2.2.5

5. **Searching**: Find friends of requester who have shared the queried file using 2.2.6 table. Compare mobility profiles of providers and the requester. Return the requester the predicted place and time of meeting.

2.4 User Mobility Profiling

A user's mobility profile is summarization of his raw mobility data in terms of latitude-longitude coordinates and timestamps. Mobility profile is denoted in terms of user's stay points, with their arrival and departure times for every day. A stay point is a location where the user spends significant amount of time. For a location to be considered as a stay point there are two parameters, *threshold distance* and *threshold time*. If a user spends more than *threshold time* within a *threshold distance* then the mean of the GPS points within that region is considered as his stay point. MobiShare uses GPS latitude longitude information of a user collected in the cloud to generate mobility profile of its users. Algorithm 1 describes how raw latitude longitude data is converted into Mobility Pattern [9].

Users mobility profile may differ according to day of the week. So MobiShare computes mobility profile day-wise. Table 2.1 shows an example of the mobility profile of a user.

Place IDs	**Monday**	**Tuesday**	**Sunday**
P_1	00:49-10:28 21:39-23:59	00:01-07:53 19:27-23:59	00:53-22:12 22:57-23:59
P_2	11:08-18:38	08:17-19:24	
P_3	19:10-21:11		

Table 2.1: User mobility profile

Algorithm 1: Pseudocode for generating Mobility Pattern

Input : GPS points G = $\{g_1, g_2, ..., g_n\}$ of a user,a distance threshold $distThreshold$ and a time threshold $timeThreshold$.
For each $g_i \in G$, g_i=$(lat_i, lng_i, timestamp_i)$

Output: Mobility profile of the user as set of Stay points=$\{S_i\}$ where $S_i = \{avgLat_i, avgLng_i, startTime_i, endTime_i\}$

begin

$GPSptsCount = |G|$;
$i = 0$;
while $(i < GPSptsCount)$ **do**
  $j = i + 1$;
  while $(j < GPSptsCount)$ **do**
    $dist = CalculateDistance(lat_i, lng_i, lat_j, lng_j)$;
    if $(dist < distThreshold)$ **then**
      $timediff = time_j - time_i$;
      if $timediff > timeThreshold$ **then**
        $sumlat = 0$;
        $sumlng = 0$;
        $pt = 0$;
        $startTime = time_i$;
        $endTime = time_j$;
        $k = i$;
        while $k < j$ **do**
          $sumlat = sumlat + lat_k$;
          $sumlng = sumlng + lng_k$;
          $pt = pt + 1$;
          $k = k + 1$;
        $StayPtslat = sumlat/pt$;
        $StayPtslng = sumlng/pt$;
        $S = (StayPtslat, StayPtslng, startTime, endTime)$;
        $Staypts.insert(S)$; $Count = Count + 1$;
        $i = j$;
        break ;
    else
      break;
    $j = j + 1$;
    $i = i + 1$;
return $StayPts$

Chapter 3

Home and Workplace location

As discussed earlier, MobiShare collects location information to generate mobility profile which are then used to predict rendezvous of users. In a recently study by Montjoye et al. [16], they observed fifteen months of human mobility data for one and a half million individuals, and found four spatio-temporal points are enough to uniquely identify 95% of the individuals.

In the following we find home and workplace location which are two important palces in the mobility profile of a user . A user's phone number along with his home and workplace location makes identification of the user significantly easier.

3.1 Procedure

We use the timestamped location information of users stored in the MobiShare cloud. We assume workplace as place where user spends most of her time between 10 am to 5 pm in weekdays. While home is the place where user stays between 12 midnight to 5 am in weekdays and whole day in weekends. These assumptions may not be the best, but the sole aim of this experiment is to prove that users' location privacy is being breached.

A step by step description on how user's home and workplace location were identified is as follows:

1. Let L= $\{l_1, l_2, ..., l_n\}$ where $l_i = \{latitude_i, longitude_i\}$ be the set of distinct latitude-longitude pairs for a user. Clustered together l_i which are within a distance of 100 m from each other in C_i.

2. For each cluster C_i, we calculate their home and workplace score as follows:

$$Workplace\ score = \frac{W(C_i)}{W(L)} \times 100$$

where,
$W(C_i)$ = Total number of occurrences of l_i, $\forall l_i \in C_i$ between 10 am to 5 pm in weekdays
$W(L)$ = Total number of occurrences of l_j, $\forall l_j \in L$ between 10 am to 5 pm in weekdays

$$Home\ score = \frac{H(C_i)}{H(L)} \times 100$$

where,
$H(C_i)$ = Total number of occurrences of l_i, $\forall l_i \in C_i$ between 12 midnight to 5 am in weekdays and whole day in weekends.
$H(L)$ = Total number of occurrences of l_j, $\forall l_j \in L$ between 12 midnight to 5 am in weekdays and whole day in weekends.

3. Take clusters with highest workplace and home score. Calculate geo-midpoint of latitude and longitude within the cluster and return them as workplace and home location.

3.2 Analysis

To check the validity of our method, we applied it on location information of five volunteer users whose home and workplace locations were already known to us. All of them are students of IIIT Delhi, one hosteler (staying in IIIT Delhi campus) and five non-hosteler. For hosteler students, home and workplace location are highly lightly to overlap as they spend most of their time in labs, even during night time. As for the non-hosteler, we expect to find distinct home and workplace location. Lastly, our home and workplace score allocation scheme is independent of each other, and hence we do not expect any problem even when they two places overlap each other.

User	Home Days	Office days	Total Days	No. of distinct latitude-longitude
U_1	10	12	12	86
U_2	6	11	13	93
U_3	7	5	10	80
U_4	2	14	14	61
U_5	3	11	15	48
U_6	6	13	17	168

Table 3.1: User Location information details

Table 3.1 shows details about the location information of the five users.U_3 is a hosteler while U_1, U_2, U_4, U_5 and U_6 are non-hosteler. Home days means number of days for which we have data for home hours i.e. 12 midnight to 5 am or weekends. Similarly, office days means number of days for which we have data for time between 10 am to 5 pm. Figure 3.1 shows the scores of the predicted home and office location for the five users.

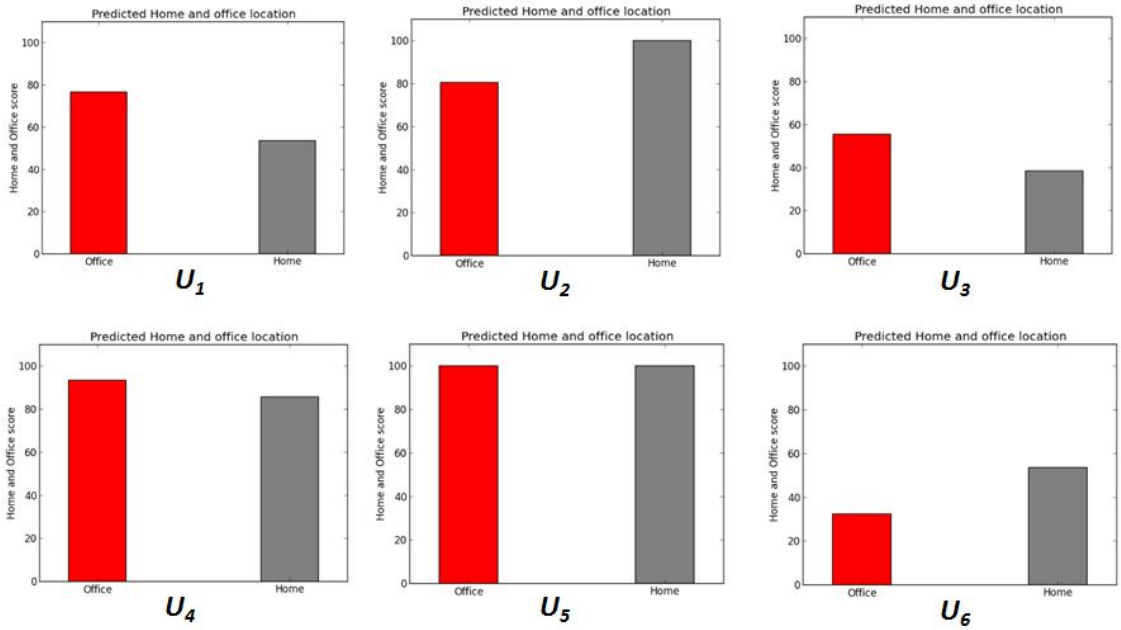

Figure 3.1: Scores of the predicted Home and Office Locations

User	Distance between actual and inferred home location (in KM)	Distance between actual and inferred office location (in KM)
U_1	0.4151	0
U_2	0.04253	0
U_3	0.1068	0.1604
U_4	7.384	0
U_5	2.24	0
U_6	0.1558	0

Table 3.2: Difference between inferred and actual home and office locations.

We consider a prediction as a successful one if the distance between predicted and actual location is less than 500 meter. Table 3.2 shows distance between actual and predicted home and office locations for the five users.

For all the users, office locations are predicted successfully except in case of user U_4 and U_5 which is possibly due to less number of *Home Days*. From Table 3.2 we find that at least 5 days of location data is required to correctly identify home and workplace location.

This experiment shows one of the possible information that can extracted from raw location data of users.

Chapter 4

Literature Review

4.1 Personal Data Vault: A locus of control for Personal data streams(PDV)

Mun et al. [12] proposed a model called *Personal Data Vault (PDV)*, in which user information are stored in a secure container, to which only the user has complete access. A cloud-based virtual machine acts as the secure container i.e. PDV and any access to user information has to be done through it. For each application, the PDV has user-defined Access Control List which specifies the *bound, precision and frequency.Bounds* decides which subset of data to be shared with third parties by limiting data space. *Precision* can be in terms of time, location, numerical value, etc. For example, in case of location information precision could be at the level of street, zip code, city, country. While *frequency* sets a limit on how frequently an application can request for user information. Depending on the ACL, only a subset of information, which user is comfortable sharing, is provided to the application. It basically allows user to control what to share with whom and also the granularity of the data being shared.

The disadvantages of this model is that providing a cloud-based virtual machine for each user can be costly. Also some amount of data is received by the service provider (even though its limited).

4.2 *Koi*:A Location-Privacy Platform for Smartphone Apps

Saikat et al. [6] proposed koi which preserves location privacy by taking out the association between the user and its locations. That is, there are two non-colluding components: 1) *combiner*: which knows the user and its location, but does not know the association between the two. 2)*matcher*: which knows the association between the (anonymized) user

and the encrypted location.

As the combiner has location information of the user in plain text, it is possible to analyze the significant location of the anonymized user. From the significant location with or without other external information, it may be possible to map the location information to its owner.

4.3 Trust No One: A Decentralized Matching Service for Privacy in Location Based Services

In [7], Jaiswal and Nandi proposed an approach wherein the matching of an end-user's location and their interests in a Location Based Service (LBS) is carried out by an entity called *Matching Service* which is unaware of the actual value (or identification) of both the location and the interest. Mobile operator partitions physical space into location grids and assigns *pseudonymized location (PL)* to user's and business locations. It builds on the idea that mobile operators already has coarse-grained information of every mobile user. LBS provider who has a list of all the users and business which are a part of the LBS and assigns each of them a *pseudonymized identifer (PI)*. The *matching service* has a set of PIs and PLs pairs for business and users. When mobile user registers in the LBS, it gets PIs corresponding to the business and users of interest. Before sending a LBS query, the mobile user contacts the mobile operator for the PL corresponding to its current location. It can then initiate a LBS query, the generic form of which is a 3-tuple <user's-PI, user's-PL, desired-PI-list >.The *matching service* then checks if there are any PIs of interest to a user, that are located in a location-grid with the same PL as the mobile user, and if yes, it creates a trigger to the end-user app notifying of nearby PIs of interest. The mobile user can decode which businesses / acquaintances correspond to the notified PIs.

The authors claim that their approach can be used for locating nearby friends is questionable. The architecture assumes that the LBS already knows the location of the entities that should be located. This assumption is reasonable only for businesses but not for cases where the concerned entity is a user. The assumption conflicts with the goal of the architecture that the LBS provider does not learn about user's location.

4.4 SMILE: Encounter-Based Trust for Mobile Social Services

SMILE [10] attempts to allow users equipped with mobile devices to build such trust relationships while preserving their privacy against potential attackers (e.g., the central storage server and other users). In SMILE, users who want to communicate with each other must prove that an encounter occurred. To do this, an interested person generates and

passively broadcasts the âĂIJencounter keyâĂİ to others within his communication range, and posts a hash of the encounter key, along with a message encrypted using the encounter key, to a centralized server. When a user sends message to a server it is encrypted with one such key and labels it with the corresponding key hash. The server forwards the encrypted message to all users that have uploaded the same key hash and only encounter participants are able to decrypt the message.

SMILE has two main weakness [11]: first that it is vulnerable to impersonation attack performed by an eavesdropper since no authentication is required during key agreement. Secondly, it is vulnerable to user collusion. A few colluding users may possess enough information about the activities of honest users for the server to unmask the identity of the communicating users.

4.5 Contrail: Enabling Decentralized Social Networks on Smartphones

Partick et al. [14] proposed a decentralized social network where privacy is achieved by encrypted end-to-end communication. The main idea of their approach is the *Contrail Filter* - an application defined function which returns true or false based on given input data. If a user C (consumer) wants to receive data from user P (producer), C sends content filter request to P. The filter gets installed if P accepts the request. The filter is evaluated by P on any new data; if it matches, the data is transmitted to C. For example, if Alice wants to get notified if her son Junior goes out of certain distance from his house, Alice installs a Contrail filter in Junior's phone. The filter checks the location data of Junior periodically and if happens to cross the set limit, it sends an alert to Alice.

It has usability issue as user will be asked to permission to install filter for each data his friend want from him and that also separately for each friend.

4.6 Spatial and Temporal Cloaking

This approach is based on the *k*-anonymity principle [15], in which the aim is to at make an anonymized record in the database indistinguishable from the records of at least *k-1* other users. Spatial and temporal cloaking makes sure that every location based service request cannot be distinguished from threshold number of requests. A trusted anonymizer [5] or a decentralized mix network [2] removes any identifier of the user and reduces the precision of either the spatial (location) or temporal information depending on inaccuracy tolerance of the LBS . For example, in case of a request from a user to find nearby restaurant response time of the service is important but precision of location can be

decreased somewhat i.e. about 100 m of accuracy can be tolerated by the LBS.

Spatial and temporal cloaking approach can be adapted to meet the requirements of different types of applications. However, if the number of users in a particular location (biggest location allowed by precision requirement of the LBS) never exceed k then it fails to provide the promised anonymity. Also research [4] has shown that it is inappropriate as a method for location privacy.

4.7 Search Me If You Can: Privacy-preserving Location Query Service

Taeho et al. [8] proposed a Privacy Preserving Distance Comparison Algorithm using Paillier's homomorphic encryption. Their algorithm allows comparison of distance between two locations against a threshold value without disclosing the actual location information. This algorithm was however not implemented on a phone.

In our work we have used their algorithm to compare user's mobility profiles, which consists of multiple location informations. We have successfully implemented the adapted protocol on Android phone with reasonable overhead.

Chapter 5

Proposed Models

We redefined the architecture of MobiShare to ensure location privacy of users. We also protect contacts of people not using MobiShare. In this chapter, we will discuss features and procedures of the proposed model. Hereafter we use the term *"requester"* for the user searching a file and *"potential provider"* for users (friends) who have shared the queried file. Also *"contacts"* of a users means her friends in MobiShare and the *"Cloud"* for the cloud part of MobiShare.

5.1 Registration and File Sharing

Registration of user for using MobiShare remains basically the same with a small addition of storing hash value of registered user's phone number and Facebook ID. Next user chooses the files he wants to advertise for sharing and attributes of these files are uploaded to Cloud as before.

5.2 Phone and Facebook Contacts

Uploading phone number and Facebook ID of friends to Cloud discloses contacts of friends who are not participating in MobiShare. Instead the new system uploads hash value of phone numbers and Facebook ID. As MobiShare has phone numbers of its registered users, it can calculate their hash value and build friend network for its users. Hash being a one-way function, contacts of those not using MobiShare remain hidden. Figure 5.1 shows a simplified example of how MobiShare cloud can identify friends of a user who are also using MobiShare without disclosing the contact information of the non-users.

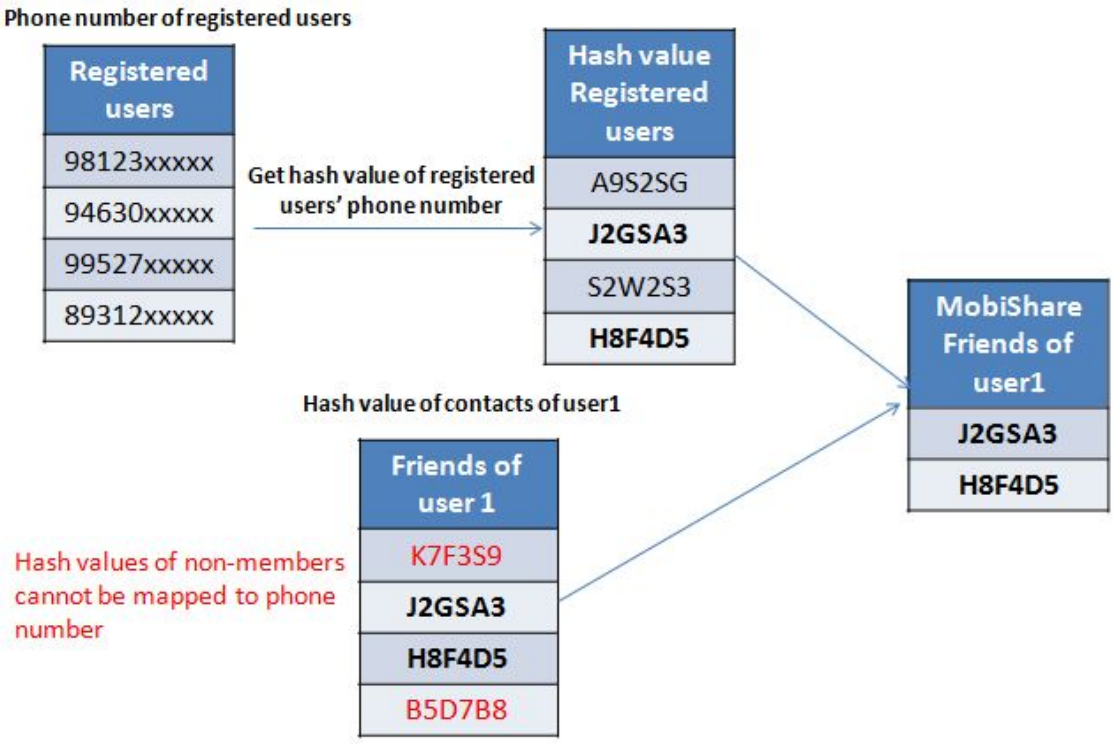

Figure 5.1: Finding MobiShare friends of a user without disclosing the contacts of non-users

5.3 Location Data

In the new system, instead of uploading user's location data to cloud it is stored in phone itself, thus giving user complete control over his data. Space requirement for storing timestamped location information is very small, only a few KBs.

5.4 Mobility Profile

The raw latitude longitude informations are converted into Mobility Pattern in the phone using the same algorithm as that in the original MobiShare. The location information in the mobility profile is encrypted and uploaded to cloud.Based on the encryption algorithm used, we propose two models which provides varying degree of location privacy to the user.

1. **Partial Location Privacy Preserving MobiShare**
   In this model the user's location are hidden from the cloud but not from his friends with who he exchange file(s).

2. **Full Location Privacy Preserving MobiShare**
   In this model user's location is hidden from the cloud as well as his friends.

We will see details about these two models in the following sections.

5.4.1 File Searching

Search query structure is same as before. Whenever a search request comes, Cloud finds out *potential providers* and helps the requester and potential providers exchange their mobility profile. As the cloud does not have access to user's location data or mobility profile, computation for predicting rendezvous point is done in phone.The two proposed models have different protocols for exchanging and comparing the mobility profile and are discussed in detail in their respective sections.

5.5 Partial Location Privacy Preserving MobiShare

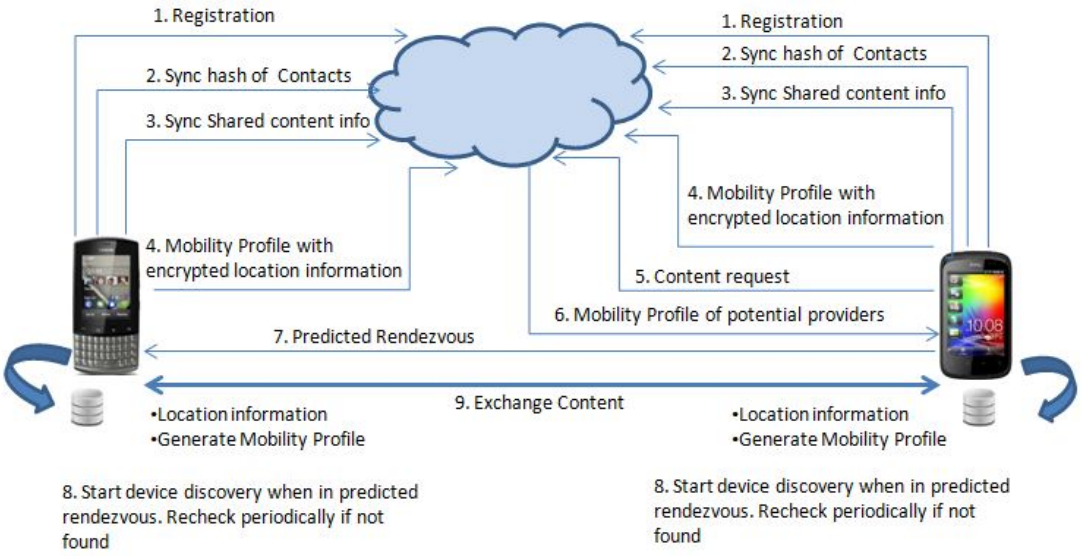

Figure 5.2: Partial Location Privacy Preserving MobiShare

5.5.1 Encryption Algorithm

Partial Location Privacy Preserving MobiShare model use *Symmetric-key algorithm* for encrypting location information. Symmetric-key algorithms are encryption systems in which the sender and receiver of a message share a single, common key that is used to encrypt and decrypt the message.

Each user has his own unique key and encrypts location information in his mobility profile using that key. He sends his secret key to his friends using *Public Key Encryption*.

Public Key Encryption consists of two keys: a public key known to everyone and a private or secret key known only to the recipient of the message. When Alice wants to send a secure message to Bob, she uses Bob's public key to encrypt the message. Bob then uses his private key to decrypt it.

The sample of the encrypted mobility profile is shown below.This mobility profile with encrypted location information is uploaded to the cloud. Once uploaded, the user need not update his mobility profile until there is a drastic change in his mobility profile.

Sample Mobility Profile for Partial Location Privacy Preserving MobiShare

```
{
              "user": "15555215554",
              "type": "MobilityPattern",
    "MobilityProfile":
    [
        {
          "placeID": "P1",
           "encLat": "ae813d66d753922b6f10c6c191688af",
           "encLon": "e0f79484f6545b437237f517dba54dg",
         "dayNtime":[
                        {"sTime":"9:30", "eTime":"12.30","day":"tue"},
                        {"sTime":"9:50", "eTime":"2.30","day":"wed"}
                    ]
         }
  ]

}
```

5.5.2 Rendezvous Predication

When a requester search a file, the Cloud sends him mobility profiles of potential providers. If the number of potential providers is more than a threshold value, then the user is first returned the list of potential providers and selects users he wants get the requested file from; cloud then sends him mobility profile of selected users only. Requester decrypts the location information in the mobility profile using their corresponding key which he already knows. He then compares the decrypted mobility profile with his own and finds out rendezvous points and expected time of delivery. The predicted rendezvous are then sent to potential providers via cloud.

Sample Predicted Rendezvous Message


```
{
              "user": "15555215554",
              "type": "Rendezvous",
        "Rendezvous": [
                        {
                                  "placeID": "P1",
                                      "day": "wed",
                                "startTime": "10:30",
                                  "endTime": "11:40"
                        }
                        {
                                  "placeID": "P3",
                                      "day": "tue",
                                "startTime": "12:10",
                                  "endTime": "1:15"
                        }
                  ]
}
```


5.5.3 Local Sharing

Whenever either user is at predicted rendezvous, MobiShare starts device discovery to check if they are nearby each other. If they discover each other, the requested content is transferred to the requester. If not device is found, they recheck periodically.

5.5.4 Summary

As location information in the mobility profile of the user is encrypted the cloud never learns about user's location information. This is a very simple approach and has comparatively low data usage and simpler computations.

This model assumes that user's friends are not malicious and user does not mind sharing his location with his friends. This assumption however may not be always true. For example a users may be not want his colleague in office to know where he goes to after office hours. The Full Location Privacy Preserving MobiShare overcomes this problem.

5.6 Full Location Privacy Preserving MobiShare

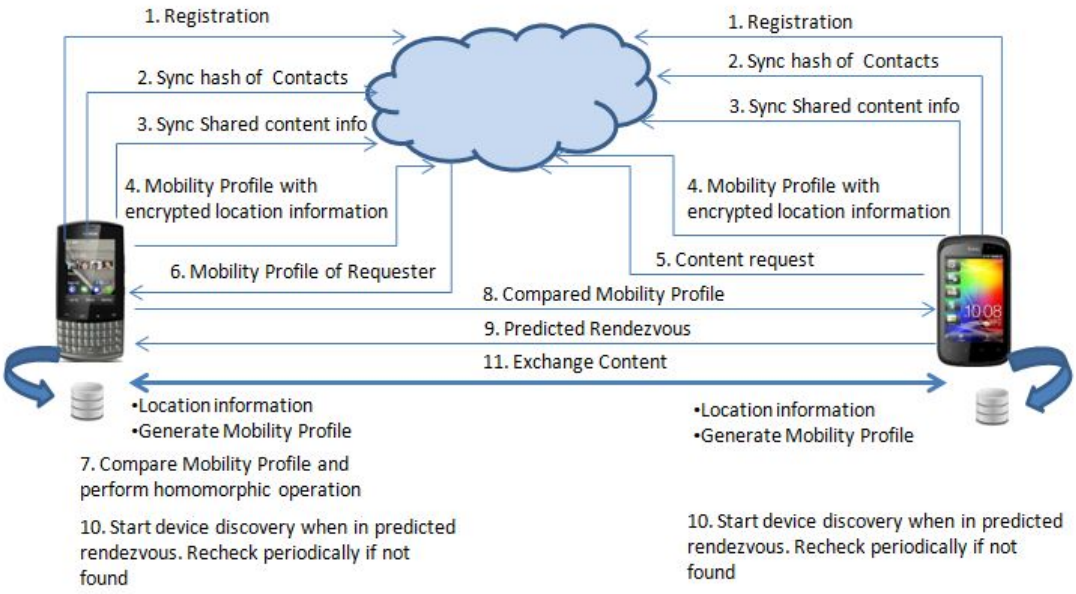

Figure 5.3: Full Location Privacy Preserving MobiShare

5.6.1 Encryption Algorithm

This model uses Paillier Encrytion which is a homomorphic encryption. Homomorphic encryption is a form of encryption which allows specific types of computations to be carried out on ciphertext and obtain an encrypted result which decrypted matches the result of operations performed on the plaintext.

Homomorphic Properties of Paillier's Cryptosystem

Paillier's encryption satisfies the following homomorphic properties:

$$\mathbb{E}(m_1, r_1).\ \mathbb{E}(m_2, r_2) = \mathbb{E}(m_1 + m_2, r_1 r_2) \bmod n^2$$
$$(\mathbb{E}(m_1, r_1))^{m_2} = \mathbb{E}(m_1.m_2, r_1^{m_2})$$

Paillier's Cryptosystem

Paillier's cryptosystem [13] consists of three algorithms - **KeyGenerate**, **Encrypt**, **Decrypt**.

1. **KeyGenerate:**
   An entity randomly chooses two large prime numbers p and q of same bit length. Then compute $n = pq$ and $\lambda = (p-1)(q-1)$. Let $g = (n+1)$ and $\mu = (\lambda \bmod n^2)^{-1} \bmod n$. Encryption key is $EK = (n, g)$ and the decryption key is $DK = (\lambda, \mu)$.

2. **Encrypt**
   The encrypter selects a random integer $r \in \mathbb{Z}_n$ and computes the ciphertext $\mathbb{E}(m, r) = g^m.r^n \bmod n^2$ and publishes it.

3. **Decrypt**
   The holder of $DK = (\lambda, \mu)$ can decrypt the ciphertext $\mathbb{E}(m, r)$. He computes the following to recover the message:

$$m = L((\mathbb{E}(m, r))^{\lambda} \bmod n^2).\ \mu \bmod n$$

   where $L(a) = (a - 1)/n \bmod n$

Note that DK can decrypt only the ciphertext encrypted with EK which pairs with it. Also the random number r in ciphertext $\mathbb{E}(m, r)$ does not contribute in encryption or homomorphic encryption.

5.6.2 Encrypting Location information Mobility Pattern

Earth is not a perfect sphere. But for the accuracy requirement of MobiShare, it is fair enough to consider earth as a spherical body with generally accepted value of radius, $R = 6371Km$ With this assumption, latitude and longitude (in radians) can be converted into three dimensional co-ordinates as follows:

$$y_1 = R * cos(latitude) * cos(longitude)$$
$$y_2 = R * cos(latitude) * sin(longitude)$$
$$y_3 = R * sin(latitude)$$

Let(y_{i1}, y_{i2}, y_{i3}) be the corresponding Euclidean co-ordinates for each $(latitude_i, longitude_i)$ pairs in the mobility profile. Then, the following Paillier encrypted values are included in the mobility profile as location information:

$$\mathbb{E}_Q\left(\sum_{i=1}^{3} y_i^2\right), \{\mathbb{E}_Q(-2y_i) \mid i = 1, 2, 3\}$$

Section 5.6.2 show a sample of encrypted mobility profile uploaded to the Cloud.

Sample Mobility Pattern for Full Location Privacy Preserving MobiShare


```
{
    "return": "pullmessage",
    "sender": "15555215554",
      "type": "MobilityPattern",
"public key": "220059214949056211314274470684043246377",
  "mProfile":[
  {
          "placeID": "P1",
           "encLoc": {
                "sqSum":"964611434836784264690804494218609138536609703277881
                   657157384462718077849666",
                "-2y1":"165256725551475099652554573949903145806730214180504 1
                   857522855791555122489974 9",
                "-2y2":"328637040900172379256636480318279041587455835486599 2
                   514749853196313240931527 9",
                "-2y3":"244185068078103283580171200764735437474773730685383 9
                   832957183240496092011180 8",
                        },
        "dayNtime": [{"sTime": "9:40","eTime": "12:30","day": "wed"}],
  }
                ]
}
```


5.6.3 Rendezvous Predication

Rendezvous Prediction in this model is done as follows:

1. When requester sends a search request, Cloud finds *potential providers* and sends them mobility profile of *requester*.
2. Potential Provider then compares his mobility profile with that of the requester's by first finding out places with time overlap.
3. If overlap exists, provider includes his corresponding location information in the encrypted location information of the requester by performing a series of homomorphic operations. This operations give encrypted Euclidean distance between the two locations. Also an encrypted threshold value is generated. Algorithm 2 describes the steps in detail.

4. Potential Provider then sends back the compared Mobility Profile consisting of encrypted euclidean distance and threshold distance along with time overlap to the requester. A sample Compared Mobility Profile message is shown below.

Sample Mobility Profile Comparison Response


```
{
              "user": "15555215556",
              "type": "ComparisionResponse",
    "ComparedMobilityProfile":
    [
        {
               "placeID": "P1",
           "encEuclnDist": "875432125551475099652554573949903145806730214180
              504185752285579155519873790 87",
           "encThreshold": "724459126751475099652554573949903145806730214180
              504185752285579155512843 9875",
         "dayNtime":[
                        {"sTime":"10:30", "eTime":"12.30","day":"tue"},
                        {"sTime":"10:50", "eTime":"2.30","day":"wed"}
                    ]
        }
  ]

}
```


5. The requester decrypts the returned values using his private key and check if

$$\delta|\mathbf{x\text{-}y}|^2 + \delta^{'} < \delta\tau^2 + \delta^{'} \Leftrightarrow |\mathbf{x\text{-}y}| < \tau$$

If the euclidean distance value is less than threshold value, then requester the place and time overlap is one of the potential rendezvous. The provider is also notified the same. The structure of the message exchanged is same as that used in Partial Location Privacy Preserving Model.

5.6.4 Local Sharing

Once the predicted rendezvous is communicated, the method for local sharing is the same as the previous model.

Algorithm 2: Privacy Preserving Distance Comparison

Input : $\mathbb{E}_Q\left(\sum_{i=1}^{3} y_i^2\right)$, $\{\mathbb{E}_Q(-2y_i) \mid i = 1, 2, 3\}$, (x_1, x_2, x_3) and public key of requester Q. (y_1, y_2, y_3) represent location of the requester and (x_1, x_2, x_3) that of the provider P.

Output: Encrypted Euclidean Distance and Encrypted threshold distance

1: P, randomly generate $\delta \in \mathbb{Z}_{2^{972}}$, $\delta' \in \mathbb{Z}_{2^{1022}}$ and executes the following homomorphic operations:

$$\{\mathbb{E}_Q(-2y_i)^{\delta x_i} = \mathbb{E}_Q(-2\delta x_i y_i) \mid i = 1, 2, 3\}$$

$$\mathbb{E}_Q(\sum_{i=1}^{3} y_i^2)^{\delta} = \mathbb{E}_Q(\delta(y_1^2 + y_2^2 + y_3^2))$$

$$\mathbb{E}_Q(1)^{\delta \sum_{i=1}^{3} x_i^2} = \mathbb{E}_Q(\delta \sum_{i=1}^{3} x_i^2)$$

$$\mathbb{E}_Q(1)^{\delta'} = \mathbb{E}_Q(\delta')$$

$$\mathbb{E}_Q(\delta \sum_{i=1}^{3} x_i^2).\ \mathbb{E}_Q(\delta') = \mathbb{E}_Q(\delta \sum_{i=1}^{3} x_i^2 + \delta')$$

2: P then computes the following:

$$\mathbb{E}_Q(\delta \sum_{i=1}^{3}) x_i^2 + \delta')\ .\ \mathbb{E}_Q(\delta \sum_{i=1}^{3} y_i^2) \prod_{i=1}^{3} (\mathbb{E}_Q(-2\delta x_i y_i))$$

$$= \mathbb{E}_Q((\delta \sum_{i=1}^{3} (x_i - y_i)^2) + \delta')$$

$$= \mathbb{E}_Q(\delta |\mathbf{x\text{-}y}|^2 + \delta')$$

$$\mathbb{E}_Q(1)^{\delta\tau^2 + \delta'} = \mathbb{E}_Q(\delta\tau^2 + \delta')$$

3: Return $\mathbb{E}_Q(\delta\tau^2 + \delta')$ and $\mathbb{E}_Q(\delta |\mathbf{x\text{-}y}|^2 + \delta')$

NOTE: The reason δ and δ' are chosen from $Z \in 2^{972}$ and $Z \in 2^{1022}$ is because otherwise the comparison is not correct due to the modular operations considering that its a 1024 bit Paillier encryption that we are using. It is explained in details in [8].

5.6.5 Summary

This model allows rendezvous prediction without disclosing the actual location information. Friends however learns about the location of the predicted rendezvous. However, if a place is there in the predicted rendezvous it means there is a high probability of the two users being their. So the users must have already seen each other at that place at that time many times and therefore, the location information is not private. That is, if Alice and Bob works in the same office, and the predicted rendezvous is their office during their normal working hour, either user does not learn any extra information about the location of each other as they have seen each other at that place and time and hence the information is no

longer private.

A possible attack by a malicious friend will be to find out the usual arrival and departure time of a user to a known place. To do this attacker can leave his phone to the known place for a week so that his mobility profile will consist of only that location and stay duration will be the whole day. When his mobility profile is compared with that of the victim, it will disclose arrival and departure time of the victim to that place.

This approach requires greater amount of message exchange and also has higher data usage. The overhead and the feasibility of the approach is discussed in Chapter 5.

Chapter 6

Performance Evaluation

The mobile application of MobiShare was implemented on Android and tested on Android 2.3, 512 RAM HTC Wildfire S. We used a local server with MySQL database as the Cloud for MobiShare. We have executed each process 1000 times and measured the runtime for each.

6.1 Storage of Location in Phone

Storing timestamped latitude-longitude information for a week in SQLite database, recorded at a frequency of 1 record per 10 minute occupy 42 KB of space. This amount of space is reasonable considering storage capacity of present day mobile phones.

6.2 Generating Mobility Pattern

The location information from the SQLite database stored in SD card was used to generate mobility profile of the user. We generated the mobility profile of a user with 720 records collected over a duration of 12 days. The average time taken to access this records and convert them into mobility profile is approximately 1.5 seconds.

6.3 Hash of phone number and Facebook ID

We used 256-bit SHA-2 algorithm to calculate the hash value. The average runtime for a single hash calculation is 1.458 ms. A single hash value occupies 105 bytes of space.

6.4 Partial Privacy Preserving MobiShare

6.4.1 Encryption and Decryption

The symmetric key encryption algorithm we used is Advanced Encryption Standard (AES). Its performance on Android phone is as follows:

- Average encryption runtime is 3.397 ms.
- Average decryption runtime is 5.979 ms.
- Space required to store an encrypted latitude longitude pair is 66 bytes.

6.5 Full Privacy Preserving MobiShare

6.5.1 Encryption

We used Java *thep* library [1] for Paillier encryption. Encryption of all the four location values were considered to calculate the average runtime.Table 6.1 shows the performance of encryption algorithm. We used tried two key sizes for the Paillier's encryption: [3] 512 bits(reasonably secure) and 1024 bits(more secure).

Key Size (bits)	Average runtime (ms)	Encrypted Data (bytes)
512	62.783	1244
1024	424.921	2470

Table 6.1: Performance of Encryption

6.5.2 Homomorphic operation to compute Euclidean distance and threshold distance

The average time required to perform homomorphic operations on the encrypted data in order to compute the euclidean distance and to encrypt the threshold distance is shown in Table 6.2.

Key Size (bits)	Time taken (ms)	Encrypted Data (bytes)
512	179.357	618
1024	952.320	1234

Table 6.2: Performance of homomorphic operations

6.5.3 Decryption

Table 6.3 shows the performance of decryption. Average runtime is average of the total time required to decrypt euclidean distance and threshold distance.

Key Size (bits)	Average runtime (ms)
512	52.278
1024	368.617

Table 6.3: Performance of Decryption

6.6 Comparison

We choose 512-bit key size Paillier's encryption for the Full Location Privacy Preserving Model as it is reasonably secure and has lesser data usage compared to 1024-bit key size.

We observed location data of 5 users for between 10 to 15 days and found the average of 9 unique locations in their mobility profile. The overhead for encrypting 9 location information in the proposed models are shown in Table 6.4

Model	Average encryption runtime(ms)	Encrypted data size	Average decryption runtime (ms)
PLPP	30.573	594 bytes	53.811
FLPP	565.047	10.92 KB	470.502

Table 6.4: Comparison of PLPP and FLPP model

As seen from the comparision, FLPP is expensive as compared to PLPP. However, if we see FLPP idependently, then 11KB of data is reasonable even for slow speed netowrk connection. A hybrid FLPP and PLFF model can also be employed where users can choose which protocol to use depending on trustability of friends.

Chapter 7

Conclusion and Future Work

In the original architecture of MobiShare, by uploading location information users are at risks of being tracked. Also, uploading phone contacts breach privacy of individuals not using MobiShare. We proposed a modified architecture of MobiShare in which the cloud never learns about location information of users.

Users have two options on how to exchange his location information with his friends. User can choose to disclose his location information to his friends and opt for the Partial Privacy Preserving Model. This approach is less expensive in terms of data usage and computation. The downside is that friends learn about locations the user generally visits but some people may not complain with this model. If user does not want to disclose any location information even to his friends, he can go for the Full Privacy Model which allows comparing the mobility profile of users without disclosing the actual location information of requester and provider. This model however demands comparatively heavier data usage. We showed the feasibility of our models and reported the overhead introduced by the privacy preserving component.

Lastly, we used a very simple concept of taking hash value of contacts to protect privacy of individuals not using MobiShare.

As a future work, we would like to preserve privacy of what files users are sharing.

Bibliography

[1] thep: The homomorphic encryption project - computation on encrypted data for the masses. https://code.google.com/p/thep/. Online accessed 4-April-2013.

[2] Beresford, A., and Stajano, F. Location privacy in pervasive computing. *Pervasive Computing, IEEE 2*, 1 (2003), 46–55.

[3] Catalano, D., Gennaro, R., and Howgrave-Graham, N. The bit security of paillier's encryption scheme and its applications. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology* (London, UK, UK, 2001), EUROCRYPT '01, Springer-Verlag, pp. 229–243.

[4] Duckham, M., and Kulik, L. A formal model of obfuscation and negotiation for location privacy. In *Proceedings of the Third international conference on Pervasive Computing* (Berlin, Heidelberg, 2005), PERVASIVE'05, Springer-Verlag, pp. 152–170.

[5] Gruteser, M., and Grunwald, D. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services* (New York, NY, USA, 2003), MobiSys '03, ACM, pp. 31–42.

[6] Guha, S., Jain, M., and Padmanabhan, V. Koi: A Location-Privacy Platform for Smartphone Apps. In *Proceedings of the 9th Symposium on Networked Systems Design and Implementation (NSDI)* (San Jose, CA, Apr 2012).

[7] Jaiswal, S., and Nandi, A. Trust no one: a decentralized matching service for privacy in location based services. In *Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds* (New York, NY, USA, 2010), MobiHeld '10, ACM, pp. 51–56.

[8] Jung, T., and Li, X.-Y. Search me if you can: Privacy-preserving location query service. *CoRR abs/1208.0107* (2012).

[9] Khetarpaul, S., Chauhan, R., Gupta, S. K., Subramaniam, L. V., and Nambiar, U. Mining gps data to determine interesting locations. In *Proceedings of the 8th International*

Workshop on Information Integration on the Web: in conjunction with WWW 2011 (New York, NY, USA, 2011), IIWeb '11, ACM, pp. 8:1–8:6.

[10] Manweiler, J., Scudellari, R., and Cox, L. P. Smile: encounter-based trust for mobile social services. In *Proceedings of the 16th ACM conference on Computer and communications security* (New York, NY, USA, 2009), CCS '09, ACM, pp. 246–255.

[11] Mohaisen, A., Vasserman, E. Y., Schuchard, M., Foo Kune, D., and Kim, Y. Secure encounter-based social networks: requirements, challenges, and designs. In *Proceedings of the 17th ACM conference on Computer and communications security* (New York, NY, USA, 2010), CCS '10, ACM, pp. 717–719.

[12] Mun, M., Hao, S., Mishra, N., Shilton, K., Burke, J., Estrin, D., Hansen, M., and Govindan, R. Personal data vaults: a locus of control for personal data streams. In *Proceedings of the 6th International Conference* (New York, NY, USA, 2010), Co-NEXT '10, ACM, pp. 17:1–17:12.

[13] Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th international conference on Theory and application of cryptographic techniques* (Berlin, Heidelberg, 1999), EUROCRYPT'99, Springer-Verlag, pp. 223–238.

[14] Stuedi, P., Mohomed, I., Balakrishnan, M., Mao, Z. M., Ramasubramanian, V., Terry, D., and Wobber, T. Contrail: enabling decentralized social networks on smartphones. In *Proceedings of the 12th International Middleware Conference* (Laxenburg, Austria, Austria, 2011), Middleware '11, International Federation for Information Processing, pp. 40–59.

[15] Sweeney, L. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 10*, 5 (Oct. 2002), 557–570.

[16] Yves-Alexandre de Montjoye, Cesar A. Hidalgo, M. V. . V. D. B. Unique in the crowd: The privacy bounds of human mobility. http://www.nature.com/srep/2013/130325/srep01376/pdf/srep01376.pdf, 2013. Online accessed 4-April-2013.