Privacy-Preserving Location Sharing System with Client/Server Architecture in Mobile Online Social Network

Xi Xiao, Chunhui Chen, Xinyu Liu, Guangwu Hu, Yong Jiang

Abstract—Location sharing is a fundamental service in mobile Online Social Networks (mOSNs), which raises significant privacy concerns in recent years. Now, most location-based service applications adopt client/server architecture. In this paper, a location sharing system, named CSLocShare, is presented to provide flexible privacy-preserving location sharing with client/server architecture in mOSNs. CSLocShare enables location sharing between both trusted social friends and untrusted strangers without the third-party server. In CSLocShare, Location-Storing Social Network Server (LSSNS) provides location-based services but do not know the users’ real locations. The thorough analysis indicates that the users’ location privacy is protected. Meanwhile, the storage and the communication cost are saved. CSLocShare is more suitable and effective in reality.

Keywords—Client/server architecture, location sharing, mobile online social networks, privacy-preserving.

I. INTRODUCTION

With the development of mobile Internet, the social network enters the era of mOSNs. In a traditional social network that supports Location-Based Services (LBSs), such as Foursquare and Gowalla, users share their locations by means of “check-in”, which helps the online website/application to record the time and the location of each user. The inefficient and inconvenient location sharing situation is not changed until the appearance of mobile devices, which can obtain the owner’s location through Global Positioning System (GPS) or cellular geolocation anywhere and anytime. Since mobile devices can be online all the time and they can be everywhere, real-time interaction of users is realized in mOSNs. More excitingly, users can conveniently use various LBSs provided by mobile devices, such as recommendation of good friends or searching Points of Interests (POIs) including hotels, hospitals, and restaurants. However, behind the convenience brought by mOSNs, there comes an indispensable security risk of privacy. In the era of data, location data does not only mean the place of an individual, but also presents his/her home, interests, physical conditions, and so on. Therefore, location records should be regarded as unusually sensitive information and they deserve a high degree of attention.

In order to flexibly share privacy-preserving location in mOSNs, Wei et al. [1] introduced MobiShare in 2012. Since MobiShare enables location sharing between both trusted social relations and untrusted strangers, it is adaptable to support a variety of location-based applications. Based on MobiShare, N-MobiShare [2], [3], MobiShare+ [4], and BMobiShare [5] have been successively put forward. N-MobiShare [2], [3] uses Social Network Server (SNS) instead of Cellular Tower (CT) to forward users’ location updates and requests to Location Based Server (LBS) without leaking anything about the location information. Shortly afterwards, MobiShare+ [4] employs dummy queries and the private set intersection protocol to prevent SNS and LBS from learning individual information from each other. In order to improve transmission efficiency, BMobiShare [5] replaces the private set intersection protocol in MobiShare+ with Bloom Filter. In fact, most of the LBS applications have client/server architecture [6]-[8]. Therefore, all these systems, which depend on the third-part LBS, are not suitable in the real world. Further, the third-part LBS generates additional communication overhead and economic costs. Moreover, if SNS and LBS work together with each other, they can easily get both users’ individual profile and location information.

In order to overcome the above drawbacks, we present a new privacy-preserving location-sharing system for client/server based applications in mOSNs, named CSLocShare. Different from MobiShare, N-MobiShare, MobiShare+, and BMobiShare, in our system, SNS and LBS have been amalgamated into one single server, i.e., LSSNS. The efficiency in communication is improved, while the protection effect on the users’ privacy is the same as the previous systems. Users’ basic social information and location data are stored in LSSNS, and every user has one real location and k-1 dummy locations. So, LSSNS cannot know the users’ real location and only CT is able to identify the real location. Meanwhile, CT no longer needs to store any users’ related records and just only does some simply computation.

There are three main contributions in our work as follows:

1. We observe that, in real life, most of the mobile social applications have client/server architecture without the third-party server such as LBS. Therefore, we design a new location-sharing system named CSLocShare, which does not depend on the third-part server and can achieve the same protection goal on the users’ privacy.

2. In our system, CT does not keep users’ relevant information, thus the storage load has considerably decreased, while, in MobiShare, MobiShare+, and BMobiShare, CT has to store users’ relevant information to ensure that SNS and LBS are able to cooperate with each other to complete users’ location updates and requests.

Xi Xiao is with Graduate School at Shenzhen, Tsinghua University, Shenzhen, China (e-mail: xiaox@sz.tsinghua.edu.cn).
3. The nearby friends’ and strangers’ locations are obtained in one entity, LSSNS, which does not require the cooperation between SNS and LBS. Thus, the encryption and transmission of the interactive data between SNS and LBS are eliminated.

This paper is organized as follows. In Section II, we state the system model and threat model for our mechanism. In Section III, we detail our improved privacy-preserving mechanism and system design. In Section IV, we present a comprehensive security analysis. In Section V, we briefly describe related work. Finally, we draw the conclusion in Section VI.

II. PROBLEM STATEMENT

In this section, we mainly introduce the privacy issues of location-sharing in mOSNs. Meanwhile, we simply analyze several main attack models in mOSNs.

A. System Model

In Fig. 1, we show the architecture of our location-sharing system in mOSNs, which consists of three entities.

1. Mobile user U is an entity that intends to share U’s real-time location and to request locations of nearby friends and strangers.

2. LSSNS is an entity that not only stores U’s social profile and location information, but also provides LBSs according to U’s requests with nearby persons’ locations.

3. CT is an entity that helps U to communicate with LSSNS and executes some simple computation.

There are manifold challenges of location-sharing in mOSN. The first challenge is how to update U’s location in a privacy-preserving way when U arrives at a new place and the location has changed. When U wants to know the nearby friends’ and strangers’ locations, how to complete U’s location request without leaking both U and the qualified users’ individual privacy is the second challenge. We formalize these issues as follows.

We formalize the whole social network as a Graph \( G = (V, E) \), where \( V \in G \) is a set of identity vertices and \( E \in G \) is a set of edges. Suppose that \( V = \{ ID_1, ID_2, \ldots, ID_n \} \) is the identity set of all the users in the social network supporting location sharing. If there is a linked edge between \( ID_1 \) and \( ID_2 \), \( ID_1 \) and \( ID_2 \) own a trusted social relationship, i.e., friendship. If we do not think the privacy concerns, LSSNS maintains a location database, in which the location records are stored in the form of \( \{(ID, (x_{ID}, y_{ID}), df_{ID}, ds_{ID})_{DEF} \} \). In the record \( \{(ID, (x_{ID}, y_{ID}), df_{ID}, ds_{ID})_{DEF} \} \), \( ID \) is the user’s identity, \( (x_{ID}, y_{ID}) \) is his/her current location, \( df_{ID} \) is the distance threshold within which the user is willing to share his/her location to the nearby friends, and \( ds_{ID} \) is the distance threshold within which the user agrees to share location with close-by strangers.

In a social network, the users’ authorized location-based operations are real-location update and locations query of nearby friends/strangers. The first security issue is how to generate/update an appropriate record to LSSNS’s location database so as to satisfy desired security goals. The second security issue is how to make an operation model according to friends'/strangers’ location queries without violating privacy of both sides.

The locations query can be formalized in two sides:

1. For friends’ locations query, a user U with identity \( ID_U \) can send a query in the form of \( \{(ID, (x_{ID}, y_{ID}), df_{ID}, ds_{ID})_{DEF} \} \) satisfying the constraint \( (ID_U, ID) \in G \) and \( dist((x_{ID}, y_{ID}), (x_{ID}, y_{ID})) \leq \min(q_{ID}, df_{ID}) \), where ‘f’ presents the query of finding friends, \((x_{ID}, y_{ID})\) is the user \( ID_U \)’s current location, \( dist(,..) \) is a function to compute Euclidean distance, \( \min(,..) \) is a function to return minimum value in its input, \( q_{ID} \) is the distance threshold within which \( ID_U \) wants to find his/her friends, and \( df_{ID} \) is the distance threshold within which \( ID_U \) allows his/her friend to find himself.

2. For strangers’ locations query, a user U with identity \( ID_U \) can submit a request in the form of \( \{(ID, (x_{ID}, y_{ID}), df_{ID}, ds_{ID})_{DEF} \} \) getting all the nearby strangers’ locations \( \{(ID, (x_{ID}, y_{ID}), df_{ID}, ds_{ID})_{DEF} \} \) satisfying with \( dist((x_{ID}, y_{ID}), (x_{ID}, y_{ID})) \leq \min(q_{ID}, ds_{ID}) \), where ‘s’ presents the query of finding strangers, \( q_{ID} \) is the distance threshold within which \( ID_U \) wants to find the nearby strangers, and \( ds_{ID} \) is the distance threshold within which the user \( ID_U \) allows the others to find himself in strangers’ locations query.

B. Adversary Model

The system consists of three entities including mobile users, CT and LSSNS. In our trusted model, CT is assumed as a trusted entity, which can obtain the users’ complete information without rising privacy concerns. Here, we focus on security risk with respect to LSSNS and mobile users. There are three main attack models as follows.

\[ \text{Fig. 1 System architecture} \]
We consider that LSSNS is “honest-but-curious”. This means that LSSNS is supposed to honestly follow our presented protocol in general but attempt to acquire the users’ information as much as possible.

2) Dishonest mobile users. The dishonest mobile users try to use authorized operation to acquire the information outside the scope of their access privileges.

3) Collusion between LSSNS and Dishonest mobile users. LSSNS may collude with these malicious users. For example, a social network company’s employee may register for the location sharing service and collude with the server to extract the users’ complete social-location topology structure.

Different from the previous systems, our system merges SNS and LBS into one server, i.e. LSSNS so that there is no threat that SNS leaks social relationships to LBS or LBS leaks location information to SNS. In fact, LSSNS has the users’ social relationship network. So, one of most important goals is to prevent LSSNS from obtaining the complete location topology structure. In fact, in our system, LSSNS cannot identify the target user’s real location even though the target user’s real location is stored in it.

III. SYSTEM DESIGN

In this section, we detail our practical system, which separates the issue of privacy-preserving location sharing into two cases, sharing locations between trusted friends and untrusted strangers. A summary of the notations used in this section is given in Table I.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDU</td>
<td>User U’s social network identifier, regarded as his/her identity</td>
</tr>
<tr>
<td>(xv, yv)</td>
<td>User U’s real location</td>
</tr>
<tr>
<td>dfU</td>
<td>User U’s friend-case distance threshold</td>
</tr>
<tr>
<td>dsU</td>
<td>User U’s stranger-case distance threshold</td>
</tr>
<tr>
<td>qv</td>
<td>User U’s Distance threshold in strangers’ locations query</td>
</tr>
<tr>
<td>G</td>
<td>A social network graph stored at LSSNS</td>
</tr>
<tr>
<td>Keys</td>
<td>A symmetric key shared by user and his/her friends</td>
</tr>
<tr>
<td>TagU</td>
<td>A encrypted string that indicates the index of real location</td>
</tr>
<tr>
<td>KeyCT</td>
<td>Location-Storing Social Network Server’s secret key, shared with cellular towers</td>
</tr>
<tr>
<td>Decrypto(.)</td>
<td>A decryption function using secret key Key</td>
</tr>
<tr>
<td>distl(.)</td>
<td>A function to compute Euclidean distance</td>
</tr>
<tr>
<td>min(.)</td>
<td>A function to return minimum value in its input</td>
</tr>
</tbody>
</table>

A. System Overview

Before we detail our proposed system, we first give an overview of location-sharing mechanism which mainly includes four processes as follows.

1) Registration: Before using the location-sharing service in the social network, the user needs to register on LSSNS to get an account ID as his/her identifier. Then, the user sends his/her location-sharing preferences to LSSNS. And LSSNS stores the user’s location-sharing preferences in the database and provides individual LBSs according to the user’s preferences. In addition, the user also has to register on the local CT to obtain the permission of authorized operation.

2) Location Updates: When the user reaches a new place, he/she must update his/her location in LSSNS’s database to ensure that LSSNS knows the user’s real-time locations.

3) Friends’ Locations Query: When a user wants to know the nearby friends’ current locations, he/she can submit a friends’ locations query to LSSNS. After receiving the user’s query, LSSNS selects the user’s friends in the database, who satisfy the distance restriction of the user, and returns these friends’ present locations.

4) Strangers’ Locations Query: When a user intends to know the nearby strangers’ current locations, he/she can send a strangers’ locations request to LSSNS. After receiving a user’s query, LSSNS selects the users in the database satisfying the distance restriction of the user and transmits these qualified users’ locations to the requester.

B. Registration

There is no doubt that LSSNS distributes U a unique identifier in the social network and builds a friend-relationship network for U. Before mobile user U uses the location-sharing service, he/she needs to register the service in CT and LSSNS and shares location-sharing preferences with LSSNS. The main registration process is described as follows (Fig. 2 shows the message transmission in this stage).

Step 1. U sends a record in the form of {IDU, ‘reg’, dfU, dsU} to CT, where ‘reg’ presents the registration query, dfU is the distance threshold within which IDU allow his friends to find himself, dsU is the distance threshold within which the user IDU allows the others to find himself in strangers’ locations query.

Step 2. After receiving U’s message, CT forwards {IDU, ‘reg’, dfU, dsU} to LSSNS.

Step 3. When LSSNS receives the registration query, LSSNS keeps a record as {IDU, dfU, dsU} in its local subscriber database. Then, LSSNS sends a reply {IDU, ‘ok’} to CT.

Step 4. Finally, CT forwards LSSNS’s message {IDU, ‘ok’} to U in order to indicate registration success. Consequently, secure communication link is established between U and CT.

![Registration diagram](https://example.com/registration_diagram.png)
C. Location Updates

Different from MobiShare, we no more employ the dummy techniques in LBS or store the mapping entries in SNS. We merge SNS and LBS into one server, i.e., LSSNS and use dummy techniques to generate k-1 dummy locations, which are stored together with user U’s real location in LSSNS. The detailed steps are as follows (Fig. 3 shows the message transmission in this stage).

Step 1. When U intends to update his/her location, he/she uploads a record in the form of \((ID_U(x, y), \text{Sess}_U(x, y))\) to CT, where \((x, y)\) is U’s current location, and Sess\(_U(x, y)\) is the location encrypted with U’s session key, which is shared with all his/her trusted social friends.

Step 2. CT randomly generates k-1 dummy locations \((x_i', y_i')\), \(i=1,\ldots,k-1\), and k-1 random strings str\(_i\), \(i=1,\ldots,k\) to imitate the encrypted locations. To anonymize the location update from U, one real location and k-1 dummy locations are sent to LSSNS in the form of \((ID_U(x_i, y_i, \text{str}_i), \ldots, (x_k, y_k, \text{str}_k), \text{Tag}_U)\), where the real location \((x_i, y_i, \text{str}_i)\) is randomly put at the \(n\)th place (1 \(\leq n \leq k\)), and \(\text{Tag}_U\) is an encrypted string that indicates the index of the real location, encrypted by CT’s secret key Key\(_{CT}\).

Step 3. When LSSNS receives the update record, it stores the record in the database and sends a success message \((ID_U, \text{‘ok’})\) to CT.

Step 4. After receiving the message from LSSNS, CT directly forwards the message to U.

In LSSNS, the database consists of a number of tables, and each table represents a geographic region. The updates of locations within a region are kept in the corresponding table, where the user’s ID is the primary key. The purpose of storage measure is aimed at improving search efficiency and reducing computation overhead. For example, given one location, to find the strangers within a range, instead of searching all the stored location records, LSSNS only needs to check the tables of the regions that overlap the queried circular area. It is remarkable that the entries in the database expire after a certain period of time.

D. Querying Friends’ Locations

When user U wants to query the nearby friends’ locations, there are four steps as follows (Fig. 4 shows the message transmission in this stage).

Step 1. When U intends to query his/her friends’ locations within a certain range, he/she can submit a query \((ID_U, \text{‘f’, qf}_U)\) to CT, where qf\(_U\) is U’s specified distance, within which he/she wants to find his/her friends.

Step 2. After receiving U’s query, CT appends a sequence number seq, which are encrypted by LSSNS’s secret key Key\(_{LS}\), and forwards the query \((ID_U, \text{‘f’, qf}_U, \text{Key}_{LS}(\text{seq}))\) to LSSNS, where seq is used to resist the replay attack and tampering attack and Key\(_{LS}\) is LSSNS’s secret key, shared with the CT.

Step 3. Upon receiving the query \((ID_U, \text{‘f’, qf}_U, \text{Key}_{LS}(\text{seq}))\), LSSNS finds U’s friends set FS consisting of all the friend identifiers ID, with \((ID, ID') \in G.E.\). For each entry \((ID, df, ds, (x_1, y_1, \text{str}_1), (x_2, y_2, \text{str}_2), \ldots, (x_n, y_n, \text{str}_n), \text{Tag})\) stored in the database, LSSNS checks whether \(dist((x_i, y_i), (x_0, y_0)) \leq \min(qf_U, df)_{\text{seq}}\), \(i=1,\ldots,k\), \(t=1,\ldots,k\), \(k=1,\ldots,\text{seq}\), \(x_0, y_0\) are the corresponding k locations of U. For all \(F \in FS\), if one or more \((x_0, y_0)\), \(i=1,\ldots,k\), satisfy the distance requirement, the corresponding record in the form of \((ID_U, \text{str}_i, \text{Tag}_U)\) inserts into the result set. For example, if A \(\in FS\) and \((x_{iA}, y_{iA}), (x_{iA}, y_{iA})\) and \((x_{iA}, y_{iA})\) meet the above restrictions, \((ID_A, (2, str_1), (3, str_2), (k, str_3), \text{Tag}_A)\) is recorded. And the result set is divided into k subsets F\(_k\), \(i=1,\ldots,k\), according to the corresponding center point \((x_{iA}, y_{iA})\), \(t=1,\ldots,k\). Finally, LSSNS replies \(\text{Key}_{LS}(\{F_{qf}\}_{qf=1,\ldots,k}, \text{Tag}_U, \text{seq})\) to CT.

Step 4. Upon receiving \(\text{Key}_{LS}(\{F_{qf}\}_{qf=1,\ldots,k}, \text{Tag}_U, \text{seq})\), CT decrypts the reply with Key\(_{LS}\) and checks whether seq corresponds to the sequence number that it previously sent, and finds the real center point (U’s real location) with the real location index \(r = \text{Decrypt}_{Key_{CT}}(\text{Tag}_U)\), where Decrypt\(_{Key_{CT}}(.)\) is a decryption function using the secret key Key\(_{CT}\). According to the real center point, CT chooses F\(_q\), where \(r\) is the real location index and discards the reminding set F\(_q\), \(i \neq r\). For each item in the set F\(_q\), say \((ID_U, (2, str_2), (3, str_3), (k, str_k), \text{Tag}_U)\), if Decrypt\(_{Key_{CT}}(\text{Tag}_U)\) indicates that the real location index is 3, then str\(_3\) is Sess\(_L(x_3, y_3)\) and \((ID_U, \text{Sess}(x_3, y_3))\) is added to the final result Ans\(_F\). If the real location index is not 2, 3 or k, CT continues to handle the next record in F\(_q\). Finally Ans\(_F\) is a set in the form of \(\{ID_U, \text{Sess}(x_i, y_i)\}_{i=1,\ldots,k} \cdot k' \leq k\), and CT sends \((ID_U, \text{Ans}_F)\) to U.

When U receives \((ID_U, \text{Ans}_F)\) and decrypts the encrypted
Step 1. User U submits a strangers’ locations query ($qs_U$) to CT, where $qs_U$ is U’s specified distance, within which he/she wants to find the nearby strangers.

Step 2. Upon receiving $qs_U$, CT decrypts the message with DecryptKeyCT and forwards the query ($qs_U$, $Tag_A$) to LSSNS. LSSNS decrypts the message with KeyLS and checks whether $Tag_A$ corresponds to the sequence number it previously sent and finds the real center point (U’s real location) by decrypting $Tag_U$. According to the real center point, CT chooses the real subset $S_q$ and discards the reminding set $S_q$, $i \neq r$. For each item in the set $S_q$, say $(ID_A, (x_{A1}, y_{A1}), (x_{A2}, y_{A2}), (x_{A3}, y_{A3}), Tag_A)$, if DecryptKeyCT ($Tag_A$) indicates that the real location index is 3, $(ID_A, (x_{A3}, y_{A3}))$ is added to the final result $Ans_S$ and if the real location index is not 2, 3 or $k$, CT continues to handle the next record in $S_q$. Finally, $Ans_S$ is a set in the form of $(ID_A, (x, y), Tag_A)$ where $(x_{A1}, y_{A1})$ are the corresponding k locations of U.

When U receives $(ID_U, Ans_S)$, U knows both the identities and the locations of the nearby strangers who are willing to share their location information.

### E. Querying Strangers’ Locations

The strangers’ locations query performs in a similar way with the friends’ locations query, but does not require LSSNS to find friends and the scope of the candidates expands to all the nearby users. There are four steps in this stage as follows (Fig. 5 shows the message transmission in this stage).

Step 1. User U submits a strangers’ locations query ($ID_U, ‘s’, qs_U$) to CT, where $qs_U$ is U’s specified distance, within which he/she wants to find the nearby strangers.

Step 2. After receiving U’s query, CT appends a sequence number, which are encrypted by LSSNS’s secret key, and forwards the query ($ID_U, ‘s’, qs_U, KeyLS(seq)$) to LSSNS.

Step 3. Upon receiving the query ($ID_U, ‘s’, qs_U, KeyLS(seq)$), for each entry $(ID, df, ds, (x_1, y_1, str_1), (x_2, y_2, str_2), \ldots, (x_n, y_n, str_n), Tag)$ stored in the database, check whether $\text{dist}(x, y, (x_{10}, y_{10})) \leq \text{min}(qs_U, ds_i)$, $i = 1, \ldots, k$, $t = 1, \ldots, k$, where $(x_{10}, y_{10})$ are the corresponding k locations of U. For each user in the database, if one or more $(x, y)$, $i = 1, \ldots, k$, satisfy the distance requirement, the corresponding record in the form of $(ID, (x, y), Tag)$ inserts into the result set. For example, for user A, $(x_{A2}, y_{A2}), (x_{A3}, y_{A3})$ and $(x_{A4}, y_{A4})$ meet the above restrictions, $(ID_A, 2, (x_{A2}, y_{A2}), 3, (x_{A3}, y_{A3}), k, (x_{A4}, y_{A4}), Tag_A)$ are recorded. And the result set is divided into k subsets $S_{qi}$, $i = 1, \ldots, k$, according to the corresponding center point $(x_{10}, y_{10})$, $t = 1, \ldots, k$. Finally, LSSNS replies $Key_{LS}({S_{qi}}) i = 1, \ldots, k, Tag_U$, seq) to CT.

Step 4. Upon receiving $Key_{LS}({S_{qi}}) i = 1, \ldots, k, Tag_U$, seq), CT decrypts the message with $Key_{LS}$ and checks whether seq corresponds to the sequence number it previously sent and finds the real center point (U’s real location) by decrypting $Tag_U$. According to the real center point, CT chooses the real subset $S_q$ and discards the reminding set $S_q$, $i \neq r$. For each item in the set $S_q$, say $(ID_A, 2, (x_{A2}, y_{A2}), 3, (x_{A3}, y_{A3}), k, (x_{A4}, y_{A4}), Tag_A)$, if DecryptKeyCT ($Tag_A$) indicates that the real location index is 3, $(ID_A, (x_{A3}, y_{A3}))$ is added to the final result $Ans_S$ and if the real location index is not 2, 3 or $k$, CT continues to handle the next record in $S_q$. Finally, $Ans_S$ is a set in the form of $(ID_A, (x, y), Tag_A)$ where $(x_{A1}, y_{A1})$ are the corresponding k locations of U.

When U receives $(ID_U, Ans_S)$, U knows both the identities and the locations of the nearby strangers who are willing to share their location information.

### IV. SECURITY ANALYSIS

In our system, we assume that CT is a trusted entity and LSSNS is ‘honest-but-curious’. In the other words, this work does not concern social-location privacy rising by CT and LSSNS is supposed to follow the agreement mechanism, but it attempts to find out as much privacy information as possible. Moreover, LSSNS may collude with the arbitrary malicious users and intend to obtain the location of a target user. Through
analyzing the following two main threats, we can prove that our system is secure.

Threat 1: LSSNS intends to obtain the target user’s location. Although all the users’ real locations are stored in plain at LSSNS, K-anonymity technique is used so that LSSNS has at most the probability of 1/k to guess the user’s real location. Though LSSNS deals with location update and query, LSSNS cannot distinguish the users’ real locations because of K-anonymity technique. If LSSNS wants to get the user’s real location, it must decrypt the encrypted tag, which is encrypted by CT. That is to say, as long as the encryption scheme using by CT is safe enough, our system is secure.

Threat 2: Malicious users, even colliding with LSSNS, want to obtain the target user’s location. User B can fake as user A to query A’s friends location, and CT sends back A’s result set. B can still not obtain A’s friends’ locations because A’s friends’ locations in the result set are encrypted by the session key, which is only sharing with A’s friends.

Additionally, each transmission message appends a sequence number, which is effectively withstanding the replay attack and the tampering attack.

V. RELATED WORK

With the rise of OSNs, location sharing becomes an increasingly significant LBS, especially in mOSN. Simultaneously, the privacy issue caused by the location-sharing has developed into a devil of a tricky problem. In order to address the location privacy issue risen by location-sharing, many researches about information privacy [9] and location privacy protection [10], [11] have been done. There are a large number of researches focusing on preventing a location server from learning users’ location when the users access the LBSs with their location information, e.g. the K-anonymity [12]-[14], the mix zones [15], [16], the pseudonym methods [17], [18], the m-unobservability [19], [20], and the location anonymity [21]-[23]. Except the methods above, in order to defend against various inference attacks, [24] present a systematic solution basing on differential privacy to preserve location privacy. In addition, [25]-[27] proposed privacy context obfuscation to obscure location information based on parameters, such as data requester, time of day, and so on.

In recent years, with the cell phones and tablets being exploding, most devices are considerably smart and capable of determining their locations through GPS or cellular geolocation. As a result, with the rapid fusion of OSNs with mobile computing, a new paradigm called mOSNs has emerged. A lot of works in mOSNs have been carried out. In 2007, SmokeScreen [28] provided a flexible and power-efficient mechanism to allow safely sharing location between both trusted social friends and untrusted strangers. Later, considering flexible privacy-preserving location sharing in mOSNs, [1] introduced MobiShare, which is adaptable to support a variety of location-based applications, in that it enables location-sharing between both trusted social relations and untrusted strangers. In MobiShare, users’ individual profiles and location information are separately stored on SNS and LBS to secure the users’ location privacy. After MobiShare [1], many improved systems were subsequently proposed, such as [2]-[5]. N-MobiShare [2], [3] assumes that CT should not be treated as a core component of the system, and instead of CT, SNS is used to forward users’ location update requests to the Location-Based Server without knowing anything about the location information. Meanwhile, N-MobiShare employs Identity- Based Broadcast Encryption (IBBE) to realize sharing key off-line to all users’ friends. Shortly afterwards, Li et al. [4] observed that users’ real fake identities would be potentially leaked to location service provider in MobiShare and then they proposed a security improved mechanism, i.e. MobiShare+, which applies dummy queries and the private set intersection protocol to prevent SNS and LBS from learning individual information from each other. In order to improve transmission efficiency, B-MobiShare [5] uses Bloom Filter, which can filter invalid information with masking the sensitive data, to replace private set intersection protocol in MobiShare+. In 2015, aiming at achieving enhanced privacy against the insider attack launched by the service providers in mOSNs, Li et al. [29] introduced a new architecture with multiple location servers and proposed a secure solution supporting sharing location between friends and strangers in location-based applications.

All the above mechanisms have to depend on the third-party location server, which greatly increases the risks of privacy leakage and the economic burden. Different from the previous systems [1-5], considering the popular client/server structure, our system amalgamates SNS and LBS into one single server, i.e. LSSNS, which has the same protection effect on the users’ privacy and the less resource consumption.

VI. CONCLUSION

In this paper, taking into account the current popular client/server architecture of location-based applications in mOSNs, we present CSLocShare, a system that provides flexible privacy-preserving location sharing with client/server architecture in mOSNs. CSLocShare does not depend on the third-party server. It achieves a higher efficiency in communication and has the same protection effect on the users’ privacy as the previous systems. CSLocShare can be applied more widely in mOSNs.

ACKNOWLEDGMENTS

This work is supported by the NSFC projects (61379041, 61402255, 61371078), the National High-tech R&D Program of China (2015AA016102), the Guangdong Natural Science Foundation (No.2015A030310492), and the R&D Program of Shenzhen (JCYJ20150630170146831, JCYJ20160531174259309, JCYJ20160301152145171, ZDSYS20140509172959989, JSGG20150512162853495, Shengfagai [2015] 986).

REFERENCES