A Search Efficient Privacy-Preserving Location-Sharing Scheme in Mobile Online Social Networks

Guangcan Yang 1,†,‡, Shoushan Luo 1,‡, Yang Xin 1,*,‡, Hongliang Zhu 1,‡, Jingkai Wang 2,‡, Mingzhen Li 1,‡ and Yunfeng Wang 1,‡

1 National Engineering Laboratory for Disaster Backup and Recovery, Information Security Center, School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China; yangguangcan@bupt.edu.cn (G.Y.); buptlou@263.net (S.L.); zhuhongliang@bupt.edu.cn (H.Z.); mzli@bupt.edu.cn (M.L.); wangyunfeng@bupt.edu.cn (Y.W.)
2 School of Mathematical Sciences, Beihang University, Beijing 100083, China; wangjingkai@buaa.edu.cn
* Correspondence: yangxin@bupt.edu.cn; Tel.: +86-1371-856-3988
† Current address: National Engineering Laboratory for Disaster Backup and Recovery, School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China
‡ These authors contributed equally to this work.

Received: 9 October 2020; Accepted: 24 November 2020; Published: 25 November 2020

Abstract: With the advent of intelligent handheld devices, location sharing becomes one of the most popular services in mobile online social networks (mOSNs). In location-sharing services, users can enjoy a better social experience by updating their real-time location information. However, the leakage of private information may hinder the further development of location-sharing services. Although many solutions have been proposed to protect users’ privacy, the privacy-utility trade-offs must be considered. Therefore, we propose a new scheme called search efficient privacy-preserving location-sharing (SELS) system. In our scheme, we create a new approach named associated grids to improve the efficiency of location-sharing systems while maintaining users’ privacy. In addition, by setting the user-defined access control policy proposed in our scheme, users’ flexible privacy-preserving requirements can be satisfied. Detailed complexity and security analysis show that the proposed scheme is a practical and efficient privacy-preserving solution. Extensive simulations are performed to validate the effectiveness and performance of our scheme.

Keywords: mobile online social networks; privacy-preserving; location sharing; access control policy

1. Introduction

The rapid advance of mobile communication technologies and modern smart mobiles is changing the way people socialize [1,2]. Fresh social paradigms named mobile online social networks (mOSNs) have become ubiquitous in recent years. In addition to providing regular services such as sharing pictures, status, and moods on traditional web-based social networks, mOSNs also supply new services such as searching Points of Interests (POIs), querying for nearby friends and so on, which offer users much convenience.

Location-based service (LBS), one of the most important components in mOSNs, plays an increasingly significant role in daily life [3,4]. With the help of network access service and positioning techniques (e.g., Global Positioning System (GPS) and cellular tower geolocation service) of mobile devices, many applications of LBS have been proposed, such as location-based mobile service recommendations and check-in games like Foursquare. LBS promotes the change from traditional social networks to mobile social networks and makes real-time location-sharing services a reality.
Due to the convenience brought by real-time location-sharing services, users can enjoy a better experience by updating their real-time location information. However, some security issues, especially location privacy, have become the focus of users. It is important to prevent personal location information from being obtained by malicious attackers since users’ geographical locations are a very sensitive factor that is related to the spatial and contextual information about users. For example, users’ job and housing locations may be inferred from public records of bicycle renting stations [5]. By collecting and analyzing the social and location-related information, adversaries can infer users’ crucial privacy information such as users’ preferences, faiths, and physical conditions. Some criminals may even launch personal tracking attacks after grasping users’ privacy information. Without the guarantee of privacy, users may not be willing to share their location information. Therefore, how to protect users’ privacy, especially in real-time location-sharing services, is one of the main challenges at present.

Fortunately, some solutions have been put forward on this issue. Wei et al. [6] proposed a location-sharing system, named MobiShare, to protect users’ location privacy for friends’ and strangers’ queries in location-sharing applications. Then, to prevent a third-party such as the location server from learning users’ sensitive information, Li et al. [7] proposed a series of privacy interaction protocols to reduce security risks. Subsequently, a solution, called N-Mobishare, was introduced by Liu et al. [8] to preserve users’ location and social information via the social network server and the location server respectively. Afterwards, Shen et al. [9] put forward a scheme named B-Mobishare that used Bloom Filter to improve the efficiency of privacy interaction protocols proposed in [7]. To prevent the disclosure of users’ privacy in social networks and improve the flexibility of privacy protection, Sun et al. [10] designed a user-defined location-sharing system. To overcome the drawback of disclosing users’ social network privacy and prevent the location server from learning users’ complete social network relationships, Li et al. [11] proposed a scheme which used multiple location servers (MLS) to protect users’ social network privacy. Further, to decrease communication overhead, Xiao et al. [12] designed a centralized architecture to address the privacy issues in location-sharing systems.

However, while protecting users’ privacy, the previous solutions seldom consider the problems of system efficiency and the flexibility of users’ privacy-preserving requirements. In location-sharing systems, a user can request a location-sharing service for querying friends’ locations. However, in the previous work, the locations (no matter real coordinates or dummy coordinates used for protecting location privacy) of all of the user’s friends must be sent to the location server for distance calculation and comparison regardless of the user’s query distance. In fact, some locations may be far beyond the query distance of the user, so the distance calculation is not required. In addition, most of the previous works only focus on the basic security requirements in location-sharing systems such as protecting users’ identities and locations. However, users may have privacy-preserving requirements for some sensitive areas or specific geofences such as workplace. Therefore, how to design user-defined access control policies to meet users’ flexible privacy-preserving requirements is also an urgent issue.

In this paper, we propose a search efficient privacy-preserving location-sharing scheme (i.e., SELS) to solve the issues mentioned above. The main contributions of this paper are as follows.

1. We create a new approach named associated grids to improve the efficiency of location-sharing systems. Based on the grid structure and a user’s query distance, we can find out the smallest set of grid cells (i.e., the associated grids) that cover the user’s query distance. By using the approach, our scheme can filter out the locations of the user’s friends that are not in the associated grids, which can reduce the burden of distance calculation and comparison on the location server.

2. We propose a new user-defined access control policy to meet users’ flexible privacy-preserving requirements. For some specific geofences or sensitive areas that users do not want to share with their friends, users can use the access control policy to prevent the leakage of privacy in social networks. Compared with paper [10], our solution can provide a more flexible
privacy-preserving way, rather than simply preventing users from sharing locations with their friends.

(3) Extensive simulations are implemented to explore the relationship between different solutions and parameters, as well as the performance of our scheme (i.e., SELS).

The remainder of this paper is organized as follows. After discussing the related work in Section 2, the notations and techniques are introduced in Section 3. Subsequently, we show our system architecture and security model in Section 4. Then, the detailed scheme description and discussion about security and efficiency are given in Sections 5 and 6, respectively. Finally, we show the experiment results and our conclusions in Sections 7 and 8.

2. Related Work

With the increasing prevalence of mobile devices, especially smartphones, mOSNs have gone through rapid development [13,14]. Location-sharing, as an increasingly important service in mOSNs, brings people great convenience. However, the privacy issues caused by location-sharing also become an urgent problem [15]. To address this issue, many privacy-preserving methods such as k-anonymity [16], dummy [17], and spatial cloaking [18] have been widely adopted. The security and system efficiency are the important two aspects in location-sharing systems. Therefore, the current location-sharing solutions are introduced based on the two aspects mentioned above.

Many solutions have been proposed to protect users’ sensitive information such as identities and locations. For example, Li et al. [7] pointed out that users’ real identities may be leaked to the location server during location-sharing services. Thus, they proposed a security mechanism named MobiShare+ which employed dummy queries to protect identity privacy. Son et al. [19] used a broadcast form to query the information of friends nearby and introduced a new cryptography primitive called the functional pseudonym to protect users’ identity information. To overcome the privacy threat caused by users’ co-location information, Olteanu et al. [20] proposed a game-theoretic framework to explore the relationship between users’ behaviors and locations. Lin et al. [21] stated that users may provide inaccurate positioning data in location-sharing systems. Hence, they designed an attack method to reveal the shortcomings of protecting location privacy in existing location-sharing mechanisms. Recently, to prevent the disclosure of users’ identities caused by users’ threshold distances, Xu et al. [22] have proposed a secure distance comparison protocol by the paillier encryption. However, the above solutions mainly focus on the basic security requirements in location-sharing systems such as protecting users’ identity privacy and location privacy, but can not meet user-defined privacy-preserving requirements. To resist attacks from friends, Sun et al. [10] created a user-defined access control policy which can allow a user to determine whether to share a location with part of his/her friends. However, the above access control policy is aimed at some trust-less friends but not some sensitive areas. If the user has a privacy-preserving requirement for some geofences such as the workplace, the user has to set the access control policy to all his/her friends each time when the user is in these geofences. Similarly, when the user moves to other places and wants to use location-sharing services, the user has to reset the access control policy. Therefore, in terms of the privacy-preserving requirements for sensitive areas, the above access control policy is not flexible enough.

While maintaining users’ privacy, some schemes are proposed to improve the efficiency of location-sharing systems. For example, the location update database is adopted in paper [6] to improve search efficiency and reduce the time of finding users’ locations. However, the location update database brings a great storage burden. In order to increase the transmission efficiency, Shen et al. [9] proposed a scheme named B-Mobishare and used Bloom Filter to filter the sensitive data in the transmission between the social network server and the location server. However, B-Mobishare suffers a high time cost and computational overhead. To prevent the location server from getting users’ complete social network relationships, Li et al. [11] designed a new architecture with multiple location servers, which also can increase the computing power of distance calculation and comparison. Unfortunately, multiple location servers mean an increase in hardware costs. To enhance the efficiency of data
transmission, Xiao et al. [12] proposed a centralized location-sharing system that integrates the social network server and the location server into one server. However, the increase of system efficiency is caused by the change of system architecture. Overall, the above schemes can not filter out some locations in advance based on an issuing user’s query distance. For example, when a user submits a query with a certain query distance and wants to request a location-sharing service with nearby friends, the social network server in [6,11,12] first finds out the locations (no matter real coordinates or dummy coordinates) of the user’ friends, and then directly sends these locations to the location server even though some locations are far beyond the query distance of the user. Thus, their solutions affect the system efficiency since the location server in their schemes has to retrieve the locations of all of the user’s friends rather than the filtered locations based on the user’s query distance.

3. Preliminaries

In this section, we explain the basic concepts used in this work. Key notations used in this paper are summarized in Table 1.

| Notation | Description |
|----------|-------------|
| ID       | A user’s identifier. |
| PID      | A user’s pseudo-identifier (pseudo-ID). |
| GID      | A grid cell’s identifier. |
| GFID     | A grid cell’s fake-identifier (fake-ID). |
| BID      | A base station’s identifier. |
| (x, y)   | A location coordinate. |
| df       | Threshold distance for friends. |
| ds       | Threshold distance for strangers. |
| dis(i, j)| The distance comparison function to compute Euclidean distance from i to j. |
| l        | A user’s query distance. |
| ts       | A time-stamp. |
| SecK<sub>B&L</sub> | Location server’s secret key, shared with base stations. |
| SecK<sub>B&S</sub> | Social network server’s secret key, shared with base stations. |
| SecS<sub>U</sub> | A user’s session key, shared with all his friends. |
| seq      | A sequence number, generated by base stations. |

3.1. Grid Structure

Before users request location-sharing services, the location-sharing system usually specifies a query area (e.g., a city or an administrative district). Therefore, the grid structure [23] can be used to divide a query area into a certain number of grid cells. By using the method of grid structure, a query area can be designed to a uniform structure. The uniform structure can be a square or a rectangle covering the entire query area, and consists of the grid cells with equal size. The size of each grid cell can be defined as \( S \times S \). Suppose that the lower-left and upper-right coordinate of a query area are denoted as \((x_a, y_a)\) and \((x_b, y_b)\), respectively, then the query area can be divided into \( n_x \times n_y \) grid cells with equal size and represented by Equation (1):

\[
\text{Query area} \leftarrow ((x_a, y_a), (x_b, y_b), n_x, n_y). \tag{1}
\]

To unify the grid structure, the lower-left and upper-right coordinates can be converted by Equation (2):

\[
\begin{align*}
x_c &= x_a - x_a; \\
y_c &= y_a - y_a; \\
x_d &= x_b - x_a; \\
y_d &= y_b - y_a;
\end{align*}
\]
Then, \( n_x = (x_d - x_c)/S \) and \( n_y = (y_d - y_c)/S \) can be calculated.

In the uniform grid structure, each grid cell can be represented by a unique identifier. Herein, the identifier \( GID \) of each grid cell is represented by \((r_i, c_i)\), where \( r_i \) denotes the row identifier of Y-axis and \( c_i \) denotes the column identifier of X-axis. Figure 1a shows an example of determining the identifier of grid cell \( GID_a \) that base station \( B \) locates in. The identifier can be calculated by Equation (3) and the identifier of \( GID_a \) can be identified as \((5, 5)\).

\[
\begin{align*}
  r_b &= \frac{y_b - y_a}{(y_d - y_c)/n_y} \\
  c_b &= \frac{x_b - x_a}{(x_d - x_c)/n_x}
\end{align*}
\]  

(3)

(a) Determination of grid cell identifier

(b) Associated grids

**Figure 1.** Grid structure and associated grids.

### 3.2. Associated Grids

Based on the grid structure and an issuing user’s query distance, we can find out a certain number of grid cells that cover the issuing user’s query distance. Thus, associated grids are the smallest set of grid cells that cover the query distance of an issuing user. Suppose that there is a user who wants to search for nearby users (e.g., friends or strangers) and submits a request with the query distance \( l \) through base station \( B \). If base station \( B \) locates in grid cell \( GID_a \), then we can lock the smallest set of grid cells that need to be searched based on the query distance \( l \). The specific steps are as follows.

1. Based on the grid cell which base station \( B \) locates in, the four coordinates (i.e., upper-left, lower-left, upper-right, and lower-right) of \( GID_a \) can be found.
2. Four circles can be formed by taking the user’s query distance \( l \) as the radius and four coordinates (i.e., upper-left, lower-left, upper-right, and lower-right) of \( GID_a \) as the center.
3. The grid cells that cover the above four circles can be figured out. Then these grid cells are named the associated grids of the user. Figure 1b shows the associated grids (i.e., green grid cells) when the user’s query distance is 2 km.

### 3.3. RSA Signature Scheme

RSA digital signature, a mature and popular signature scheme, has been used in many studies. The key point of RSA digital signature is to use the characteristics of asymmetric encryption; that is, the private key is used to encrypt a message (i.e., forming signature) and the public key is used to decrypt the encrypted message (i.e., verifying signature). There are three algorithms in a RSA digital signature scheme, i.e., \( \text{KeyGen} \), \( \text{Sig} \), and \( \text{Ver} \). The description of algorithms is as follows.
(1) **KeyGen** is a key pair generation algorithm. Firstly, select two large prime numbers \( p \) and \( q \). Then, compute \( N = p \times q \) and choose \( e \) so as to \( 1 < e < \lambda(N) \) and \( \gcd(e, \lambda(N)) = 1 \), where \( \lambda(N) = (p-1)(q-1) \) and \( \gcd \) is the greatest common divisor. Finally, compute \( d = e^{-1}(mod(\lambda(N))) \) and then \((N, e)\) is defined as the public key and \((N, d)\) is defined as the private key.

(2) **Sig** is a signing algorithm. Suppose \( M \) is the message to be signed, then the ciphertext \( C = M^e (mod N) \) is the signature that will be sent and used for verification.

(3) **Ver** is a verification algorithm. Given a signature \( C \), if the decrypted \( M' = C^d (mod N) \) is the same as the original message \( M \), then the \( C \) can be accepted as a valid signature.

4. System Architecture and Security Model

In this section, we first give the system architecture and the security model. Then, security goals for our scheme are identified and listed.

4.1. Architecture

The architecture of our location-sharing system consists of four entities: users, base stations, the social network server, and the location server. Figure 2a,b shows the system architecture of our scheme and the workflow of location query, respectively.

![System Architecture](image)

**Figure 2.** System architecture and workflow of location query.

(1) The entity of users, with mobile terminals, can send a request for location-sharing with nearby friends or strangers. Within a specified query distance, an issuing user can know nearby friends or strangers’ locations if some pre-set access control policies are satisfied. Users can communicate with base stations and online social network server directly via their mobile devices. One user only has a unique identity in the online social network server.

(2) Base station (B) is an entity that has a certain computation and acts as a connecting thread. Users can communicate with servers through base stations.

(3) The social network server (SNS) is an entity that stores users’ social network topology, in the form of social network graph \( G = (V, E) \). \( V \) is a set of vertices which represent users’ identities and \( E \) is a set of edges that indicate the friends’ relationships. If two vertices are connected by an edge, the two corresponding users have a social relationship (i.e., they are friends). In addition, the social network server is also responsible for user registration and provides social network services based on the requirements of users’ queries.
The location server (LS) is an entity that stores the location information of users and dummies. In addition, it also takes charge of providing location-related services such as computing and comparing the Euclidean distance of two locations based on their coordinates.

4.2. Security Model

In our security model, the base station (B) is assumed to be a trusted entity, which means it knows the transmitted information but does not leak any sensitive information. Herein, we mainly focus on the threats caused by users, SNS, and LS.

Users are assumed to be dishonest and try to get unauthorized information beyond the scope of their access privileges as much as possible. Thus, by querying other users’ locations in location-sharing systems, they want to infer some sensitive information (e.g., locations) of their friends or strangers such as home address.

The social network server (SNS) is supposed to be “honest-but-curious”, which means SNS will execute the proposed protocol honestly but try to get sensitive information such as users’ actual locations from the interactive communications. Previous researches such as [6,12] also use base stations in their system architectures. In addition, the researches above also assume that the social network server can not identify base stations by observing their IP addresses in the connections. However, if a system insider (e.g., an employee) colludes with SNS, the approximate region of users who send requests for location-sharing services can be figured out since the locations of base stations are fixed. Thus, with the help of insiders, we assume that SNS can infer the approximate region of users since users have to use base stations to submit location-sharing services.

The location server (LS) is also assumed to be “honest-but-curious”, which means LS will execute the proposed protocol honestly but try to get sensitive information such as users’ social network topology or identities.

However, the collusion between SNS and LS, which means they collude together to gain users’ sensitive information at the same time, is beyond our security assumption. In addition, information leakage and eavesdropping during transmission are also beyond the scope of our scheme.

4.3. Security Goals

According to the security model, the security goals are given as follows.

Users’ actual location information should be kept secret from SNS and unauthorized users. In addition, users’ location information also can not be revealed to friends or strangers who do not match their pre-set access control policies.

The social network server (SNS) should be prevented from obtaining users’ actual location information though it can collude with malicious insiders to get users’ approximate regions.

The location server (LS) is prevented from getting users’ real social network information, location information, and identity information.

5. Search Efficient Privacy-Preserving Location-Sharing (SELS) Scheme

To preserve users’ privacy and improve the efficiency of location-sharing systems, the proposed scheme utilizes the grid structure and anonymous technologies. The details of each step are given below.

Initialization. SNS assigns a unique identifier ID to each user and builds social network graphs $G = (V,E)$ for users. Then, SNS creates a mapping relationship set $\{ID, PID_{list} = (PID_1, PID_2, \ldots, PID_k), ind\}$ for each user. $PID_{list}$ is a pseudo-ID set generated by the pseudo-random function [24]. $ind$ is an index to indicate the real pseudo-ID in $PID_{list}$, where $1 \leq ind \leq k$ and $k$ is the number of pseudo-IDs. Finally, SNS shares the mapping relationship with B.

In addition, B builds a grid structure for the entire query area of the location-sharing system and creates a mapping relationship between $GID = \{GID_1, GID_2, \ldots, GID_n\}$ and $GFID = \{GFID_1, GFID_2, \ldots, GFID_n\}$, where $GID$ is the identifier set of grid cells and $GFID$ is the fake-ID
set generated by the cryptographic hash function SHA-1. Note that one grid cell identifier in $GID$ corresponds to one fake-ID in $GFID$. Afterwards, $B$ shares the mapping relationship with $SNS$.

Registration. Firstly, user $U$ sends a registration request in the form of $(ID, df, ds, ts, \text{Sig}(ID, ts))$ to $SNS$, where $ts$ is a time-stamp used to prevent replay attacks, $df$ and $ds$ represent $U$’s distance threshold in friends’ and strangers’ location query respectively, and $\text{Sig}(ID, ts)$ is a signature generated with $U$’s private key over the time-stamp $ts$. After receiving the registration request of $U$, $SNS$ uses $U$’s public key to verify the correctness of the signature. If the decrypted $ID$ is the same as $U$’s identifier assigned by $SNS$ in advance, $SNS$ will send a message $\text{OK}$ to $U$. Secondly, $SNS$ will send a notification in the form of $(ID, df, ds, reg)$ to $B$, where $reg$ is a notice which means this $ID$ is a registered one. After receiving the notification, $B$ replies a message ‘OK’ to $SNS$ and stores the corresponding information in the database. Figure 3 shows the message transmission about the registration.

![Figure 3. Registration.](image)

Update. The update process includes mapping relationship update and location update. Figure 4 shows the process of mapping relationship update. The detailed steps are as follows:

1. $B$ generates a new mapping relationship $\{GFID = GFID_1, GFID_2, \ldots, GFID_n\}$ between grid cells’ identifiers and their fake-IDs. Then, the new fake-ID set (i.e., $\{GFID = GFID_1, GFID_2, \ldots, GFID_n\}$) is encrypted with the secret key $SecK_{B&S}$ and sent to $SNS$.
2. $SNS$ uses the random number generator [25] to generate a new $ind$. Then, the new $ind$ and identifier $ID$ of each user are encrypted with the secret key $SecK_{B&S}$ and sent to $B$.
3. To prevent the users who are no longer friends from getting messages, $SNS$ updates the session key $SecS_U$ for each user based on social network graphs.

![Figure 4. Mapping relationship update.](image)

Figure 5 shows the process of location update. The detailed steps are as follows:

1. When user $U$ updates his/her location, $U$ needs to submit a message in the form of $(ID_U, (x, y), \text{SesS}_U(x, y), A/F)$ to $B$ (i.e., $(x, y)$ is the $U$’s current location and $\text{SesS}_U(x, y)$ is the encrypted location information with $\text{SesS}_U$). $A/F$ is the grid cell access control policy, which means the current grid cell is accessible/inaccessible (i.e., $U$ allows or forbids the current grid cell where $U$ locates can be searched by the location-sharing requests of his/her friends). When getting the message, $B$ will record the access control policy in the form of $(ID_U, GID, A/F)$ in the database. Note that $A$ is the default state and $F$ means only the current grid cell can not be searched by the location-sharing request of his/her friends (i.e., the current grid cell is a sensitive area that $U$ does not want his/her friends to know). However, if $U$ is no longer in this grid cell, $U$ can be searched by the location-sharing requests of his/her friends.
2. If $A$ is included in $U$’s submitted message, $B$ will send a message in the form of $SecK_{B&S}(ID_U, GFID)$ to $SNS$, where $GFID$ is the fake-ID of the grid cell where $B$ locates.
For example, if \(U\) offers update information through base station \(B\), \(B\) will send \(SecK_{B\&L}(ID_U, GFID_a)\) to \(SNS\). After receiving the update information, \(SNS\) will reply a message \(OK\) to \(B\) which then forwards \(OK\) to \(U\). If \(U\)’s submitted message includes \(F\), \(B\) will do nothing.

(3) \(B\) sends a message in the form of \((\text{PID}_1, x_1, y_1, \text{str}_1, \text{df}_1, \text{ds}_1), \ldots, (\text{PID}_k, x_k, y_k, \text{str}_k, \text{df}_k, \text{ds}_k)\) to \(LS\), where \(\text{PID}\) is the pseudo-ID set of \(U\) and \(\text{str}\) is a random string to imitate the encrypted location. Herein, suppose that \(\text{ind} = 1\) in the last mapping relationship update, then \((\text{PID}_1, x_1, y_1, \text{SecK}_{j1}(x_1, y_1), \text{df}_1, \text{ds}_1)\) represents the real information of \(U\), and \((\text{PID}_2, x_2, y_2, \text{str}_2, \text{df}_2, \text{ds}_2), \ldots, (\text{PID}_k, x_k, y_k, \text{str}_k, \text{df}_k, \text{ds}_k)\) indicates the dummies of \(U\). Then \(LS\) stores the information in its database and sends a response in the form of \('OK'\) to \(B\) which then forwards \('OK'\) to \(U\).

Figure 5. Location update.

Querying friends’ locations. Users can request a location query for searching friends’ locations. If user \(U\) wants to request a location sharing with nearby friends, \(U\) needs to submit a request for querying friends’ locations through a nearby base station. Note that the base station and the user are assumed to be in the same grid cell. Figure 6 shows the specific process of querying friends’ locations. The detailed steps are as follows:

1. \(U\) needs to submit a query in the form of \((ID_U, f, l)\) to \(B\), where \(f\) means this query is a request for querying friends’ locations and \(l\) indicates the specified query distance such as 2 km.

2. When receiving \(U\)’s query, \(B\) first figures out the associated grids of \(U\) (see Section 3.2) based on \(U\)’s query distance \(l\), and form a list \(GFID_{list}\) which contains the fake-ID set of the associated grids. The purpose of finding out the list \(GFID_{list}\) is to help \(SNS\) filter out \(U\)’s friends that are not in the associated grids. Then, \(B\) encrypts its identifier \(BID\) and \(seq\) with \(SecK_{B\&L}\) to form \(SecK_{B\&L}(BID, seq)\), where \(seq\) is a sequence number to resist the replay attack. Finally, \(B\) sends the message \((ID_U, f, l, SecK_{B\&L}(BID, seq), GFID_{list})\) to \(SNS\).

3. According to \(U\)’s social network graph and users’ location update information, \(SNS\) finds out the \(U\)’s friends who are in \(GFID_{list}\) (i.e., seeking out \(U\)’s friends whose locations are in \(GFID_{list}\)). Suppose that there are \(m\) users meeting the above condition, where \(m \leq n\) and \(n\) is the total number of \(U\)’s friends. Then \(SNS\) will collect these users’ \(PID_{list}\) to form a set \(\{\text{PID}_{ij}\}\), where \(1 \leq i \leq k\) and \(1 \leq j \leq m\). The purpose of forming \(\{\text{PID}_{ij}\}\) is to perturb the identity information of \(U\)’s friends. Finally, \(SNS\) sends a query in the form of \((\text{PID}_{ind}, l, \{\text{PID}_{ij}\}, \text{SecK}_{B\&L}(BID, seq))\) to \(LS\), where \(\text{PID}_{ind}\) represents the real pseudo-ID of \(U\).

4. After getting of the query from \(SNS\), \(LS\) runs the distance comparison function \(dis((x_{ind}, y_{ind}), (x_{v}, y_{v})) \leq min(l, df_v)\), where \(dis\) is a function to compute Euclidean distance and \((x_{ind}, y_{ind})\) is the coordinates of \(\text{PID}_{ind}\). \((x_{v}, y_{v})\) represents the corresponding coordinates of pseudo-IDs in \(\{\text{PID}_{ij}\}\), where \(1 \leq v \leq (k \times m)\). Suppose that there are \(w\) \(\text{PIDs}\) satisfying the
distance comparison function, then LS collects these \( w \) PIDs to form a set \( \{ PID_f \} \) and finds the corresponding \( str \), where \( 1 \leq f \leq w \) and \( \{ PID_f \} \subseteq \{ PID_{ij} \} \). Finally, LS replies a message in the form of \( \{ \{ PID_f, SecK_{B\&L}(str_f) \} \}_{f=1,...,w} \), \( SecK_{B\&L}(seq) \) to SNS.

5. Upon receiving \( \{ \{ PID_f, SecK_{B\&L}(str_f) \} \}_{f=1,...,w} \), SNS recovers the real identifiers from \( \{ PID_f \} \) to form a result set \( \{ ID_g \} \) (i.e., the friends’ real identifiers of \( U \)) based on the \( \text{inds} \) of \( U \)’s friends. The purpose of forming \( \{ ID_g \} \) is to filter out dummies of \( U \)’s friends. Suppose that there are \( t \) users’ IDs recovered from the result set, then SNS sends a message in the form of \( \{ \{ ID_g, SecK_{B\&L}(str_g) \} \}_{g=1,...,t} \), \( SecK_{B\&L}(seq) \) to \( B \). Note that if \( t = 0 \), SNS will send \( \emptyset \) to \( B \).

6. Upon the reception of the response from SNS, \( B \) first checks the sequence number \( seq \) and then decrypts \( \{ SecK_{B\&L}(str_g) \} \) to obtain \( \{ SecS_{U}(x_g, y_g) \} \). Finally, \( B \) replies the result \( res \) in the form of \( \{ \{ ID_g, SecS_{U}(x_g, y_g) \} \}_{g=1,...,t} \) to user \( U \).

7. After obtaining \( res \), \( U \) uses session key \( SecS_{U} \) to decrypt the corresponding coordinates of his/her friends.

**Figure 6. Friend query.**

Querying strangers’ locations. Users also can request a location query for searching strangers’ locations. The process of querying strangers’ locations is similar to that of friends. Figure 7 shows the specific process of querying strangers’ locations and detailed steps are as follows:

1. \( U \) needs to submit a query in the form of \( \{ ID_{U}, s, l \} \) to \( B \), where \( s \) means this query is a request for querying strangers’ locations and \( l \) indicates the specified query distance such as 2 km.

2. After receiving \( U \)’s query, \( B \) figures out the associated grids based on \( U \)’s query distance \( l \), and forms a list \( GFID_{list} \). Then, \( B \) sends the message \( \{ ID_{U}, s, l, SecK_{B\&L}(BID, seq), GFID_{list} \} \) to SNS.

3. Based on \( GFID_{list} \) and \( U \)’s social network graph, SNS first eliminates the users who have friendship with \( U \). Then, SNS will randomly collect a certain number of users’ \( PID_{list} \) to form a set \( \{ PID_{ij} \} \), where \( 1 \leq i \leq k, 1 \leq j \leq rand \), and \( rand \) is a number of randomly selected users. Finally, SNS sends a query in the form of \( \{ PID_{ind}, l, PID_{ij}, SecK_{B\&L}(BID, seq) \} \) to LS, where \( PID_{ind} \) represents the real pseudo-ID of \( U \).

4. When receiving the query from SNS, LS runs the distance comparison function \( dis((x_{ind}, y_{ind}),(x_{ij}, y_{ij})) \leq \min(l,v_{seq}) \). \( (x_{ij}, y_{ij}) \) indicates the corresponding coordinates of pseudo-IDs in \( \{ PID_{ij} \} \), where \( 1 \leq v \leq (k \times \text{rand}) \). Suppose that there are \( w \) PIDs satisfying the distance comparison function, then LS collects these \( w \) PIDs to form a set \( \{ PID_{i} \} \) and finds the corresponding \( (x, y) \), where \( 1 \leq s \leq w \) and \( \{ PID_{i} \} \subseteq \{ PID_{ij} \} \). Finally, LS replies a message in the form of \( \{ \{ PID_{i}, SecK_{B\&L}(x, y) \} \}_{s=1,...,w} \), \( SecK_{B\&L}(seq) \) to SNS.

5. Upon receiving \( \{ \{ PID_{i}, SecK_{B\&L}(x, y) \} \}_{s=1,...,w} \), SNS recovers the real identifiers from \( \{ PID_{i} \} \) to form a result set \( \{ ID_{g} \} \) (i.e., the users’ real identifiers) based on users’ \( ind \). Suppose that there are \( t \) users’ IDs recovered from the result set, then SNS sends \( \{ \{ ID_{g}, SecK_{B\&L}(x, y) \} \}_{g=1,...,t} \), \( SecK_{B\&L}(seq) \) to \( B \).

6. After getting the response from SNS, \( B \) first checks the sequence number \( seq \) and then decrypts \( \{ SecK_{B\&L}(x, y) \} \) to obtain \( \{ (x, y) \} \). Finally, \( B \) replies the result \( res \) in the form of \( \{ \{ ID_{g}, (x, y) \} \}_{g=1,...,t} \) to user \( U \).
6. Discussion

In this section, the proposed scheme will be analyzed in terms of efficiency and security.

6.1. Efficiency Analysis

In previous studies such as [6, 10–12], their schemes also adopted the method of pseudo-IDs to protect the privacy of users’ identities. When an issuing user $U$ sends a request for querying friends’ locations, the social network server (SNS) will send a certain number pseudo-IDs of the user’s friends to the location server (LS), and then LS will operate the distance comparison function for each pseudo-ID sent from SNS. Therefore, the number of pseudo-IDs sent from SNS not only can reflect the size of the communication traffic between SNS and LS, but also can be a measure for the computational cost of LS. Based on the times of operating the distance comparison function (i.e., $dis(i, j)$) in LS, we compare the time complexity among our scheme and previous solutions. The comparison results are given in Table 2. Note that the time complexity focuses on the process of querying friends’ locations. The time complexity of querying strangers’ locations is related to the number of strangers’ pseudo-IDs randomly selected by SNS, so we omit it here.

**Table 2. Compared with previous solutions.**

| Scheme         | Time Complexity |
|----------------|-----------------|
| MobiShare [6]  | $O(kf)$         |
| UDPLS [10]     | $O(kf_1)$       |
| MLS [11]       | $O(\sum w_i(S_i f_i))$ |
| CenLocShare [12]| $O(kf)$         |
| SELS (Our scheme) | $O(kf_2)$     |

In Table 2, $k$ is the number of pseudo-IDs for each user (i.e., $PID_{list}$), and $f$ is the total number of an issuing user $U$’s friends. In addition, MobiShare, MLS, and CenLocShare do not consider attacks from friends (i.e., not consider the disclosure of privacy among friends), which means SNS has to send all the pseudo-IDs of $U$’s friends to LS. However, the number of pseudo-IDs that SNS sends to LS is different in MobiShare, MLS, and CenLocShare. In MobiShare and CenLocShare, SNS directly sends all the pseudo-IDs of $U$’s friends to LS, so the number of pseudo-IDs sent from SNS is $kf$. In MLS, to prevent the location server side (i.e., LS) from getting the complete social network relationship of $U$, the social network server (SNS) first divides all the pseudo-IDs of $U$’s friends into a certain number of subsets and then sends these subsets to multiple location servers separately. Herein, according to scheme MLS, we can assume that the number of subsets is $w$, a subset is represented by $S_i$ and the number of pseudo-IDs in subset $S_i$ is $f_i$. Therefore, the number of pseudo-IDs sent from SNS can be represented by $\sum w_i(S_i f_i)$. In the schemes that do not consider attacks from friends, we can conclude that: (1) $kf \leq \sum w_i(S_i f_i)$; and (2) $kf = \sum w_i(S_i f_i)$ when $w = 1$ (i.e., when there is only one set which contains all the pseudo-IDs of $U$’s friends). From the above conclusions, it is easy to deduce that: (1) the time complexity of scheme MLS (i.e., $O(\sum w_i(S_i f_i))$) is usually greater than that of scheme MobiShare and CenLocShare (i.e., $O(kf)$); (2) in terms of the size of the communication traffic between SNS and LS, scheme MLS is usually bigger than MobiShare and CenLocShare; (3) in terms of the computational cost of LS, MLS is usually bigger than MobiShare and CenLocShare.

In Table 2, $f_1$ means the number of $U$’s friends who agree to share their locations with $U$ [10], and $f_2$ indicates the number of $U$’s friends who locate in $U$’s associated grids and agree to share their
locations with \( U \). Note that both \( f_1 \) and \( f_2 \) are usually not the total number of \( U \)'s friends. The reason is that user-defined privacy location-sharing system (UDPLS) proposed in paper [10] and our SELS both consider attacks from friends and design corresponding access control policy; that is, SNS only sends the pseudo-IDs of \( U \)'s friends that meet \( U \)'s privacy-preserving requirements to \( LS \). Therefore, if \( U \) uses the access control policy for preventing the disclosure of privacy among friends in UDPLS and our SELS, the number of pseudo-IDs sent from SNS in UDPLS and our SELS is bound to be less than that in schemes MobiShare, MLS, and CenLocShare. Through the above analysis, we can deduce that: (1) the time complexity of the schemes that do not consider attacks from friends is usually greater than that of the schemes that do; (2) in terms of the size of the communication traffic between SNS and \( LS \), the schemes that do not consider attacks from friends are usually bigger than the schemes that do; (3) in terms of the computational cost of \( LS \), the schemes that do not consider attacks from friends are usually bigger than the schemes that do.

From Table 2, we know that the number of pseudo-IDs sent from SNS in UDPLS is \( kf_1 \) and the number of pseudo-IDs sent from SNS in our SELS is \( kf_2 \). Therefore, we can compare \( f_1 \) and \( f_2 \) to know the number of pseudo-IDs sent from SNS by setting \( k \) as a constant. In UDPLS, \( f_1 \) means the number of \( U \)'s friends who agree to share their locations with \( U \), i.e., \( f_1 = f - f_5 \), where \( f \) represents the total number of \( U \)'s friends and \( f_5 \) represents the number of \( U \)'s friends who do not agree to share their locations with \( U \) in the entire query area of the location-sharing system. In our SELS, \( f_2 \) indicates the number of \( U \)'s friends who locate in \( U \)'s associated grids and agree to share their locations with \( U \), i.e., \( f_2 = f - f_4 - f_5 \), where \( f_4 \) represents the number of \( U \)'s friends who are not in \( U \)'s associated grids and \( f_5 \) means the number of \( U \)'s friends who do not agree to share their locations with \( U \) in \( U \)'s associated grids. With the above description, we can know that: (1) when \( U \)'s associated grids are the entire query area of the location-sharing system, \( f_4 = 0 \); and (2) when \( f_4 = 0 \) and the number of \( U \)'s friends who do not agree to share their locations with \( U \) is set to be the same constant in UDPLS and SELS (i.e., \( f_3 = f_5 \)), then \( f_1 = f_2 \). In general, \( U \)'s associated grids are part of the entire query area of the location-sharing system, and the possibility that all \( U \)'s friends are in \( U \)'s associated grids is very small. Thus, it can be concluded that: (1) the time complexity of UDPLS (i.e., \( O(kf_1) \)) is usually greater than that of our SELS (i.e., \( O(kf_2) \)); (2) in terms of the size of the communication traffic between SNS and \( LS \), UDPLS is usually bigger than SELS; (3) in terms of the computational cost of \( LS \), UDPLS is usually bigger than SELS.

Through the above analysis, we also can summarize that: (1) the time complexity of our scheme is lower than that of other solutions; (2) when \( k \) is set to be a constant, bigger \( f \) (i.e., the number of an issuing user’s friends) requires more resources of location-sharing systems (i.e., the size of the communication traffic between SNS and \( LS \) and the computational cost of \( LS \)).

### 6.2. Security Analysis

In our security model, \( B \) is a trusted entity while SNS and \( LS \) are “honest-but-curious”. In addition, SNS and \( LS \) are assumed not to be able to collude with each other to obtain users’ sensitive information. Thus, our scheme mainly focuses on the proposed security goals.

Access control. There are two kinds of access control policies in our scheme. One is users’ defined threshold distance access control, and the other is users’ defined grid cell access control. Since SNS and \( LS \) are assumed to be “honest-but-curious”, that is, the access control policies will be implemented honestly. Thus, users’ defined access control policies can be achieved.

Identity privacy. The identity privacy of users does not need to be considered on SNS since SNS has users’ identity information. Therefore, we only need to analyze whether \( LS \) can obtain a user’s real identity or not.

In the location update process, all the users’ pseudo-IDs are uploaded to \( LS \) and each user’s real identity has been anonymized by a pseudo-ID set (i.e., \( PID_{\text{list}} \)). Thus, the probability of getting a user’s real identity is \( \frac{1}{n} \), where \( n \) is the total number of users and \( k \) is the number of pseudo-IDs for each user. In the location query process, suppose that the number of pseudo-IDs that are sent to \( LS \)
is \( w \), the probability of identifying a user’s real identity is \( \frac{1}{w} \). Since \( n \) and \( w \) are usually a very large number, it is impossible for \( LS \) to get the real identity of a user. In addition, when a user performs the location update, the pseudo-ID which represents the user’s real identity randomly changes with the index \( ind \). Therefore, \( LS \) can not get users’ real identities.

Sensitive area privacy. Unlike the previous studies that do not consider the disclosure of privacy among friends (i.e., a curious user may infer the privacy information of his/her friends such as workplace by querying friends’ locations), and different from the access control policy proposed in paper [10] which directly shuts up the location-sharing service for friends, our scheme focuses on the privacy-preserving requirements for sensitive areas. In our scheme, users are assumed to be curious and want to infer other users’ privacy by requesting location-sharing services, such as home address and workplace. Suppose that there is a user \( U \) who does not want to share some specific geofences or locations with his/her friends in location-sharing systems. In the location update process, if \( U \)’s current location or geofence is a sensitive area, then \( U \) can use the grid cell access control policy proposed in our scheme to set the current grid cell as inaccessible. In this way, \( U \)’s friends can not get location information of \( U \)’s sensitive area by their location-sharing services. Therefore, \( U \)’s privacy-preserving requirements of sensitive areas can be satisfied.

Location privacy. In our scheme, both \( LS \) and \( SNS \) are assumed to be curious and want to obtain users’ locations. Thus, we need to analyze whether \( LS \) and \( SNS \) can get a user’s real location.

From the perspective of \( LS \), the probability of identifying a user’s location is the same as that of getting the user’s identity in the location update process, because each user’s pseudo-ID is associated with a corresponding location. In the location query process, \( LS \) can not still distinguish a users’ real location since \( LS \) can not get each user’s real identity which randomly changes with the index \( ind \). Therefore, \( LS \) can not get users’ locations.

From the perspective of \( SNS \), the coordinates replied from \( LS \) are protected by symmetric encryption scheme (i.e., \( SecK_{B&L} \)) in the location query process, so \( SNS \) has no chance to get the users’ locations unless \( SNS \) can decrypt the encrypted coordinates. In the location update process, \( SNS \) can get the fake-IDs of base stations. As mentioned in the security model (i.e., Section 4.2), we suppose that the malicious users may be system insiders who can collude with \( SNS \). Thus, with the help of these insiders, \( SNS \) can obtain users’ rough location range since base stations can locate the subscribed cell phones with an accuracy of 50 to 300 m. However, \( SNS \) can not get actual locations of users and can not always make sure of users’ rough location range since the mapping relationship between grid cells’ identifiers and their fake-IDs is different in each update cycle. Therefore, \( SNS \) can not get users’ actual locations.

Social Network Privacy. Since \( SNS \) manages users’ social network graphs, we do not consider social network privacy on \( SNS \), and only need to analyze whether there is an issue of disclosing social network privacy on \( LS \).

In the location query process, \( LS \) has nothing information about a user’s query type (i.e., querying friends’ locations or querying strangers’ locations). Therefore, suppose that the number of pseudo-IDs sent from \( SNS \) is \( m \), the probability of inferring the user’s social network privacy for \( LS \) can be computed as \( \frac{1}{m} \). In the location update process, all the users’ pseudo-IDs are uploaded to \( LS \) and the index \( ind \), which is used to indicate the real pseudo-ID, is randomly changed. Therefore, the probability of inferring the user’s social network privacy can be calculated as \( \frac{1}{mk} \), where \( n \) is the total number of users and \( k \) is the number of pseudo-IDs for each user. Since \( m \) and \( n \) are usually large numbers, it is impossible for \( LS \) to obtain the social network privacy. Therefore, users’ social network privacy can be well protected. In addition, since \( SNS \) sends a subset of an issuing user’s friends (i.e., part of the user’s friends) to \( LS \) each time [11], our scheme can also prevent \( LS \) from inferring the complete social network relationship of the user.
7. Simulation and Results

We have conducted extensive simulations to evaluate the performance of our scheme. In this section, we first describe the simulation environment and then give the simulation results and analysis.

7.1. Simulation Setup

The simulations are implemented by using JAVA programming language and conducted on a Windows machine with an Intel Core-i5 2.6 GHz, 16 GB RAM, and Microsoft Windows 7 OS. For pseudo-random function, we choose the hash function SHA-256 with an output size of 32 byte. Since the efficiency of querying strangers’ locations directly relates to the number of strangers randomly selected by SNS, we focus the research on the performance of querying friends’ locations. In general, the grid structure is generated by the system and it does not change frequently. In addition, for an issuing user \( U \), the friend threshold distance of \( U \) is also usually set as a constant since it is meaningless to adjust \( d_f \) frequently when his/her privacy is not threatened. Therefore, in the process of querying friends’ locations, we mainly focus on four parameters that directly relate to the performance of location-sharing systems. These four parameters are the number of users, the number of pseudo-IDs for each user, the number of friends for each user, and the query distance. The specific parameter settings of our simulations are shown in Table 3. In Table 3, \( n \) is the total number of users, and \( k \) is the number of pseudo-IDs for each user. \( GS \) represents the grid structure composed of \( 20 \times 20 \) grid cells and the size of each grid cell is \( 1 \text{ km} \times 1 \text{ km} \). \( d_f \) indicates the threshold distance for friends. \( l \) means the query distance and \( f \) is the number of friends for each user.

| Item | Attribute | Specification |
|------|-----------|---------------|
| \( n \) | Variate | 1000–2500 |
| \( k \) | Variate | 10–30 |
| \( GS \) | Constant | \( 20 \times 20, 1 \text{ km} \times 1 \text{ km} \) |
| \( d_f \) | Constant | 5 km |
| \( l \) | Variate | 1–5 km |
| \( f \) | Variate | 50–90 |

The performance metrics used in this paper are the query time and the effective users. The effective users are the number of users’ pseudo-IDs sent from SNS to LS, and the query time is the time cost on distance comparison function based on the number of the effective users. In addition, UDPLS allows an issuing user \( U \) to set the access control policy to resist attacks from trust-less friends, which leads to the result that a small number of \( U \)’s friends can not share locations with \( U \), so we set the proportion of \( U \)’s trust-less friends to be 1%. To compare the performance of our scheme with that of other schemes (i.e., MobiShare, UDPLS and CenLocShare), we set different parameter variables as follows:

(1) Under the condition of querying friends’ locations, ten locations are randomly selected as the initial locations of an issuing user. The total number of users \( n \) is set to be 1000 and the users are randomly distributed throughout the grid structure. The number of pseudo-IDs for each user \( k \) and the query distance \( l \) of the issuing user are set as 10 and 5 km, respectively. The number range of friends for each user \( f \) changes from 50 to 90. We execute 100 times and then calculate the average number of effective users and the average time cost under different \( f \). The above settings are named scenario-1, which is used to explore the effect caused by \( f \).

(2) Under the condition of querying friends’ locations, ten locations are randomly selected as the initial locations of an issuing user. The total number of users \( n \) is set to be 1000 and the users are randomly distributed throughout the grid structure. The number of pseudo-IDs for each user \( k \) and the number of friends for each user \( f \) are set as 10 and 50, respectively. The query distance \( l \) of the issuing user changes from 1 km to 5 km. We execute 100 times and then calculate...
the average number of effective users and the average time cost under different $l$. The above settings are named scenario-2, which is used to research the effect caused by $l$.

(3) Under the condition of querying friends’ locations, ten locations are randomly selected as the initial locations of an issuing user. The total number of users $n$ is set to be 1000 and the users are randomly distributed throughout the grid structure. The query distance $l$ of the issuing user and the number of friends for each user $f$ are set as 5 km and 50, respectively. The number of pseudo-IDs for each user $k$ changes from 10 to 30. We execute 100 times and then calculate the average number of effective users and the average time cost under different $k$. The above settings are named scenario-3, which is used to study the effect caused by $k$.

(4) Under the condition of querying friends’ locations, ten locations are randomly selected as the initial locations of an issuing user. The total number of users $n$ changes from 1000 to 2500 and the users are randomly distributed throughout the grid structure. The query distance of the issuing user $l$, the number of pseudo-IDs for each user $k$, and the number of friends for each user $f$ are set as 5 km, 10, and 50, respectively. We execute 100 times and then calculate the average number of effective users and the average time cost under different $k$. The above settings are named scenario-4, which is used to research the effect caused by $n$.

7.2. Simulation Results

Figure 8 shows the results under scenario-1. From Figure 8a, it can be known that the number of effective users increases with $f$ no matter which solution is used. The reason is that the increasing $f$ leads to the increasing number of users’ friends and the corresponding dummies. The number of effective users by using our scheme is smaller than other solutions since our SELS can efficiently filter out the issuing user’s friends who are not in the issuing user’s associated grids. However, other solutions have to provide all the issuing user’s friends and the corresponding dummies to LS, which inevitably increases the computational burden of LS. From Figure 8b, it can be seen that: (1) the query time increases with $f$ in all schemes; (2) when $50 \leq f \leq 90$, the query time of our SELS is at least $1/3$ of that of other schemes, i.e., the efficiency of our scheme is at least 3 times better than other solutions due to the same reason as above. In addition, it can be summarized that the effect of $f$ on the query time is linear in all schemes. The reason is that when other parameters are fixed, the number of users (including friends and dummies) that need to be compared for distance with the issuing user is almost fixed, and the only factor that affects the query time is a linearly increasing number $f$.

![Figure 8. The simulation results for scenario-1.](image)

Figure 9 shows the results under scenario-2. From Figure 9a, it can be seen that when using our scheme, the number of effective users increases with $l$, while the number of effective users is almost stable when using other solutions. The reason is that the number of the effective users increases
with the query distance $l$ which determines the number of associated grids in our scheme. However, for other solutions, the friends who are not in the issuing user’s associated grids can not be filtered out, which leads to the result that all the friends have to be retrieved by $LS$. Thus, it can be concluded that the efficiency of our SELS is better than other solutions since the number of effective users in our scheme is smaller than other solutions under different query distances. From Figure 9b, it can be known that: (1) the query time increases with $l$ in all schemes; (2) when the query distance $l$ changes from 1 km to 5 km, the query time of our SELS is at least 1/3 of that of other schemes, i.e., the query efficiency is at least 3 times better than other solutions since a smaller number of effective users need less computational cost. In addition, we can also summarize that the effect of $l$ on the query time is quadratic in all schemes. The reason is that when the query distance changes, the number of users (including friends and dummies) within the issuing user’s query distance also changes, and the change of these users is quadratic. Furthermore, the reason for different curves in Figure 9b among our scheme and other solutions is as follows. When other parameters are fixed, different $l$ leads to the change in the number of effective users and the number of users (including friends and dummies) within the issuing user’s query distance in our scheme. While in other schemes, different $l$ only leads to the change in the number of users (including friends and dummies) within the issuing user’s query distance.

![Figure 9a: The effective users](image1)

![Figure 9b: The query time](image2)

**Figure 9.** The simulation results for scenario-2.

Figure 10 shows the results under scenario-3. From Figure 10a, it can be known that the number of effective users increases with $k$ in all schemes. The reason is that increasing $k$ leads to an increasing number of friends’ dummies. The number of effective users by using our scheme is smaller than other solutions since our SELS can efficiently filter out the issuing user’s friends who are not in the issuing user’s associated grids. However, other solutions have to provide all the issuing user’s friends and the corresponding dummies to $LS$, which inevitably increase the computational burden of $LS$. From Figure 10b, it can be seen that: (1) the query time increases with $k$ in all schemes; (2) when $10 \leq k \leq 30$, the query time of our SELS is at least 1/3 of that of other schemes, i.e., the efficiency of our scheme is at least 3 times better than other solutions due to the same reason as above. In addition, it also can be concluded that the effect of $k$ on the query time is quadratic in all schemes. The reason is that when the number of pseudo-IDs for each user changes, the number of effective users and the number of users (including friends and dummies) within the issuing user’s query distance both changes, and the above two kinds of users changes in multiples. While in other schemes, different $k$ only leads to the change in the number of users (including friends and dummies) within the issuing user’s query distance.
Figure 10. The simulation results for scenario-3.

Figure 11 shows the results under scenario-4. From Figure 11a, it can be seen that the number of effective users is stable with different $n$ in all schemes. The reason is that when other parameters are fixed, the increasing $n$ cannot affect the number of effective users since the number of the issuing user’s friends does not change. Thus, the number of effective users that needs to be sent to LS has no change. However, the number of effective users in our SELS is smaller than other solutions since LS in our scheme only needs to run the distance comparison function for the friends are located in the issuing user’s associated grids. From Figure 11b, it can be seen that: (1) the query time increases with $n$ in all schemes; (2) when the total number of users $n$ changes from 1000 to 2500, the query time of our SELS is at least 1/3 of that of other schemes, i.e., the efficiency of our scheme is at least 3 times better than other solutions since a smaller number of effective users need less computational cost. In addition, it can be concluded that the effect of $n$ on the query time is linear in all schemes. The reason is that when other parameters are fixed, the only factor that affects the query time is a linearly increasing number $n$, which increases the number of dummies within the issuing user’s query distance.

Figure 11. The simulation results for scenario-4.

8. Conclusions

In this paper, while maintaining users’ privacy, we explore the issues of utility in location-sharing systems. To address the problems of system efficiency and the flexibility of users’ privacy-preserving requirements in location-sharing systems, we propose a search efficient privacy-preserving location-sharing solution (i.e., SELS). Specifically, based on the grid structure and users’ query distances, we design a novel approach called associated grids to improve the efficiency of location-sharing
systems while maintaining users’ privacy. In addition, our scheme also provides a user-defined access control policy to meet users’ flexible privacy-preserving requirements, which can effectively prevent the disclosure of users’ privacy in social networks. The detailed efficiency analysis proves that our SELS can improve the efficiency of location-sharing systems. The security analysis also shows the privacy-preserving ability of our scheme. Under the condition of setting different parameters, we explore the effect of these parameters on location-sharing systems and compare our SELS with other schemes. Extensive experiments validate the practicability and performance of our scheme comprehensively.

However, there are also some limitations in our SELS, such as the regional rationality of grid structure and the computing power cost of base stations. In the future, we will think of ways to lessen the limitations.

Author Contributions: Methodology, G.Y. and S.L.; software and validation, Y.X. and H.Z.; formal analysis, resources, and data curation, M.L. and Y.W.; writing—original draft preparation and writing—review and editing, G.Y. and Y.X.; supervision, S.L. and J.W.; funding acquisition, Y.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by the National Key R&D Program of China under Grant 2017YFB0802300, BUPT Excellent Ph.D. Students Foundation (CX2019229), National statistical scientific research project of China (2018LY61), BUPT Ph.D. Students Short term overseas study exchange program, National Natural Science Foundation of China (No.61902361).

Conflicts of Interest: All the authors declare that we have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References
1. Vijayakumar, P.; Obaidat, M.S.; Azees, M.; Islam, S.K.H.; Kumar, N. Efficient and Secure Anonymous Authentication With Location Privacy for IoT-Based WBANs. *IEEE Trans. Ind. Inform.* 2020, 16, 2603–2611. [CrossRef]
2. Sun, G.; Chang, V.; Ramachandran, M.; Sun, Z.; Li, G.; Yu, H.; Liao, D. Efficient location privacy algorithm for Internet of Things (IoT) services and applications. *J. Netw. Comput. Appl.* 2017, 89, 3–13. [CrossRef]
3. Yang, G.; Luo, S.; Zhu, H.; Xin, Y.; Li, M.; Wang, Y. An Efficient Approach for LBS Privacy Preservation in Mobile Social Networks. *Appl. Sci.* 2019, 9, 316. [CrossRef]
4. Yang, G.; Luo, S.; Zhu, H.; Xin, Y.; Xiao, K.; Chen, Y.; Li, M.; Wang, Y. A Mechanism to Improve Effectiveness and Privacy Preservation for Review Publication in LBS. *IEEE Access* 2019, 7, 156659–156674. [CrossRef]
5. Xiaoying, S.; Zhenhai, Y.; Qiming, F.; Quan, Z. A Visual Analysis Approach for Inferring Personal Job and Housing Locations Based on Public Bicycle Data. *Int. J. Geo Inf.* 2017, 6, 205.
6. Wei, W.; Xu, F.; Li, Q. MobiShare: Flexible privacy-preserving location sharing in mobile online social networks. In Proceedings of the 2012 Proceedings IEEE Infocom, Orlando, FL, USA, 25–30 March 2012; pp. 2616–2620.
7. Li, J.; Li, J.; Chen, X.; Liu, Z.; Jia, C. MobiShare+: Security Improved System for Location Sharing in Mobile Online Social Networks. *J. Internet Serv. Inf. Secur.* 2014, 4, 25–36.
8. Liu, Z.; Luo, D.; Li, J.; Chen, X.; Chunfu, J. N-Mobishare: new privacy-preserving location-sharing system for mobile online social networks. *Int. J. Comput. Math.* 2016, 93, 384–400. [CrossRef]
9. Shen, N.; Yuan, K.; Yang, J.; Jia, C. B-Mobishare: Privacy-Preserving Location Sharing Mechanism in Mobile Online Social Networks. *Comput. Stand. Interfaces* 2016, 44, 102–109. [CrossRef]
10. Sun, G.; Xie, Y.; Liao, D.; Hongfang, Y.; Chang, V. User-Defined Privacy Location-Sharing System in Mobile Online Social Networks. *J. Netw. Comput. Appl.* 2017, 86, 34–45. [CrossRef]
11. Li, J.; Yan, H.; Liu, Z.; Chen, X.; Huang, X.; Wong, D. Location-Sharing Systems With Enhanced Privacy in Mobile Online Social Networks. *IEEE Syst. J.* 2017, 11, 439–448. [CrossRef]
12. Li, J.; Yan, H.; Liu, Z.; Chen, X.; Huang, X.; Wong, D. CenLocShare: A centralized privacy-preserving location-sharing system for mobile online social networks. *Future Gener. Comput. Syst.* 2018, 86, 863–972.
13. Liu, J.; Fu, L.; Wang, X.; Tang, F.; Chen, G. Joint Recommendations in Multilayer Mobile Social Networks. *IEEE Trans. Mob. Comput.* 2020, 19, 2358–2373. [CrossRef]
14. Lai, I.K.W.; Liu, Y. The Effects of Content Likeability, Content Credibility, and Social Media Engagement on Users' Acceptance of Product Placement in Mobile Social Networks. *J. Theor. Appl. Electron. Commer. Res.* 2020, 15, 1–19. [CrossRef]

15. Erdemir, E.; Dragotti, P.L.; Gunduz, D. Privacy-Aware Location Sharing with Deep Reinforcement Learning. *IEEE Trans. Inf. Forensics Secur.* 2019, 16, 1–6.

16. Sweeney, L. k-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* 2002, 10, 557–570. [CrossRef]

17. Wu, Z.; Li, G.; Shen, S.; Lian, X.; Chen, E.; Xu, G. Constructing dummy query sequences to protect location privacy and query privacy in location-based services. *World Wide Web Internet Web Inf. Syst.* 2020. [CrossRef]

18. Cui, N.; Yang, X.; Wang, B. A Novel Spatial Cloaking Scheme Using Hierarchical Hilbert Curve for Location-Based Services. In Proceedings of the 17th International Conference on Web-Age Information Management (WAIM), Nanchang, China, 3–5 June 2016; pp. 15–27.

19. Son, J.; Kim, D.; Tashakkori, R.; Tokutaz, A.; Obx, H. A New Mobile Online Social Network Based Location Sharing with Enhanced Privacy Protection. In Proceedings of the 25th IEEE International Conference on Computer Communications and Networks (ICCCN), Waikoloa, HI, USA, 1–4 August 2016; pp. 59–68.

20. Olteanu, A.M.; Humbert, M.; Huguenin, K.; Hubaux, J.P. The (Co-)Location Sharing Game. *Proc. Priv. Enhancing Technol.* 2019, 2019, 5–25. [CrossRef]

21. Lin, T.L.; Chang, H.Y.; Li, S.L. A Location Privacy Attack Based on the Location Sharing Mechanism with Erroneous Distance in Geosocial Networks. *Sensors* 2020, 20, 918. [CrossRef]

22. Xu, C.; Xie, X.; Zhu, L.; Sharif, K.; Guizani, M. PPLS: A privacy-preserving location-sharing scheme in mobile online social networks. *China Inf. Sci.* 2020, 63, 56–68. [CrossRef]

23. Zhang, S.; Choo, K.K.R.; Liu, Q.; Wang, G. Enhancing privacy through uniform grid and caching in location-based services. *Future Gener. Comput. Syst.* 2018, 86, 881–892. [CrossRef]

24. Lu, R.; Lin, X.; Shi, Z.; Shao, J. PLAM: A Privacy-Preserving Framework for Local-Area Mobile Social Networks. In Proceedings of the 33rd IEEE Annual Conference on Computer Communications (IEEE INFOCOM), Toronto, QN, Canada, 27 April–2 May 2014; pp. 763–771.

25. Sunar, B.; Martin, W.J.; Stinson, D.R. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Trans. Comput.* 2007, 56, 109–119. [CrossRef]

**Publisher’s Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).