User-Defined Privacy Location-Sharing System in Mobile Online Social Networks

Gang Sun[1, 2], Yuxia Xie[1], Dan Liao[1,3], Hongfang Yu[1,2], Victor Chang[4]

[1]Key Lab of Optical Fiber Sensing and Communications (Ministry of Education), University of Electronic Science and Technology of China, Chengdu, China

[2]Center for Cyber Security, University of Electronic Science and Technology of China, Chengdu, China

[3]Guangdong Institute of Electronic and Information Engineering, UESTC, China

[4]Xi'an Jiaotong Liverpool University, Suzhou, China

Abstract: With the fusion of social networks and location-based services, location sharing is one of the most important services in mobile online social networks (mOSNs). In location-sharing services, users have to provide their location information to service provider. However, location information is sensitive to users, which may cause a privacy-preserving issue needs to be solved. In the existing research, location-sharing services, such as friends' query, does not consider the attacks from friends. In fact, a user may not trust all of his/her friends, so just a part of his/her friends will be allowed to obtain the user's location information. In addition, users' location privacy and social network privacy should be guaranteed. In order to solve the above problems, we propose a new architecture and a new scheme called User-Defined Privacy Location-Sharing (UDPLS) system for mOSNs. In our scheme, the query time is almost irrelevant to the number of friends. We also evaluate the performance and validate the correctness of our proposed algorithm through extensive simulations.

Key words: Location privacy; Mobile online social networks; Friend attacks; Anonymity; Location sharing

1. INTRODUCTION

Location-based services employ GPS, WLAN, Cellular network technologies to obtain location information of the mobile terminal, and to provide location-based services to the mobile terminal through the wireless network [1]. Due to the development of Internet technology, the well-known dominant mOSNs such as Facebook, YouTube, Twitter have been growing rapidly in both of size and popularity. In these traditional online social networks, users can conveniently exchange information, and share blog, video, images, etc.

When mOSNs and location-based services are integrated together, many location-based services such as near friends' query, "check-in", and simple location sharing can be provided by mOSNs. For example, users can get some preferential service through "check-in" services. In addition, users can query their friends and strangers which close to the current position and obtain their location information. After Facebook integrated with location-based services, they attract a large number of users from starting operations, and the number of users is still growing rapidly [2].

Location-based service (LBS) is one of the most important components in mOSNs, which provides services to users based on the geographical position of the mobile device. With mobility and ever-present Internet connectivity of the world, a great amount of users take the advantage of LBS to query information based on their location. In LBS, users can query the near hospitals, supermarkets, bars and so on, which provides users much convenience.

As LBSs and mOSNs grow in popularity, many new services are spawned, such as friends and travel routes recommendation. However, there are also some challenges need to be solved. Location information is one of the most sensitive privacies to users, and thus it is very valuable. For example, if mOSNs collect users' much location information, they may provide it to third parties since the commercial purpose, which will leak users' location privacy. In addition, much sensitive information can be inferred from location information, when more sophisticated analysis is employed. For example, attackers may deduce that the user's physical health from the data of in hospital. Also, attackers may infer that a user is a drunkard, if the user frequently query the nearest bars.

It is important that keep personal location information from being obtained by malicious attackers. Location privacy includes published time of location information, the spatial location and location service request content. Especially, spatial location is most concerned issue of location privacy in mOSNs. Users' geographical location information mainly relate to the spatial location, which is one of the main concern of this paper.

In mOSNs, the query from friend or stranger in the user-specified range is a typical application of location sharing services. Location-based social networking systems with location sharing services rely on a central server that can obtain all users' detail movement profile, which raises privacy concerns [3-5]. If users' location privacy is not well protected, users are likely to reject to use the location sharing services [6]. Therefore, the development of location sharing services will face many challenges.

Recently, in mONSs, several methods [7-11] have been proposed to protect users' location privacy in friends' and strangers' queries in the user-specified distance. In the research of [7-10], user's location privacy is protected by adding dummy identities. The location service provider can't obtain complete information of users' identities and location. In Ref. [11], a architecture with multiple location servers is proposed. The user's friend set in each friend's query submitted to the location servers is divided into multiple

subsets by the social network server randomly. Query results are sent to social servers through encryption and digital signatures, so social networking server cannot obtain location information of the user. However, theses solutions do not consider the attacks from friends. The user may not trust all of his/all friends, and thus may not want to share location with all of his/her friends.

Based on the reviews mentioned above, in this work, we propose a new solution to achieve user-defined location privacy and social network privacy. This solution allows user to choose whether to provide accurate position in friends' query or share location with a part of his/her friends.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 gives the preliminaries and problem statements. Section 4 presents descriptions on motivation, system model and the specific implementation. Section 5 gives detailed descriptions on security analysis of our UPLS scheme. The simulation results are given in Section 6. Section 7 presents the research contributions and discussion. Section 8 concludes the paper.

2. RELATED WORK

In this section, we survey the privacy-preserving techniques for location-based services and location sharing services.

2.1 Location-based Services

K-anonymity is one of the key technologies to solve the current problem of location privacy in LBS, which can ensure that the probability of user's real position recognized by attacker is at most 1/*k*. Spatio-temporal anonymous method [12-14] is one of the means of achieving *k* anonymity. In the spatio-temporal anonymous method, a trusted third party called central location anonymous server [15-17] is deployed between users and LBS servers. User's location privacy is protected by central location anonymous server.

However, the location privacy-preserving techniques mentioned above for LBS have several drawbacks. For example, it requires a fully-trusted third party, offering limited privacy guarantees, and is prone to single point of failure. If too many users simultaneously request location anonymity, which will lead to performances bottlenecks of the central location anonymous server, a crash may occur. In addition, once an attacker controls the central location anonymous server, the attacker will get all users' detail location information, and thus users' privacy will also be subsequently exposed.

Fake location method [18-21] is proposed to address the issues mentioned above. In the fake location method, users directly communicate with LBS server without a trusted third party, which mainly rely on users' terminal to achieve location privacy. In addition to the true location of user, the fake location method is to find other *k*-1 fake locations for the user. And then the *k* locations will be sent simultaneously to LBS server.

In addition, policy-based and encryption methods are then proposed. In [22], users firstly identify a query area that includes the user's true location. Query area is divided into grid cells with equal size. User encrypts information of queried area and gird configuration, and encrypts the space intersecting with each grid cell as an encrypted identifier. All points of interest within the specified range will be discovered by location-based server. Query results matching user's true location will be sent to users.

2.2 Location Sharing Services

Restricting location sharing to established social relations makes a large class of mobile social applications, such as Serendipity [23]. Previous work [24-25, 27] discussed the problem of sharing locations between established relations in a privacy-preserving way. Novak et al. proposed a scheme allowing two parties to share location information [26]. For example, when a user named Alice queries the location of the other user named Bob, homomorphic encryption is used by the system to detect whether location of Alice matches Bob's access policy. This strategy does not rely on any trusted third party server. A solution solving the problem of *k* nearest neighbor queries is proposed in Ref. [28]. A client-server solution [29] for proximity detection is based on encryption and multi-level partitions of spatial domain. Vicinity region proposed in [29] is a region where the solution will notify a user if any friend of users enter the user's specified area. All work mentioned above just address the issue of location sharing with user's friends. The authors in [30] proposed a system called SmokeScreen, which allow users to share their location with both friends and strangers.

In research [7] and [11], both location server and social network server assumed to be untrusted. Location based service provider may unauthorized obtain the user's complete social relations, when users share location information among trusted social relations. However, different dynamic pseudonyms each query cannot protect users' social network privacy on location server, because the same social network may be related to the same user. In order to address the issue mentioned above, a system with multiple location servers is proposed in Ref. [11].

Each location server can only obtain a portion of user's friends. It is noteworthy that the query time increases linearly with the number of friends and strangers. However, all of the researches mentioned above do not consider the attacks from friends. The user may not prefer their friends to know his/her precise location or just would like to share location with a part of his/her friends.

3. PRELIMINARIES AND PROBLEM STATEMENTS

3.1 The Basic Concepts and Definitions

In this subsection, we explain the main concepts and definitions used in this work. Key notations used in this section are summarized in Table 1.

Total users: It refers to all of users within user-specified distance, where any user meets condition: $dis((x, y),(x_i, y_i))\leq l$. The number of total users is called the total number of users.

Effective users: Any user in total users meeting the condition of $dis((x, y), (x_i, y_i))\leq dif_i$ is called effective user.

Access control: Access control mentioned in this paper includes *distance control* and *identity control*. *Distance control* refers to that query users who specify query distance and queried distance. *Identity control* refers to that users can specify a part of his/her friends to inquiry their location information.

Table 1 Summary of Key Notations

Notation	Description
ID	User's social network identifier.
pid_{ID}	The pseudonym of user ID.
l	Distance threshold in friends' location query.
dif_{ID}	Friend-case threshold distance of user ID .
d	Distance threshold in strangers' location query.
S_{ID}	Stranger-case threshold distance of user ID .
(PK, SK)	User's public key and secret key for social network registration.
(x_{ID}, y_{ID})	User's real location.
$dis((.,.))$	A function for returning distance between inputs.
$min(.,.)$	A function to calculate the minimum value.

3.2 RSA Digital Signature

RSA is a typical asymmetric encryption. Asymmetric encryption algorithm requires two key parts: a public key and a private key. If the public key is used to encrypt data, only the corresponding private key can decrypt; if the private key is used to encrypt data, only the corresponding public key can decrypt. RSA digital signature includes following three main steps:

(1) Key Pair Generation

Select two large prime numbers p and q. Assume the number $M = p \times q$ and $\phi(M) = (p\text{-}1) \times (q\text{-}1)$. Select a number e, which satisfy the Equation (1).

$$gcd(e, \phi(M)) = 1, \quad 1 < e < \phi(M) \tag{1}$$

Function $gcd(.,.)$ returns the maximum common divisor of inputs.

And we define the number d as in Equation (2).

$$d = e^{-1} (mod\ (\phi(M))) \tag{2}$$

Based on Equations (1) and (2), two numbers e and d have been computed. We define (e, M) as the public key, and (d, M) as the private key.

(2) Message encryption and signature

We use n to denote the plaintext, and then the cryptograph can be implemented by the following formula:

$$c = n^e (mod\ M) \tag{3}$$

where the message c is cryptograph, and (n, c) used to denote a signature and will be sent for authorizing.

(3) Message decryption and authentication

After receiving a signature, the cryptograph c can be decrypted by the following formula:

$$m = c^d (mod\ M) \tag{4}$$

where m is the deciphered information of c. If m is same as original message n, it will be accepted as a valid signature. Otherwise, it will be recognized as an invalid signature.

3.3 Pseudonym Generation

In this work, we assume that different pseudonyms have different IDs, i.e., one-to-one mapping relationship must be established between ID and pseudonym.

Mapping rule randomly generates a number by using linear congruential method for ensuring the uniqueness of pseudonym. We call the random number as *pseudorandom number* in this paper.

Random number is generated by the following recursive formula:

$$N_{j+1} = (A \times N_j + B)(mod\ M) \tag{5}$$

where, A, B, and M are constants, and they satisfy the following constraints:

$$gcd(B, M) = 1 \tag{6}$$

$$P = \prod_{c_i \in U_3} c_i\ , P(mod\ \ (A-1)) = 0 \tag{7}$$

where,

$U_3 = \{a_i |\ M(mod\ a_i) = 0, gcd(a_i, b_i) = 1,\ a_i \in Z,\ a_i < M,\ b_i \in Z, b_i < a_i\}$.

$$M(mod\ \ 4) = 0,\ (A\text{-}1)(mod\ \ 4) = 0 \tag{8}$$

$$A < M, B < M, N_0 < M \tag{9}$$

$$A > 0, B > 0 \tag{10}$$

$$\sum_j O(N_j) = 1 \tag{11}$$

Constraints (5)-(10) are to ensure correctly generating random numbers according to linear congruential method. Constraint (11) is used to ensure that each random number is unique, which means that the pseudonyms are unique.

3.4 Friends Addition and Removal

Social network server manages user's social relationship, in the form of social network graph $G=(V, E)$. V is a set of identity vertices and E is a set of edges. If two identity vertices are connected by an edge, the corresponding users are friends.

As shown in the Figure 1, User 1 has friends including User 2, User 3, and User 4. User 2 has friends including User 1 and User 5. This social network graph shows all users' complete social networks.

When a user adds or removes friends from his/her social network, the social network graph will be updated. For

example, if User 1 adds User 5 to his/her social network, an edge connecting User1 and User 5 will be added in Figure 1. If User 1 removes User 2 from his/her social network, the edge connecting User 1 and User 2 will be removed. Thus, social network server dynamically updates all users' social networks. In location-sharing services, user's friends who have been removed from social network cannot share location information with the user. However, the newly established friends can share location information with each other.

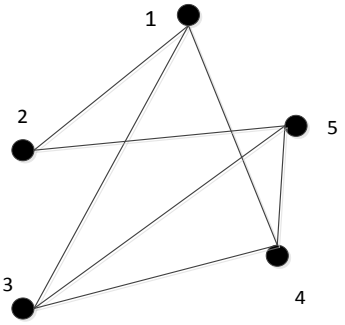


Fig. 1 An example of social network graph

3.5 Threat Model

Location Server (LS) and Social Network Server (SNS) are both assumed to be "honest-but-curious", which means that they will perform users' query scheme and send query results to users correctly, but they may try to obtain as much sensitive information of users as possible.

Users are assumed to be dishonest. Since they try to get as much unauthorized information as possible and may leak location privacy to attackers.

LS and *SNS* cannot collude together to obtain users sensitive information. Information leaks during transmission such as eavesdropping is beyond the scope of this paper yet.

3.6 Security Goals

In this work, the security goals include the following three areas for the location-sharing system in online mobile social networks: *i*) *SNS* is prevented from obtaining users' location information; *ii*) The users' location information cannot be accessed by such friends and strangers if their access control does not match the users' access control; *iii*) *LS* is prevented from getting users' social network information.

4. Motivation and System Model

In this section, we give descriptions on the motivation and a new system model to solve our researched problem.

4.1 Motivation

The authors in [11] had proposed the framework of multiple location servers to prevent the location server from obtaining the users' complete information of social networks. However, the problems below are not considered in [11].

(1) Location servers may obtain users' historical location information because user's fake identifier includes his/her real identifier which is a constant;

(2) The whole set of friends of users may be obtained by location servers if all location servers collude, although the list of user's friends has been divided into multiple parts;

(3) Policy of multiple location servers uses symmetric encryption algorithm and digital signature algorithm in the query processing. The overhead of query time is almost proportional to the number of friends, which consumes much time due to a large number of friends;

(4) Multiple location servers store the same information, which results in wasting of resources;

(5) The user may want to share their location with part of friends matching his/her access control or just want their identity information to be queried in certain situations.

Taking all of the above into account, we propose a new architecture and a new scheme namely User-Defined Privacy Location-Sharing system (UDPLS) in this work.

4.2 System Model

The system model for privacy preserving is composed of three parts: the entity of users, the entity of online social network server and the entity of location server.

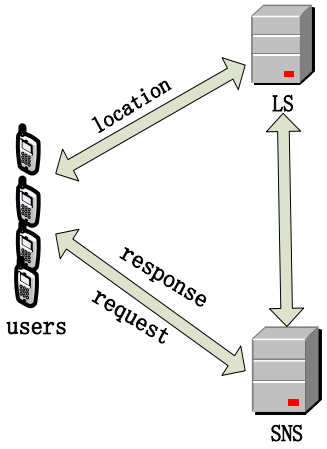


Fig. 2 System model for privacy preserving

The entity of users, carrying a mobile terminal, can share their location with his/her friends, which means that they can execute friends' query within the user-specified distance. Users can communicate with online social network server and location server. Each user has a unique social identity in online social network server.

The entity of online Social Network Server (SNS) manages user's profiles, friend list and information for authentication. It also provides online social network service to users based on the given identity-based information.

The entity of Location Server (LS) stores all users' pseudonyms and corresponding location information in order to provide location-related services.

As shown in Figure 2, the proposed framework for privacy protection can be divided into three steps: *user registration*, *location update* and *request submission*.

1) User Registration

User registration includes the registration at *SNS* and registration at *LS*.

Registration at SNS: Social network server stores the user's ID and corresponding friend set. We assume that user's social network database in the form of {ID, *pid*, *G*, PK} is maintained by *SNS*, where $G=(V, E)$, as defined in Section 3.4. In each registration on social network server, user generates a key pair (*pk*, *sk*), where *pk* is the public key, *sk* is the private key. Users send identity and authentication information to the social network server in the form of {ID, *ts*, $\text{Sig}_{SK_{ID}}$ (ID, *ts*), *pk*}, where $\text{Sig}_{SK_{ID}}$ (ID, *ts*) denotes a signature generated with user's secret key SK_{ID} over the timestamp *ts*. Then social network server needs to find the same ID, and corresponding public key PK_{ID} that user has registered before. And then social network server verifies the correctness of the signature $\text{Sig}_{SK_{ID}}$ (ID, *ts*) with PK_{ID}. If the user is valid, then the SNS randomly generates a pseudonym *pid* for uniquely identifying the user. Each time a user register at social network server, SNS will randomly generate a pseudonym for the user to prevent the *LS* to obtain user's real identity. Finally, the social network server sends *pid* to the user. At the same time, social network server also sends information to the *LS* in form of (*pid*, *pk*).

Registration at LS: We assume that a location database in the form of {(*pid*, *pk*, (*x*,*y*), dif_{ID}, s_{ID})} is kept by *LS*. After receiving *pid*, the user sends the information to the *LS*, in the form of {*pid*, *ts*, Sig_{sk} (*pid*,*ts*), (*x*,*y*), dif_{ID}, s_{ID}}, where Sig_{sk} (*pid*,*ts*) is a signature generated with user's secret key *sk* over the timestamp *ts*. *LS* then uses the user's public key *pk* to verify the correctness of the signature.

2) Location update

When a user's location changes, a new pseudonym *pid'* will be assigned to the user by social network server. The user's pseudonym *pid'* will be updated in the social network server. And the user will regenerate a key pair (*pk'*, *sk'*) for registration at location server. User sends information to social network in the form of {ID, *ts*, $\text{Sig}_{SK_{ID}}$ (ID,*ts*), *pk'*}. If SNS verifies that the user is valid, it will send *pid'* to the user, and send (*pid'*, *pk'*) to *LS*. Then the user sends the information to the LS, in the form of {*pid'*, $\text{Sig}_{sk'}$ (*pid'*,*ts*), (*x*, *y*), dif_{ID}, s_{ID}}, where (*x*, *y*) is the user's current location. Once LS verifies that the user is a valid user with the public key *pk'*, the user's location (*x*, *y*) and (dif_{ID}, s_{ID}) will be updated.

3) Request submission

Request submission includes *friends' query request submission* and *strangers' query request submission*.

Friends' location query request submission: User sends friends' query to the social network server, in the form of (ID, *Y/N/ <friends-set>*), and also sends information to *LS*, in the form of (*pid*, *l*, *Y/N*). Where *N* represents that only the user's ID can be obtained by the users who meet the user's access policy; *Y* represents all users who meet the user's access policy can obtain the user's ID and location information. If *<friends-set>* is not empty, indicating that the user is only prefer to share location information with specified friends who meet the user's access policy in *<friends-set>*.

After receiving the inquiry request, *LS* checks all of pseudonym within the distance *l*. Then *LS* executes access control based on these users' friend-case threshold of distance. Assume a pseudonym set is PID, and PID= {pid_1,…, pid_i,…,pid_n}. All pseudonyms in PID must meet the following condition:

$$dis\left((x,y),(x_i,y_i)\right) \le min(l, dif_i)$$

Users in PID who have sent *N* to location server will be put into set PID_1, otherwise set PID_2, where PID_1= {pid_{11},…, pid_{1i},…,pid_{1n}}, and PID_2={pid_{21},…,pid_{2i},…,pid_{2n}}. *LS* sends PID_1 to the user, and sends PID_2 and corresponding location information to the user, in the form of {(pid_{21}, (x_{21},y_{21})),…, (pid_{2i}, (x_{2i},y_{2i})),…, (pid_{2n}, (x_{2n},y_{2n})).

Assume that a user's friend set is represented by set *Friend-Set*, and thus *Friend-Set* = {(ID_1,pid_1),…,(ID_i, pid_i),…, (ID_n,pid_n)}. If location information of friends in *Friend-Set* is allowed to access by the user, these friends' ID and corresponding pseudonyms will be added to the collection *Friend-Set'*.

Assume *Friend-Set'*= {(ID_1', pid_1'), …,(ID_i', pid_i'), …, (ID_n', pid_n')}. Social network server sends *Friends-set'* to the user. The user then matches the received information from location server and social network server. And then the query results can be obtained by the user. The matching process can be described as: For any pseudonym pid_i belonging to PID_1, if $pid_i \in$ *Friend-Set'*, the user obtains the corresponding ID_i; for any pseudonym pid_i belonging to PID_2, if $pid_i \in$ *Friend-Set'*, the user obtains the corresponding ID_i and location information (x_i, y_i).

Strangers' location query request submission: User sends strangers query request to social network server, as well sends information to *LS* in the form of (*pid*, *d*, *s*). Assume a pseudonym set is PID, and PID={pid_1,…, pid_i,…,pid_n}. All pseudonyms in PID must meet the following condition:

$$dis\left((x,y),(x_i,y_i)\right) \le min(d, s_i)$$

Since location information of pseudonyms is added into PID, we update the PID as {(pid_1, (x_1,y_1)),…,(pid_i, (x_i, y_i)) ,…, (pid_n, (x_n,y_n))}. And *LS* directly sends PID to the user.

Assume another pseudonym set is PID', any pseudonym in PID' is randomly selected by *LS* and is different from that in PID. PID' is added to PID for updating PID. Then the updated PID is sent to social network server. Social network server cannot determine which pseudonym matches the user's access policy, thus privacy protection is enhanced.

After receiving PID', social network server removes pseudonyms that belong to the friends' pseudonym of the user from PID'. Thus, PID' is updated to pseudonym set PID_1. Social network server sends PID_1 and corresponding ID to the user, in the form of {(ID_1, pid_1), …, (ID_i, pid_i), …, (ID_n, pid_n)}. Here we denote the set PID_2 as PID_2 = {(ID_1, pid_1), …, (ID_i, pid_i), …, (ID_n, pid_n).

The user matches the received information from location server and social network server. For any pseudonym pid_i belonging to PID, if $pid_i \in PID_2$, the user obtains the corresponding ID_i and location information (x_i, y_i).

4.3 Algorithm for Friends' Location Query

The execution of friends' query in UDPLS has shown above. We give the detailed pseudo code of algorithm in the Figure 3.

***Algorithm* 1: Friends' location query**

Input:1. The user's location, (x, y);
2. The user's query distance, l.
Output: Friends' location and ID.
1: **for** any pseudonym k within l
2: **if** ($dis((x, y),(x_k, y_k)) <= min(l, dif_k)$)
3: Add k to set U;
4: **end if**
5: **end for**
6: **for** any pseudonym k in U
7: **if** (k have sent N to LS)
8: Add k to set U_1;
9: **else**
10: Add k to set U_2;
11: **end if**
12: **end for**
13: **for** any pseudonym k in U_2
14: Add the location information (x_k, y_k) to U_2;
15: **end for**
16: **for** any pseudonym k in U
17: **if** (k is a pseudonym of the user's friends)
18: Add k and its ID to set U_3;
19: **end if**
20: **end for**
21: **for** any pseudonym k in U_1
22: **if** (k in U_3)
23: Add the ID to result;
24: **end for**
25: **for** any pseudonym k in U_2
26: **if** (k in U_3)
27: Add the ID and (x_k, y_k) to result;
28: **end if**
29: **end for**
30: **return** result

Fig.3 Algorithm for friends' location query in UDPLS

4.4 Algorithm for Strangers' Location Query

The process of strangers' query has been described in previous section. In this subsection, we give the detailed pseudo code of algorithm in the Figure 4.

***Algorithm* 2: Strangers' location query**

Input: 1. The user's location, (x, y);
2. The user's query distance, d.
Output: strangers' location and ID.
1: **for** any pseudonym k within d
2: **if** ($dis((x, y),(x_k, y_k)) <= min(d, s_k)$)
3: Add k and (x_k, y_k) to set U;
4: Add k to set U_1;
5: **end if**
6: **end for**
7: Randomly add some other pseudonyms to U_1;
8: **for** any pseudonym k in U_1
9: **if** (k is not a pseudonym of the user's friends)
10: Add k and its ID to set U_2;
11: **end if**
12: **end for**
13: **for** any pseudonym k in U
14: **if** (k in U_2)
15: Add the ID and (x_k, y_k) to result;
16: **end if**
17: **end for**
18: **return** result

Fig.4 Algorithm for strangers' location query in UDPLS

5. SECURITY ANALYSIS

As we mentioned in previous sections, *LS* and *SNS* are both assumed to be "honest-but-curious", and cannot collude. We need to prevent the *LS* to obtain the user's social network, but also to prevent the *SNS* to obtain the user's location information. Furthermore, the user's friends and strangers who did not meet the conditions for access control cannot obtain the user's location information. Therefore, security analysis is necessary for the following aspects.

5.1 Privacy of User's Identity

Social network server manages user's profiles and friend list, so we do not consider privacy of user's identity on the social network server, only need to analyze that whether there is an issue of disclosing user's identity on *LS*. While a user submits a query request, social network server will randomly generate a unique pseudonym for the user, even when the user changes his/her location.

Assume the total number of users is n, then the probability to link users' identity and pseudonym can be computed as:

$$P_{UDPLS}^{1} = 1/n.$$

where n is a large number, thus the probability of identity exposure is very small.

In the MLS framework [11], pseudonym includes user's real ID. Assume the length of pseudonym is l, and the length of user's real ID is i ($1 \leqslant i < l$). The probability that a user exposes his/her identity can be computed as:

$$P_{MLS}^{1} = 1/(l-1)$$

For example, we consider a case in which n = 500, and l =30, and it can be known that $1/n = 1/500 < 1/29$. Therefore, the probability of identifying a user's identity in this paper is much smaller than that of MLS proposed in [11].

5.2 Privacy of Friends' Information

Social network server manages user's friend list, so we do not consider privacy of user's friend information on social network server, only need to analyze whether *LS* can obtain friends' information of the user. In our framework, *LS* will send all pseudonyms meeting distance control and location information corresponding to the user. There may be no friends nor parts of friends in pseudonyms.

Assume the number of pseudonyms is *w*, and then the probability to obtain friends' information is as follows:

$$P_{UDPLS}^{2} = 1/2^{w}.$$

We denote the probability to crash the secret key of RSA is *p*. The probability to get friends' information in Ref. [11] can be computed as:

$$P_{MLS}^{2} = \max(1/2^{w}, p),$$

where the function max(.,.) returns the maximum number of inputs. Obviously, $1/2^{w} \leqslant \max(1/2^{w}, p)$, i.e., the probability in our proposed solution is smaller than that of MLS [11].

5.3 Privacy of User's Location

The analysis above shows that the probability for *LS* to link user's identity information and pseudonym is much small. Assume the total number of different location is *k*. Since pseudonym and location information are associated, it can be known that the probability for *LS* to infer user's location information is the probability to obtain users' identity or 1/*k*. Thus, we can see the probability for *LS* to infer user's location can be calculated as:

$$P_{UDPLS}^{3} = \max(1/k, 1/n).$$

The probability to identify user's location in [11] can be shown as:

$$P_{MLS}^{3} = \max(1/k, 1/(l-1)).$$

Usually, *n* is larger than *l*-1, we can infer the probability in this paper is less than or equal to the probability in [11].

We also mainly consider whether user's location privacy on social network server has been secured. In our framework, user's location information will be directly sent to *LS*, therefore, social networks cannot obtain any location information of user from *LS*. So for *SNS*, the probability to obtain user's location can be demonstrated as:

$$P_{UDPLS}^{4} = 1/k,$$

where *k* must be a larger number, which means the probability is negligibly small.

And the probability in [11] can be computed as:

$$P_{MLS}^{4} = \max(1/k, p).$$

Obviously, $1/k \leqslant \max(1/k, p)$, so we can get the conclusion that the probability to obtain user's location in our framework is no greater than the probability in [11].

5.4 Social Network Privacy

Since social network server manages user's friend set, we do not consider social network privacy preserving on social network server. When a user updates his/her location, social network server will re-generate a pseudonym for the user.

Assume the total number of users is *n* in *LS*. As analyzed above, *LS* can not sure which pseudonym is user's friends, so the probability to infer user's social network privacy can be calculated as:

$$P_{UDPLS}^{5} = 1/2^{n}$$

In practical applications, *n* may be a big number, so the probability is negligibly small.

User's friend set that randomly divided into multiple subsets are sent to multiple location servers in Ref. [11]. At the same time, each subset is added with some incorrect pseudonyms. We denote the total number of pseudonyms (include all friends' pseudonyms and incorrect pseudonyms) is *r* and the number of friends in friend set is *t*. The probability to infer user's social network can be computed as:

$$P_{MLS}^{5} = \sum_{i=0}^{r-t} (C_{r-t}^{i} / C_{t+i}^{i}) / \sum_{j=0}^{r} C_{r}^{j}.$$

Usually, *n* is much larger than *r* and *t*. Considering a case in which *n* =5000, *r* =200, and *t* =150, we have that the probability in the solution proposed in this work is smaller than that in [11].

5.5 Authorized Access

Two kinds of access control are included in this paper. One is distance control, the other is identity control. Distance control includes user-specified query distance and user-specified queried distance. Identity control allows user to specify a part of friends to share location with them or his/her ID to be queried. We have assumed that social network server and location server are "honest-but-curious". "Honest" means location server and social network server execute query according to our designed framework and scheme. Therefore, user's access control can be achieved.

6. Simulation Results And Analysis

For evaluating the effectiveness of our proposed UDPLS framework, we have conducted extensive simulations. In this section, we first describe the simulation environment, and then give the simulation results and analysis.

6.1 Simulation Environment

We have implemented the proposed UDPLS and multiple location servers (MLS) framework in [11] by using C++ programming language in our simulations. For cryptographic functions, we employ the Crypto++ library (http://www.cryptopp.com/). We generate 7 scenarios for experiments as shown below in order to compare the performance of

UDPLS with that of MLS. The simulations have been run on a 64-bit machine with 3.3GHz Intel CPU and 8 GB RAM.

Scenario-1: The total number of users within query range is 10,000. The number of effective users change among 0-9000, in which the number of friends is 0. The number of location servers is 1.

Scenario-2: The total number of users within query range is 10,000. The number of effective users changes among 0-9000, in which the number of friends is 40. The number of location servers is 1.

Scenario-3: The total number of users within query range changes among 2000-16000. The number of effective users is 2000, in which the number of friends is 0. The number of location servers is 1.

Scenario-4: The total number of users within query range changes among 2000-16000. The number of effective users is 2000, in which the number of friends is 10. The number of location servers is 1.

Scenario-5: The total number of users is 6000 within query range. The number of effective users is 3000, in which the number of friends changes among 0-90. The number of location servers is 1.

Scenario-6: The total number of users is 6000 within query range. The number of effective users is 3000, in which the number of friends is 10. The number of location servers varies among 1-9.

Scenario-7: The total number of users is 3000 within query range. The number of effective users is 70. The number of strangers in effective users varies among 0-30. The number of location servers is 1.

Performance metrics. We measure the performance of our UDPLS and MLS framework [11] in terms of the *query time*, and the *computation time* of query on the client side, at location server and at social network server, respectively. We also measure *registration time* of users.

6.2 Simulation Experiment Results

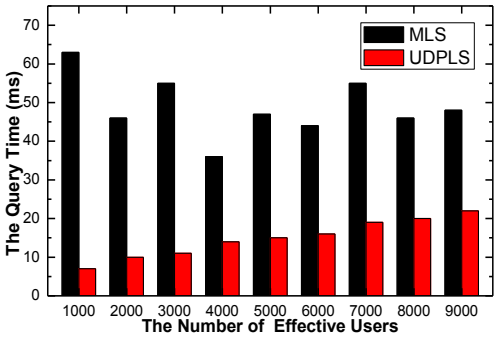
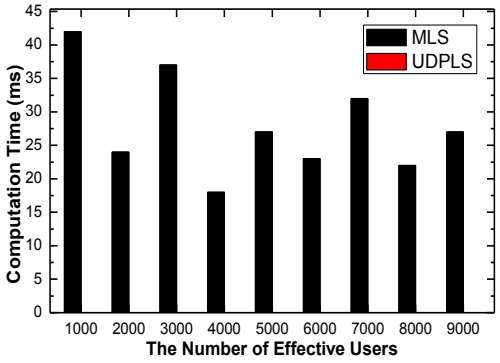
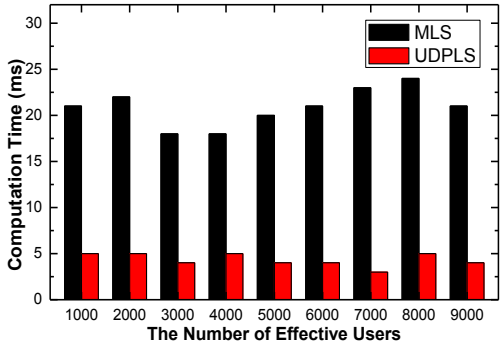


Fig.5 Simulation results on the query time under *Scenario-1*

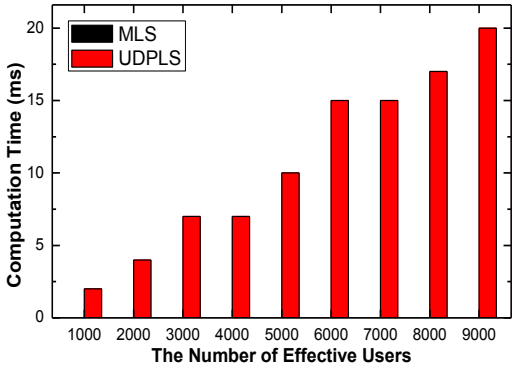
Figure 5 shows the simulation results on the query time for two compared methods under *Scenario-1*. From Figure 5, we can see that query time of our proposed UDPLS framework is shorter than that of MLS proposed in [11] for friends' location query, when there are no friends. It worth noting that the query time does not include registration time. The query time of MLS fluctuates, this is because that the number of friends is 0 in the simulation environment and MLS construction will still generate a one-time key pair, which will cost some time uncertain. Therefore, the query time fluctuates within a certain range. The query time in UDPLS increases with the growth of the number of effective users. This is because that the time for matching friends' identity and pseudonyms is proportional to the number of effective users. However, it just shows a slowly increasing since the proposed UDPLS algorithm is time efficient.



(a) Computation time consumed on client



(b) Computation time consumed on location server



(c) Computation time consumed on social network server

Fig.6 Simulation results on computation time under *Scenario-1*

From Figure 6(a), we can see that the computation time on client in UDPLS is 0. This is because that the number of friends in query distance is 0, so no time is needed to match user's friend identity and pseudonyms. However, computation time on client in MLS framework fluctuates, which is consistent with our previous analysis.

Figure 6(b) shows that the computation time on location server in UDPLS system is shorter than that in MLS system. That is because that location server in MLS has to decrypt the user's location information and find the user's friends with user-defined distance, and thus consumes more time.

From Figure 6(c), we can see that the computation time on social network server in MLS framework is 0. Since nothing is needed to be computed on social network server in MLS. However, social network server has to find friends' pseudonyms among effective users. Therefore, the computation time on social network in UDPLS shows slow growth with the increasing of the number of effective users.

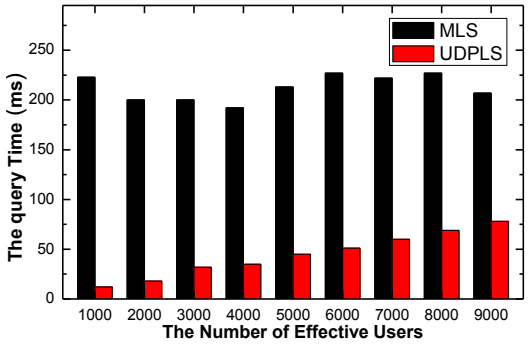
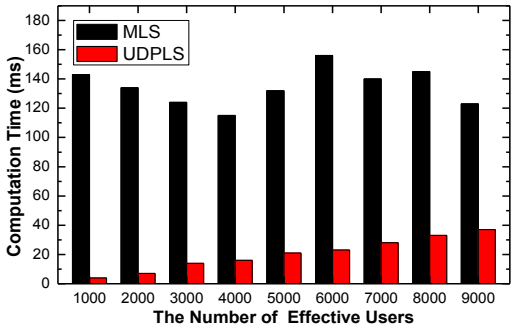
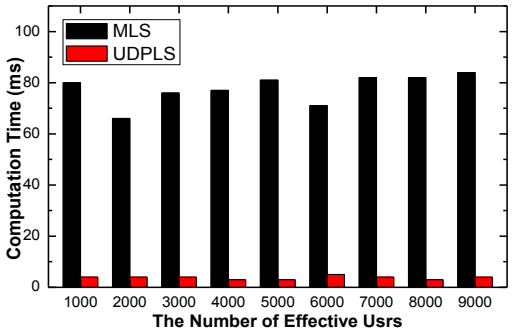


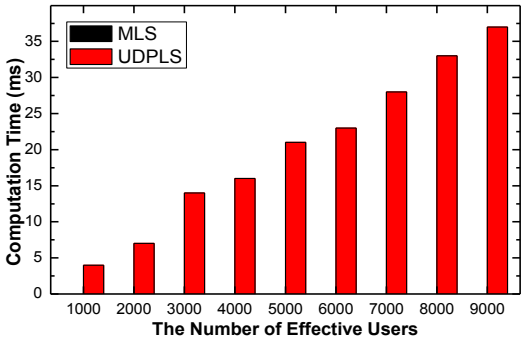
Fig.7 The effective users including 40 friends



(a) Computation time consumed on client



(b) Computation time consumed on location server



(c) Computation time consumed on social network server

Fig.8 The simulation results under *Scenario-2*

Figure 7 shows the simulation results on the query time for two compared methods under *Scenario-2*. It is clear that the query time of MLS is much higher than that of UDPLS. The query time of MLS fluctuates between 180ms-250ms, and there is no significant increase or decrease trend. The query time of UDPLS shows a slow growth. Figure 7 is similar to Figure 5, because just the number of friends is different in these two scenarios.

From Figure 8(a), we can see that computation time on client in MLS system is much higher than UDPLS system. This is because friends' location information has to be decrypted by the user, which means that more computation time on client is needed. The computation time on client in UDPLS is very short and shows slow growth. Since the user has to match friends' identities and pseudonym. Figure 8(b) shows that the computation time on location server is almost stable in both MLS and UDPLS. That means computation on location almost has no business with the number of effective users. Figure 8(c) demonstrates that the computation time on social network in MLS framework is 0, and the computation time on social network in UDPLS framework is growing linearly that is similar to Figure 6(c).

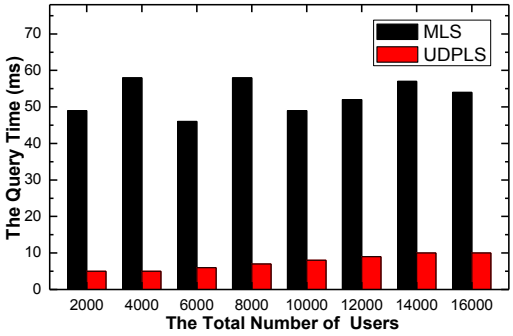
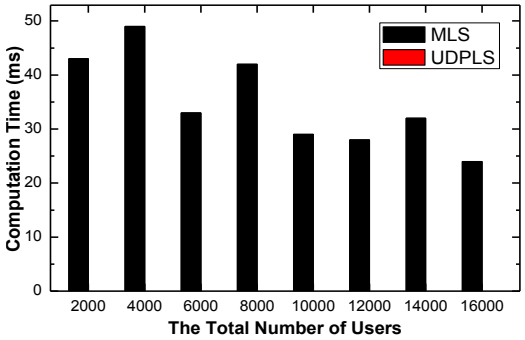


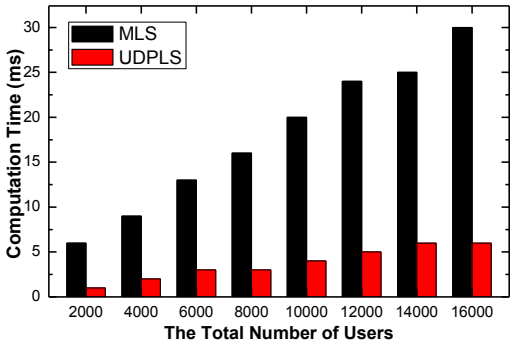
Fig.9 The total number of users including 0 friends versus the query time

Figure 9 shows the simulation results on the query time for two compared methods under *Scenario-3*. From Figure 9, we can see that the query time in UDPLS system is shorter than that in MLS system. Furthermore, the query time in MLS framework fluctuates among 40-60ms, and the query time in UDPLS framework shows a slight growth.

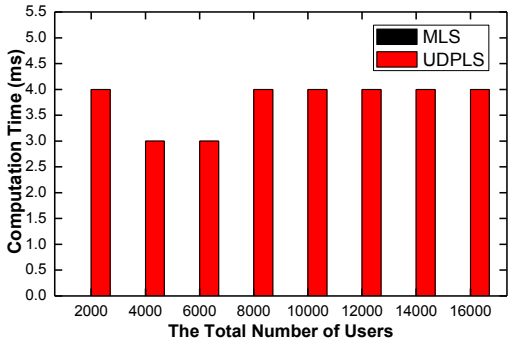
From Figure 10(a), we can see that the computation time on client in MLS system fluctuates among 20-50ms. However, the computation time on client in UDPLS framework is 0. The reason is similar with the description for Figure 6(a). Figure 10(b) shows that computation times on location server in SML system and UDPLS system are both increased with the growth of the total number of users. This is because more time is needed to find effective users from total users. Furthermore, from Figure 10(c), we can see that the computation time on social network in UDPLS is stable, and computation time on social network in MLS is 0. The reason is that the computation time on social network server almost has no business with the total number of users when the number of effective users is constant.



(a) Computation time on client



(b) Computation time on location server



(c) Computation time on social network server

Fig.10 The simulation results for *Scenario-3*

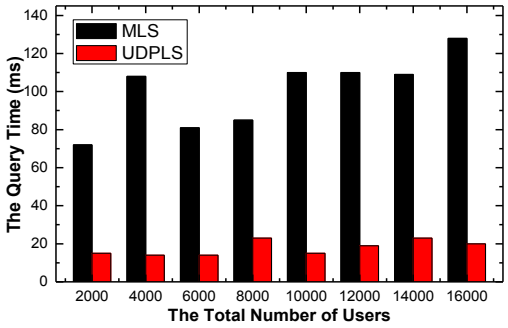
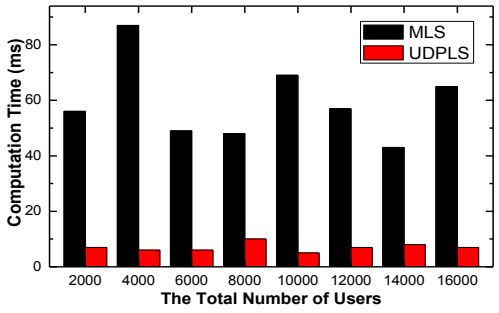


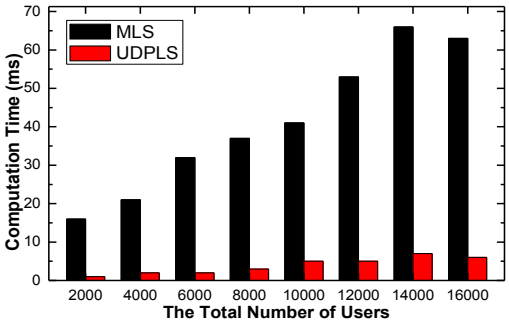
Fig.11 The total number of users including 10 friends

Figure 11 shows the simulation results on the query time for two compared methods under *Scenario-4*. We can see that the Figure 11 is similar to the Figure 9, but the difference between the query time in MLS framework and UDPLS framework is bigger, since the number of friends is increased to 10.

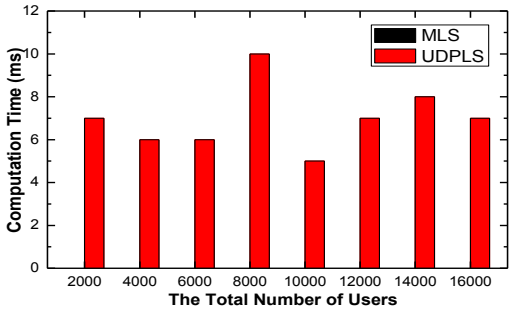
Comparing Figure 12 with Figure 8, we can see they are similar in a large extent, but the computation time difference is larger in Scenario-4. Thus, we can conclude that the number of friends is an important factor which will affect the query time in MLS system. However, the query time in UDPLS framework almost has no business with the number of friends.



(a) Computation time on client



(b) Computation time on location server



(c) Computation time on social network server

Fig.**12** The simulation results for *Scenario-4*

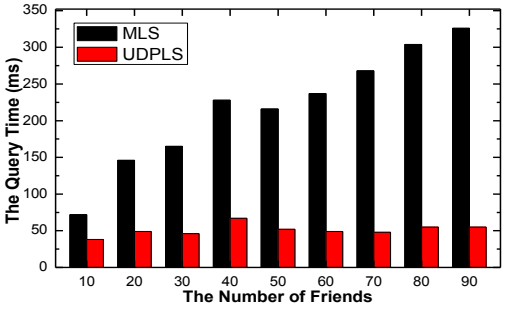
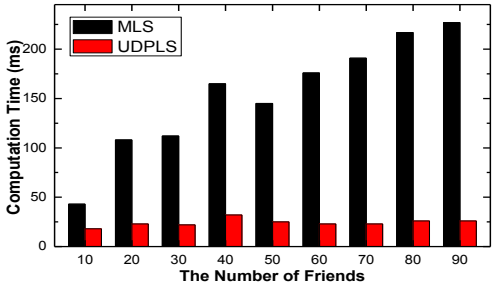
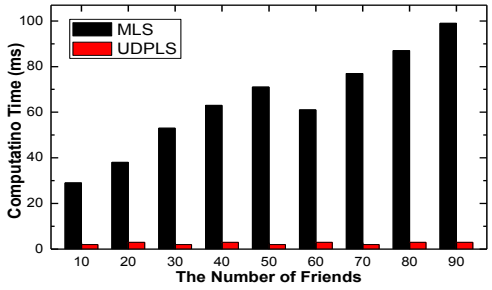


Fig.13 The number of friends versus the query time

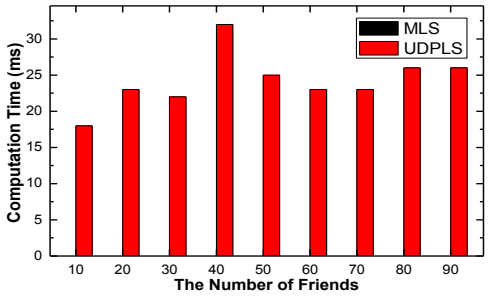
Figure 13 shows the simulation results on the query time for two compared frameworks under *Scenario-5*. From Figure 13, we can see that the query time in MLS system increased with the growth of the number of friends. However, the query time in UDPLS system still keeps stable and is much shorter than that in MLS system. In addition, the query time difference is increased with the growth of the number of friends. From the point of the query time, there is a distinct advantage in UDPLS system compared with MLS system. This is because the more number of friends means that more time of encryption and decryption is needed in MLS framework.



(a) Computation time on client



(b) Computation time on location server



(c) Computation time on social network server

Fig.**14** The simulation results for *Scenario-5*

From Figure 14(a), we can see that computation time on client is growing with the increasing of the number of friends. The reason is that friends' location information is decrypted on client. However, the query time of our proposed system is stable and much little. Figure 14(b) shows that computation time on location server in MLS system is also growing with the increasing of the number of friends. The computation time on location server in UDPLS framework is stable and much less than MLS framework. From Figure 14(c), we can see that computation time on social network server is 0, which is similar to results shown before. This is because that the number of location server is 1, so user's friend list does not need to be divided. Thus, social network server has nothing to compute. Computation time on social network server in UDPLS system fluctuates among 10-35ms.

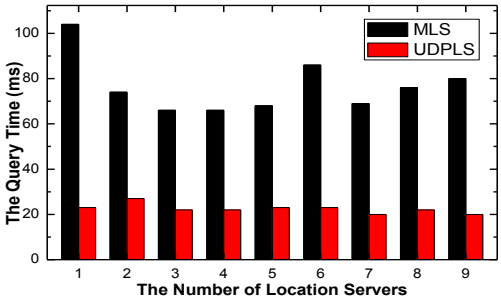


Fig.15 The number of location servers versus the query time

The Figure 15 shows the simulation results on the query time for two compared methods under *Scenario-6*. Assume that the number of the user's friends in friends list is 100, and the 10 friends satisfying access policy are evenly distributed to multiple location servers. We select maximum encryption time on multiple location servers as encryption time instead of the sum of encryption time on location server.

We can see that the query time in MLS system fluctuates among 60-110ms. The query time in UDPLS system is stable and much less than in MLS system. The 10 friends are evenly distributed to multiple location servers, which means the contents having to be encrypted are distributed. However, the query time in MLS framework does not show a downward trend. This is because that the decrease of encryption time is not enough to affect the trend of query time. Thus, the number of location servers is not a key factor for reducing the query time in MLS system. In addition, our proposed system

does not involve division of friend list, so it has no business with the number of location servers.

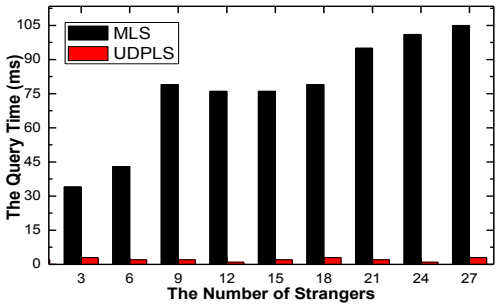


Fig.16 The number of strangers versus the query time

Figure 16 shows the simulation results on the query time for two compared frameworks under *Scenario-7*. From Fig. 16 we can see that the query time of UDPLS is much shorter than that of MLS. It is clear that query time of MLS fluctuates when the number of strangers is among 0-10, but the query time shows an upward trend when the number of strangers is among 10-25. This is because that when the number of strangers is not enough, the query time is greatly affected by the generation time of key pair. However, when the number of strangers is enough, the greater the number of strangers, the more time of encryption and decryption consumed. The query time is much short and stabilized in UDPLS, since the numbers of effective users and strangers are few and thus matching calculation is quick.

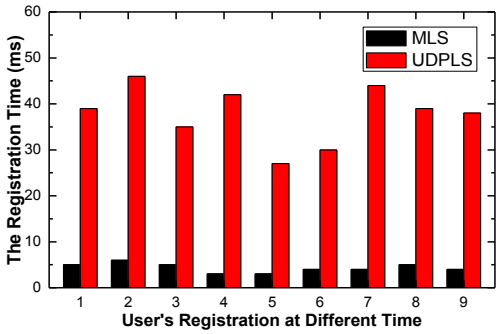


Fig.17 User's registration time at both *LS* and *SNS*

We also show the simulation results of registration time when the user registered at different time, shown in Figure 17. Registration time in MLS is less than that in UDPLS, and registration time difference between UDPLS and MLS is 30ms-40ms. However as shown before, it is smaller compared with the query time difference.

Furthermore, we conduct extensive simulations in order to prove the correctness and feasibility of our proposed solution. We present a large number of simulation results in the following tables. In this set of simulations, we assume that the information can correctly transmit among location server, social network server, and users, and the response time is limited at 300ms. Table 2 shows the result of friends' location query when a user shares his/her location with all of his/her friends. Table 3 gives the result of friends' location query when a user shares his/her location with part of his/her friends specified by the user. Table 4 displays the results of strangers' location query.

Table 2 Results for sharing with all friends

Number of friends	700	800	900	1000	1100
Number of queried friends	700	800	900	953	1002
Number of error IDs or locations	0	0	0	0	0
Accuracy ratio	100%	100%	100%	95%	91%

Table 3 Results for sharing with parts of friends

Number of friends specified by user	700	800	900	1000	1100
Number of queried friends	700	800	900	921	936
Number of error IDs or locations	0	0	0	0	0
Accuracy ratio	100%	100%	100%	92%	85%

Table 4 Results on strangers' location query

Number of strangers	700	800	900	1000	1100
Number of queried strangers	700	800	900	962	986
Number of error IDs or locations	0	0	0	0	0
Accuracy ratio	100%	100%	100%	96%	89%

As shown in Table 2, Table 3 and Table 4, the solution we proposed has good effectiveness when users perform friends' location query or strangers' location query. We can see the accuracy ratio is very high even under a large-scale scenario. It should be noted that if the response time were set up to 1000ms, the accuracy ratio would rise to 100%.

7. RESEARCH CONTRIBUTIONS AND DISCUSSION

We propose a new solution to achieve user-defined location privacy and social network privacy. Social network server is prevented from obtaining users' location information, and location server cannot get users' social network information in our proposed solution. Moreover, users' location information cannot be accessed by people who does not match their access control. Our main research contributions are described as follows.

- We design an efficient algorithm to preserve user's location privacy and network privacy on location server, and preserve user's location privacy on social network server. Our proposed algorithm is suitable for both friends' and strangers' location queries, and not leak user's location information on location servers.
- For preserving privacy, we construct a novel system model which needs only one location server to execute friends' location queries and strangers' location queries.
- Considering that the user may not trust all of his friends, our proposed system model allows the user share his/her location with a part of his friends.

- We evaluate our algorithm by conducting extensive simulations under various scenarios. Simulation results show that our proposed algorithm incurs a lower time complexity than existing approaches.

However, for protect user's location privacy and social network privacy, when the user query friends' or strangers' location, all pseudonyms and their location information that match their access control will be sent to the user by location server, which will result in additional traffic overhead. Moreover, when users change their location, their identities will be certified by location server and the time consumption cannot be ignored.

8. CONCLUSION

In this paper we study the problem of protecting users' privacy in location sharing services, such as nearby friends query and strangers query. We propose a new framework and a new query algorithm (UDPLS) to protect user's location privacy on social network server and user's social network privacy on location privacy. Users can share location with specified-friends instead of all of his friends. It is noteworthy that query time of our framework almost has no business with the number of fiends in friend query. We match pseudonym of user's friends and ID in the user terminals. We conduct extensive simulation experiments to evaluate the performance of our system and algorithm. The simulation results show that the proposed algorithm outperforms the existing approach.

The solution proposed in this work will result in additional traffic overhead, our future work is to reduce the traffic overhead without compromising users' privacy.

ACKNOWLEDGEMENT

This research was partially supported by the National Grand Fundamental Research 973 Program of China under Grant (No. 2013CB329103), Natural Science Foundation of China (61571098), China Postdoctoral Science Foundation (2015M570778), Guangdong Science and Technology Foundation (2013A040600001, 2013B090200004, 2014B090901007, 2015A040404001, 2013B040300001).

REFERENCES

[1] K. Virrantaus, J. Markkula, A. Garmash, et al. Developing GIS-supported location-based services. The 2nd International Conference on Web Information Systems Engineering, 66-75, 2001.

[2] N. Li, G. Chen. Sharing location in online social networks. IEEE Network, 24(5), 20-25, 2010.

[3] L. Barkhuus, B. Brown, M. Bell, et al. From awareness to repartee: Sharing location within social groups. The SIGCHI Conference on Human Factors in Computing Systems, 497-506, 2008.

[4] E. Toch, J. Cranshaw, et al. Empirical models of privacy in location sharing. the 12th ACM International Conference on Ubiquitous Computing, 129-138, 2010.

[5] S. Consolvo, I. Smith, T. Matthews, et al. Location disclosure to social relations: Why, when, &what people want to share. The SIGCHI Conference on Human Factors in Computing Systems, 81-90, 2005.

[6] L. Barkhuus, A. K. Dey. Location-based services for mobile telephony: A study of users' privacy concerns. The 13th International Conference on Human-computer Interaction, 709-712, 2003.

[7] W. Wei, F. Xu, Q. Li. Mobishare: Flexible privacy-preserving location sharing in mobile online social networks. IEEE INFOCOM, 2616-2620, 2012.

[8] J. Li, J. Li, X. Chen, Z. Liu, et al. Mobishare+: Security improved system for location sharing in mobile online social networks. Journal of Internet Services and Information Security, 4(1), 25-36, 2013.

[9] Z. Liu, J. Li, X. Chen, et al. New privacy-preserving location sharing system for mobile online social networks. International Conference on P2P, Parallerl, Grid, Cloud and Internet Computing, 214-218, 2013.

[10] Z. Liu, D. Luo, J. Li, et al. N-Mobishare: New privacy preserving location-sharing system for mobile online social networks. International Journal of Computer Mathematics, 93(2), 384-400, 2016.

[11] J. Li, H. Yan, Z. Liu, et al. Location-sharing systems with enhanced privacy in mobile online social networks. IEEE Systems Journal, 1-10, 2015.

[12] A. Hossain, A. Hossain, S. J. Jang , et al. Privacy-aware cloaking technique in location-based services. IEEE International Conference on Mobile Services (MS), 9-16, 2012.

[13] J. Zheng, X. Tan, C. Zou, et al. A cloaking-based approach to protect location privacy in location-based services. The 33rd Chinese Control Conference (CCC), 5459-5464, 2014.

[14] B. Ying, D. Makrakis. Protecting location privacy with clustering anonymization in vehicular networks. IEEE INFOCOM Workshops, 305-310, 2014.

[15] M. F. Mokbel, C. Y. Chow, W. G. Aref. The new Casper: query processing for location services without compromising privacy. International Conference on Very Large Data Bases, 763-774, 2006.

[16] T. Xu Y. Cai. Location anonymity in continuous location-based services. The 15th annual ACM international symposium on Advances in geographic information systems, 1-8, 2007.

[17] T. Xu, Y. Cai. Exploring historical location data for anonymity preservation in location-based services. IEEE INFOCOM, 547-555, 2008.

[18] B. Niu, Q. Li , X. Zhu, et al. Achieving k-anonymity in privacy-aware location-based services. IEEE INFOCOM, 754 -762, 2014.

[19] B. Niu, Z. Zhang, X. Li, et al. Privacy-area aware dummy generation algorithms for location-based services. IEEE International Conference on Communications (ICC), 957-962, 2014.

[20] X. Liu, K. Liu, L. Guo, et al. A game-theoretic approach for achieving k-anonymity in location based- services, IEEE INFOCOM, 2985-2993, 2013.

[21] B. Niu , X. Zhu, H. Chi , H. Li. 3PLUS: Privacy-preserving pseudo-location updating system in location-based services. IEEE Wireless Communications and Networking Conference (WCNC), 4564-4569, 2013.

[22] R. Schlegel, C. Y. Chow, Q. Huang, et al. User-defined privacy grid system for continuous location-based services. IEEE Transactions on Mobile Computing, 14(10), 2158-2172, 2015.

[23] N. Eagle, A. Pentland. Social serendipity: Mobilizing social software. IEEE Pervasive Computing, 4(2), 28-34, 2005.

[24] K. Puttaswamy, B. Zhao. Preserving privacy in location-based mobile social applications. The 11th Workshop on Mobile Computing Systems & Applications, 1-6, 2010.

[25] G. Zhong, I. Goldberg, et al. Louis, Lester and Pierre: Three protocols for location privacy. Privacy Enhancing Technologies, 62-76, 2007.

[26] E. Novak, Q. li. Near-pri: Private, proximity based location sharing. IEEE INFOCOM, 37-45, 2014.

[27] R. Schlegel, C. Chow, Q. Huang, et al. Privacy-preserving location sharing services for social networks. IEEE Transactions on Services Computing, 1-14, 2016.

[28] S. Papadopoulos, S. Bakiras, D. Papadias. Nearest neighbor search with strong location privacy. Proceedings of the VLDB Endowment, 619-629, 2010.

[29] L. Šikšnys, J. R. Thomsen, S. Šaltenis, M. L. Yiu. Private and flexible proximity detection in mobile social networks. The 11th International Conference on Mobile Data Management (MDM), 75-84, 2010.

[30] L. Cox, A. Dalton, V. Marupadi. SmokeScreen: Flexible privacy controls for presence-sharing. The 5th International Conference on Mobile Systems, Applications and Services, 233-245, 2007.