PAPER

Towards Privacy-Preserving Location Sharing over Mobile Online Social Networks

Juan CHEN[†a)], ***Nonmember***, **Shen SU**[†b)], ***Member***, ***and*** **Xianzhi WANG**[††c)], ***Nonmember***

SUMMARY Location sharing services have recently gained momentum over mobile online social networks (mOSNs), seeing the increasing popularity of GPS-capable mobile devices such as smart phones. Despite the convenience brought by location sharing, there comes severe privacy risks. Though many efforts have been made to protect user privacy during location sharing, many of them rely on the extensive deployment of trusted Cellular Towers (CTs) and some incur excessive time overhead. More importantly, little research so far can support complete privacy including location privacy, identity privacy and social relation privacy. We propose SAM, a new System Architecture for mOSNs, and P^3S, a Privacy-Preserving Protocol based on SAM, to address the above issues for privacy-preserving location sharing over mOSNs. SAM and P^3S differ from previous work in providing complete privacy for location sharing services over mOSNs. Theoretical analysis and extensive experimental results demonstrate the feasibility and efficiency of the proposed system and protocol.

key words: *privacy-preserving protocol, location sharing, mOSNs, system architecture*

1. Introduction

Mobile online social networks (mOSNs) are widely used in various applications to support next generation social networks that provide easy accessibility and location-aware services [1].

With the wide adoption of GPS-capable mobile devices, people can now easily exchange ideas, current statuses, and locations with their friends at real-time through mOSNs. The technological shift of the mobile Internet from the traditional 2nd generation network (2G) to the faster 3rd and 4th generation networks (3G and 4G) have further motivate tremendous traditional social network applications, such as Weibo[*] and Twitter[**], to transit towards providing location-based service (LBS). Along with this transition process, location sharing, as the fundamental means of enabling people to share their locations with designated friends, has always been a critical building block of implementing LBSs over mOSNs. For example, Foursquare[***] is one of the most popular geosocial service providers that allow users to register their current locations and share their location information with nearby friends or strangers [2]. According to a recent report [3], LBS is envisioned to become an over 10-billion-per-year business by the year 2017.

Despite the convenience brought by location sharing in mOSNs, however, there comes an indispensable risk of privacy. Most location-sharing applications require updating users' location information to provide better services without considering the related privacy issue. This raise the possibility of disclosing users' location information [4]. For example, users usually need to disclose their identity and location information to obtain personalized location-based services. Location privacy violation usually puts users in many potentially unpleasant situations such as unwanted advertisements, spams, or adversary tracking of users' daily life by malicious parties. For example, the disclosed information might reveal a user's private activities such as visiting a bank or going to a hospital and be used for targeted advertisements against users' willingness [5]; a user's trajectory may be inferred by the location server based on the user's identity, which can be analyzed from its social relationship (friends' information). All the above risks suggest an urgent need of protecting user privacy for location-sharing services in mOSNs.

Although the privacy issue has received much attention in recent years regarding location sharing services over mOSNs [6]–[28], many of them rely on the extensive deployment of trusted Cellular Towers (CTs), whose deployment, however, is generally unfeasible in the real world. In addition, some of the previous work incur excessive time overhead, especially in the transmission process. More importantly, most of the previous work protect user privacy from just a single perspective and little research so far can support full-scale privacy. To cover the complete privacy of users in this study, we identify three requirements of the privacy-preserving location sharing over mOSNs:

- ***Location Privacy:*** the current location of a user should not be tracked by any unauthorized parties including service providers.
- ***Identity Privacy:*** users' identities should be completely hidden from the location server.
- ***Social Relationship Privacy:*** users' friends information should be completely hidden from the location server.

We propose a novel system architecture and the corre-

Manuscript received May 23, 2018.
Manuscript revised September 16, 2018.
Manuscript publicized October 18, 2018.

[†]The authors are with the Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006 China.
[††]The author is with the School of Software at University of Technology Sydney, Australia.
a) E-mail: juanchencs@gmail.com
b) E-mail: johnsonhit@163.com
c) E-mail: xianzhi.wang@uts.edu.au

DOI: 10.1587/transinf.2018EDP7187

[*]http://www.weibo.com/
[**]https://twitter.com/?lang=en
[***]https://foursquare.com/

sponding privacy-preserving protocol to address the above challenges for the privacy-preserving location sharing over mOSNs. Distinguishing from the previous work, our approach features a complete privacy protection for location sharing services over mOSNs. In a nutshell, we make the following contributions:

(1) We propose SAM, a new **S**ystem **A**rchitecture for **m**OSNs, to provide the infrastructural support for complete privacy protection. Compared with existing work, our system not only deploys single location server (LS), but also requires no CTs.
(2) We propose P^{3}S, a **P**rivacy-**P**reserving **P**rotocol based on **S**AM, to achieve the complete privacy protection for location sharing services over mOSNs. P^{3}S outperforms the previous work in the following three aspects:
   - P^{3}S provides a high level of location privacy protection. In particular, a fake location generation scheme combining two algorithms has been proposed to avoid generating 'stupid fake locations', i.e., locations where a person rarely goes to? Though a 'stupid fake locations' can easily be identified, none of the previous work can avoid generating 'stupid fake locations'.
   - P^{3}S can provide social relation privacy protection by a social relation concealment scheme based on the bloom filter. Specifically, the SNS adds a user's friend list into the bloom filter, which is then sent to the LS. In this way, LS can filter strangers and find the friends of a user by the bloom filter, without needing to know the exact friend information of the user.
   - P^{3}S supports many more functions. P^{3}S can support not only nearby friends and strangers search which is also supported by existing work, but also support real-time location sharing between friends.
(3) We prove the effectiveness of the proposed approach in preserving the location privacy, identity privacy, and social relationship privacy under the security architecture of SAM. Extensive experimental results demonstrate the feasibility (in terms of both execution efficiency and convenience) of applying the proposed approach on the current mobile devices.

The remainder of this paper is organized as follows. The related work is discussed in Sect. 2. We introduce the SAM architecture and the corresponding threat model in Sect. 3. Section 4 presents P^{3}S, the privacy-preserving protocol in detail, followed by the security analysis in Sect. 5. Section 6 reports the experimental results, and finally, we give some concluding remarks in Sect. 7.

2. Related Work

Privacy protection has received tremendous attention in recent years for mOSNs, such as privacy strengthening during propagation [29] and privacy-preserving text analysis [30]. As location-sharing becomes an increasingly significant service, especially in the mobile online social network, the privacy issue caused by location-sharing has developed into a devil of a tricky problem. In order to address this issue, many researches about information privacy [31], [32] and location privacy protection [33], [34] have been done. There are a number of works focusing on preventing the location server from learning users' locations when users access the location-based services. The K-anonymity [11], [12], the mix zones [13], [14], the pseudonym methods [15], [16], the m-unobservability [17] and the location anonymity [18], [19] are the typical solutions. Moreover, some people investigated privacy preservation policies about location privacy in distributed social network [20], [21]. Besides, in order to defend against various inference attacks based on differential privacy, Xiao et al. [22] have presented a systematic solution to preserve location privacy. And Sun et al. [23] have introduced a location-label based approach for location-aware location privacy protection problem. Aiming at the same problem, Wang et al. presented [24] several efficient heuristics. In addition, Li et al. [25], [26] and Rahman et al. [27] have proposed privacy context obfuscation to obscure location information based on data requester, time of day, and so on.

In 2007, earlier researchers Cox et al. [28] introduced a representative mechanism for location sharing between both trusted social friends and untrusted strangers. Subsequently, an improved system MobiShare [6] provides flexible privacy protection for location sharing services in mOSNs by accessing users' location information through a location server. This design allows users to share their location information but meanwhile avoids both SNS providers and the location server from having the complete knowledge of a user's identity and location. However, Mobishare relies on the extensive deployment of trusted CTs, which can be difficult to implement in the real world. Also, Mobishare cannot achieve the identity privacy as it cannot prevent the location server from linking the queries from the same user to extract sensitive information. N-MobiShare [7] improves Mobishare by assuming CTs as non-core components of the system and forwarding update requests of user locations to the location server instead of CTs to avoid the extensive deployment of CTs. This approach, however, cannot preserve social relationship privacy as location server can obtain a user's friend list in the query phrase. MobiShare+ [8] aims to address the social relationship privacy by building n-degree polynomials to filter users' real friends. However, it is still possible for the location server to obtain users' friend lists because the fake identifiers of a user are already indicated by the common set of zero points in the corresponding n-degree polynomial. In addition, MobiShare+ incurs excessive time overhead, especially during the transmission process.

To improve transmission efficiency, BMobiShare [9] adopted Bloom Filter to replace the private set intersection protocol in MobiShare+. Whenever a user queries a nearby friend' location in BMobiShare, the user sent a real query along with k-1 dummy queries to the SNS. Since the final result set returned to the user is the merge of the k result sets that satisfy respectively the totally k queries, the final result set actually includes the real user's friends located both around the real user and around the k-1 dummy users and therefore is inaccurate. Recently, a new architecture that employs with multiple location servers [10] is proposed to support the location sharing between friends and strangers in location-based applications. The introduction of multiple location servers lowers the degree of knowledge of each individual server on a user, but at the same time, adds to the difficulty of location privacy protection by making it possible for the location servers to collude, which is likely to happen typically when these servers are owned by the same service provider.

To summarize, though many efforts have been made to protect user's privacy for the location sharing in mOSNs, none of the previous work can protect the complete privacy of users with respect to the location privacy, identity privacy, and social relationship privacy at the same time. That forms the major motivation of our study in this paper.

3. System Architecture and Threat Model

In this section, we describe the system architecture and threat model as the foundation of our approach. A summary of the notations used in this paper is listed in Table 1.

3.1 System Architecture

The basic components of SAM (Fig. 1) include mobile users, a social network server (SNS), and a location server (LS), where SNS and LS store users' profile (including user identities and social relations such as friend lists) and location information, respectively. The separate storage of users' profile and location information provides the basic support for preventing the linkage between the two types of information, i.e., SNS cannot obtain users' real locations while LS cannot infer the users' identities and social relations. We introduce each component of SAM as follows:

- *Mobile users*. Every user can register to the social network with their personal information and obtain their unique identities. They can then communicate with both SNS and LS via GPS-enabled mobile devices to enjoy various location sharing services.
- *SNS*. The social network server is deployed and managed by the social network provider such as Facebook or Weibo. SNS stores the users' personal profiles and protects the user privacy by anonymizing its identity, concealing its social relations, and generating fake locations.
- *LS*. The location sharing server manages user locations and supports the retrieval of nearby locations of users.

We configure the three components as follows to support complete privacy protection:

- Each user, say u, generates its own public/private key pair (puk_u, prk_u). Then, it shares the public key puk_u with SNS and LS, and its symmetric key sk_u, named 'friend key' with its friends. Specifically, two users who are friends, say u and v, exchange their 'friend keys' following these steps: first, SNS sends puk_v to u; then, u encrypts sk_u by puk_v and sends the encrypted key to v through SNS; finally, v decrypts the encrypted key by its private key and obtains sk_u. In a similar way, u can obtain sk_v.
- The SNS is pre-loaded with a public/private key pair (puk_S, prk_S), a hash function H, and a bloom filter B. Then, the SNS shares its puk_S with the LS and all the registered users, followed by its generating the pseudo-IDs and real/fake location tags using H and concealing of the social relations using the bloom filter. In particular, a bloom filter, without having a set of m elements, is a data structure which can efficiently determine whether an element is possibly a member of a set or not. A bloom filter consists of two components: a set of p independent hash functions, $\{h_i()|1 \leq i \leq p\}$, and an array of e bits, $D[e]$, all initialized to 0. Each hash function returns a value that maps to a position in $D[e]$, say y_i, satisfying $y_i < e$. A bloom filter supports two basic operations: *adding* and *query*. To add an element, say $f \in FR$, to a bloom filter, p hash functions are used to generate p indices into the array with the corresponding bits of the array all set to 1. A query is positive if all p referenced bits are 1 while a negative query indicates that the element is not in the bloom filter. Specifically in our system, the SNS uses the bloom filter to record a user's friend set FR and send it to the LS. To check whether an element, say f, exists in FR, LS simply applies the p hash-functions on f to generate p values and concludes that f doesn't belong to FR if any of the p indices in the bit array $D[e]$ is set to 0 (otherwise, f belongs to FR). In this way, the bloom filter allows for the checking of users' friendship relations without letting LS know the exact elements of the friend set FR.
- The LS is pre-loaded its public/private key pair (puk_L, prk_L). Then, LS shares its puk_L with SNS and the user.

3.2 Threat Model

We assume three sources of privacy violation caused by a malicious user, honest but curious SNS, and honest but curious LS, respectively, to form a strong threat model.

- **Malicious user**
  A malicious user usually obtains a target user's identity and location information by performing unauthorized operations. We assume that a malicious user may collude with the SNS or LS while a user's friends who will never collude with the SNS or LS to acquire the user's sensitive information.

Table 1 Notations

Notation	Description
ID_u	Identity of user u
PID_u	Pseudo identity of u
sk_u	A symmetric key, named 'friend key' shared between u and its friends
(puk_u, prk_u)	u's public/private key pair
$E_k(p)$	Encrypt data p by key k
F	u's friend set including its friends' information
df_u	The distance within which u would like to share its location with friends
ds_u	The distance within which u would like to share its location with strangers
sig_u	u's signature
l_u	u's real location
tag_i	The location tag which is used to identify whether the i-th location is real or not.

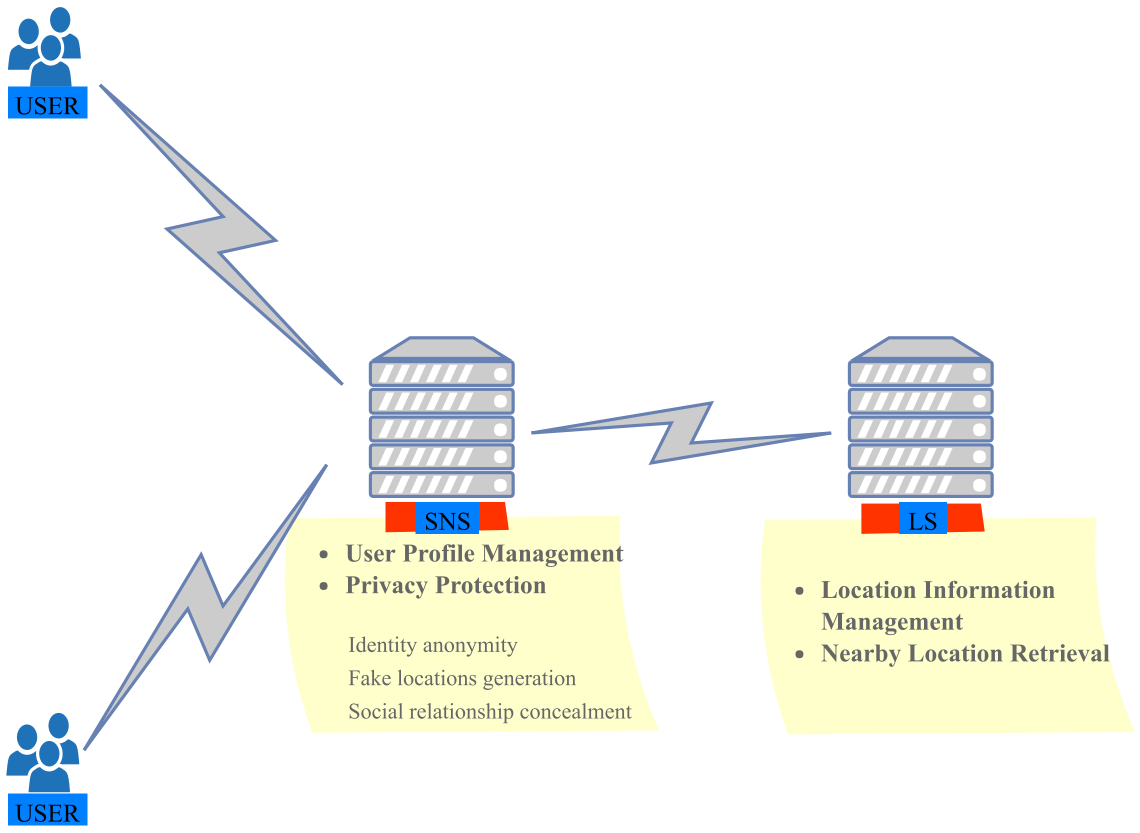

USER
USER
SNS
User Profile Management
Privacy Protection
Identity anonymity
Fake locations generation
Social relationship concealment
LS
Location Information Management
Nearby Location Retrieval

Fig. 1 System architecture.

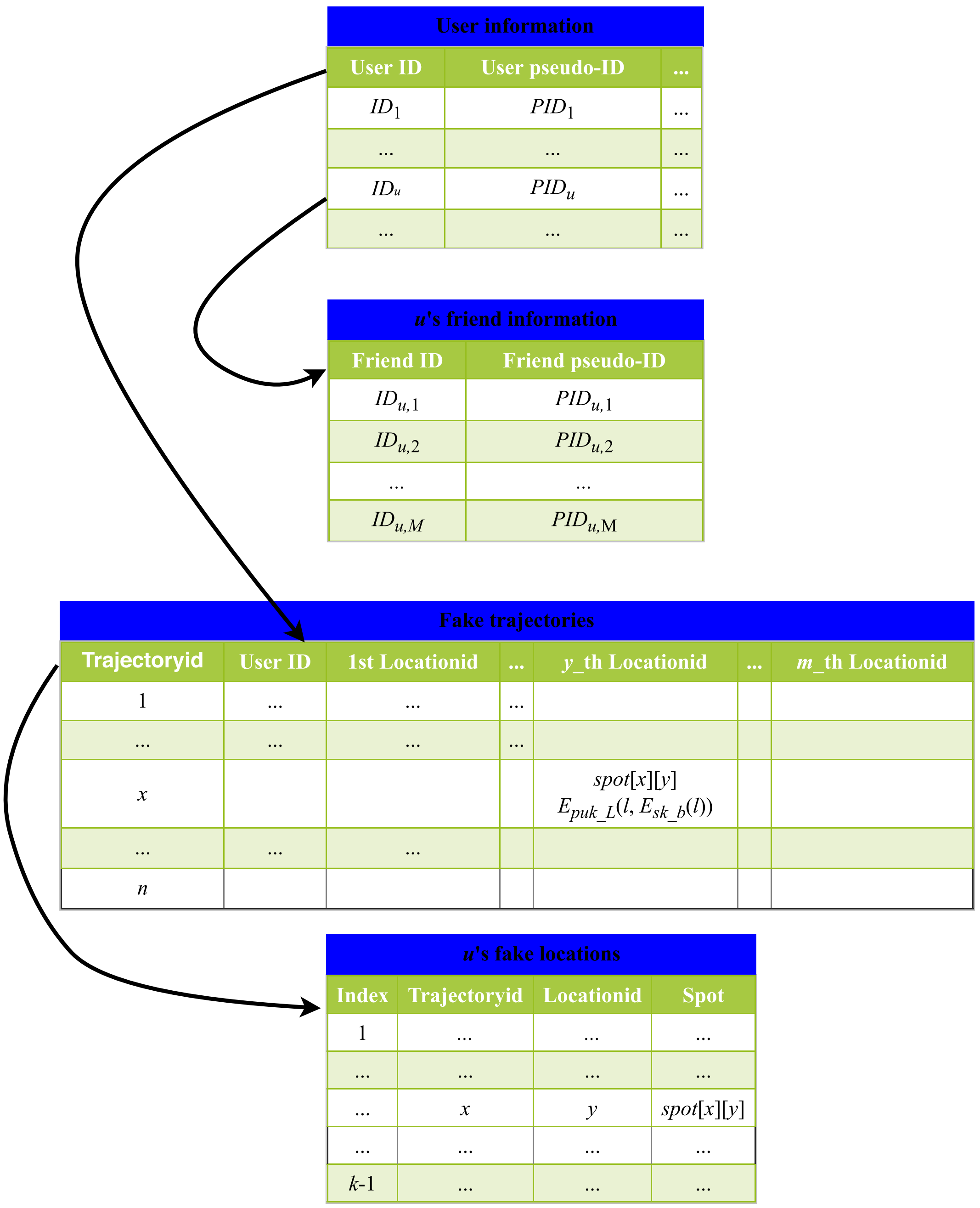
- **Honest but curious SNS**
  We assume the SNS generally follows the protocol but may attempt to obtain users' location information attracted the potential business benefits held by combining user identity and trajectory information. Also, we assume the LS and SNS do not collude as that would make the threat model too strong to be beaten in practice.
- **Honest but curious LS**
  We assume the LS generally follows the protocol but may attempt to obtain users' identity and trajectory for the same reasons as the SNS.

4. Privacy-Preserving Protocol Based on SAM

The privacy-preserving protocol based on SAM (P^3S) aims to manage user identity and social relation information separately by SNS and LS, while prevent them from obtaining user information from each other. P^3S includes five sub-protocols: user registration, location update, nearby friends query, nearby strangers query, and real-time location sharing.

4.1 User Registration

Before using a location sharing service, a mobile user, say u, has to register by providing its personal profile and friend



User information

User ID	User pseudo-ID	...
ID_1	PID_1	...
...	...	...
ID_u	PID_u	...
...	...	...

u's friend information

Friend ID	Friend pseudo-ID
$ID_{u,1}$	$PID_{u,1}$
$ID_{u,2}$	$PID_{u,2}$
...	...
$ID_{u,M}$	$PID_{u,\mathrm{M}}$

Fake trajectories

Trajectoryid	User ID	1st Locationid	...	*y*_th Locationid	...	*m*_th Locationid
1	...	...	...			
...	...	...	...			
x				$spot[x][y]$ $E_{puk\_L}(l, E_{sk\_b}(l))$		
...	...	...				
n						

u's fake locations

Index	Trajectoryid	Locationid	Spot
1	...	...	...
...	...	...	...
...	*x*	*y*	$spot[x][y]$
...	...	...	...
k-1	...	...	...

Fig. 2 Data storage structure of SNS.

information to the SNS. The registration process (Fig. 3) is as follows:

1. User u sends a registration request to the SNS.
2. SNS replies to u by returning a message $< UR, ID_u, puk\_S >$, where UR denotes the message type, ID_u is a unique ID assigned by the SNS to u.
3. u sends a message $< UR, puk\_u, FR, df_u, ds_u >$ to the SNS, where $FR = \{ID_{u,i} | 1 \leq i \leq M\}$ is the friend set of u, M is the total number of friends of u, $ID_{u,i}$ is the ID of the i-th friend of u, df_u and ds_u are the distances within which u would like to share its locations with friends and strangers, respectively.
4. u exchanges its 'friend key' with all its friends by the method introduced in Sect. 3.1.
5. SNS inserts u's personal profile into the user information table (Fig. 2) and friend information into u's friend information table (Fig. 2).

4.2 Location Update

The location update sub-protocol aims to conceal users' real identities from the LS, which might infer their identities based on their locations or trajectory information, whenever

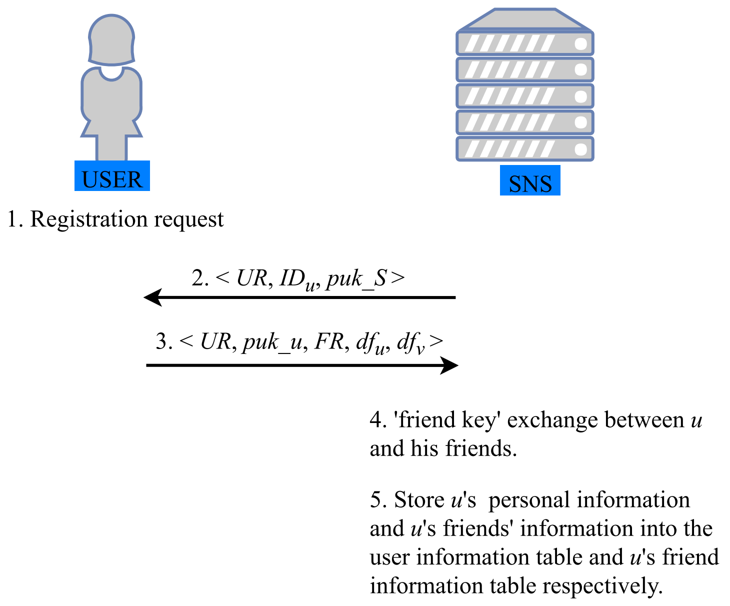

USER
SNS
1. Registration request
2. < UR, ID_u, puk_S >
3. < UR, puk_u, FR, df_u, df_v >
4. 'friend key' exchange between u and his friends.
5. Store u's personal information and u's friends' information into the user information table and u's friend information table respectively.

Fig. 3 User registration.

Algorithm 1 Fake Trajectories Generation

```
Input: $T_p, spot$    // $T_p$ (See Fig. 2) is the fake trajectories table. $spot$ is $u$'s current location.
Output: $T_p$
1: $x$=Trajectoryid($T_p$)    // Trajectoryid($T_p$) returns the number of trajectories in $T_p$
2: $y$=Locationid($ID_u, T_p$)    // Locationid($ID_u, T_p$) is used to find whether part of $u$'s history trajectory is recorded in $T_p$. If $u$ exists in $T_p$, Locationid($ID_u, T_p$) returns the number of $u$'s history locations recorded in $T_p$. If $u$ does not exist in $T_p$, Locationid($ID_u, T_p$)=-1
3: if $x < n$ then
4:     Insert($ID_u, spot, T_p$)    // Insert $u$'s location to $T_p$
5: end if
6: return $T_p$
```

users need to update their location information in the LS (e.g., when they move to new places).

The location update sub-protocol works as follows: first, users encrypt and send their current locations to the SNS; Then, the SNS anonymizes user identities; Finally, the SNS generates k-1 fake locations and sends k locations (including one real and $k-1$ fake ones) to the LS to mislead the LS. The above procedure still faces three challenges:

a) How to prevent the SNS from inferring the users' real locations?

b) How to avoid generating 'stupid fake locations', i.e., locations where a person rarely goes to? Though 'stupid fake locations' can easily be identified, none of the previous work can avoid generating 'stupid fake locations'.

c) How to avoid generating 'stupid fake trajectories', which include 'stupid fake locations' and cannot fit user movements in the real world? Similar to 'stupid fake locations', the previous work can identify yet cannot prevent 'stupid fake trajectories' from being

Algorithm 2 Fake Locations Generation

```
Input: $T_p, F[k-1]$    // $T_p$ is the fake trajectories table (See Fig. 2). $F[k-1]$ is used to store the information about the $k-1$ fake locations. Specifically, $F[i].trajectoryid$, $F[i].locationid$ and $F[i].spot$ are the trajectoryid (See Fig. 2), locationid and the $i$-th fake location respectively, where $0 \le i < k-1$.
Output: $F[k-1]$
1: if $Empty(F[k-1])$ then    // $u$ updates its location for the first time
2:     for $i = 0; i < k-1; i++$ do    // Choose $k-1$ different fake trajectories randomly from table $T_p$.
3:         $a = 1$
4:         while $a == 1$ do
5:             $x = rand(1, n)$
6:             $a = 0$
7:             for $j = 0; j < i; j++$ do    // Make sure that the $x$-th fake trajectory has never been selected before.
8:                 if $F[j].trajectoryid == x$ then
9:                     $a = 1$
10:                    break
11:                end if
12:            end for
13:        end while
14:        $F[i].trajectoryid = x$    // Choose a location randomly from the $x$-th fake trajectory in table $T_p$.
15:        $y = rand(1, m)$
16:        $F[i].locationid = y$
17:        $F[i].spot$ =GetLocation($x, y, T_p$)    // GetLocation($x, y, T_p$) returns the location whose trajectoryid and locationid are $x$ and $y$ respectively in table $T_p$.
18:    end for
19: else
20:    for $i = 0; i < k-1; i++$ do    // Update $k-1$ fake locations for $u$.
21:        $x = F[i].trajectoryid$
22:        $y = mod[(F[i].locationid + 1)/m]$    // Choose the $i$-th fake location from the same fake trajectory one by one in order
23:        $F[i].locationid = y$
24:        $F[i].spot$ =GetLocation($x, y, T_p$)
25:    end for
26: end if
27: return $F[k-1]$
```

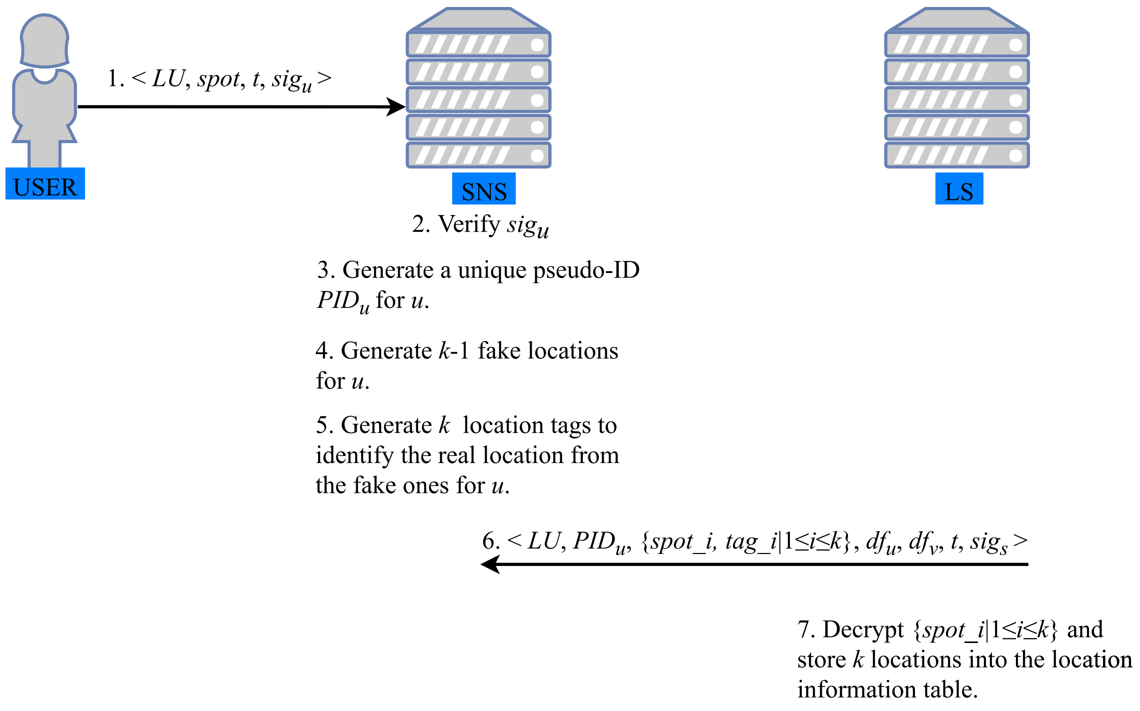

USER
SNS
LS
1. < LU, spot, t, sig_u >
2. Verify sig_u
3. Generate a unique pseudo-ID PID_u for u.
4. Generate k-1 fake locations for u.
5. Generate k location tags to identify the real location from the fake ones for u.
6. < LU, PID_u, {spot_i, tag_i|1≤i≤k}, df_u, df_v, t, sig_s >
7. Decrypt {spot_i|1≤i≤k} and store k locations into the location information table.

Fig. 4 Location update.

generated.

The location update sub-protocol address the above challenges as follows: 1) each user u needs to send encrypted locations, say *spot*, instead of its original locations, say l_u, to the SNS, where $spot = E_{puk\_L}(l_u, E_{sk\_u}(l_u))$, to address challenge **a**); 2) a **F**ake **LO**cation gene**R**a**T**ion scheme named FLORT is employed to to deal with challenge **b)** and **c**). FLORT generates fake locations based on the real trajectories of users—it first records users' historical trajectories and generates a fake trajectories table (Fig. 2) with n fake trajectories; and then the SNS selects $k-1$ trajectories from the table by Algorithm 1; finally, the SNS generates $k-1$ fake locations from the selected trajectories by Algorithm 2.

In particular, a fake trajectories table (Fig. 2) has n rows and m columns, where both n and m are predefined. Each row records a historical trajectory containing m continuous locations of a user. The SNS updates a fix portion of trajectories in the fake trajectories table on a regular basis, e.g., 10 percent of the fake trajectories by Algorithm 1. Fake trajectories are generated in a way that repeats some of the historical trajectories of users in the fake trajectories table—$k-1$ historical trajectories are randomly selected when a user updates its location for the first time, and then for each location update of the user, the next location in each of the $k-1$ historical trajectories is used as a fake location.

The location update sub-protocol is detailed as follows (Fig. 4), which is a 7-step procedure:

1 Once u moves to a new place l_u, it sends to the SNS a location update notification message $< LU, spot, t, sig_u >$, where LU, $spot$, t and sig_u are the message type, encrypted location, timestamp, and user signature, respectively. Specifically, *spot* is of the form $E_{puk\_L}(l_u, E_{sk\_u}(l_u))$, the timestamp is used to defend against replay attack, and the signature is of the form $E_{prk\_u}(ID_u, t)$.
2 SNS verifies sig_u.
3 SNS generates a unique pseudo-ID, $PID_u = H(ID_u \oplus t')$ for u, where t' is the timestamp.
4 SNS generates $k-1$ fake locations using Algorithm 2.
5 SNS generates k location tags, $\{tag\_i|1 \le i \le k\}$, to identify the real location from those fake ones: if $tag\_i = H(ID_u)$, the location related with tag_i is real; if $tag\_i = H(ID_u \oplus i)$, the related location is fake.
6 SNS sends a message $< LU, PID_u, \{spot\_i, tag\_i|1 \le i \le k\}, df_u, ds_u, t, sig_S >$ containing k locations to the LS, where $spot\_i$ and tag_i are the i-th location and its corresponding tag, respectively. Specifically, $spot\_i = E_{puk\_L}(loc\_i, E_{sk\_u}(loc\_i))$ and $sig_S = E_{prk\_S}(PID_u, t)$.
7 By decrypting $\{spot\_i|1 \le i \le k\}$ from the received message, the LS obtains k locations $\{(loc\_i, E_{sk\_u}(loc\_i)|1 \le i \le k\}$ and stores them in Table 2.

Table 2 Location information

User pseudo-ID	Location	Encrypted location	Location tag
PID_u	loc_1	$E_{sk\_u}(loc\_1)$	tag_1
…	……	…	…
PID_u	loc_k	$E_{sk\_u}(loc\_k)$	tag_k
…	……	…	…

Note that the location updates may produce an infinite number of pseudo-IDs as each location update comes with a new pseudo-ID. Therefore, the LS regularly removes old en-

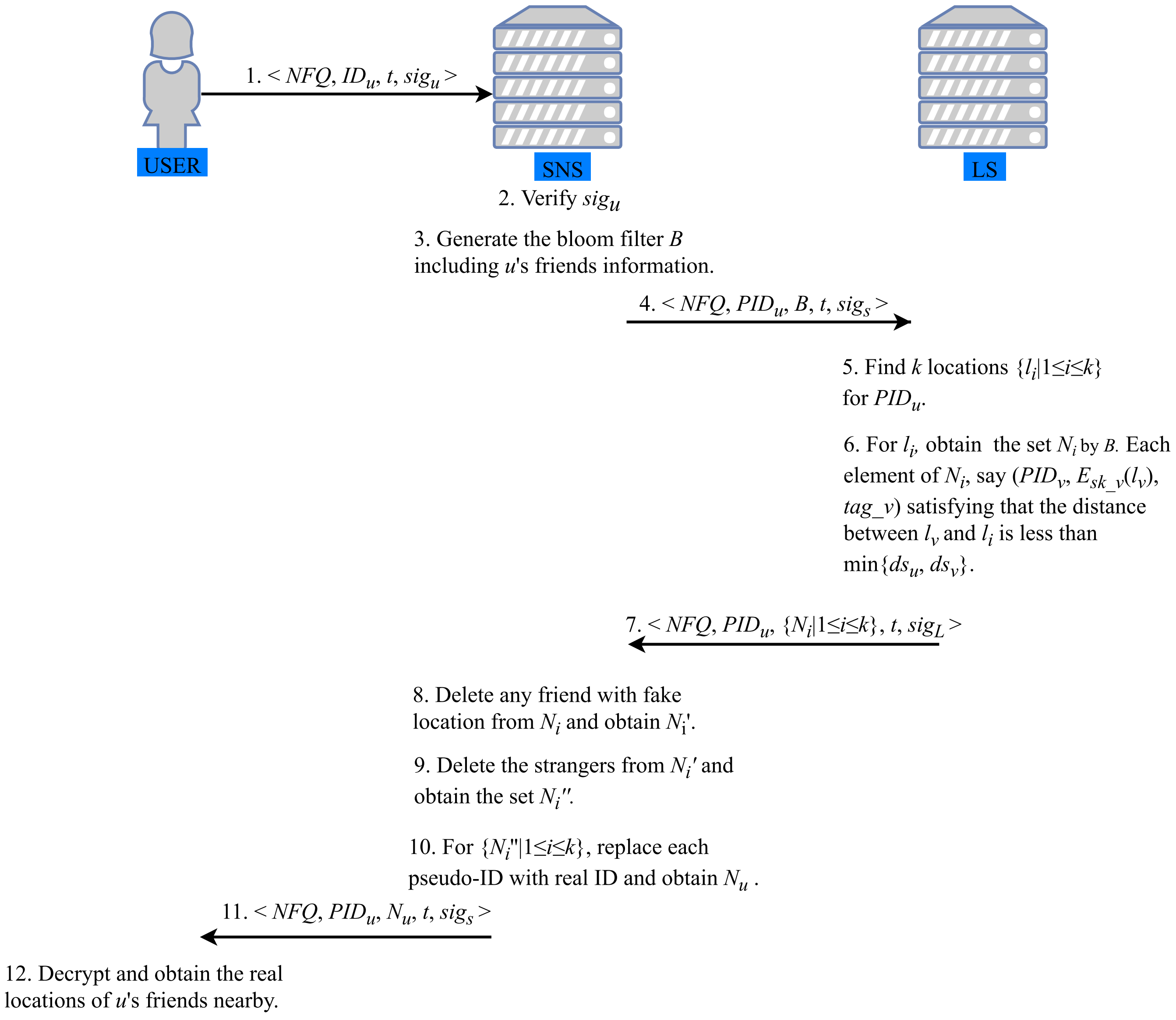

USER
SNS
LS
1. $< NFQ, ID_u, t, sig_u >$
2. Verify sig_u
3. Generate the bloom filter B including u's friends information.
4. $< NFQ, PID_u, B, t, sig_s >$
5. Find k locations $\{l_i|1 \leq i \leq k\}$ for PID_u.
6. For l_i, obtain the set N_i by B. Each element of N_i, say $(PID_v, E_{sk\_v}(l_v), tag\_v)$ satisfying that the distance between l_v and l_i is less than $\min\{ds_u, ds_v\}$.
7. $< NFQ, PID_u, \{N_i|1 \leq i \leq k\}, t, sig_L >$
8. Delete any friend with fake location from N_i and obtain N_i'.
9. Delete the strangers from N_i' and obtain the set N_i''.
10. For $\{N_i''|1 \leq i \leq k\}$, replace each pseudo-ID with real ID and obtain N_u .
11. $< NFQ, PID_u, N_u, t, sig_s >$
12. Decrypt and obtain the real locations of u's friends nearby.

Fig. 5 Nearby friends query.

tries from the location information table to avoid excessive storage.

4.3 Nearby Friends Query

The nearby friends query sub-protocol aims to complete users' location requests without leaking their privacy. The sub-protocol focuses on addressing two challenging issues: 1) how to conceal the user identity and social relations from the LS? 2) how to protect the user locations from the SNS?

Basically, a social relation concealment scheme based on the bloom filter has been proposed to protect the user's friends' information. Specifically, the SNS adds a user's friend list into the bloom filter B, which is then sent to the LS. In this way, LS can filter strangers and find the friends of a user by B, without needing to know the exact friend information of the user.

The nearby friends query sub-protocol (Fig. 5) follows 12 steps to query the nearby friends of a user u:

1. u sends a request message $< NFQ, ID_u, t, sig_u >$ to the SNS, where NQF is the message type.
2. SNS verifies sig_u.
3. SNS generates the bloom filter B that includes the friend information of u.
4. SNS sends a query message $< NFQ, PID_u, B, t, sig_S >$ to the LS.
5. LS retrieves k locations of PID_u, say $\{l_i|1 \leq i \leq k\}$.
6. For each location, say l_i, LS finds u's friends around l_i by B and obtains a set N_i. Each element of N_i is of the form $(PID_v, E_{sk\_v}(l_v), tag\_v)$, satisfying the distance between l_v and l_i is less than $min\{df_u, df_v\}$.
7. LS sends all the nearby friends, $< NFQ, PID_u, \{N_i|1 \leq i \leq k\}, t, sig_L >$, to the SNS, where $sig_L =$

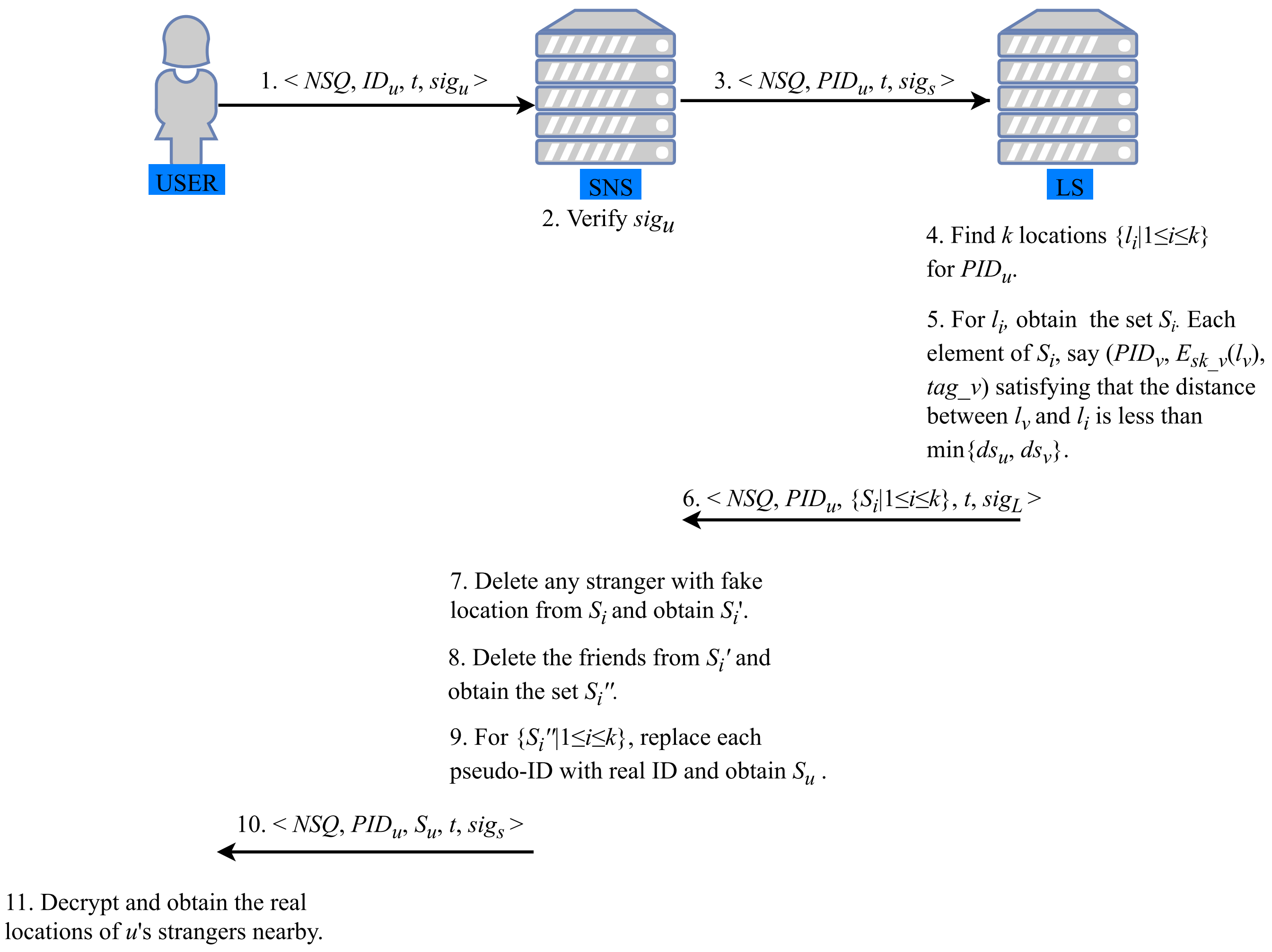

USER
SNS
LS
1. $< NSQ, ID_u, t, sig_u >$
2. Verify sig_u
3. $< NSQ, PID_u, t, sig_s >$
4. Find k locations $\{l_i|1\leq i\leq k\}$ for PID_u.
5. For l_i, obtain the set S_i. Each element of S_i, say (PID_v, $E_{sk\_v}(l_v)$, tag_v) satisfying that the distance between l_v and l_i is less than $\min\{ds_u, ds_v\}$.
6. $< NSQ, PID_u, \{S_i|1\leq i\leq k\}, t, sig_L >$
7. Delete any stranger with fake location from S_i and obtain S_i'.
8. Delete the friends from S_i' and obtain the set S_i''.
9. For $\{S_i''|1\leq i\leq k\}$, replace each pseudo-ID with real ID and obtain S_u .
10. $< NSQ, PID_u, S_u, t, sig_s >$
11. Decrypt and obtain the real locations of u's strangers nearby.

Fig. 6 Nearby strangers query.

$E_{prk\_L}(PID_u, t)$.

8. SNS deletes the elements with fake locations from N_i and obtains $N_i^{'}$.
9. SNS deletes the erroneous results (i.e., strangers) from $N_i^{'}$ according to u's friend information table (Fig. 2) and thereby obtains the real friend set $N_i^{''}$.
10. SNS replaces each pseudo-ID in $\{N_i^{''}|1 \leq i \leq k\}$ with the corresponding real ID and thereby obtains $N_u = \{ID_j, E_{sk\_j}(l_j)|1 \leq j \leq q\}$, where q is the number of u's nearby friends.
11. SNS sends N_u to u.
12. u decrypts N_u and obtains the real locations of its friends nearby.

4.4 Nearby Strangers Query

The nearby strangers query sub-protocol (Fig. 6) performs the following steps to find the strangers around a user while protecting its privacy:

1. u sends a request message $< NSQ, ID_u, t, sig_u >$ to the SNS, where NSQ is the message type.
2. SNS verifies sig_u.
3. SNS sends a request message $< NSQ, PID_u, t, sig_S >$ to the LS.
4. LS finds k locations of PID_u, say $\{l_i|1 \leq i \leq k\}$.
5. For each location l_i, LS finds the people around l_i and obtains a set S_i. Each element of S_i is of the form $(PID_v, E_{sk\_v}(l_v), tag\_v)$, satisfying the distance between l_v and l_i is less than $min\{ds_u, ds_v\}$.
6. LS sends $< NSQ, PID_u, \{S_i|1 \leq i \leq k\}, t, sig_L >$ that contains strangers around u to the SNS.
7. SNS deletes all the people with fake locations from S_i and obtains $S_i^{'}$.
8. SNS deletes u's friends from $S_i^{'}$ according to u's friend information table (Fig. 2) and obtains the stranger set $S_i^{''}$.
9. For each pseudo-ID in $\{S_i^{''}|1 \leq i \leq k\}$, SNS replaces it with the corresponding real ID and thereby obtains $S_u = \{ID_j, E_{puk\_u}(l_j)|1 \leq j \leq Q\}$, where Q is the number of u's nearby strangers.
10. SNS sends S_u to u.
11. u decrypts S_u and obtains the real locations of its nearby strangers.

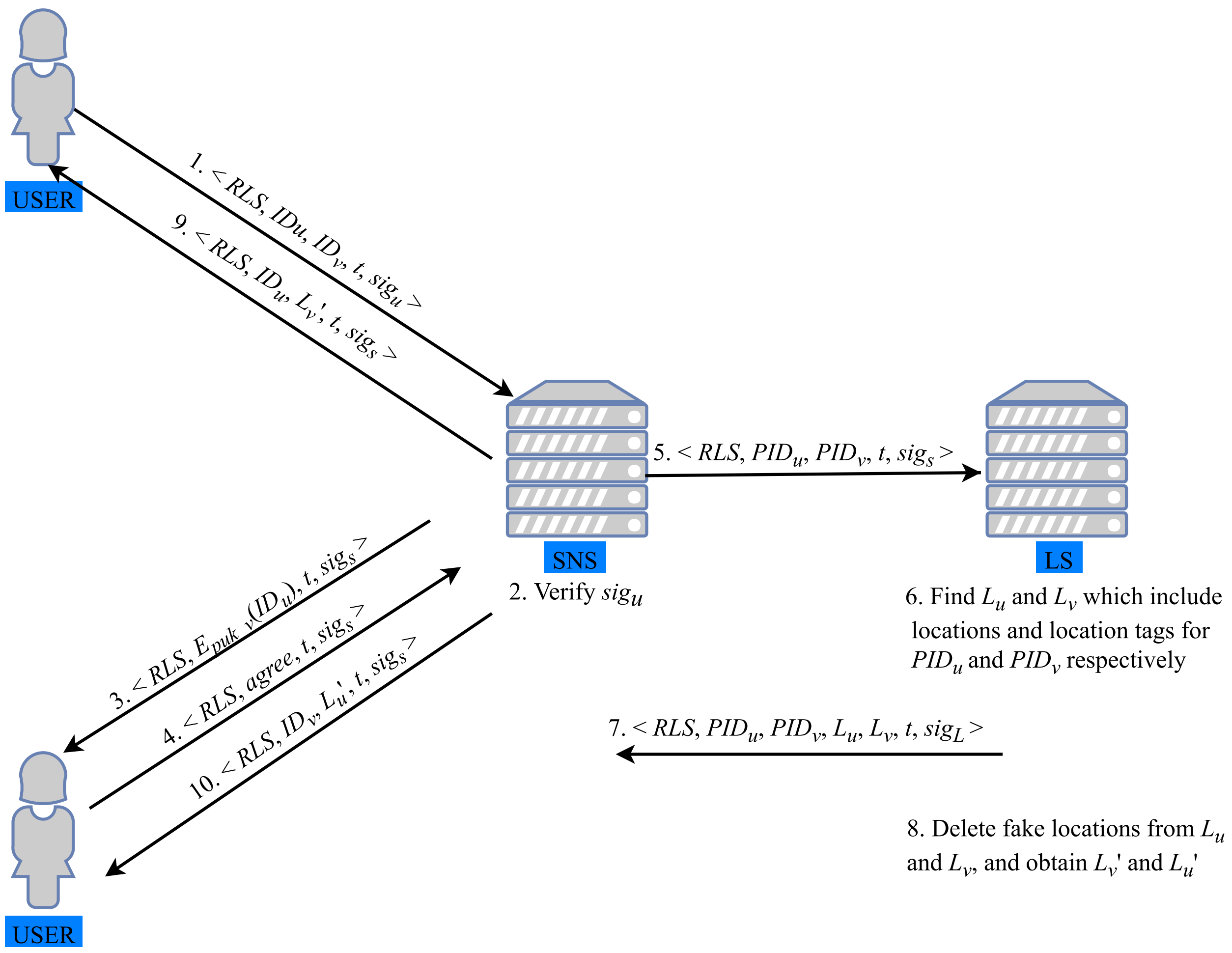

USER
1. < RLS, IDu, IDv, t, sigu >
9. < RLS, IDu, Lv', t, sigs >
SNS
2. Verify sigu
5. < RLS, PIDu, PIDv, t, sigs >
LS
6. Find Lu and Lv which include locations and location tags for PIDu and PIDv respectively
7. < RLS, PIDu, PIDv, Lu, Lv, t, sigL >
8. Delete fake locations from Lu and Lv, and obtain Lv' and Lu'
3. < RLS, Epuk_v(IDu), t, sigs >
4. < RLS, agree, t, sigs >
10. < RLS, IDv, Lu', t, sigs >
USER

Fig. 7 Real-time location sharing.

4.5 Real-Time Location Sharing

The real-time location sharing sub-protocol (Fig. 7) performs the following steps to complete the real-time location sharing between two friends (e.g., u and v):

1. u sends a request message $< RLS, ID_u, ID_v, t, sig_u >$ to the SNS, where LS is the message type.
2. SNS verifies sig_u.
3. SNS sends a request message $< RLS, E_{puk\_v}(ID_u), t, sig_s >$ to v.
4. v responds to the SNS with a message $< RLS, agree, t, sig_v >$, where $sig_v = E_{prk\_v}(agree, t)$, if it agrees to share its location in real-time with u.
5. SNS sends a message $< RLS, PID_u, PID_v, t, sig_s >$ to the LS to request for the locations of u and v.
6. LS obtains L_u and L_v from Table 2, where L_u and L_v include k encrypted locations and location tags of PID_u and PID_v, respectively.
7. LS sends L_u and L_v to the SNS.
8. SNS deletes the fake locations in L_u and L_v and thereby obtains u's real location $L_u^{'} = E_{sk\_u}(l_u)$ and v's real location $L_v^{'} = E_{sk\_v}(l_v)$.
9. SNS sends $L_v^{'}$ to u.
10. SNS sends $L_u^{'}$ to v.

Note that some operations may still need to be performed during the real-time location sharing process between two friends (e.g., u and v): once a person u changes its location, the SNS needs to perform the location update sub-protocol (introduced in Sect. 4.2), send a location request message to the LS, and return the result to v. Specially, no location will be sent to any of the two people by the SNS when a person stops sharing its real-time location with a friend.

5. Security Analysis

We analyse the security performance of SAM and P^{3}S from three aspects: location privacy, identity privacy and social relationship privacy.

5.1 Location Privacy

The proposed approach can protect users' location privacy for the following reasons:

- SNS cannot obtain the users' real locations for two reasons: 1) each location sent by a user to the SNS is encrypted by the public key of the LS; 2) each location sent by the LS to the SNS is encrypted by a user's 'friend key'.
- Without the location tag, the LS cannot identify a user's real location from the fake ones for two reasons: 1) each location tag is computed by a user's real ID, which the LS does not know; 2) it is difficult for the LS to infer the real location from $k-1$ fake locations as fake locations are generated from real historical trajectories in P^3S.
- A malicious user cannot obtain the target user's real locations even if it colludes with either the SNS or the LS— the malicious user can only obtain the locations encrypted by the target user's 'friend key' rather than the target user's original locations through the collusion with the SNS; similarly, it can neither identify the target user nor obtain the user's real location from the LS's databases (which requires knowing the user's real ID) through collusion with the LS.

5.2 Identity Privacy and Social Relationship Privacy

The proposed approach can protect users' identity privacy and social relationship privacy for the following reasons:

- LS cannot obtain a user's identity and social relations for two reasons: 1) LS cannot infer a user's real ID, as each location query sent to the LS uses different pseudo-IDs even for the same users; 2) LS cannot obtain a user's social relations as the LS uses the bloom filter rather than the real friend list to find the user's friends.
- A malicious user cannot infer the target user's ID through collusion with the LS, even though they can find the user's pseudo-IDs according to the target user's real locations.

6. Implementation and Evaluation

We examined the acceptability and feasibility of P^3S on mobile devices, which are generally more resource-constrained when compared with wired devices. The cryptographic operations performed on mobile phones in P^3S involve symmetric encryption and decryption, asymmetric encryption and decryption, signature, and verification. Therefore, we chose AES with 128-bit keys in CBC mode for the symmetric cryptography and RSA with 2048-bit length keys for both asymmetric cryptography and signature. All experiments are executed on Huawei NEM-AL10 smart phone running Android 6.0 operation system with a 2.0GHZ CPU. Specifically, AES, RSA and signature is implemented by the android cryptography class javax.crypto.cipher which is provided by Android SDK. We obtain the location of the mobile device by the getLastKnownLocation() function provided by the LocationManager class.

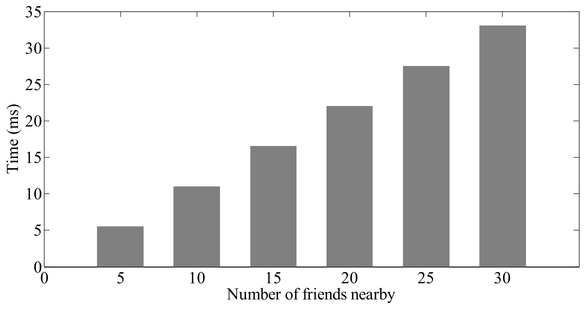


Fig. 8 AES decryption time

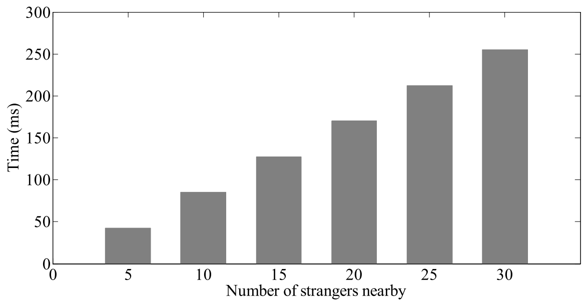


Fig. 9 RSA decryption time

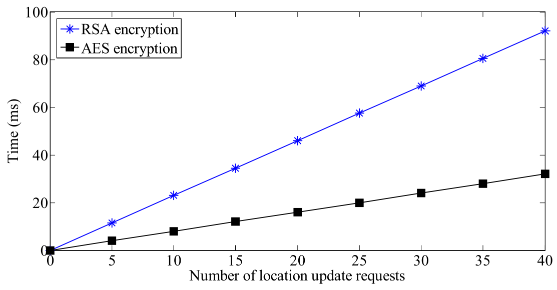


Fig. 10 Encryption execution time

Figures 8 and 9 show the average time for data decryption with AES and RSA, respectively. The results in Fig. 8 show that AES takes more time on decryption as the number of nearby friends of a user grows. Given 30 friends around the user, we observed no more than $35ms$ taken by AES, which is acceptable within current mobile environment. Similarly, we observed from Fig. 9 that RSA took a longer time for decryption as the number of strangers nearby increased. When the number of strangers around the user was 30, the time cost for RSA was still less than $300ms$.

We also tested the average time for encrypting data with AES and RSA (Fig. 10). The results show that the encryption time required by RSA and AES increases as the number of location update requests grows, and RSA takes more time to encrypt data than AES. On the other hand, the performance of both encryption methods is acceptable, with both RSA and AES take no more than $100ms$ to encrypt data when there are as many as 40 location update requests from a single person.

We further studied the signing and verifying operations on a 128-bytes data set by experiments. As we can see from Fig. 11, the time cost for signature and verification grows as the number of 128-bytes data increases. Figure 11 also indicates the average time needed by one-time signature and verification is about 2.7*ms* and 0.8*ms* which is acceptable for most mobile phones.

Figure 12 shows the time needed by a user to exchange his friend key with his friends' friend keys. It is observed that the execution time increases almost linearly with the number of the user's friends grows. However, even that the number of friends is as high as 100, the execution time is less than 1s. Since the friend keys' exchange process is one-time, the execution time is acceptable.

We compare our system with three typical location-sharing systems for mOSN: Mobishare in 2012 [6], N-Mobishare in 2014 [7], Multi-location servers in 2015 [10] and our system. Table 3 lists the performances of different systems.

- **About the cellular towers**
  Different from Mobishare, our system and the other two typical systems are more flexible since they do not need cellular towers.
- **About the mobile device performance**
  Mobile devices in the four systems perform similar operations such as location computation, data encryption and decryption operation.
- **About the SNS requirements** Compared with Mobishare and N-Mobishare, our system requires a more powerful social network server to generate fake locations and conceal the social relationship by bloom filter. However, the Multi-location server system deploys the most powerful social network server since it encrypts much more locations which will then be sent to multiple location server for privacy protection consideration.
- **About the number of Location Servers** Mobishare, N-Mobishare and our system store all of the locations into single server. On the contrary, the multi-location server system stores locations to multi-servers.
- **About security performance**
  Our system provides a higher level of security than other systems. Specifically, our system can achieve the location privacy, identity privacy and social relationship privacy while others cannot.
- **About system functions**
  Our system supports many more functions than the others. Specifically, both our system and the other three systems can search nearby friends' and strangers' locations. Furthermore, our system can support real-time location sharing between friends.

The above experimental and evaluation results suggest that the presented P^3S protocol under the SAM system provides a higher level of security and supports many more functions than the others. Furthermore, our system is feasible and efficient for use in a general mobile devices.

7. Conclusion

In this paper, we have proposed SAM, a new system architecture for mOSNs, and a corresponding privacy-preserving protocol based on SAM (P^3S), as the overall solution to the privacy protection issue during the location sharing over mOSNs. Distinguishing from previous approaches, our solution features a full-scale privacy protection of users that covers location privacy, identity privacy, and social relationship privacy under the strong security architecture of SAM. Extensive experimental results demonstrate the feasibility and effectiveness of the proposed approach on the state-of-the-art mobile devices.

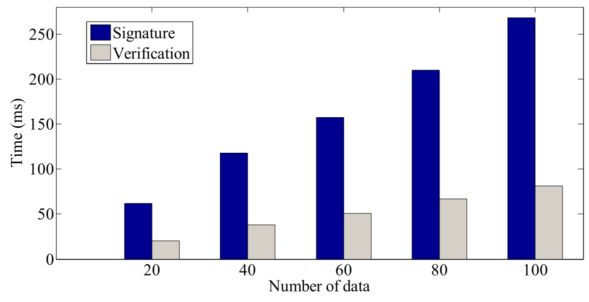


Fig. 11 Execution time for signature and verification

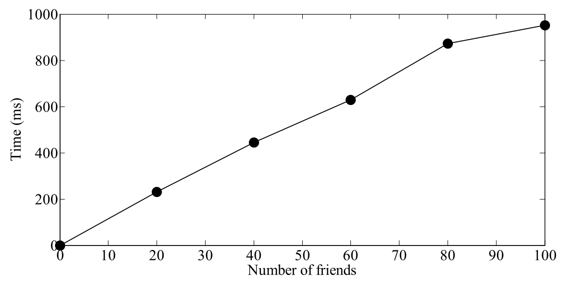


Fig. 12 Execution time for 'friend key' exchange

Table 3 Performance comparison of different systems

Items	**Mobishare** [6]	**N-Mobishare** [7]	**Multi-location server** [10]	**Our system**
Cellular Tower	*Need*	*No*	*No*	*No*
Mobile Device	*Similar*	*Similar*	*Similar*	*Similar*
SNS Hardware Requirements	*Low*	*Low*	*High*	*Middle*
Number of Location Servers	*Single*	*Single*	*Multiple*	*Single*
Security	*Middle*	*Low*	*Low*	*High*
System function	*General*	*General*	*General*	*Powerful*

Acknowledgments

This research is supported in part by the Natural Science Foundation of China under grants No. 61572153 and No. 61702223; by Scientific Research Staring Foundation for the Ph.D. in Liaoning Province No. 201601081; by Scientific Research Projects from Education Department in Liaoning Province No. L2015056.

References

[1] H. Li, H. Zhu, S. Du, X. Liang, and X. Shen, "Privacy leakage of location sharing in mobile social networks: Attacks and defense," IEEE Transactions on Dependable and Secure Computing, 2016.

[2] J. Son, D. Kim, R. Tashakkori, A.O. Tokuta, and H. Oh, "A new mobile online social network based location sharing with enhanced privacy protection," 2016 25th International Conference on Computer Communication and Networks (ICCCN), pp.1–9, IEEE, 2016.

[3] P. Nitesh, "The10 billion rule: location, location, location." http://www.strategyanalytics.com/default.aspx?mod=reportabstractviewer&a0=6355.

[4] K.G. Shin, X. Ju, Z. Chen, and X. Hu, "Privacy protection for users of location-based services," IEEE Wireless Communications, vol.19, no.1, 2012.

[5] W. Karim, "The privacy implications of personal locators: Why you should think twice before voluntarily availing yourself to GPS monitoring," Wash. UJL & Pol'y, vol.14, p.485, 2004.

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

[7] Z. Liu, D. Luo, J. Li, X. Chen, and C. Jia, "N-mobishare: new privacy-preserving location-sharing system for mobile online social networks," International Journal of Computer Mathematics, vol.93, no.2, pp.384–400, 2016.

[8] J. Li, J. Li, X. Chen, Z. Liu, and C. Jia, "Mobishare+: Security improved system for location sharing in mobile online social networks," Journal of Internet Services and Information Security, vol.4, no.1, pp.25–36, 2014.

[9] N. Shen, J. Yang, K. Yuan, C. Fu, and C. Jia, "An efficient and privacy-preserving location sharing mechanism," Computer Standards & Interfaces, vol.44, pp.102–109, 2016.

[10] J. Li, H. Yan, Z. Liu, X. Chen, X. Huang, and D.S. Wong, "Location-sharing systems with enhanced privacy in mobile online social networks," IEEE Syst. J., 2015.

[11] J. Maheswaran, D. Jackowitz, E. Zhai, D.I. Wolinsky, and B. Ford, "Building privacy-preserving cryptographic credentials from federated online identities," Proc. Sixth ACM Conference on Data and Application Security and Privacy, pp.3–13, ACM, 2016.

[12] J. Maheswaran, D.I. Wolinsky, and B. Ford, "Crypto-book: an architecture for privacy preserving online identities," Proc. Twelfth ACM Workshop on Hot Topics in Networks, p.14, ACM, 2013.

[13] A.R. Beresford and F. Stajano, "Location privacy in pervasive computing," IEEE Pervasive computing, vol.2, no.1, pp.46–55, 2003.

[14] A.R. Beresford and F. Stajano, "Mix zones: User privacy in location-aware services," Proc. Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004, pp.127–131, IEEE, 2004.

[15] Y. Ouyang, Z. Le, Y. Xu, N. Triandopoulos, S. Zhang, J. Ford, and F. Makedon, "Providing anonymity in wireless sensor networks," ICPS, pp.145–148, 2007.

[16] S.M.M. Rahman, A. Inomata, M. Mambo, and E. Okamoto, "Anonymous on-demand position-based routing in mobile ad-hoc networks," Information and Media Technologies, vol.1, no.2, pp.1191–1203, 2006.

[17] Z. Chen, X. Hu, X. Ju, and K.G. Shin, "Lisa: Location information scrambler for privacy protection on smartphones," 2013 IEEE Conference on Communications and Network Security (CNS), pp.296–304, IEEE, 2013.

[18] J. Li and Y. Wang, "Universal designated verifier ring signature (proof) without random oracles," Emerging directions in embedded and ubiquitous computing, pp.332–341, 2006.

[19] S. Rass, R. Wigoutschnigg, and P. Schartner, "Doubly-anonymous crowds: Using secret-sharing to achieve sender- and receiver-anonymity," JoWUA, vol.2, no.4, pp.27–41, 2011.

[20] Y. Wang, D. Xu, and F. Li, "Providing location-aware location privacy protection for mobile location-based services," Tsinghua Science and Technology, vol.21, no.3, pp.243–259, 2016.

[21] M. Li, S. Yu, N. Cao, and W. Lou, "Privacy-preserving distributed profile matching in proximity-based mobile social networks," IEEE Trans. Wireless Commun., vol.12, no.5, pp.2024–2033, 2013.

[22] Y. Xiao and L. Xiong, "Protecting locations with differential privacy under temporal correlations," Proc. 22nd ACM SIGSAC Conference on Computer and Communications Security, pp.1298–1309, ACM, 2015.

[23] G. Sun, D. Liao, H. Li, H. Yu, and V. Chang, "L2p2: A location-label based approach for privacy preserving in lbs," Future Generation Computer Systems, 2016.

[24] S. Veluru, Y. Rahulamathavan, B. Gupta, and M. Rajarajan, "Privacy preserving text analytics: Research challenges and," Handbook of Research on Securing Cloud-Based Databases with Biometric Applications, p.364, 2014.

[25] J. Li, K. Kim, F. Zhang, and X. Chen, "Aggregate proxy signature and verifiably encrypted proxy signature," International Conference on Provable Security, pp.208–217, 2007.

[26] J. Li, F. Zhang, and Y. Wang, "A new hierarchical id-based cryptosystem and cca-secure pke," Emerging directions in embedded and ubiquitous computing, pp.362–371, 2006.

[27] F. Rahman, M.E. Hoque, F.A. Kawsar, and S.I. Ahamed, "Preserve your privacy with pco: A privacy sensitive architecture for context obfuscation for pervasive e-community based applications," 2010 IEEE Second International Conference on Social Computing (SocialCom), pp.41–48, IEEE, 2010.

[28] L.P. Cox, A. Dalton, and V. Marupadi, "Smokescreen: flexible privacy controls for presence-sharing," Proc. 5th international conference on Mobile systems, applications and services, pp.233–245, ACM, 2007.

[29] B. Gupta, D.P. Agrawal, and S. Yamaguchi, Handbook of research on modern cryptographic solutions for computer and cyber security, IGI Global, 2016.

[30] P. Stuedi, I. Mohomed, M. Balakrishnan, Z.M. Mao, V. Ramasubramanian, D. Terry, and T. Wobber, "Contrail: Decentralized and privacy-preserving social networks on smartphones," IEEE Internet Comput., vol.18, no.5, pp.44–51, 2014.

[31] R. Jiang, R. Lu, and K.K.R. Choo, "Achieving high performance and privacy-preserving query over encrypted multidimensional big metering data," Future Generation Computer Systems, 2016.

[32] S. Canard and J. Devigne, "Highly privacy-protecting data sharing in a tree structure," Future Generation Computer Systems, vol.62, pp.119–127, 2016.

[33] X. Ju and K.G. Shin, "Location privacy protection for smartphone users using quadtree entropy maps," Journal of Information Privacy and Security, vol.11, no.2, pp.62–79, 2015.

[34] U.P. Rao and H. Girme, "A novel framework for privacy preserving in location based services," 2015 Fifth International Conference on Advanced Computing & Communication Technologies (ACCT), pp.272–277, IEEE, 2015.

Juan Chen is currently an associate professor in the Cyberspace Institute of Advanced Technology at Guangzhou University, since 2018. Prior to Guangzhou University, Juan Chen served as a lecturer at the School of Computer Science at Dalian Maritime University since 2013. She received the Ph.D. degree in computer science from Harbin Institute of Technology in 2013. Her research interests include wireless network security, deep learning and machine learning for network security.

Shen Su received his Ph.D. degree in computer science and technology from Harbin Institute of Technology, in 2016. Since 2018, he has been a lecturer at Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, China. His research interests include interdomain routing modeling and analysis, sentiment analysis, wireless sensor networks, and service computing.

Xianzhi Wang is a lecturer with the School of Software at University of Technology Sydney. He holds a Ph.D. degree and a M.E. degree, both from Harbin Institute of Technology, and a B.E. degree from Xi'an Jiaotong University, all in Computer Science. His research interests include Internet of Things, machine learning, crowdsourcing, and services computing. He received the ARC Discovery Early Career Researcher Award (DECRA) in 2017 and IBM Ph.D. Fellowship Award in 2013. He is a member of the IEEE and the ACM.