PROS: A Privacy-Preserving Route-Sharing Service via Vehicular Fog Computing

MENG LI¹, (Student Member, IEEE), LIEHUANG ZHU¹*, (MEMBER, IEEE), ZIJIAN ZHANG¹*, (MEMBER, IEEE), XIAOJIANG DU², (SENIOR MEMBER, IEEE), AND MOHSEN GUIZANI³, (Fellow, IEEE)

¹School of Computer Science and Technology, Beijing Institute of Technology (e-mail: {menglibit, liehuangz, zhangzijian}@bit.edu.cn)
²Department of Computer and Information Sciences at Temple University (e-mail: dxj@ieee.org)
³Department of Electrical and Computer Engineering at University of Idaho (e-mail: mguizani@ieee.org)

Co-Corresponding Authors: Liehuang Zhu (e-mail: liehuangz@bit.edu.cn) and Zijian Zhang (email: zhangzijian@bit.edu.cn).

This work is partially supported by China National Key Research and Development Program (Grant No. 2016YFB0800301) and National Natural Science Foundation of China (Grant Nos. 61872041 and 61871037).

ABSTRACT In vehicular networks, route sharing is a novel cloud-computing service, where users form groups with their friends, set up a common destination, and share real-time locations and routes. However, to form these groups, users have to upload their group formation or participation requests to a remote service provider, which consumes a significant amount of network bandwidths and incur an increase in the response delay. Meanwhile, users’ sensitive information, such as identity and location, along with their group privacy such as social graphs are entirely exposed to the service provider, thereby causing critical user privacy threats. Herein, we first introduce fog computing into a route-sharing service model in vehicular networks, where fog nodes preprocess user data. Second, group privacy is defined and three new attacks are considered, namely member impersonation attack, group affiliation attack, and unlimited participation attack, under this model. Third, the privacy-preserving route-sharing (PROS) scheme is proposed to protect user and group privacy by utilizing improved anonymous authentication, rate limiting pseudonyms, modified privacy-preserving equality test, and location geo-indistinguishability. Finally, the security and privacy of the PROS scheme is analyzed and extensive experiments are conducted to compare its efficiency with existing schemes.

INDEX TERMS Vehicular Networks, Route-Sharing Service, Fog Computing, User Privacy, Group Privacy

I. INTRODUCTION

Location-sharing applications [1] such as Google Maps [2], [3] are popular service providers in vehicular networks [4], [5] that are based on location-based service (LBS). These applications can detect the users’ locations using wireless communication mobile devices equipped with GPS modules [6]. These applications can further help the user in finding a highly recommended restaurant or detect a friend’s location. LBS has been combined with MSN, where users share their geo-tagged information with people who are a part of a social network. For example, friends share location-embedded pictures on Facebook [7] and families share real-time locations on Glympse [8].

Recently, Gaode Maps [9], a cloud-computing service that provides detailed geographical information and locations, has launched a novel application, namely Route Sharing [10].

Users in route sharing can be vehicles or pedestrians, and both of them can be a group leader. First, a user forms a group, known as the group leader, and adds several members in the group, who can see each other’s profile photo and location. Then, the group leader sets up a destination for the group, and each member will travel to this rendezvous for a meeting or date, among others, while sharing their real-time locations and routes, as shown in Fig. 1. Route sharing provides extra advantage to users because the users can track the locations of all group members accurately when they are going to the destination. Meanwhile, they can also interact with group members or provide directions to someone who cannot read a map.

However, users need to upload their group formation or participation requests to a remote server [11], [12], which consumes a significant amount of bandwidth and increases
Figure 1: Example of route sharing in Gaode Maps

the response delay. To overcome this limitation, Bonomi et al. proposed [13] fog computing to extend the cloud computing's capabilities to a network edge and allow fog nodes to preprocess users' data. The advantages of fog computing are geo-distribution, location awareness, and low latency [14], [15]. Moreover, fog computing has already been adopted in vehicular networks [15].

However, a significant amount of risks to user privacy is associated with route sharing [6], [16]–[18] because existing applications [9], [10] do not have a security mechanism. Additionally, such applications in the future will require stronger privacy properties owing to the users' increasing awareness of privacy protection. First, identities of users are exposed to a service provider (SP), which is a direct privacy violation because they do not need the SP to know whether they have logged into a system and have requested pertinent services [6]. Second, the starting locations and destinations of all the users are revealed to the SP. Locations, such as home address and military institution, are considered as sensitive [19]. Furthermore, they can be easily correlated to a lot of other information that a user intends to protect. For instance, Alice has been visiting a cancer hospital for two months and a malicious SP will know that Alice has cancer with a high probability that this location is leaked to itself. By continuously collecting and processing accurate locations, inferring a user's sexual preferences, political views, and religious inclinations is possible. Third, we claim that group privacy should be protected in three ways as follows. 1) After being formed for route sharing, each group must have an identifier similar to a user. If this group is formed on the same platform for several times with a fixed identifier. 2) Each group includes noisy locations [21] of more than one user; thus, the possibility of inferring their common destination will be eliminated. 3) The social graph of a group is extremely important to the group members' activities and social relations [20]. For instance, when Mike and Charlie have formed a group and their destination is a cinema hall, an adversary can assume that either they are on a date or they are in a relationship. This type of inference attack can cause more unwanted consequences and user safety may be endangered if one's privacy is divulged.

Fourth, we identify three novel attacks, namely member impersonation attack, group affiliation attack, and unlimited participation attack. Specifically, adversaries can 1) impersonate a legal group member and attempt to join this group to obtain the information or data of other members, 2) monitor group formation process to verify if a user belongs to an existing group, and 3) join groups arbitrarily within a period of time, which breaks fairness [4]. Therefore, implementing a privacy-preserving route-sharing scheme to prevent such privacy leakage is necessary.

To address these issues, a Privacy-preserving ROute-Sharing (PROS) scheme via vehicular fog computing is proposed to protect user and group privacy. The main contributions of this study are as follows:

1) We introduce fog computing into the route-sharing service model where fog nodes i.e., road-side units (RSUs) [22] preprocess the users' group formation/participation requests for the SP. We provide the definition of group privacy. Additionally, we establish a corresponding security model for the new system model, where the SP and RSUs are semi-honest, a small number of users are malicious, and new attacks, such as member impersonation, group affiliation, and unlimited participation, can be launched.

2) We propose the PROS scheme with the defined system and security models. Specifically, we improve the anonymous authentication [23] to protect identity privacy, leverage rate limiting pseudonyms [24] to limit the participation number and guarantee fairness, modify privacy-preserving equality test [25] to form a group and protect group privacy, and utilize geo-indistinguishability [19] to request route service and protect location privacy.

3) We formally analyze the security and privacy of the PROS scheme. Extensive experiments are conducted to compare its efficiency with existing schemes.

The rest of this paper is organized as follows. First, we discuss related work in Section II. Second, we illustrate the system model, security model, and design goals in Section III. Third, we revisit the preliminaries in Section IV. Fourth, we present the proposed PROS scheme in Section V, analyze the security and privacy in Section VI, and evaluate the scheme performance in Section VII. Finally, we conclude this paper in Section VIII.

II. RELATED WORK

Several works are available that achieved location preservation [7], [16] and location sharing [6], [26].
Most applications resort to group-based access control on released locations. Li et al. [7] claimed that it is not easy for a user to clearly determine a group, where his/her location is visible only to the members of the group, and binary access control is not sufficient to configure the privacy setting properly. Hence, researchers have proposed a fine-grained privacy-preserving location query protocol (PLQP) to allow querying for location information without violating users’ location privacy. For example, in their level-2 query, a querier q can learn whether distance(q, p) ≤ τ when the q’s attributes satisfy the publisher p’s conditions and q cannot detect the p’s location. However, they did not consider the group formation.

Puttaswamy et al. [16] introduced LocX to protect location privacy. The main idea of LocX can be understood by following example. Suppose a user, Alice, transforms her location from (x, y) to (x’, y’), generates a random index i, and encrypts it to obtain E_sym(i). Then, she stores [(x’, y’), E_sym(i)] on an index server and location data [i, E_sym(data)] (e.g., a restaurant review) on a data server. Thereafter, Alice’s friend Bob who knows her secret key initially sends a query that includes a list of transformed locations to the index server, then fetches and decrpts all the encrypted indices, and finally queries the data server for the location data he wanted to know. However, their system model consists of two servers [27]. Thus, management complexity increases and queriers must send multiple queries, thereby increasing the computational costs.

Schlegel et al. [6] proposed a new encryption notion called order-retrievable encryption for privacy-preserving location-sharing (PPLS) services in social networking applications. Specifically, their design allows a group of friends to share their location and receive the exact location of their friends without requiring any direct communication among users or multiple rounds of communication between a user and server. It can update locations dynamically and specify a maximum distance to provide personalized privacy protection within a group of friends. However, if each user in a group wants to know the locations of other group members, then the database server should perform massive distance comparisons.

Xiao et al. [26] proposed a centralized privacy-preserving location-sharing system (CLS). CLS uses dummy locations and dedicated mapping protocols between one location-storing social network server and cellular tower to share privacy-preserving locations. Basically, a user queries his nearby friends’ locations by sending a query (IDS, f, qf) to the cellular tower, which will query the server for his friends’ identities and locations. However, the actual locations are exposed to the server and the k-anonymity cannot provide rigorous security proof.

### III. PROBLEM STATEMENT

In this section, we illustrate the proposed system model, security model, and design goals.

---

**A. SYSTEM MODEL**

The system model of the proposed scheme PROS comprises four entities, namely certificate authority (CA), SP, RSUs, and users (vehicles and pedestrians), as shown in Fig. 2.

**FIGURE 2: System Model of Route Sharing**

CA is responsible for initializing a route-sharing system by generating public parameters and registering RSUs, which are vehicles and pedestrians. It will remain offline after registrations until a targeted user’s real identity is revealed.

SP is a route-sharing platform that collects user location and group formation information from distributed RSUs and responds to users’ route queries by providing specific routes from the current location to the destination for users.

RSUs are deployed near the road and deemed as fog nodes with computational and communication capabilities. Each RSU initially authenticates user’s identity in an anonymous manner, collects their queries and group formation/participation requests, verifies data integrity and participation qualification, execute the equality test to form corresponding groups, and finally uploads user and group information to the SP.

Users are equipped with a smartphone or OBU. Each vehicle uploads a group formation as a group leader or a group participation request as a group member. Each group shares a secret key [28]–[30] prior to group formation through a secure channel. Each user can send a route request, including a noisy location, to the cloud server via RSUs for route planning or share route with a group member if their starting locations are close.

We notice that when a group is being formed, members are at different locations, which can be close or away from each other. Table 1 shows the key notations.

---

**B. SECURITY MODEL**

CA is fully trusted and infeasible to be compromised. Meanwhile, SP and RSUs may attempt to infer users’ privacy, such as personal identity, location, destination, and group identity.

Most users will send their group formation/participation requests automatically. Meanwhile, other users are malicious, and they can launch three types of attacks, namely member impersonation attack, group affiliation attack, and unlimited participation attack. First, they can attempt to impersonate
TABLE 1: Key Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_1, G_2, G_T, G_1, G_2; q$</td>
<td>Cyclic group, group order</td>
</tr>
<tr>
<td>$y_1, y_2, y_i; H, H_2, H_3$</td>
<td>Group generator, hash function</td>
</tr>
<tr>
<td>$MC$</td>
<td>Max formation/participation count</td>
</tr>
<tr>
<td>$\alpha, \beta, \Lambda$</td>
<td>Master secret key, master public key</td>
</tr>
<tr>
<td>$dID_1, dID_2, S$</td>
<td>Anonymous key</td>
</tr>
<tr>
<td>$\sigma_1, \sigma_2, \sigma_3; A_1, A_2, A_3$</td>
<td>Secret key, public key</td>
</tr>
<tr>
<td>$\delta, \kappa$, $\Delta$</td>
<td>Secret key, public key, state information</td>
</tr>
<tr>
<td>$U, R_j$</td>
<td>Real identity of a user and an RSU</td>
</tr>
<tr>
<td>$\text{Cert}$</td>
<td>Verifies the validity of message, signature, and anonymous certificate.</td>
</tr>
<tr>
<td>$\text{CertGen}$</td>
<td>Generates a corresponding public key, an anonymous certificate, and a hash value $C$.</td>
</tr>
<tr>
<td>$\text{KeyGen}(\text{paras}_1, u_i)$</td>
<td>Takes as input public parameters and an identity $u_i$ and outputs two dummy identities $dID_1$ and $dID_2$ and an anonymous key $AK_i$.</td>
</tr>
<tr>
<td>$\text{CertGen}(\text{paras}_1, dID_1, dID_2, AK)$</td>
<td>Given system parameters, two dummy identities, and an anonymous key, respectively. It outputs a temporary private key and a corresponding public key, an anonymous certificate $\text{Cert}$, and a hash value $C$.</td>
</tr>
<tr>
<td>$\text{Sign}(\text{paras}_1, m, \sigma, \text{Cert})$</td>
<td>Given system parameters and a message $m$ and outputs a signature $\sigma$.</td>
</tr>
<tr>
<td>$\text{Open}(\text{paras}_1, \text{Cert})$</td>
<td>Takes as input system parameters and an anonymous certificate and outputs an identity $u_i$.</td>
</tr>
</tbody>
</table>

A legal member of a group where an imposter does not belong to and then obtain the information, such as personal identity, location, trajectory, and social graph, of the members in this group. Second, they can attempt to verify if a user belongs to an existing group. Third, they can attempt to expand their participation qualification and simultaneously join different groups, thereby breaking the system security.

We notice that common neighborhood attacks [31] are not applicable to our model because social graphs originating from group formation are complete graphs, which are different from classic social networks. Therefore, the entire information of the members’ social relationships is not leaked.

C. DESIGN GOALS

User privacy. Each user’s privacy contains identity, starting location, destination, and route, all of which must be preserved. Unlinkability should be guaranteed to ensure that each user’s privacy is unlinkable on the basis of the two queries or requests.

Group Privacy. Each group’s privacy contains group identity, group destination, and social graphs, all of which must be preserved, and unlinkability is the same as discussed previously.

Fairness. Only a permitted user can join his/her desired group, and any user cannot simultaneously participate in more than one group.

IV. PRELIMINARIES

This section revisits the preliminaries used to construct the PROS scheme, such as anonymous authentication [23], rate limiting pseudonyms [24], privacy-preserving equality test [25], and geo-indistinguishability [19].

A. ANONYMOUS AUTHENTICATION

An anonymous authentication scheme [23] is a collection of five algorithms as follows:

- **Setup($1^{k_1}$)** takes as input a security parameter $1^{k_1}$ as input, chooses bilinear parameters $H_1$, $H_2$, and two master secret keys $aAndb$. It computes two corresponding master public keys $AAndB$ and outputs system parameters $paras_1$.
- **KeyGen(\text{paras}_1, u_i)** takes as input public parameters and an identity $u_i$ and outputs two dummy identities $dID_1$ and $dID_2$ and an anonymous key $AK_i$.
- **CertGen(\text{paras}_1, dID_1, dID_2, AK)** are given system parameters, two dummy identities, and an anonymous key, respectively. It outputs a temporary private key and a corresponding public key, an anonymous certificate $\text{Cert}$, and a hash value $C$.
- **Sign(\text{paras}_1, m, \sigma, \text{Cert})** is given system parameters and a message $m$ and outputs a signature $\sigma$. It verifies the validity of $\text{Cert}$ and $\sigma$ and outputs 1 if the verification passes or zero otherwise.
- **Open(\text{paras}_1, \text{Cert})** takes as input system parameters and an anonymous certificate and outputs an identity $u_i$.

B. RATE LIMITING PSEUDONYMS

A rate limiting pseudonym scheme is a collection of the following three algorithms:

- **Setup($1^{k_2}$)** uses a security parameter $1^{k_2}$ as input and outputs system parameters $paras_2$.
- **KenGen(\text{paras}_2, u_i)** is used as the input system parameter and outputs a secret user key $uk$.
- **Request(\text{paras}_2, state, MC, uk)** takes as input system parameter, state information, max count, and user key. It outputs a rate limiting pseudonym $rnym$ and a zero-knowledge proof of knowledge (ZKPK) proof [32] $P$.
- **Verify(\text{paras}_2, state, count, rnym, P)** takes as input system parameter, state information, count, rate limiting pseudonym, and proof. It outputs 1 if count is not larger than $MC$ and proof verification passes or zero otherwise.

C. PRIVATE EQUALITY TEST

A private equality test scheme [25] is a collection of the following six algorithms:

- **Setup($1^{k_3}$)** takes as input a security parameter $1^{k_3}$, chooses bilinear parameters $paras_3$ and two hash functions $H_1$, $H_2$, and outputs the system parameter $paras_3$.
- **KenGen(\text{paras}_3)** takes as input system parameters. It initially selects three private keys, $a_1$, $a_2$, and $A_3$, and then computes three public keys $A_1$, $A_2$, and $A_3$; and finally outputs a key pair ($sk$, $pk$).
- **Encrypt(\text{paras}_3, m, pk)** takes as input system parameters, a message $m$, and a public key and outputs a ciphertext $c$.
- **Decrypt(\text{paras}_3, c, sk)** takes input system parameters, ciphertext, and secret key. It outputs a message $m$.
- **Type-2 Authorization(\text{paras}_2, c, sk)** takes as input system parameters, ciphertext, and secret key. It outputs a trapdoor $td$. 

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2878792, IEEE Access
Test\((c_1, td_2, c_2, td_2)\) takes as input system parameters, two ciphertexts, and two trapdoors. It verifies whether \(m_1 = m_2\) and outputs 1 if verification holds or zero otherwise.

**D. GEO-INDISTINGUISHABILITY**

The mechanism that protects location privacy with geo-indistinguishability guarantee includes the following three steps:

- **Noisy Point Generation.** For an actual location \(l_0 \in \mathbb{R}^2\), a mechanism for the continuous plane generates a random noise \(\text{noi} \) according to the noise function. After \(\text{noi}\) is added to \(l_0\), the reported location \(l_1\) should be as follows: the probabilities of reporting a point around \(l_1\), when the real locations are \(l_0\) and \(l'_1\), differ at most by \(e^{-cd(l_0, l')}\).

- **Discretization.** The discretization mechanism \(\mathcal{K}_e : \mathcal{G} \rightarrow \mathcal{P}(\mathcal{G})\) initially draws a point \((\text{range}, \theta)\) and then remaps the point to the closest point on the grid \(\mathcal{G}\).

- **Truncation.** The last step is a truncated variant of the discretized mechanism, which generates points only within a specified region and fully satisfies geo-indistinguishability. The truncation mechanism \(\mathcal{P}_L_e : \mathcal{A} \rightarrow \mathcal{P}(\mathcal{A} \cup \mathcal{G})\) runs with a difference that the point generated in the last step is remapped to the closest point in \(\mathcal{A} \cup \mathcal{G}\).

**V. THE PROPOSED PROS SCHEME**

In this section, we propose the privacy-preserving route-sharing scheme PROS to protect user and group privacy for route-sharing applications. The PROS scheme involves seven phases, namely system initialization, entity registration, group formation requesting, group participation requesting, group formation, route sharing, and user tracking.

**A. SYSTEM INITIALIZATION**

First, given a security parameter \(1^{k_1}\), CA generates three (multiplicative) cyclic groups, \(G_1, G_2, G_T\), with the same order \(q\) and two generators, \(g_1, g_2\) for \(G_1\) and \(G_2\), respectively. \(e_1\) is a bilinear map: \(e_1 : G_1 \times G_2 \rightarrow G_T\). CA chooses two random numbers \(a, b \in \mathbb{Z}_q^*\) as master secret keys and computes two corresponding public keys \(A = g_1^a, B = g_2^b\). It selects a secure hash function \(H : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*\). Second, given a security parameter \(1^{k_2}\), CA generates public group parameters \((\Gamma, \rho, g)\) and a max formation/participation count \(MC\). Third, given a security parameter \(1^{k_3}\), CA generates two groups, \(G_1, G_2\), of prime order \(q\) with a generator \(g\) for \(G_1\) and a bilinear map \(e_2 : G_1 \times G_1 \rightarrow G_2\). TA chooses two hash functions, \(H_1 : G_1 \rightarrow \{0, 1\}^{2|G_1|}\) and \(H_2 : G_2 \rightarrow \{0, 1\}^{2|G_2|}\).

Finally, CA published public parameter are \(para = (G_1, G_2, G_T, e_1, q, A, B, H, \Gamma, \rho, g, MC, G_1, G_2, e_2, H_1, H_2)\).

**B. ENTITY REGISTRATION**

First, a user with identity \(U_i\) registers to CA. Second, CA generates random numbers, \(\{uk_i\}_{i=1}^{MC}, x_i, y_i\), and two dummy identities, \(dID_{i1} = g_1^{x_i + a}\) mod \(q\), \(dID_{i2} = g_1^{y_i}\) mod \(q\), and computes \(S_i = g_1^{x_i - a}\). Third, it stores \((U_i, dID_{i1}, dID_{i2}, S_i^\text{ty})\) in a tracking list. Fourth, it selects three privates keys, \(a_{i1}, a_{i2}, a_{i3}\), and computes three public keys, \(A_{i1} = g_1^{a_{i1}}, A_{i2} = g_1^{a_{i2}}\), \(A_{i3} = g_1^{a_{i3}}\). Finally, it returns a secret user key set, \(\{uk_i\}_{i=1}^{MC}\), an anonymous key, \(AK_i = (dID_{i1}, dID_{i2}, S_i)\), and a key pair, \((a_{i1}, a_{i2}, a_{i3}, A_{i1}, A_{i2}, A_{i3})\) to \(U_i\).

An RSU with identity \(R_j\) registers to CA and obtains a secret key \(sk_{R_j}\) and a public key \(pk_{R_j} = g_1^{sk_{R_j}}\).

**C. GROUP FORMATION REQUESTING**

Each RSU \(R_j\) broadcasts a public key \(pk_j\), and a time-varying state information \(\Delta\) within its coverage area. A user \(U_i\) in the RSU \(R_j\)'s coverage area intends to form a group with his/her friends as a group leader. \(U_j\) initially establishes a group name \(GN_i\), a destination \(dest_i\), and a group communication key \(k_{i, \text{com}}\) with his/her group members through a secure channel.

First, \(U_i\) forms a rate limiting pseudonym \(rlp_i\) [24] with 1 \(\leq\) \(count\) \(\leq\) \(MC\) as follows:

\[
gr_{rlp_i} = H(\Delta|\{\text{count}\})^{(\Gamma-1)/\rho},
\]

\[
rlp_i = g_1^{uk_{\text{count}}} \mod \Gamma.
\]

Second, \(U_i\) generates a ZKPK \(P_i\) of \(uk_{\text{count}}\) as follows:

\[
\{SPK(uk_i) : g_1^{uk_{\text{count}}} \mod \Gamma\}.
\]

Thus, the proposed route-sharing system has guaranteed that \(U_i\) can participate in more than \(MC\) groups at a time. If \(U_i\) wants to join another group, then \(U_i\) can wait for the state information to change or apply for another secret user key.

Third, \(U_i\) chooses two random numbers, \(r_1, and r_2 \in \mathbb{Z}_q^*\), and encrypts \(GN_i\) using \((A_{i1}, A_{i2}, A_{i3})\) to obtain \(c_i = (c_{i1}, c_{i2}, c_{i3})\) [25] as follows:

\[
c_{i1} = g_1^{r_1} pk_{R_j},
\]

\[
c_{i2} = g_1^{r_2},
\]

\[
c_{i3} = (GN_i^{r_1}|(GN_i \cdot A_{i2})^{r_1}) \oplus H_1(A_{i3}^{r_1}),
\]

where \(U_i\) generates a trapdoor

\[
td_i = [H_1(A_{i3}^{r_1})]^{2|G_1| - 1}.
\]

In this way, \(U_i\) has enabled \(R_j\) to form \(GN_i\) based on messages from \(U_i\) and other group members later, and other entities cannot form a group. We note that \(c_{i1}\) is different from the original design [25] because we multiply \(g_1^{r_1}\) with \(R_j\)'s public key. Hence, only \(R_j\) can perform a private equality test later, thus achieving the designated testing goal.

Thereafter, \(U_i\) generates a noisy starting location \(loc_i^0\) and a noisy destination \(dest_i^0\) according to geo-indistinguishability [19] and ready to send an encrypted (with the SP's public key) route service request to the SP. Finally, \(U_i\) forms a message \(m_i = (gr_{rlp_i}, rlp_i, P_i, c_i, td_i, loc_i^0, dest_i^0)\). Then, \(U_i\) generates a temporary key pair.
(tki, TKi = g2tki), selects four random numbers (i.e., η, τ1, τ2, and τ3), and sends an anonymous certificate certi = (dID1i, dID2i, TKi, γ1, γ2, C, δ1, δ1, δ2), as well as a signature σi to Rj, as follows:

\[ dID1i = dID1 \cdot g^\eta_i, \quad dID2i = dID2 \cdot g^\eta_i, \]
\[ \gamma_1 = B^\tau_1, \quad γ_2 = S_i \cdot A_{Ti}, \quad δ_1 = (τ_1 + tk_i) \mod q, \]
\[ δ_2 = γ_1^{τ_1+τ_2}, \quad δ_3 = γ_1^{τ_1+τ_2}, \]
\[ C_i = H(dID1_i || dID2_i || TK_i || A_i || B_i || γ_1 || γ_2 || δ_2 || δ_3), \]
\[ θ_1 = (d_1 - τ_2) \mod q, \quad θ_1 = (d_1 - τ_3) \mod q, \]
\[ σ_i = g_1^{γ_1^{τ_1+τ_2} \cdot r_{i,m}}. \]

We express the group formation requesting phase in Algorithm 1.

**Algorithm 1 Group Formation Requesting.**

**Input:**
- RSU’s state information Δ and public key pkRj;

**Output:**
- Group name GNi, group communication key ki, message mi, secret key tk, public key TKi, anonymous certificate certi, and signature σi
1. Choose a number Count to be the number of groups that the leader group should form, s.t. 1 ≤ Count ≤ MC.
2. For count = 1, 1 ≤ count ≤ Count:
3. Form a rate limiting pseudonym rlp1.
4. Generate a ZKPK P1 of ukcount.
5. Choose two random numbers r1, r2.
6. Encrypt GNi to obtain ci = (c11, c12, c13).
7. Compute a trapdoor td1.
8. Generate a noisy location loc1 and a destination dest1.
9. Form a message m1 = (g1rlp1, rlp1, P1, c1, td1, loc1).
10. Select a secret key tk, compute a public key TKi.
11. Compute anonymous certificate certi, a signature σi.
12. return GNi, ki, mi, tk, TKi, certi, σi.

---

**D. GROUP PARTICIPATION REQUESTING**

A user Uj intends to join a group GNi and already receives three public keys, (A1i, A2i, and A3i), and a group communication key, ki. Generally, Uj is also used within the coverage area of Uj’s RSU Rj. However, if Uj does not share the same RSU with Ui, then Uj will send Rj’s identity received from Ui to a local RSU and Rj will perform the group formation later.

First, Uj forms a rate limiting pseudonym rlp1: g1rlp1 = H(δ || 1)P−1/r, rlp1 = ukrlp1 mod Γ and generates a proof P1: {SPK(ukj) : g1ukrlp1 mod Γ}.

Second, Uj chooses two random numbers, r1 and r2, ∈ Zq, and encrypts GNi using (A1j, A2j, and A3j) to obtain ci = (c1j, c2j, c3j, c4j) : c1j = g1c1jpkRj, c2j = g1c2j,
c3j = (GNi1 | (GNi1 ∙ A2j)1) ⊕ H1(A2j), and c4j = H2(c1j2 | c2j3 | A2j2) ⊕ (GNi1 | r1j). tdj = {H1(c2j1)}2[0,1]−1.

Third, Uj generates a noisy starting location loc1 and a noisy destination dest1. We notice that because the SP can monitor the information flow of users within the same group, encrypting the actual locations and sending them to other group members are not necessary for members. Here, we require that SP only relays users’ obfuscated locations to their group members. However, the users have the options to share locations (encrypted by himself/herself and decrypted by other group members) within a group, thereby providing better user experience.

Finally, Uj forms a message, mj = (g1rlp1, rlp1, Pj, c1j, tdj, loc1, dest1), and sends an anonymous certificate certj and a signature σj to the local RSU.

---

**E. GROUP FORMATION**

After receiving a message mi from the user Ui, the validity of certi and σi is initially verified as follows:

\[ dIDj' = dID1 \cdot dID2, \]
\[ δ'1 = γ_1^{δ_1} \cdot γ_1^{θ_1}, \]
\[ δ'2 = γ_1^{δ_1} \cdot γ_2^{θ_1} \cdot γ_2^{δ_2}, \]
\[ C_i' = H(dID1_i || dID2_i || TK_i || A_i || B_i || γ_1 || γ_2 || δ'2 || δ_3'), \]

where Rj verifies dIDj' = A and Cj' = Cj. If they both hold true, then whether e1(σ1, TKi ∙ g2r(mi)) = e1(g1, g2) is verified. Meanwhile, if only one holds true, then Rj accepts mi and continues.

Uj verifies P1 and checks if the parameters are valid, i.e., the Δ is the state information is the correct one and whether 1 ≤ count ≤ MC. If the proof verification passes, then Rj accepts mi and continues.

Thereafter, given another ciphertext c1j and trapdoor tdj from the user Uj who wants to join in the group GNi, initiated by the user Ui, Rj tests their group equality by computing

\[ c_{j+1} ⊕ tdj = GN_{i1}, \]
\[ c_{j+1} ⊕ tdj = GN_{i1}, \]
\[ c_{j+1} = c_{j+1}g^{skRj}, \]
\[ c_{j+1} = c_{j+1}g^{skRj}, \]

and allows Uj to joins GNi if e2(c1j1, GNi1) = e2(c1j1, GNi1) holds true.

Finally, Rj uploads the group name, noisy locations, and noisy destinations to the SP, which will return corresponding route services to the users. We express the group formation phase in Algorithm 2.
Algorithm 2 Group Formation.

Input:
Message \( m_i \), anonymous certificate \( cert_i \), signature \( \sigma_i \) from a group leader, set of messages, anonymous certificates and signatures \( \{ M_j, Cert_j, \pi_j \}_{j=1}^{num} \) from \( num \) group members.

Output:
A set of \( GN_i \) or \( \perp \).

1. Verify \( cert_i \) and \( \sigma_i \).
2. If passes, then continue.
3. Else, choose another group leader, and proceed to 1.
4. Create a group \( GN_i = \{ dID_{11} \} \).
5. For \( o = 1, 1 \leq o \leq num \):
6. Verify \( Cert_o \) and \( \pi_o \).
7. If passes, then continue.
8. Else, break.
9. Test the equality of \( c_i \) and \( C_i \).
10. If equals, then incorporate \( dID_{11} \) into \( GN_i \).
11. Else, continue.
12. Send \( GN_i, \{ loc_{u}^{\prime} \}_{u=1}^{GN_i}, \{ dest_{u}^{\prime} \}_{u=1}^{GN_i} \) to the SP.
13. return \( a; \)

F. ROUTE SHARING

After the group \( GN_i \) is formed, through encryption and decryption with the group communication key \( k_{comm}^i \), all group members will share their real-time locations, destinations, and routes to detect their friend’s location and estimate their arrival time at the destination.

Additionally, group members can also communicate with other group members and share information, such as current traffic condition on a specific road [33], picking up some pizza when passing by a store, or asking for a direction if a group member cannot read the route on a map. The four main phases of the PROS scheme are depicted in Fig. 3.

G. USER TRACKING

When a user \( U_i \) misbehaved (e.g., impersonates another user, participates in more than \( K \) groups) in the route-sharing system, CA uses the anonymous certificate \( cert_i \) generated by \( U_i \) to reveal \( U_i \)'s real identity by computing

\[
\frac{\gamma_2}{\gamma_1} = \frac{(S_i \cdot A^{\gamma_1})^b}{(B^{\gamma_1})^a} = \frac{S_i^b \cdot A^{\gamma_1 b}}{B^{\gamma_1 a}} = \frac{S_i^b \cdot g_1^{\gamma_1 b}}{g_1^{\gamma_1 a}} = S_i.
\]

Then, it discloses the real identity of \( U_i \) by searching \( S_i \) in its tracking list.

VI. SECURITY AND PRIVACY ANALYSIS

In this section, we demonstrate that our scheme can achieve all the security goals defined in section III-C.

User privacy. First, users’ real identities are not included in their requests and starting locations, and destinations are randomized by noises according to geo-indistinguishability [19]. Their social graphs are hidden because the group name is secured by private equality test. Second, the two dummy identities are randomized in each request, any two rate limiting pseudonyms are not connected owing to the zero proof of knowledge [24], all of which ensure unlinkability.

Theorem 1: The proposed PROS scheme is one-way under chosen-ciphertext attacks (OW-CCA) secure for type-2 authorization against an adversary based on Computational-Diffie-Hellman (CDH) assumption in a random oracle model [25].

Proof: Here, the type-I adversary cannot recover the plaintext from the challenge ciphertext given the type-2 trapdoor information in a type-2 authorization. Let \( A_1 \) be the PPT type-I adversary breaking the PROS scheme. Assume that \( A_1 \) can make at most \( q_1 H_1 \) hash queries, \( q_2 H_2 \) hash queries, \( q_k \) key retrieve queries, \( q_d \) decryption queries, and \( q_A \) authorization queries for the type-2 authorization. The advantage \( Adv^{ow-cca_{type-2}}_{PKEET-FA,A_1} (\lambda) \) of \( A_1 \) is the probability \( Pr[M_1 = M_2] \) where 1) \( M_1 \) is the \( A_1 \)'s output and 2) for the challenger-chosen \( M_t \) does not appear in the key retrieve queries and \( c_i^* \) does not appear in the encryption queries. Hence, it could be deduced that \( Adv^{ow-cca_{type-2}}_{PKEET-FA,A_1} (\lambda) \leq 2Adv^{CDH} + \frac{q_d}{2^{161 + |\pi|}} \). Given that the CDH is a hard problem, \( Adv^{CDH} \) is negligible, and \( \frac{q_d}{2^{161 + |\pi|}} \) is a constant number. In addition, the advantage of \( A_1 \) in winning the game is also negligible. This completes the proof. \[Q.E.D.\]

Theorem 2: The proposed PROS is indistinguishable under chosen-ciphertext attacks (IND-CCA) secure for the type-2 authorization based on the CDH assumption in the random oracle model [25].

Proof: Here, the type-II adversary cannot decide that \( c_i \) is the encryption of which message without the type-2 trapdoor information in a type-2 authorization. Let \( A_2 \) be the PPT type-II adversary attacking the PROS scheme. Assume that \( A_2 \) can make the same queries as \( A_1 \) for type-2 authorization. The advantage \( Adv^{ind-cca_{type-2}}_{PKEET-FA,A_1} (q_1, q_2, q_d, q_k, q_A) \) of \( A_1 \) is the probability \( Pr[b' = b] \) in a game where 1) \( b' \) is \( A_1 \)'s output and 2) for the challenger-chosen \( b \), \( t \) does not appear in key retrieve queries and
c^* does not appear in decryption queries. Moreover,

\[ Ad_{PKEET}^c\cdot F_{\mathcal{A}_2}(q_{1},q_{2},q_{d},q_{k},q_{A}) \leq 2Ad_{CDH}^c + \frac{q_{d}^2}{4q_{1}^2} + \frac{q_{k}^2}{(4q_{1}^2)q_{A}}. \]

If the advantage of \( \mathcal{A}_2 \) attacking the PROS scheme is not negligible, then the CDH problem can be solved using the PPT adversary, which is not possible. This completes the proof. \( \square \)

**Group Privacy.** First, the group’s identity, which is the group name, is encrypted and secured on the basis of the CDH problem, and it is also different in each group formation. Second, a group destination is a value encrypted by the group communication key, which is distributed through a secure channel. Even if SP can see all the noisy destinations of a group, it cannot identify the actual common destination.

**Fairness.** First, only permitted user can form a group and an outsider does not have the group name or communication key. Hence, the outsider cannot successfully pass the equality test to join in the group. Second, the rate limiting pseudonym technique with state information and a secret user key can ensure that any user may only join in less than two groups without being detected.

In a member impersonation attack, an adversary cannot successfully become a legal member in a group without the previously co-established group name \( GN_i \). Meanwhile, a group affiliation attack is prevented using our proposed message encryption design in Equation (3) to ensure that only the local RSU can test the equality of two users’ membership. Moreover, the group communication key \( k_{com} \) secures the group members’ following communications. An unlimited participation attack is excluded by the utilization of the time-varying state information \( \Delta \) and unique user key \( uk \), indicating that any user should wait for \( \Delta \) to change to join in another group. Finally, we compare the security and privacy properties in Table 3.

**TABLE 2:** Comparison of Security Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>PLQP</th>
<th>LocX</th>
<th>PPLS</th>
<th>CLS</th>
<th>PROS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidentiality</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Integrity</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>Authentication</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>User Privacy</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Group Privacy</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Fairness</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>Traceability</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

**VII. PERFORMANCE ANALYSIS**

In this section, we evaluate the performance of the PROS scheme regarding the computational cost and communication overhead.

**A. EXPERIMENTAL ENVIRONMENT**

We perform the experiments using a laptop with 8 GB memory, Intel Core i7-7500 CPU @ 2.70 GHz, and Microsoft Windows 7 64-bit home operating system and a desktop with a 12 GB memory, two Intel Xeon-E5620 processors, and Microsoft Windows 7 professional operating system. We use two tools, which are Miura [34] and Crypto++ [35]. The elliptic curve is defined as \( y^2 = x^3 + 1 \) over \( \mathbb{F}_p \), where \(|p| = 512 \) bits. The detailed explanations of the experimental parameters are listed in Table 3. We simulate the users on the laptop and the RSUs on the desktop. The number of users \( N_1 \) is from 100 to 1000 and the number of RSUs \( N_2 \) is 10.

**TABLE 3:** Experimental Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Definition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N_1, N_2 )</td>
<td># of users/RSUs</td>
<td>([100, 1000], 10)</td>
</tr>
<tr>
<td>( q, p )</td>
<td>Big prime number</td>
<td>(</td>
</tr>
<tr>
<td>( H, H_1, H_2 )</td>
<td>Hash function</td>
<td>SHA256</td>
</tr>
<tr>
<td>( f, d )</td>
<td># of query locations; dimension</td>
<td>10</td>
</tr>
</tbody>
</table>

**B. COMPUTATIONAL COSTS**

Exp, \( Exp_1 \), \( Mul_1 \), \( Mul_T \), \( Div_1 \), \( Div_T \), \( BP \), Hash, SE, AE, Ex, Ad, Su, Mu, and Di denote the operations of exponentiation in \( G_1 \), exponentiation in \( G_T \), multiplication in \( G_1 \), multiplication in \( G_T \), division in \( G_1 \), division in \( G_T \), bilinear pairing, hash function, symmetric encryption, asymmetric encryption, exponentiation in \( Z_q \), addition in \( Z_q \), subtraction in \( Z_q \), multiplication in \( Z_q \), and division in \( Z_q \), respectively. Each group leader or member conducts \( 4Exp_1+2Mul_1+3Exp+Ad+Su+Mu+Di+4TH+AE \) in group formation/participation requesting. Meanwhile, each user conducts \( 8Exp_1+3Mul_1+2Div_T+2TH+5Ad+3T_3+Di \) in generating an anonymous credential and a signature. Each RSU performs \( 7Exp_1+4Mul_1+2Div_T+TH+BP \) to verify a user’s message.

We compared the PROS proposed scheme with PLQP [7], LocX [16], PPLS [6], and CLS [26] from the viewpoint of computational costs through group formation requesting, group participation requesting, and group formation. The experimental results in Fig. 4(a) indicate that the PROS scheme is not significantly advantageous in group formation/participation requesting because we aim to protect the security and privacy of users simultaneously. Additional operations will inevitably incur more computational investments.

Then, we compared the PROS scheme with BBS signature [36], ECPP [37], and KPSD [38] from the viewpoint of computational costs in anonymous credential generation and message verification. The PROS scheme outperforms the other schemes, as shown in Figs. 4(b) and 4(c). The reason for choosing these three schemes is that PLQP [7], LocX [16], PPLS [6], and CLS [26] do not consider the problem of user authentication, and BBS signature [36], ECPP [37], and KPSD [38] are the three classic utilization of an anonymous authentication.

For PLQP [7], we choose their distance comparison algorithm because it is similar to group formation. For the group formation phase in LocX, we choose \( f = 10 \). Meanwhile, for the group formation and participation phase in PPLS, we choose \( d = 10 \).
model where, three types of attacks from malicious users were considered. The PROS scheme not only provides users with route planning and group formation but also protects user and group privacy. Finally, we proved the security and privacy of the PROS scheme and evaluated its performance by comparing it with existing work.

In the future work, we will consider the possibility that the RSUs can be compromised. If an RSU is compromised, then its group formation process is interfered and the group formation result is not accurate. To defend this attack from a compromised RSU, we will design a privacy-preserving route-sharing scheme to support group formation verification without violating user and group privacy.

ACKNOWLEDGMENT
This work is partially supported by China National Key Research and Development Program No. 2016YFB0800301 and National Natural Science Foundation of China No. 61872041 and No. 61871037.

REFERENCES


MENG LI (menglibit@bit.edu.cn) is currently working toward the Ph.D. degree with the School of Computer Science and Technology, Beijing Institute of Technology and he is a visiting Ph.D. student in Wilfrid Laurier University. He has received the National Graduate Student Scholarship in 2011 and he has been sponsored by China Scholarship Council in 2017. His research interests are applied cryptography, security and privacy, VANET, fog computing and blockchain.

LIEHUANG ZHU (liehuangz@bit.edu.cn) is a Professor in the School of Computer Science and Technology at Beijing Institute of Technology. He is selected into the Program for New Century Excellent Talents in University from Ministry of Education, P. R. China. His research interests include cryptographic algorithms and secure protocols, Internet of Things security, cloud-computing, big data privacy, mobile and Internet security, and trusted computing.

ZUIJIAN ZHANG (zhangzijian@bit.edu.cn) is an Associate Professor in the School of Computer Science and Technology at Beijing Institute of Technology. He was a visiting scholar in the Computer Science and Engineering Department of the State University of New York at Buffalo in 2015. His research interests include cryptography, differential privacy, smart grid, data privacy, mobile security, and data mining.

XIAOJIANG DU (dxj@ieee.org) is currently a Professor in the Department of Computer and Information Sciences at Temple University. He received his B.E. degree from Tsinghua University, China in 1996, and his M.S. and Ph.D. degrees from the University of Maryland, College Park in 2002 and 2003, respectively, all in Electrical Engineering. His research interests are security, systems, wireless networks, and computer networks. He has published over 300 journal and conference papers in these areas. He serves on the editorial boards of three international journals.

MOHSEN GUIZANI [S’85, M’89, SM’99, F’09] (mguizani@ieee.org) is currently a professor and the ECE Department chair at the University of Idaho. He received the bachelor’s (with distinction) and master’ degrees in electrical engineering, the master and doctorate degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987 and 1990, respectively. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, security and smart grid. He currently serves on the editorial boards of several international technical journals and the founder and the editor-in-chief of Wireless Communications and Mobile Computing Journal (Wiley). He was the chair of the IEEE Communications Society Wireless Technical Committee and the chair of the TAOS Technical Committee. He served as the IEEE Computer Society Distinguished Speaker from 2003 to 2005.