Journal of Parallel and Distributed Computing 135 (2020) 156–168

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

# Privbus: A privacy-enhanced crowdsourced bus service via fog computing☆

Yuanyuan He [a,b,d], Jianbing Ni [e], Ben Niu [b,*], Fenghua Li [b,c], Xuemin (Sherman) Shen [d]

[a] School of Cyber Science and Engineering, Huazhong University of Science and Technology, China
[b] Institute of Information Engineering, Chinese Academy of Sciences, China
[c] School of Cyber Security, University of Chinese Academy of Sciences, China
[d] Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada
[e] Department of Electrical and Computer Engineering, Queen's University, Kingston, ON, Canada

ARTICLE INFO

*Article history:*
Received 29 April 2019
Received in revised form 18 August 2019
Accepted 17 September 2019
Available online 28 September 2019

*Keywords:*
Privacy preservation
Crowdsourced bus service
Fog computing
Data clustering

ABSTRACT

Crowdsourced bus service provides the customized bus for a group of users with similar itineraries by designing the route based on the users' trip plans. With crowdsourced bus service, the users with similar trips can enjoy the customized bus route efficiently and inexpensively. However, serious privacy concerns (e.g., the exposure of users' current and future locations) have become a major barrier. To protect users' itineraries, we propose Privbus, a privacy-enhanced crowdsourced bus service without hampering the functionality of bus route planning. Specifically, Privbus improves the performance of clustering itineraries due to the assistance of fogs. Then, Privbus executes the fog-assisted density peaks clustering operations on ciphertexts of users' travel plans to protect the users' trips. By doing so, the clustering operation is removed from users' smart devices to fog nodes, so as to enable the users to be offline after they submit their travel plans. According to the clustering results, Privbus uses a route planning method to optimize the bus routes. The optimization reduces the time cost on travel of users, while guaranteeing the good profit and the wide coverage of crowdsourced bus service. Finally, through the performance evaluation and extensive experiments, we demonstrate that Privbus has the advantage of low computational and communication overhead, while providing high security and precision guarantees.



## 1. Introduction

Crowdsourced Bus Service (CBS) customizes a bus route for each group of users with similar itineraries after analyzing the trip requests crowdsourced from users. Since CBS can improve the vehicle occupancy rate and provide relatively fast trips in crowded cities compared with public buses, it has been widely expected as a green and efficient way to commute and travel in metropolises. A number of people have changed the way to get to work or school due to various CBS applications, such as Bridj[1], Jiewo[2], GoOpti[3] and Bus Pooling[4]. It is reported that Jiewo has offered more than 200 routes in Beijing and served more than 20,000 passengers per day; among them, 30% of whom previously drive to work or travel[5].

Unfortunately, the privacy concerns about users' personal information have arisen with the popularity of CBS. As the personal itineraries are closely related to user behavior including the location and time of the event [18,35], such exposure poses threats to the user's sensitive information, such as home address, workplace, financial situation, personal preference and even personal safety[6]. According to reports, more than 200,000 users deleted their Uber accounts in August 2017, since Uber had released 2

☆ Part of this research work was presented in the 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (IEEE WiOpt 2018) .

* Corresponding author.
*E-mail addresses:* yuanyuan_cse@hust.edu.cn (Y. He), jianbing.ni@queensu.ca (J. Ni), niuben@iie.ac.cn (B. Niu), lifenghua@iie.ac.cn (F. Li), sshen@uwaterloo.ca (X. Shen).

1 https://www.bridj.com

2 http://www.jiewo.com/

3 https://www.goopti.com/en/

4 http://www.bus-pooling.com/

5 http://www.scmp.com/news/china/society/article/1934853/far-madding-crowd-cui-ruis-chinese-bus-sharing-app-offers-beijing

6 http://www.bbc.com/news/world-europe-38948281 https://toronto.ctvnews.ca/privacy-commissioner-opens-formal-investigation-into-uber-data-breach-1.3716688

https://doi.org/10.1016/j.jpdc.2019.09.007
0743-7315/

billion pieces of trip data collected from people in more than 450 cities[7] to promote the platform, Uber Movement. Therefore, serious privacy concerns have become a major barrier against the further growth of CBS.

While the privacy of itineraries gains great importance, CBS requires the use of detailed trip information, including users' ID, source locations, destination locations and expected arrival time [17,18,27]. These trip requests are essential for trip clustering and bus route planning in the process of offering CBSs. If users' itineraries are completely hidden, performing clustering computation for designing bus routes is infeasible [10]. Therefore, the major technical challenge is how to enable privacy-preserving trip clustering for CBSs while preserving sensitive information about users' trips [7,15,25].

The existing Privacy-Preserving Clustering (PPC) schemes are based on either randomization or cryptography techniques. The former [1,16,29] protects sensitive data by adding noise, which is computationally efficient, but the extra noise reduces clustering quality in most randomization-based schemes, resulting in inaccurate cluster centers and improper bus routes. The latter [2,33,37] can achieve the relatively strong privacy preservation without sacrificing clustering quality. Nevertheless, applying these PPC approaches to CBS has several limitations: (1) Users need to execute encryption and decryption operations in each iteration to guarantee privacy and clustering quality in most methods, which put heavy computational and communication overhead on the users' resource-constrained mobile devices. (2) Users' trips have the characteristic of positive spatial auto-correlation [12] in a manner of spatial statistics, which is conductive to quickly identify the geographically proximate users. Unfortunately, it is difficult to make the positive spatial auto-correlation characteristic work when the trips are encrypted. (3) The service provider has enough motivations to merge some clusters into a bus route, since it is able to increase profit by using fewer buses to carry more passengers, whereas a route of this type is usually long, resulting in the conflict between the length of routes and the user demand of fast bus trips.

To address these issues, we introduce fog computing [26,28] into CBSs. Fog nodes are deployed between the users' terminal devices and CBS server, and have the ability to provide data storage and network computing services. Thus, each fog node can collect encrypted trip plans from all users in its service region [36], and then performs density peaks (DP) trip clustering on the crowdsourced ciphertexts [38,39] instead of the users. This makes it feasible to reduce the computing and communication cost on the users' smart devices and even support the offline users. Furthermore, even though the trip information is encrypted, the positive spatial auto-correlation characteristic [12] of crowdsourced trips works for improving the performance of implementing DP trip clustering [34] without losing the clustering quality due to the data locality property of fog nodes. Fog nodes are only allowed to access the encrypted trips, since fog nodes are not fully trusted. Therefore, in this paper, we propose Privbus to provide the privacy-preserving CBS in the fog-assisted architecture, including fog-assisted DP clustering, privacy-preserving trip clustering and bus route planning. Specifically, the main contributions of our paper are threefold:

- We propose the fog-assisted DP clustering mechanism, which enhances the performance of implementing DP clustering without losing clustering quality due to the data locality property of fog nodes. Given $n$ trips in CBS system, compared with $O(n^2)$ distances in the original DP clustering, only $O(n_i n)$ distance values need to be calculated, where $n_i \ll n$ holds and $n_i$ denotes the number of users in a fog node's service region.
- We design a novel privacy-preserving trip clustering mechanism to perform the fog-assisted DP clustering with privacy protection techniques. In addition to protect each user's sensitive trip, the mechanism assigns the calculation task of privacy-preserving clustering to fog nodes instead of users. Thus the mechanism reduces the computing cost on the users' smart devices, and even allows the users to be offline after submitting their trip requests.
- We present an approach of route planning based on the clustering outcome, which customizes and optimizes bus routes for the users. In order to increase the service provider's profit and extend access to CBS, CBS server optimizes the customized bus routes by merging some clusters into a bus route, subject to the user demands of walking distance to bus stops, acceptable waiting time, expected arrival time and fast bus trips.

The remainder of this paper is organized as follows. In Section 2, we present the system model and privacy model, and identify the design goals. In Section 3, we review the preliminaries. Then, we propose Privbus and discuss its security in Section 3.3, followed by the performance evaluation in Section 4. Finally, we review the related works in Section 5 and draw the conclusion in Section 6.

## 2. Preliminaries

In this section, we present the system model and privacy model, and identify our design goals for Privbus.

### 2.1. System model

CBS system consists of three entities: a server, users with terminal devices, and fog nodes, as depicted in Fig. 1.

CBS server is a platform to provide CBS for a metropolitan, which has strong computational capability and sufficient storage space. CBS server cooperates with fog nodes to gather similar trips in a cluster without learning the detailed trips. Based on the clustering result, CBS server designs routes for providing on-demand bus trips and optimizes bus routes to increase its profit and the service coverage on the premise of satisfying user demand. As shown in Fig. 1, CBS server recommends route 1 to users, since 3 clusters ($a_0 \rightarrow a_1$, $b_0 \rightarrow b_1$ and $c_0 \rightarrow c_1$) are merged into route 1, which enables a bus to serve more passengers.

The individual users buy the bus tickets from CBS server. Users' trip requests hide information about their trips and the adjacent fog nodes that cover their $d_c$-length neighbors. After submitting a trip request to a nearby fog node, each user is allowed to be offline until she downloads the result of CBS and pays for the ticket. Thus, users' mobile devices with the limited resources can save resource to accomplish other important tasks.

Fog nodes can be cellular base stations, WiFi access points or femtocell routers [23], having computational capability and storage space, which are deployed at the edge of the Internet. Each fog node can serve users holding mobile devices within the scope of the area it covers. In the fog-assisted architecture, fog nodes crowdsource users' trip requests, exchange the crowdsourced data with adjacent fog nodes, and then perform DP clustering with CBS server. Fog nodes are able to improve the performance of clustering calculation without losing clustering quality and users' privacy.

[7] https://futurism.com/uber-releases-a-staggering-2-billion-trips-worth-of-traffic-data/

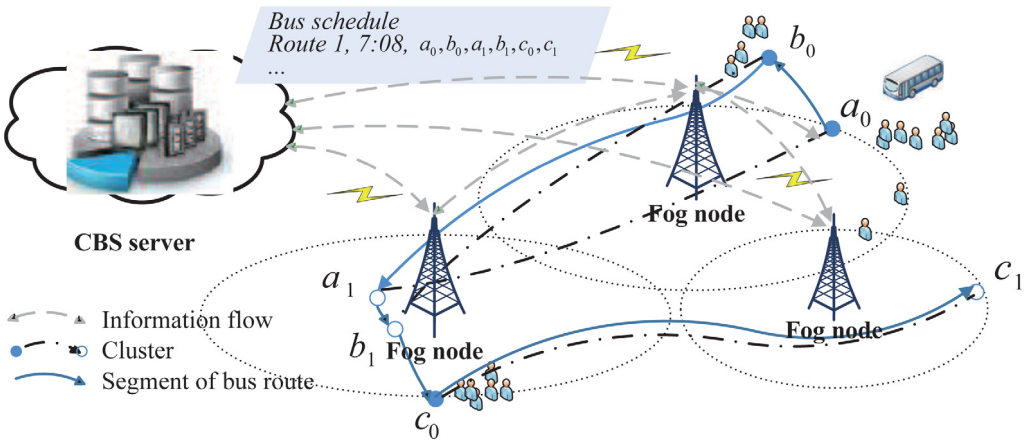



**Fig. 1.** Crowdsourced Bus Service (CBS).

### 2.2. Privacy model

CBS server is assumed to follow the agreements between it and users, such as privacy policy, which usually requires CBS server to keep users' trip data private. However, it is impossible to fully trust the service provider, since CBS server has a strong incentive to infer detailed information about users' trips and even capture personal information hidden in the intermediate values and the final results. Hence, CBS server is honest-but-curious in our work. The fog nodes are honest-but-curious as well. They follow the protocol honestly, but are interested in individual users' detailed trips for business benefits. Besides, CBS server does not actively collude with fog nodes, since they are business driven entities from different companies in realistic scenarios [31], and the collusion would impact their reputation and give others the witness of their misbehaviors.

Individuals honestly submit their real trip requests. If users report fake trip requests, the customized bus routes would be imprecise and it would be a waste of money to buy these bus tickets.

The global eavesdroppers are curious about the target users' privacy and try to infer their itineraries through the analysis of all collected messages.

### 2.3. Design goal

Our design goal is to provide CBS with the protection of users' trip information under the aforementioned system model and privacy model. To be specific, the goal of Privbus can be divided into two folds:

- Privacy Preservation. Users' sensitive trips should be encrypted before being submitted to fog nodes for privacy preservation. CBS server and fog nodes are required to perform trip clustering and bus route planning without learning detailed trips. Privbus protects users' trips and bus schedules against global eavesdroppers as well. When Privbus ends, CBS server only learns bus schedules, and the fog nodes cannot learn anything about users' trips. Meanwhile, each user only has information about her own departure time, pick-up and drop-off bus stops.
- Functions. Firstly, Privbus should improve the performance of implementing trip clustering without sacrificing clustering quality. Furthermore, Privbus saves online traffic for users by removing the clustering computation from users' smart devices to fog nodes. Finally, Privbus optimizes the routes of customized buses on the premise of satisfying user demand to increase the service provider's profit and improve the coverage of CBS.

### 2.4. Technologies used in privbus

In this section, we review the Paillier-based cryptosystem [21] and the DP clustering algorithm [34], which are used to design Privbus.

#### 2.4.1. The paillier-based cryptosystem

The Paillier cryptosystem with threshold decryption (PCTD) cryptosystem follows the idea in [11] and separates the private key into different shares [21] to support threshold decryption, consisting of the following algorithms:

① Key Generation, $KeyGen(\kappa)$. An entity chooses two $\kappa$ bit primes $p_0$, $p_1$, computes $N = p_0 p_1$, $\lambda = lcm(p_0 - 1, p_1 - 1)$ and defines function $L(x) = \frac{(x-1)}{N}$. Then, it chooses a generator $g \in \mathbb{Z}^*_{N^2}$ [9] of order $(p_0 - 1)(q_0 - 1)/2$. The public key is $pk = (N, g)$ and the corresponding private key is $sk = \lambda$.

② Encryption, $E(m, pk)$. Let $m \in \mathbb{Z}_N$ be a plaintext and $r \in \mathbb{Z}_N$ be a random number. The ciphertext $C$ can be generated as $C = E(m, pk) = g^m r^N \bmod N^2 = (1 + mN) r^N \bmod N^2$.

③ Decryption, $D(C, sk)$. Given the ciphertext $C$ and the corresponding private key $sk$, due to $gcd(\lambda, N) = 1$, the plaintext can be derived as $m = D(C, sk) = L(C^\lambda \bmod N^2)\lambda^{-1} \bmod N$.

④ Private key splitting, $KeyS(sk)$. The private key $sk = \lambda$ can be randomly separated into $k$ different shares. In particular, when $k = 2$, the partially private keys are denoted as $sk^{(s)} = \lambda_s$, $s = 1, 2$, where both $\lambda_1 + \lambda_2 = 0 \bmod \lambda$ and $\lambda_1 + \lambda_2 = 1 \bmod N^2$ hold [22].

⑤ Partial decryption, $D_{sk^{(s)}}(C)$. The ciphertext $C$ can be partially decrypted with the partially private key $sk^{(s)}$ as $D_{sk^{(s)}}(C) = C^{sk^{(s)}} = (1 + mN\lambda_s) r^{\lambda_s N} \bmod N^2$.

⑥ Threshold decryption, $D_T(C)$. Given $D_{sk^{(s)}}(C)(s = 1, 2)$, the original message can be obtained by $m = D_T(C) = L(D_{sk^{(1)}}(C) D_{sk^{(2)}}(C))$.

For any $m_a, m_b, r' \in \mathbb{Z}_N$ under the same $pk$, we have:

$$E(m_a, pk)E(m_b, pk) = E(m_a + m_b, pk) \bmod N^2, \quad (1)$$

$$[E(m_a, pk)]^{m_b} = E(m_a m_b, pk) \bmod N^2, \quad (2)$$

$$E(m_a, pk) r'^N = E(m_a, pk) \bmod N^2, \quad (3)$$

$$[E(m_a, pk)]^{N-1} = E(-m_a, pk) \bmod N^2. \quad (4)$$

The PCTD has been proven to be semantically secure assuming the intractability of the Partially Discrete Logarithm (PDL) problem [21,22].

#### 2.4.2. Density peaks (DP) clustering algorithm

The DP algorithm [34] is based on two observations. Firstly, any cluster center should be surrounded by neighboring points with lower local densities. Secondly, each cluster center is far away from other points with high local densities. Thus, DP algorithm [34] defines two metrics, $\rho_i$ and $\delta_i$, to quantify the local

density of a data point $i$ and the minimum distance from any other points with high densities to the data point $i$. These two metrics are used to locate density peaks that are indeed the cluster centers.

Given a data point $i$, the local density $\rho_i$ and the distance $\delta_i$ of the point are computed as:

$$\rho_i = \Sigma_j \chi(d_{i,j} - d_c),\ \chi(x) = \begin{cases} 1 & \text{if } x < 0, \\ 0 & \text{otherwise,} \end{cases} \qquad (5)$$
$$\delta_i = min_{j:\rho_j > \rho_i}(d_{i,j}),$$

where $d_c$ is the cutoff distance. In other words, $\rho_i$ is equal to the number of data points within cutoff distance $d_c$. Suppose $j' = argmin_{j:\rho_j > \rho_i}(d_{i,j})$, the point $i$ is assigned to the point $j'$ which means the upslope point of the point $i$. If point $i$ has the highest density among all data points, the point is called the absolute density peak and the corresponding parameter $\delta_i$ is set as $\delta_i = max_j(d_{i,j})$. As the parameter values $\gamma_i = \rho_i\delta_i$ [8,34] of cluster centers are much higher than that of other points, the cluster centers can be distinguished from other points. After selecting cluster centers, we can assign the remaining points to the corresponding cluster centers straightforwardly according to their upslope points.

## 3. Proposed privbus

In this section, we propose Privbus for the purpose of providing privacy-preserving CBS and discuss its security.

### 3.1. Overview

Privbus is organized as follows: initialization, fog-assisted DP clustering, privacy-preserving trip clustering, and bus route planning. Table 1 shows the notations throughout the paper.

In order to realize Privbus, additive homomorphic technique appears to be a suitable solution for our system. Additive homomorphic encryption schemes, such as the Paillier cryptosystem [30], Bresson cryptosystem [5] and BGN cryptosystem [4], allow other parties to perform some additive calculations over the ciphertext. Compared with Bresson's scheme, Paillier's scheme is more efficient in terms of bandwidth and decryption procedure [5,30]. BGN cryptosystem only supports limited number of additive homomorphic operations [4]. Thus we utilize PCTD [21] which is based on Paillier cryptosystem as the building block of Privbus. Meanwhile, it is important to ensure that the outsourced encrypted data cannot be manipulated to compromise the privacy of users by fog nodes or the server. The PCTD [21] follows the idea in [11] and separates the private key into 2 different shares to support (2, 2) threshold decryption.

In initialization, Privbus is bootstrapped by using the key generation algorithm of the PCTD cryptosystem for all entities. The cutoff distance $d_c$ is defined for DP clustering. The public parameters of bus route optimization include $\eta$, $\beta_k$, $\alpha$ and $\mu$. Users $(u_{i,j}, i = 1, \ldots, m, j = 1, \ldots, n_i)$ create tuples $(\{tr_{i,j}\}_{(i,j)=(1,1)}^{(m,n_i)})$ to denote their trip requests.

In fog-assisted DP clustering, fog nodes partition the set of all users into many disjoint subsets according to the data locality property and execute comparison operations inside each partition, hence the number of comparison operations for DP clustering is reduced. Meanwhile, to guarantee the clustering quality, each user creates a set of fog nodes that are neighbors of the user's source location, called the neighboring fog node set. Since such neighboring fog node sets can make comparison operations self-contained within each fog node's crowdsourced dataset, the correctness of indicator $(\rho, \delta)$ calculation is guaranteed.

Privacy-preserving trip clustering is to perform the fog-assisted DP clustering with privacy-preserving techniques. As shown in Fig. 2, this privacy-preserving mechanism of trip clustering includes the following 7 steps.

(a) Each user $u_{i,j}$ hides trip $tr_{i,j}$ and its neighboring fog node set $F_{i,j}$ in her trip request $B_{i,j}$, and then submits it to fog node $f_i$.

(b) After crowdsourcing trip requests in its service region, the fog node preprocesses these requests to collect more fresh encrypted trips from its $d_0$-length neighbors outside its service region, for the purpose of maintaining the clustering quality. Then, each fog node refreshes all encrypted trips and forwards them to CBS server.

(c) CBS server adds random messages to these encrypted trips, partially decrypts them and sends to the fog node.

(d) The fog node obtains noise-added trips by running $D_T(\cdot)$ algorithm with its partially private key, and then uses them to calculate a set of ciphertexts of the distance values ($B_4^{(i)}$) between any noise-added trip in $U_i$ and another one in $U_i \cup U_i'$.

(e) After receiving the ciphertext, CBS server removes noises hidden in the encrypted distances by utilizing additive homomorphism property of the PCTD cryptosystem.

(f) CBS server further uses a randomization rule supporting comparison on any two encrypted distances and outputs $B_6^{(i)}$. Thus, fog node $f_i$ calculates $\rho_{i,j}$ by directly comparing the randomized distances after decrypting ciphertext $B_6^{(i)}$ and obtaining randomized distances.

(g) CBS server and fog node finally run DP-based Trip Clustering Algorithm (Algorithm 1) together to obtain cluster centers and the member sets belonging to these clusters.

According to the clustering result, CBS server cooperates with fog nodes to provide the customized bus routes for users, and further optimizes these routes on the purpose to increase the service provider's profit and the service coverage. Each user's ticket information is encrypted under the user's public key, and then is sent to the corresponding fog node. When users are online, they download and decrypt it to obtain bus information. If users accept the bus routes, they could pay for the ticket and enjoy the trip provided by CBS.

### 3.2. The detailed privbus

**(1) Initialization.**

Given the security parameter $\kappa$, the Key Generation Center (KGC) generates a public–private key pair of the PCTD cryptosystem by running the algorithm $(pk, sk) \leftarrow KeyGen(\kappa)$. The private key $sk$ is randomly separated into two partially private keys $m$ times by using $KeyS(sk)$, denoted as $sk^{(1,i)}$ and $sk^{(2,i)}$, $i = 1, 2, \ldots, m$, where $sk^{(1,i)}$ is the partially private key of fog node $f_i$ and $sk^{(2,i)}$ is that of CBS server. The KGC also runs $KeyGen(\kappa)$ to generate a public–private key pair of any fog node $f_i \in F$ and that of each user $u_{i,j}$, denoted as $(pk_i, sk_i)$ and $(pk_{i,j}, sk_{i,j})$, respectively.

The server sets the public parameters $d_c = (d_0, d_0, t_0)$ and $\eta$. $d_c$ indicates the cutoff distance used in the subsection of DP-based trip clustering. $d_0$ and $t_0$ denote the maximum values of any user's reasonable walking distance and waiting time, e.g., $d_0 = 500$ m and $t_0 = 12$ min, respectively. $\eta$ refers to the upper bound of the ratio of traveling miles of bus trips provided by Privbus to that of taxi trips, e.g., $\eta = 1.5$. $\eta$ is used to avoid long traveling miles brought by merging clusters into a route.

The fog node set is denoted by $F = \{f_i\}_{i=1}^m$. The locations of fog nodes are public information. The coverage radius of a fog node generally ranges from 500m to 3km [23]. The fog node $f_i$ serves users holding mobile devices within the scope of cell it covers, who are denoted as $U_i = \{u_{i,j}\}_{j=1}^{n_i}$.

The set $U = \cup_{i=1}^m U_i$ includes all users in CBS system. User $u_{i,j}$ sets bus trip $tr_{i,j} = (s_{i,j}, d_{i,j}, t_{i,j})$, where $s_{i,j} = (s_{x,i,j}, s_{y,i,j})$ and $d_{i,j} = (d_{x,i,j}, d_{y,i,j})$ denote the coordinates of the source and destination points, respectively, and $t_{i,j}$ refers to the expected arrival time.

**Table 1**
Notations.

| Notations | Descriptions |
|---|---|
| $\kappa$ | The security parameter |
| $\mathbb{Z}_{N^2}^*$, $\mathbb{Z}_N$ | The cyclic multiplicative groups of $N^2$ and $N$ in PCTD |
| $f_i$ | A fog node $f_i \in F = \{f_i\}_{i=1}^m$ |
| $u_{i,j}$ | The pseudonym of a user within $f_i$'s service region |
| $pk_{i,j}$, $sk_{i,j}$ | The $u_{i,j}$'s public and private key pair |
| $pk_i$, $sk_i$ | The $f_i$'s public and private key pair |
| $pk$, $sk$ | The system's public and private key pair |
| $sk^{(s,i)}$ | The $sk$ is separated into different sharing, $s = 1, 2$ |
| $tr_{i,j}$ | $u_{i,j}$'s trip, $tr_{i,j} = (s_{i,j}, d_{i,j}, t_{i,j})$, denoting source, destination and the expected arrival time |
| $d_c$, $\overline{d_c}$, $\overline{d_c'}$ | The cutoff distance, its squared value and the randomized $(d_c)^2$ |
| $d(tr_{i,j}, tr_{\tau,\varsigma})$ | The distance vector between two trips |
| $\overline{d}(\cdot,\cdot)$, $\overline{d'}(\cdot,\cdot)$ | A squared distance vector and randomized $\overline{d}(\cdot,\cdot)$ |
| $\rho_i^{(j)}$, $\delta_{i,j}$, $\gamma_{i,j}$ | The local density of $tr_{i,j}$, the minimum distance from others with higher density to $tr_{i,j}$, and $\gamma_{i,j} = \rho_i^{(j)}\delta_{i,j}$ |
| $F_{i,j}$ | The $u_{i,j}$'s neighboring fog node set |
| $B_{i,j}$ | The $u_{i,j}$'s bus trip request |
| $B_2^{(i)}$ | The crowdsourced trip requests from users inside and outside the $f_i$'s coverage, $B_2^{(i)} = U_i \cup U_i'$ |
| $\sigma_{i,j}^{(i)}$ | The noise added in the ciphertext of $tr_{i,j}$ |
| $B_3^{(i)}$ | The ciphertexts of the noise-added trips in $B_2^{(i)}$ |
| $B_4^{(i)}$ | The ciphertexts of the distances between randomized trips in $B_3^{(i)}$ |
| $B_5^{(i)}$ | The ciphertexts of the distances in $B_4^{(i)}$ after removing the noise |
| $C_{i,j,\tau,\varsigma}^{(0)}$, | The fresh encrypted distance $\overline{d}(tr_{i,j}, tr_{\tau,\varsigma})$ |
| $B_6^{(i)}$ | The ciphertexts of $\overline{d_c'}$ and $\overline{d'}(\cdot,\cdot)$ from $f_i$ |
| $c_k$ | The $k$th cluster center $c_k = tr_{i_k,j_k}$, $k = 1, 2, \ldots, K$ |
| $\Gamma_k$ | All trips belonging to the cluster $c_k$, $\lvert\Gamma_k\rvert = L_k$ |
| $\triangle K_{p-p}$ | The clusters served by point–point buses |
| $b_k'$, $b_{\cdot,\cdot}'$ | The results of point–point bus and optimized shuttle bus services |
| $\beta_k$, $\alpha$, $\mu$ | The ticket price per seat, the cost per unit distance per seat and the fixed cost for running a bus |
| $R_m$ | A direct path on the bus routing graph, $R_m \in \mathcal{R}$ |
| $\mathcal{R}^*$ | The result of routes optimization for CBS |
| $e(a, b)$ | A direct edge on a path in $\mathcal{R}$ |
| $p_a^l$ | The passenger load of $e(a, b)$ |
| $p$ | The maximum passenger load of a bus |

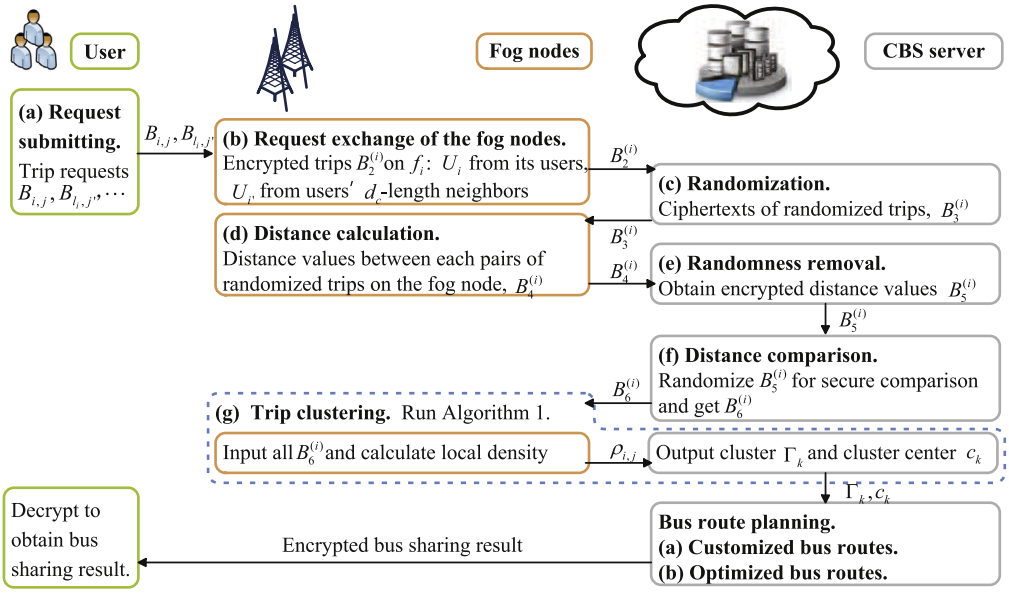



**Fig. 2.** Information Flow of Privbus.

For briefness, the range of each component is mapped into a set of integers.

**(2) Fog-assisted DP Clustering**.

In this subsection, we combine DP clustering and fog computing to improve the performance of implementing DP clustering without the clustering quality loss.

The DP clustering algorithm needs to compute $\lceil \frac{|U|(|U|-1)}{2} \rceil$ distances for the computation of the local density $\rho$ and the distance $\delta$ defined in Eq. (5). The computational cost is quadratic with respect to the total number of users $|U| = \sum_{i=1}^m |U_i|$. In order to improve the performance of clustering computation, we use the data locality property of the fog node to partition the set of all users $U$ into $m$ disjoint subsets $\{U_i\}_{i=1}^m$. If a user's source location is located in a fog node $f_j$'s service region, the user is within the partition $U_i = \{u_{i,j}\}_{j=1}^{n_i}$. If $f_i$ only calculates the distances

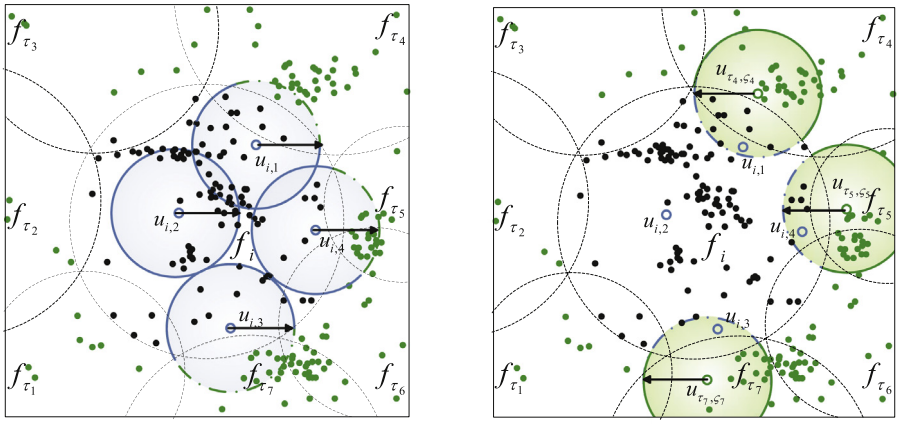



**Fig. 3.** Users' neighboring fog node sets: (a) Users inside the service area of fog node $f_i$; (b) Users outside the service area of fog node $f_i$.

between users' trips inside $U_i$, the computational cost would be significantly reduced.

However, the calculation of $\rho$ and $\delta$ fails to be self-contained within a partition generated by a fog node, since some users' $d_c$-length neighbors are usually not in the corresponding partition. As shown in Fig. 2(a), when sources of users $u_{i,1}$, $u_{i,3}$ and $u_{i,4}$ are located near the border line of $f_i$, a large number of their $d_c$-length neighbors are located outside $f_i$'s service area. If only these $d_c$-length neighbors inside $U_i$ are counted, the users' local density values ($\rho_{i,1}$, $\rho_{i,3}$ and $\rho_{i,4}$) would be incorrect.

To guarantee the correctness of $\rho$ and $\delta$ in fog-assisted DP clustering, $u_{i,j}$ should create a neighboring fog node set, denoted as $F_{i,j} = \{f_i, f_{l_{i,j}}, l = 1, 2, \ldots, d_f - 1\}$. The adjacent fog nodes in $F_{i,j}$ cover users whose source and $u_{i,j}$'s source are $d_0$-length neighbors. According to the general distribution of fog nodes presented in Fig. 2(a), the range of $d_f$ is $[1, 4]$. The neighboring fog node sets help guarantee the correctness of implementing DP clustering according to the following analysis:

**The Correctness of** $\rho$. As shown in Fig. 2(a), users $u_{i,1}$, $u_{i,3}$ and $u_{i,4}$ create their neighboring fog node sets $F_{i,1} = \{f_i, f_{\tau_4}\}$, $F_{i,3} = \{f_i, f_{\tau_1}, f_{\tau_6}, f_{\tau_7}\}$ and $F_{i,4} = \{f_i, f_{\tau_5}, f_{\tau_6}\}$, respectively. If each user hides the information about her neighboring fog node set in her trip request, $f_i$ would enable these neighboring fog nodes to obtain the user's fresh encrypted trips. Meanwhile, users $u_{\tau_4,\varsigma_4}$, $u_{\tau_5,\varsigma_5}$ and $u_{\tau_7,\varsigma_7}$ create $F_{\tau_4,\varsigma_4} = \{f_{\tau_4}, f_i\}$, $F_{\tau_5,\varsigma_5} = \{f_{\tau_5}, f_{\tau_4}, f_{\tau_6}, f_i\}$ and $F_{\tau_7,\varsigma_7} = \{f_{\tau_7}, f_{\tau_1}, f_i\}$, respectively, as shown in Fig. 2(b). Thus, $f_i$ obtains the ciphertext of $tr_{\tau_4,\varsigma_4}$, $tr_{\tau_5,\varsigma_5}$ and $tr_{\tau_7,\varsigma_7}$ under the public key $pk$. In this way, for each $u_{i,j} \in U_i$, $f_i$ is able to crowdsource the ciphertexts of all $d_0$-length neighbors of the user's source location $s_{i,j}$. Besides, if two users' trips are $d_c$-length neighbors, their source locations would be $d_0$-length neighbors according to $d_c = (d_0, d_0, t_0)$. Thus, all of $u_{i,j}$'s $d_c$-length neighbors are included in the crowdsourced dataset on $f_i$. That makes it self-contained to the calculation of $\rho$, hence the correctness of $\rho$ is guaranteed.

**The Correctness of** $\delta$. When a user's $\delta$ value is less than $d_c$, the user's upslope point would be found from her $d_c$-length neighbors. In this case, the parameter $\delta$ can be calculated correctly by ranking $\rho$ values, since the correctness of $\rho$ is guaranteed. When a user's $\delta$ is larger than $d_c$, the user definitely has the highest local density among all of her $d_c$-length neighbors. This is because the radius of a cluster should not be larger than $d_c$, where $d_c = \{d_0, d_0, t_0\}$ refers to the upper bound of any user's walking distance and that of waiting time in our scenario. In that way, the user should be its own upslope point and a cluster center. Hence, $\delta$ is self-contained in the crowdsourced trip dataset stored on each fog node.

Therefore, the correctness of the indicator $(\rho, \delta)$ calculation is guaranteed in fog-assisted DP clustering due to the users' neighboring fog node sets. In other words, the correctness of implementing DP clustering is guaranteed.

$B_{i,j} \leftarrow \text{BusGen}(tr_{i,j}, pk, pk_i, pk_{l_{i,j}}, F_{i,j})$ on $u_{i,j}$

Input trip $tr_{i,j} = \{s_{i,j}, d_{i,j}, t_{i,j}\}$, adjacent fog nodes' public keys $\{pk_{l_{i,j}}\}_{i=1}^{d_f-1}$, $f_{l_{i,j}} \in F_{i,j}$, the fog node's public key $pk_i$, the server's public key $pk$ and random integer vector $r_{i,j}^{(3)}$.

Output trip request $B_{i,j} = \{pk_{i,j}, C_{i,j}^{(1)}, C_{i,j}^{(2)}, C_{i,j}^{(3)}\}$:

$C_{i,j}^{(1)} \leftarrow E(tr_{i,j}, pk)$,

$C_{i,j}^{(2)} \leftarrow E(r_{i,j}^{(3)}, pk_i), C_{i,j}^{(3)} \leftarrow \{E(r_{i,j}^{(3)} \| h(r_{i,j}^{(3)}), pk_{l_{i,j}})\}_{l=1}^{d_f-1}$

**Fig. 4.** Request generation.

**(3) Privacy-preserving Trip Clustering**.

In this subsection, the fog-assisted DP clustering is performed with privacy preserving techniques.

**(a) Request Submitting**. As shown in Fig. 4, the user $u_{i,j}$ generates bus trip request $B_{i,j}$ by encrypting trip $tr_{i,j}$, random $r_{i,j}^{(3)}$ and $r_{i,j}^{(3)} \| h(r_{i,j}^{(3)})$ under the public key $pk$, $pk_i$ and $pk_{l_{i,j}}$ ($f_{l_{i,j}} \in F_{i,j}$), respectively. After submitting trip request $B_{i,j}$ to $f_i$, $u_{i,j}$ can be offline and wait the bus ticket.

**(b) Request Exchange of Fog Nodes**. This step is to further crowdsource the fresh encrypted trips outside the fog node's service region, which would make it self-contained to calculate local density $\rho$. It includes operations $\text{RanC}(B_{i,j}, sk_i)$ and $\text{Fresh}(C_{i,j}^{(3)}, C_{i,j}^{(4)}, sk_{l_i})$ as presented in Fig. 5.

In $\text{RanC}(B_{i,j}, sk_i)$, $f_i$ decrypts ciphertext $C_{i,j}^{(2)} \in B_{i,j}$ to obtain secret random integer $r_{i,j}^{(3)}$ and calculates $C_{i,j}^{(4)} = g^{r_{i,j}^{(3)}} C_{i,j}^{(1)}$. Then ciphertext $\{pk_i, C_{i,j}^{(3)}, C_{i,j}^{(4)}\}$ is sent to adjacent fog nodes.

$f_{l_{i,j}}$ denotes one of $f_i$'s adjacent fog nodes. In $\text{Fresh}(C_{i,j}^{(3)}, C_{i,j}^{(4)}, sk_{l_{i,j}})$, if at least one equation $m'_{l,i,j} = h(m'_{l,i,j})$ holds in range $l \in [1, d_f - 1]$, $f_{l_{i,j}}$ can remove $g^{r_{i,j}^{(3)}}$ from $C_{i,j}^{(4)}$ to obtain $u_{i,j}$'s fresh ciphertext $C_{i,j}^{(1)}$ and refresh it according to Eq. (3). Meanwhile, $f_i$ executes $\text{Fresh}(\cdot)$ to obtain some users' fresh encrypted trips, whose source locations are $d_0$-length neighbors of $f_i$'s users and inside the adjacent fog nodes' service regions.

$C_{i,j}^{(4)} \leftarrow \mathrm{RanC}(B_{i,j}, sk_i)$ on $f_i$

Input trip request $B_{i,j}$ and fog node $f_i$'s private key $sk_i$.
Output randomized trips $C_{i,j}^{(4)}$ :

$$r_{i,j}^{(3)} = D(C_{i,j}^{(2)}, sk_i), C_{i,j}^{(4)} = g^{r_{i,j}^{(3)}} C_{i,j}^{(1)} \bmod N^2.$$

$C_{i,j}^{(1)} \leftarrow \mathrm{Fresh}(C_{i,j}^{(3)}, C_{i,j}^{(4)}, sk_{l_{i,j}})$ on $f_{l_{i,j}}$

Input fog node $f_{l_{i,j}}$'s private key $sk_{l_{i,j}}$, randomized trip $\{C_{i,j}^{(3)}, C_{i,j}^{(4)}\}$ from adjacent fog node $f_i$ :
Calculate
$m_{l,i,j} = m'_{l,i,j} \left\| m''_{l,i,j} = D(C_{l,i,j}^{(3)}, sk_{l_{i,j}}), C_{l,i,j}^{(3)} \in C_{i,j}^{(3)}.\right.$
If $m''_{l,i,j} = h(m'_{l,i,j})$ holds, $r_{i,j}^{(3)} = m'_{l,i,j}$.
Remove randomness and obtain $C_{i,j}^{(1)} = r_{i,j}'^{N} C_{i,j}^{(4)} / g^{r_{i,j}^{(3)}} \bmod N^2$.
Output encrypted trip $C_{i,j}^{(1)}$.

**Fig. 5.** Fog node-to-fog node interaction.

$f_i$ divides the crowdsourced dataset into two disjoint subsets

$$U_i = \{(u_{i,j}, pk_{i,j}, C_{i,j}^{(1)})\}_{j=1}^{n_i}, U_i' = \{(u_{l,j'}, pk_{l,j'}, C_{l,j'}^{(1)})\}_{l \neq i}$$

for users inside and outside $f_i$'s service region, respectively. The fog node sends $B_2^{(i)} = \{U_i, U_i'\}$ to CBS server.

**(c) Randomization**. For each user $u_{\tau,\varsigma} \in U_i \cup U_i'$ in $f_i$, CBS server chooses random values $\sigma_{\tau,\varsigma}^{(i)} \in (\mathbb{Z}_N)^5$ and employs additive homomorphism encryption to calculate

$$C_{\tau,\varsigma}^i = E(tr_{\tau,\varsigma} + \sigma_{\tau,\varsigma}^{(i)}, pk) = C_{\tau,\varsigma}^{(1)} E(\sigma_{\tau,\varsigma}^{(i)}, pk).$$

Then CBS server sends $B_3^{(i)} = \{C_{\tau,\varsigma}^i, D_{sk^{(2,i)}}(C_{\tau,\varsigma}^i)\}_{u_{\tau,\varsigma} \in U_i \cup U_i'}$ to $f_i$, where $D_{sk^{(2,i)}}(C_{\tau,\varsigma}^i)$ is obtained by using the partially decryption algorithm.

**(d) Distance Calculation**. The Euclidean distance $d(\cdot)$ is used to measure the distance between $tr_{i,j}$ ($u_{i,j} \in U_i$) and $tr_{\tau,\varsigma}$ ($u_{\tau,\varsigma} \in U_i \cup U_i'/\{u_{i,j}\}$), i.e.,

$$d(tr_{i,j}, tr_{\tau,\varsigma}) = (d(s_{i,j}, s_{\tau,\varsigma}), d(d_{i,j}, d_{\tau,\varsigma}), d(t_{i,j}, t_{\tau,\varsigma})). \tag{6}$$

For easy reading, we use squared Euclidean distance value $\overline{d}(\cdot,\cdot)$ instead of distance $d(\cdot,\cdot)$.

$f_i$ partially decrypts $C_{\tau,\varsigma}^i$ with its partially private key $sk^{(1,i)}$ and uses $D_T(\cdot)$ to obtain the randomized trip $m_{\tau,\varsigma}^{(i)} = tr_{\tau,\varsigma} + \sigma_{\tau,\varsigma}^{(i)}$. Then, $f_i$ calculates the squared distance $\overline{d}(m_{i,j}^{(i)}, m_{\tau,\varsigma}^{(i)})$ and encrypts it with the public key $pk$ to obtain the ciphertext of the randomized distance

$$C_{i,j,\tau,\varsigma}^{(i)} \leftarrow E(\overline{d}(m_{i,j}^{(i)}, m_{\tau,\varsigma}^{(i)}), pk).$$

$f_i$ sends $B_4^{(i)} = \{C_{i,j,\tau,\varsigma}^{(i)} | u_{i,j} \in U_i, u_{\tau,\varsigma} \in U_i \cup U_i'/\{u_{i,j}\}\}$ to CBS server.

**(e) Randomness Removal**. This step is to remove random noise hidden in the ciphertext $B_4^{(i)}$.

Since

$$\begin{aligned} &\overline{d}(tr_{i,j}, tr_{\tau,\varsigma}) - \overline{d}(m_{i,j}^{(i)}, m_{\tau,\varsigma}^{(i)}) \\ &= -2(\sigma_{\tau,\varsigma}^{(i)} - \sigma_{i,j}^{(i)})(tr_{\tau,\varsigma} - tr_{i,j}) - \overline{d}(\sigma_{i,j}^{(i)}, \sigma_{\tau,\varsigma}^{(i)}), \end{aligned} \tag{7}$$

the server calculates $\pm\sigma_{i,j,\tau,\varsigma}^{(i)} = \pm 2(\sigma_{\tau,\varsigma}^{(i)} - \sigma_{i,j}^{(i)})$ and

$$C'_{i,j,\tau,\varsigma} \leftarrow E(\overline{d}(\sigma_{i,j}^{(i)}, \sigma_{\tau,\varsigma}^{(i)}), pk),$$

$$C_{i,j,\tau,\varsigma}^0 = C_{i,j,\tau,\varsigma}^{(i)} (C_{i,j}^{(1)})^{\sigma_{i,j,\tau,\varsigma}^{(i)}} (C_{\tau,\varsigma}^{(1)})^{-\sigma_{i,j,\tau,\varsigma}^{(i)}} C'_{i,j,\tau,\varsigma},$$

where $C_{i,j,\tau,\varsigma}^0$ is equal to the encrypted $\overline{d}(tr_{i,j}, tr_{\tau,\varsigma})$.

**(f) Distance Comparison**. This step is to support comparison on any two encrypted distances without learning the true values.

For three arbitrary integers $r, r'$ and $r''$ such that $r > r' > r'' \geqslant 0$, inequality $0 < \frac{r'-r''}{r} < 1$ holds. Hence, the inequality

$$d' < d'' \Leftrightarrow d' + \frac{r' - r''}{r} < d'' \Leftrightarrow rd' + r' < rd'' + r'' \tag{8}$$

holds, if $d'$ and $d''$ are integers. Similarly, if $r > r_1', r_2' \geqslant 0$, $-1 < \frac{r_1'-r_2'}{r} < 1$ holds. Hence, for integers $d_1$ and $d_2$, we have

$$rd_1 + r_1' < rd_2 + r_2' \Leftrightarrow d_1 < d_2 + \frac{r_2' - r_1'}{r} \Rightarrow d_1 \leqslant d_2. \tag{9}$$

For 3-tuple $\overline{d}(tr_{i,j}, tr_{\tau,\varsigma})$, CBS server randomly chooses secret parameters $r_{i,j} > r'_{i,j,\tau,\varsigma} > r''_{i,j,\tau,\varsigma} \in (\mathbb{Z}_N)^3$ and sets

$$\overline{d_c'} = r_{i,j}\overline{d_c} + r''_{i,j,\tau,\varsigma}, \overline{d'}(tr_{i,j}, tr_{\tau,\varsigma}) = r_{i,j}\overline{d}(tr_{i,j}, tr_{\tau,\varsigma}) + r'_{i,j,\tau,\varsigma}.$$

According to Eqs. (8) and (9), we have

$$\overline{d'}(tr_{i,j}, tr_{\tau,\varsigma}) < \overline{d_c'} \Leftrightarrow \overline{d}(tr_{i,j}, tr_{\tau,\varsigma}) < \overline{d_c}, \tag{10}$$

$$\overline{d'}(tr_{i,j}, tr_{\tau,\varsigma}) < \overline{d'}(tr_{i,j}, tr_{\tau',\varsigma'}) \Rightarrow \overline{d}(tr_{i,j}, tr_{\tau,\varsigma}) \leqslant \overline{d}(tr_{i,j}, tr_{\tau',\varsigma'}). \tag{11}$$

Thus CBS server encrypts $\overline{d'}(tr_{i,j}, tr_{\tau,\varsigma})$ and $\overline{d_c'}$ by calculating

$$C'_{i,j,\tau,\varsigma} \leftarrow E(\overline{d'}(tr_{i,j}, tr_{\tau,\varsigma}), pk) = (C_{i,j,\tau,\varsigma}^0)^{r_{i,j,\tau,\varsigma}} E(r'_{i,j,\tau,\varsigma}, pk),$$

$$C''_{i,j,\tau,\varsigma} \leftarrow E(\overline{d_c'}, pk).$$

The server sends $B_6^{(i)}$ to $f_i$, where

$$B_6^{(i)} = \{D_{sk^{(2,i)}}(C'_{i,j,\tau,\varsigma}), D_{sk^{(2,i)}}(C''_{i,j,\tau,\varsigma}) | u_{i,j} \in U_i, u_{\tau,\varsigma} \in U_i \cup U_i'/\{u_{i,j}\}\}.$$

**(g) Trip Clustering**. $f_i$ runs lines 1–9 (Algorithm 1) to obtain the user $u_{i,j}$'s local density and his upslope trip $tr_{l_0,j_0}$. After receiving the user's local density and ciphertexts of the upslope trip, CBS server then runs lines 10–14 (Algorithm 1) to calculate the user's distance $\delta_{i,j}$ and the parameter $\gamma_{i,j}$. Finally, CBS server selects the cluster center $c_k = tr_{i_k,j_k}$ according to the parameter $\gamma_{i,j}$ and obtains the corresponding member set $\Gamma_k$, $|\Gamma_k| = L_k$, $k = 1, 2, \ldots, K$ on lines 15–20 (Algorithm 1). $tr_{i,j} \in \Gamma_k$ means that the trip $tr_{i,j}$ belongs to the $k$th cluster. The distance between a trip and its cluster center is required to be less than the cutoff distance value $d_c$.

**(4) Bus Route Planning**.

In this phrase, CBS server and fog nodes provide the customized bus routes for CBS according to the trip clustering result, and further optimize the bus routes with the objective of maximizing its profit.

The source locations and destination locations of all cluster centers are potential bus stops, forming a set $\mathcal{C} = \{s_{l_k}, d_{l_k}\}_{k=1}^K$. Bus stops candidates in $\mathcal{C}$ serve as vertices of a graph. The weight of an edge from $a$ to $b$ in $\mathcal{C}$ refers to the driving miles from $a$ to $b$, denoted as $d(a, b)$. $L_k$ denotes the passenger load on the trip from source point $s_{l_k}$ to destination point $d_{l_k}$. CBS server plans bus routes on the graph and arranges buses to pick users up.

**(a) Customized Bus Routes.** CBS server customizes bus routes to serve users in the clusters such that $\triangle K_{p-p} = \{k | L_k \geq p, k = 1, \ldots, K\}$, where $p$ is the maximum number of passengers a bus can carry. The source point and destination point of every cluster center $\{c_k\}_{k \in \triangle K_{p-p}}$ can be the candidates of bus stops. CBS server arranges $b_k = \lceil \frac{L_k}{p} \rceil$ buses for $b_k n_c$ members, who are chosen from the $k$th cluster ( $k \in \triangle L_{p-p}$ ) in chronological or random order.

To be specific, CBS server communicates with fog nodes $\{f_{i_k}\}_{k \in \triangle K_{p-p}}$ and requires information of these cluster centers, i.e., $\{C_{i_k,j_k}^{(1)}, D_{sk^{(1,i)}}(C_{i_k,j_k}^{(1)})\}_{k \in \triangle K_{p-p}}$. Then, CBS server decrypts it and obtains each $c_k = tr_{i_k,j_k} = D_T(C_{i_k,j_k}^{(1)})$. For each $k \in \triangle K_{p-p}$, CBS server further estimates the route length $d_k = d(s_{i_k,j_k}, d_{i_k,j_k})$ and the departure time $t_{i_k,j_k}^0$ to ensure that the selected users can arrive at $d_{i_k,j_k}$ before the expected arriving time $t_{i_k,j_k}$. CBS server encrypts the bus ticket $b_k' = (s_{i_k,j_k}, d_{i_k,j_k}, t_{i_k,j_k}^0)$ under each purchaser's

**Algorithm 1** DP-Based Trip Clustering Algorithm

**Input:** The public–private key pair $(pk, sk)$, cutoff distance $d_c$, all received ciphertexts for comparison $\{B_6^{(i)}\}_{f_i \in F}$;
**Output:** Cluster $U_k$ and cluster center $c_k$, $k = 1, 2, \ldots, K$;
**Fog node $f_i$ works as:**
**for** each user pair$(u_{i,j}, u_{\tau,\varsigma})$ **do**
  $\overline{d'}(tr_{i,j}, tr_{\tau,\varsigma}) = D_T(C'_{i,j,\tau,\varsigma})$, $\overline{d'_c} = D_T(C''_{i,j,\tau,\varsigma})$;
  $\chi(d(tr_{i,j}, tr_{\tau,\varsigma}) - d_c) = \chi(\overline{d'}(tr_{i,j}, tr_{\tau,\varsigma}), \overline{d'_c})$; % Count the number of the user's $d_c$-length neighbors based on Eq. (10)
  $\rho_{i,j} = \sum_{u_{\tau,\varsigma} \in U_i \cup U'_i/\{u_{i,j}\}} \chi(d(tr_{i,j}, tr_{\tau,\varsigma}) - d_c)$; % $u_{i,j}$'s local density
**end for**
**for** each user $u_{i,j}$ **do**
  $u_{l_0,j_0} = arg(min_{u_{\tau,\varsigma} \in \Delta}\{\overline{d'}(tr_{i,j}, tr_{\tau,\varsigma})\})$, $\Delta = \{u_{\tau,\varsigma} \in U_i \cup U'_i/\{u_{i,j}\} : \rho_{\tau,\varsigma} > \rho_{i,j}\}$; % Pick the upslope trip $tr_{l_0,j_0}$ of trip $tr_{i,j}$ according to Eq. (11)
  Send $\{C^0_{i,j,l_0,j_0}, D_{sk^{(2,i)}}(C^0_{i,j,l_0,j_0}), \rho_{i,j}\}$ to CBS server;
**end for**
**CBS server works as:**
**for** each user $u_{i,j}$ **do**
  $\overline{d}(tr_{i,j}, tr_{l_0,j_0}) = D_T(C^0_{i,j,l_0,j_0})$; % Recover the distance value
  $\delta_{i,j} = d(tr_{i,j}, tr_{l_0,j_0}) = \overline{d}(tr_{i,j}, tr_{l_0,j_0})^{\frac{1}{2}}$; % $u_{i,j}$'s distance
  $\gamma_{i,j} = \rho_{i,j} \times \delta_{i,j}$
**end for**
Selects clustering centers $\{c_k = tr_{i_k,j_k}\}_{k=1}^K$ according to $\gamma_{i,j}$,
$\Gamma_0 = U, k = 1$;
**while** $\Gamma_0 \neq \phi$ **do**
  $c_k = tr_{i_k,j_k} = arg(max_{u_{i,j} \in \Gamma_0} \gamma_{i,j})$; % Center of the $k$-th cluster
  set $\Gamma_k = \{u_{i,j} | \chi(d(tr_{i,j}, c_k) - d_c) = 1\}$; % Trips belong to the $k$-th cluster center
  $\Gamma_0 = \Gamma_0 - \Gamma_k, k = k + 1$ ;
**end while**

public key and sends it to fog node $f_{i_k}$. The purchaser receives it from $f_{i_k}$ and decrypts it with her private key to obtain the bus information $b'_k$.

With the customized bus routes, the users can get fast bus trips with no intermediate bus stops. CBS server can gain profit $\sum_{k \in \triangle K_{p-p}} [(\beta_k - \alpha d_k)p - \mu]b_k$ from CBS, where $\beta_k$, $\alpha$ and $\mu$ denote the ticket price per seat, the cost per unit distance per seat, and the fixed cost of running a bus, respectively.

**(b) Optimized Bus Routes.** As a CBS provider, CBS server has a strong incentive to optimize bus routes and maximize its profit.

Assuming clusters are served by some bus routes in $\mathcal{R}$. The route $R_m \in \mathcal{R}$ is a direct path on the bus routing graph:

$$R_m = \{(s_{i_k,j_k}, p^o_{s_{i_k,j_k}}, p^f_{s_{i_k,j_k}}, t^0_{s_{i_k,j_k}}) \ldots, (d_{i_k,j_k}, p^o_{d_{i_k,j_k}}, p^f_{d_{i_k,j_k}}, t^0_{d_{i_k,j_k}}), \ldots (d_{i_{k'},j_{k'}}, p^o_{d_{i_{k'},j_{k'}}}, p^f_{d_{i_{k'},j_{k'}}}, t^0_{d_{i_{k'},j_{k'}}})\}.$$

A bus route should ① start with the source point of any cluster center and end with the destination point of any cluster center; ② visit the source point of any cluster center before arriving at the corresponding destination point; ③ include both source points and destination points of some cluster centers.

Any two neighboring vertices in the route $R_m$ form an edge, i.e., $e(a, b), a, b \in R_m$. The weight of the edge $e(a, b)$ is $d(a, b)$, so the length of route $R_m$ is expressed by $\sum_{e(a,b) \in R_m} d(a, b)$. The numbers of passengers getting on and off the bus at location $a$ are denoted as $p^o_a$ and $p^f_a$, respectively. The passenger load of edge $e(a, b)$ is $p^l_a$. $t^0_a$ denotes that the bus leaves the location $a$ at the time $t^0_a$. The routes optimization problem can be formulated as the following system of equations:

$$max \sum_{R_m \in \mathcal{R}} [\sum_{e(a,b) \in R_m} (\beta d_a p^o_{R_m,a} - \alpha p d(a, b)) - \mu]$$
$$\begin{cases} p^l_a \leqslant p, \quad \sum_{R_m \in \mathcal{R}} p^o_a \leqslant L_a, \\ t^0_{d_{i_a,j_a}} - \sigma \leqslant t_{i_a,j_a}, \\ R_m.d(s_{i_a,j_a}, d_{i_a,j_a}) \leqslant \eta d(a), \\ d(a, b) \leqslant d(a, d_{i_{a'},j_{a'}}), d(b, s_{i_{a'},j_{a'}}) \leqslant d(a, s_{i_{a'},j_{a'}}), \\ \text{for } \forall R_m \in \mathcal{R}, \forall a', a, b \in \mathcal{C}, \text{ and} \\ \forall R_m.arg(s_{i_{a'},j_{a'}}) \leqslant R_m.arg(s_{i_a,j_a}) < R_m.arg(d_{i_{a'},j_{a'}}). \end{cases} \quad (12)$$

The first constraint ensures no overcrowding at each edge and limits the number of passengers getting on the bus at all sources; the second constraint requires the bus to arrive at each destination location before the corresponding expected arrival time, where $\sigma(\sigma = 5)$ denotes the average value of bus dwell time; the third constraint uses the public parameter $\eta$ $(\eta = 1.5)$ to avoid the long mileage brought by merging many clusters into a route, where $R_m.d(s_{i_a,j_a}, d_{i_a,j_a})$ represents the traveling distance from $s_{i_a,j_a}$ to $d_{i_a,j_a}$ in the route $R_m$ ; the fourth constraint forces the bus to get closer to the destination location of a cluster center during choosing the next bus stop; the last constraint ensures that all bus routes are directed paths on the graph, where $R_m.arg(s_{i_{a'},j_{a'}})$ denotes the serial number of the point $s_{i_{a'},j_{a'}}$ in the route $R_m$.

CBS server uses a greedy approach to solve the above routes optimization subject to the constraints of user demands in Eq. (12), and obtains the result $\mathcal{R}^* \subset \mathcal{R}$. The source and destination locations of any cluster center $c_{k^*_m}$ in the route $R^*_m \in \mathcal{R}^*$ are the pick-up and drop-off locations of passengers who belong to $c_{k^*_m}$, respectively. CBS server estimates the driving time of each route and arranges buses to serve the users. Then, $u_{i_{k^*_m},j_{k^*_m}}$ obtains her bus ticket result $b'_{i_{k^*_m},j_{k^*_m}}$, including her pick-up location, drop-off location, departure time and arrival time, the same way a user gets the result in (a). If the users accept the recommendation of bus tickets, they could enjoy a relatively fast and cheap bus trip. Compared with the customized bus routes in (a), the optimized bus routes increase the service provider's profit and improve its service coverage.

### 3.3. Security remarks

We discuss the security of Privbus and demonstrate that it achieves the goal of privacy preservation.

For any user $u_{i,j}$, CBS server learns her fresh encrypted trip $C^{(1)}_{i,j} \leftarrow E(tr_{i,j}, pk)$ and executes either modular multiplication or modular exponentiation on these ciphertexts. The private key is separated into two different shares and CBS server only obtains a partially private key [21]. Owning to the semantic security of the PCTD cryptosystem [22], the server cannot learn trip information hidden in these ciphertexts without knowing another partially private key which belongs to a fog node. For the purpose of customizing bus routes, CBS server is allowed to learn coordinates of each cluster center $c_k$, the local density $\rho_k$ and the minimum distance $\delta_k$ from the $k$th cluster center to any other points with higher density. Since each fog node is able to randomize ciphertexts before sending to CBS server, CBS server cannot link the encrypted trip or bus schedule to a specific user.

Each fog node only holds its own partially private key [21], which cannot decrypt the crowdsourced ciphertexts without CBS server's partially private key. Given the user pair $u_{i,j} \in U_i, u_{\tau,\varsigma} \in U_i \cup U'_i/\{u_{i,j}\}$, $f_i$ can gather information about their trips $tr_{i,j}$ and $tr_{\tau,\varsigma}$ shown in Eq. (13). From the perspective of the fog node, there are 29 unknown parameters in Eq. (13), consisting of 3 multivariate quadratic equations and 13 multivariate one power

**Table 2**
Complexity analysis.

| Schemes | | Privbus | | | NFog | | XHYCZ [37] | |
|---|---|---|---|---|---|---|---|---|
| Entity | | A user | Fog node $f_i$ | CBS server | A user | Server | A user | Server |
| Offline Comp. | $exp_1$ | $O(d_f)$ | – | – | $O(1)$ | – | – | – |
| | $exp_2$ | $O(1)$ | – | – | $O(1)$ | – | – | – |
| | $mul_2$ | $O(d_f)$ | – | – | $O(1)$ | – | – | – |
| | $h$ | $O(d_f)$ | – | – | – | – | – | – |
| Online Comp. | $exp_1$ | – | $O(n_i^2)$ | $O(n_i n)$ | $O(n)$ | $O(n^2)$ | $O(IK)$ | – |
| | $exp_2$ | – | $O(n_i^2)$ | $O(n_i n)$ | $O(n)$ | $O(n^2)$ | – | $O(IK)$ |
| | $mul_1$ | – | – | – | – | – | $O(IK)$ | $O(IK^2)$ |
| | $mul_2$ | – | $O(n_i^2)$ | $O(n_i n)$ | $O(n)$ | $O(n^2)$ | $O(IK(s-1))$ | $O(IK(n-1))$ |
| | $h$ | – | $O(n_i d_f)$ | – | – | – | – | – |
| Comm. trans (× 2048 bits) | Users | – | $O(n_i)$ | – | $O(n)$ | $O(n^2)$ | $O(IK(s-1)/s)$ | $O(IK^2 n)$ |
| | Fog node $f_i$ | $O(d_f)$ | $O(n_i d_f)$ | $O(n_i^2)$ | – | – | – | – |
| | Server | – | $O(n_i^2)$ | – | $O(n)$ | – | $O(IK)$ | – |

equations. Therefore, it is difficult for the fog node to identify their detailed trip information, i.e., $tr_{i,j}$, $tr_{\tau,\varsigma}$, $\overline{d}(tr_{i,j}, tr_{\tau,\varsigma})$.

$$\begin{cases} m_{i,j}^{(i)} = tr_{i,j} + \sigma_{i,j}^{(i)}, \\ m_{\tau,\varsigma}^{(i)} = tr_{\tau,\varsigma} + \sigma_{\tau,\varsigma}^{(i)}, \\ r_{i,j}\overline{d_c} + r''_{i,j,\tau,\varsigma} = \overline{d'_c} \\ \overline{d'}(tr_{i,j}, tr_{\tau,\varsigma}) = r_{i,j}\overline{d}(tr_{i,j}, tr_{\tau,\varsigma}) + r'_{i,j,\tau,\varsigma} \\ u_{i,j} \in U_i, u_{\tau,\varsigma} \in U_i \cup U'_i/\{u_{i,j}\}, \\ tr_{i,j}, tr_{\tau,\varsigma} \neq c_k, k = 1, 2, \ldots, K \end{cases} \tag{13}$$

The global eavesdroppers are only allowed to capture the information (including trips, random vectors, randomized trips, randomized distance values, and users' bus stops) encrypted in the PCTD cryptosystem. Therefore, the privacy of users' trips would not be exposed to the global eavesdroppers due to the semantic security of the PCTD cryptosystem [21,22].

Since the proposed fog-assistant approach does not need users to participate in the processes of trip clustering and route planing after users submit their trip requests, each user only can learn her own pick-up location, drop-off location, departure time and arrival time.

In conclusion, Privbus achieves our privacy-preserving goal.

## 4. Performance evaluation

In this section, we evaluate the performance of Privbus in the aspects of computation complexity, clustering quality and crowdsourced bus results. We implement Privbus on a DELL R720 server (Xeon E5-2690, 2.9 GHz, 32 GB RAM, OS Ubuntu 14.04.), several Nexus 5 smartphones (2.3 GHz CPU, 2G RAM, 16G ROM, Android 6.0 system, Bluetooth v4.0) and laptops (Intel Core-i7-4710MQ CPU, 4 GB RAM, Ubuntu 14.04.3 64bit OS). The laptops work as fog nodes in our simulations. We assume that $N$ and $g$ are of 1024 and 160 bits for the sufficient semantic security of the PCTD cryptosystem [21,22]. Under this assumption, a public key $(N, g)$ is of 1184 bits, a ciphertext is of $2\log_2 N$ =2048 bits.

In the subsection of complexity analysis, the computation cost and communication overhead of Privbus are evaluated using a custom simulator built in C. We collect 15,000 rides from the traffic network in New York[8] for the simulation. The cutoff distance is $d_c = (d_0, d_0, t_0) = (500, 500, 6)$. The radius of each fog node's coverage is set to 1 km.

In the subsection of clustering quality, we run the DP-based trip clustering algorithm in Privbus and the privacy-preserving method based on k-means clustering in XHYCZ [37] on a small sized dataset S4,[9] including 5000 instances from 15 clusters. The clustering quality of these approaches is evaluated by precision and recall. Precision and recall [40] are defined as $precision(C) = |\mathrm{TP}|/(|\mathrm{TP}| + |\mathrm{FP}|)$ and $recall(C) = |\mathrm{TP}|/(|\mathrm{TP}| + |\mathrm{FN}|)$, where $C$ is the clustering result. TP, FP and FN denote true positive, false positive and false negative probability, respectively.

In the subsection of crowdsourced bus results, we run trip clustering operation in Privbus on the collected commuting rides and obtain 1430 clusters. Then we choose 10 random clusters which consist of long and popular commuting rides for the simulations of route planning. Parameters used in the simulations are set to be $\alpha = 0.001$, $p = 70$, $\mu = 200$, $\eta = 1.5$, $cen(l) = C_l$, $\beta(l) = \begin{cases} 10, d(l) <= 10 \text{ km}, \\ 10^{-3} d(l), d(l) > 10 \text{ km}. \end{cases}$

### 4.1. Complexity analysis

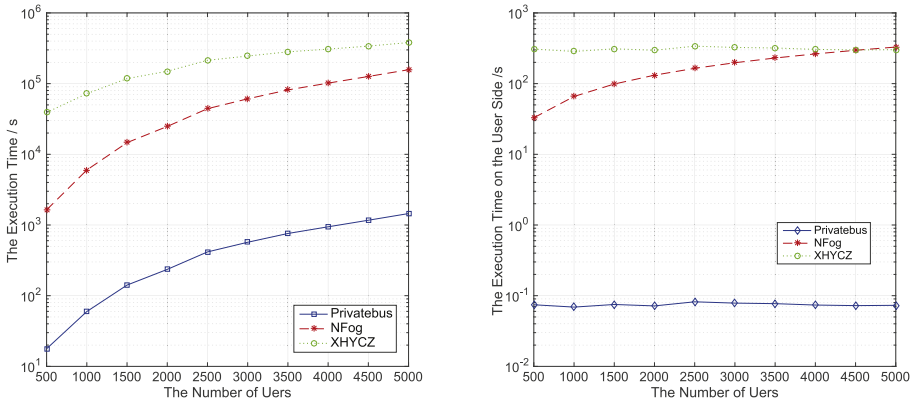
Table 2 shows the comparison result of Privbus, NFog and XHYCZ [37] in the aspect of computational complexity, where $n$ and $K$ are the number of users and clusters, respectively. NFog represents a basic Privbus version without fog nodes, where all operations executed on fog nodes in Privbus are run on their users' smart devices in NFog. XHYCZ [37] is a privacy-preserving method based on k-means clustering.

The computational cost is measured by counting the number of time-consuming cryptographic operation. Assume that $h$, $mul_1$, $mul_2$, $exp_1$ and $exp_2$ represent a keyed hash function of SHA-1, a 1024 bit modular multiplication, a 2048- bit modular multiplication, a 1024 bit modular exponentiation and a 2048- bit modular exponentiation, respectively. The online part starts with sending trip requests to the fog nodes and ends with receiving the encrypted bus-sharing result. The communication cost is evaluated by computing bits transmitted between entities. In XHYCZ [37], $I$ represents the number of iterations to achieve convergence and $s$ denotes that the encrypted bus trip request is divided into $s$ components during transmission. In Privbus, since a fog node can be a cellular base station and is generally adjacent to $a = 6$ fog nodes, the size of user $u_{i,j}$'s neighboring fog node set can be $|F_{i,j}| = d_f = 4$. The number of users served by fog node $f_i$ is $n_i$.

To perform k-means clustering, XHYCZ [37] requires $I$ iterations and allocates a large number of $exp_1$ and $mul_2$ operations to every user in each iteration, hence the user's terminal device should provide large computational and bandwidth resources. As a result, the users need to stay online all the time in XHYCZ [37]. Privbus supports offline users after they submit encrypted trip requests to fog nodes, whereas users should execute many operations online in NFog and XHYCZ [37]. Hence, Privbus is friendly

8 https://github.com/toddwschneider/nyc-taxi-data

9 http://cs.joensuu.fi/sipu/datasets/




(a) (b)

**Fig. 6.** (a) The execution time of schemes; (b) The execution time of schemes on the user side, on the smartphones and the server.

to resource-constraint mobile terminals due to the assistance of fog nodes.

To make trip clustering computation self-contained within each fog node's partition, any fog node $f_i$ collects bus trip requests of its users' $d_c$-length neighbors from adjacent fog nodes and sets $U_i'$ for these neighbors' requests. We assume the sum of all $U_i'$'s size is $\sum_{i=1}^{m} |U_i'| = \sum_{i=1}^{m} \omega_i = \omega$. In a manner of spatial statistics, the total size $\omega$ can be evaluated as $\omega = \sum_{i=1}^{m} \omega_i \approx (\frac{\pi(1000+500)^2}{\pi 1000^2} - 1) \sum_{i=1}^{m} n_i = 1.25n$, where $d_0 = 500$m or 1000m is the radius of a fog node's service region. Hence, the computation complexity of Privbus on a fog node side and CBS server side can be denoted as $O(max(n_i^2, n_i w_i)) \sim O(n_i^2)$, $O(max(n_i n, n_i w)) \sim O(n_i n)$ in Table 2, respectively. Compared with $\lceil n(n-1)/2 \rceil \sim O(n^2)$ distances in original DP clustering, only $O(n_i n)$ distance values should be calculated for $n$ users in Privbus. Therefore, fog-assisted DP clustering improves the performance of initial DP clustering due to the assistance of fog nodes.

We choose 10 random subsets from the collected 15,000 trips. The size of these subsets varies from 500 to 5000. Fig. 6(a) illustrates that the execution time of Privbus, NFog and XHYCZ [37] increases with the increasing size of user set. Since CBS server has strong computational capability and sufficient storage space, the execution time on a user' resource-limited device has big impact on that of the whole schemes. As shown in Fig. 6(b), Privbus has the lowest execution time on the user side, because users only need to submit and download message once. Meanwhile, the time spent on the user side remains stable in Privbus and XHYCZ [37]. NFog and XHYCZ require users to stay online and execute many operations, resulting in much more execution time on the user side, as presented in Fig. 7.

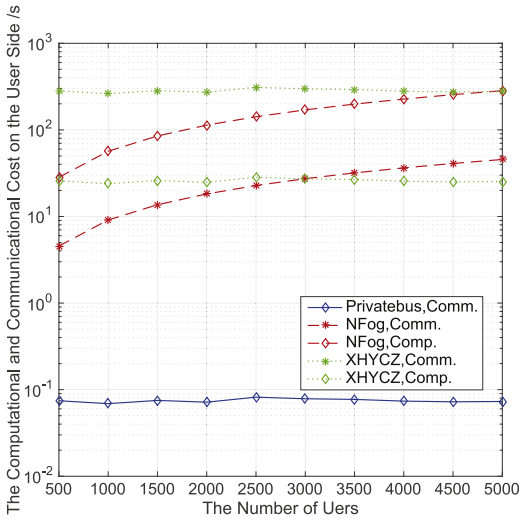



**Fig. 7.** The computational and communicational cost on the user side.

### 4.2. Clustering quality

The clustering results of Privbus and XHYCZ [37] are shown in Fig. 8. Table 3 further compares their precisions and recalls in detail. There are 15 clusters in the ground truth shown in Fig. 8(a). Fig. 8(b) presents that XHYCZ [37] fails to identify clusters which are close to each other, leading to low precision and recall of clusters in the overlapping areas, e.g., cluster 9 and cluster 11 as shown in Table 3. Although the k-means clustering algorithm correctly identifies 14 clusters in XHYCZ [37], it makes mistakes with cluster 9, which just achieves 3.24% and 2.08% in aspects of precision and recall, respectively. Fog-assisted DP used in Privbus correctly identifies all the 15 clusters which have a lot of overlap, hence Privbus performs better than XHYCZ [37] on the clustering quality.

**Table 3**
Evaluation of the Clustering Results.

| Schemes | k-Means, XHYCZ | | DP, Privbus | |
|---|---|---|---|---|
| Cluster | Precision % | Recall % | Precision % | Recall % |
| $C_1$ | 80.27 | 78.67 | 78.02 | 84 |
| $C_2$ | 92.24 | 33.86 | 82.67 | 78.48 |
| $C_3$ | 82.34 | 96.94 | 81.79 | 97.55 |
| $C_4$ | 85.44 | 82.50 | 87.96 | 82.19 |
| $C_5$ | 61.44 | 58.20 | 63.41 | 56.35 |
| $C_6$ | 88.46 | 92.28 | 88.92 | 91.67 |
| $C_7$ | 65.44 | 70.64 | 79.05 | 71.56 |
| $C_8$ | 85.27 | 80.95 | 86.16 | 81.55 |
| $C_9$ | 3.24 | 2.08 | 71.18 | 71.81 |
| $C_{10}$ | 82.20 | 97.97 | 87.27 | 97.67 |
| $C_{11}$ | 44.93 | 62.54 | 70.06 | 71.47 |
| $C_{12}$ | 67.12 | 70.57 | 72.97 | 69.43 |
| $C_{13}$ | 82.61 | 76.22 | 81.01 | 78.22 |
| $C_{14}$ | 80.39 | 94.86 | 86.91 | 94.86 |
| $C_{15}$ | 75.82 | 79.71 | 83.85 | 77.14 |

### 4.3. Crowdsourced bus results

The chosen 10 clusters are shown in Fig. 9(a). Fig. 9(b) shows the distribution of source and destination locations when the red rectangle area in Fig. 9(a) is zoomed in to the maximum. Based on these clusters, we obtain 10 bus routes and 5 more optimized

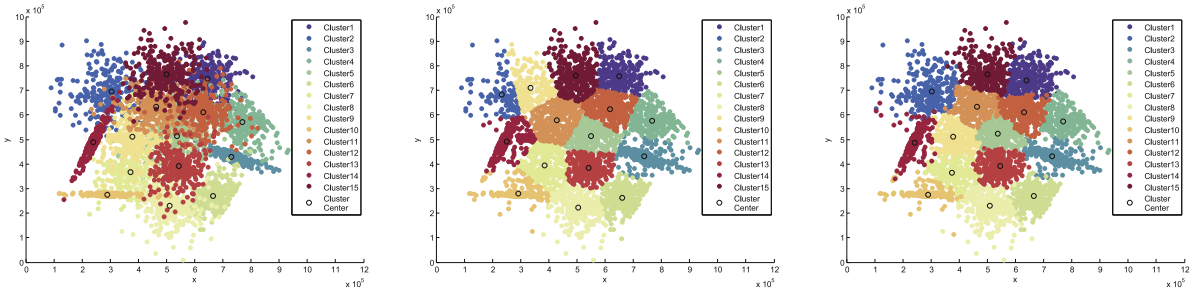



**Fig. 8.** Clustering results of # for location dataset S4 (5000 instances, 2 dimensions, 15 clusters), (a) ground truth; (b) k-Means result, XHYCZ; (c) DP, privbus.

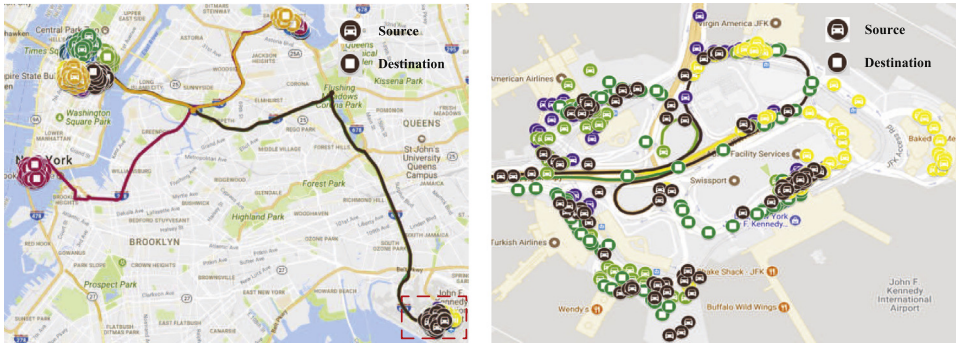



**Fig. 9.** (a) The chosen 10 clusters; (b) The distribution of source and destination locations in the red rectangle area.

**Table 4**
Bus routes for point–point bus service.

| Route | DT | Profit | TM (km) | PL |
|---|---|---|---|---|
| $C_1$ | 8:10 | 1351.5 | 10.32 | 210 |
| $C_2$ | 7:45 | 3328.8 | 20.79 | 210 |
| $C_3$ | 7:30 | 2100.3 | 19.84 | 140 |
| $C_4$ | 8:10 | 893.5 | 10.27 | 140 |
| $C_5$ | 8:00 | 911.7 | 10.41 | 140 |
| $C_6$ | 7:50 | 1127.2 | 21.07 | 70 |
| $C_7$ | 7:50 | 1124.5 | 21.02 | 70 |
| $C_8$ | 8:05 | 686.2 | 14.07 | 70 |
| $C_9$ | 7:40 | 1038.2 | 19.65 | 70 |
| $C_{10}$ | 8:15 | 456.0 | 10.41 | 70 |

**Table 5**
Bus Routes For Optimized Shuttle Bus Service.

| Route | DT | Profit | PL |
|---|---|---|---|
| $R_1^*$ : $C_3.s$, $C_5.s$, $C_3.d$, $C_5.d$ | $C_3$ 7:10, $C_5$ 7:58 | 846.3 | 70 |
| $R_2^*$ : $C_6s$, $C_4.s$, $C_4.d$, $C_6.d$ | $C_6$ 7:30, $C_4$ 7:32 | 909.0 | 70 |
| $R_3^*$ : $C_7.s$, $C_9.s$, $C_5.s$, $C_9.d$, $C_5.d$, $C_7.d$ | $C_7$ 7:08, $C_9$ 7:12, $C_5$ 8:00 | 1042.4 | 70 |
| $R_4^*$ : $C_8.s$, $C_4.s$, $C_8.d$, $C_4.d$ | $C_8$ 7:05, $C_4$ 7:35 | 442.9 | 70 |
| $R_5^*$ : $C_{10}.d$, $C_4.d$, $C_{10}.d$, $C_4.d$ | $C_{10}$ 8:00, $C_4$ 8:05 | 217.0 | 25 |

routes as shown in Tables 4 and 5, respectively, where DT, TM and PL denote departure time, traveling miles and passenger load, respectively.

Table 4 illustrates that the customized bus routes are profitable with the full passenger load and the long traveling miles. $C_{10}$ gains the lowest profit as it has the least passenger load and the shortest path. The coverage of this service reaches to 77.52% if the full passenger load is guaranteed.

When the bus routes are optimized with the objective of maximizing the service provider's profit, the full passenger load could not be ensured all the time. As shown in Table 5, $R_5^*$ gains the lowest profit as it has the long path with the most vacant seats, while $R_1^*$ and $R_2^*$ serve 2 clusters with full passenger load, resulting in high profit. Thus, the bus routes with full passenger load and short traveling miles are usually efficient. After the route optimization, the number of users can be served increases by 305, and the whole coverage of CBS rises from 77.52% to 88.39%. We compare the expected traveling miles of bus trips provided by two types ($R_m.d(C_l)$, $d(C_l)$) in Table 6. The ratio of them is less than 1.5, hence Privbus satisfies the user demand of a fast bus trip.

**Table 6**
Comparison of Traveling Miles.

| Cluster | TM in $R_m$ (km) | TM in $R_m^*$ (km) |
|---|---|---|
| $C_1$ | 10.32 ($R_1$) | – |
| $C_2$ | 20.79 ($R_2$) | – |
| $C_3$ | 19.84 ($R_3$) | 24.68 ($R_1^*$) |
| $C_4$ | 10.27 ($R_4$) | 10.27 ($R_2^*$), 10.47 ($R_5^*$), 20.35 ($R_4^*$) |
| $C_5$ | 10.41 ($R_5$) | 10.42 ($R_3^*$),10.54 ($R_1^*$) |
| $C_6$ | 21.07 ($R_6$) | 26.00($R_2^*$) |
| $C_7$ | 21.02 ($R_7$) | 26.88 ($R_3^*$) |
| $C_8$ | 14.07 ($R_8$) | 16.12 ($R_4^*$) |
| $C_9$ | 19.65 ($R_9$) | 25.22 ($R_5^*$) |
| $C_{10}$ | 10.41 ($R_{10}$) | 11.06 ($R_5^*$) |

To deploy Privbus in CBS applications, such as Bridj, Jiewo, and Bus Pooling, fog nodes should be deployed between the users' terminal devices and CBS server. Fog nodes can be base stations, WiFi access points or femtocell routers [23] in the real world. Besides, the encryption and decryption procedures of PCTD Cryptosystem should be executed on each entity in the CBS system. Every party will have less runtime with a smaller $N$ which implies a narrower plaintext range, resulting in a lower level of security. Thus it is necessary to choose an appropriate $N$ to strike a balance between computational overhead and security level

in a real-world implementation. Finally, further optimization is expected to reduce the online computation time in practice, as our simulations use the publicly available C implementation of Paillier cryptosystem[10] without any optimization.

## 5. Related works

The existing privacy-preserving clustering schemes can be mainly divided into two categories [2,3]: randomization-based [1,16,29] and cryptography-based solutions [33,37–39]. Agrawal and Srikant [1] firstly proposed a PPC scheme over a dataset by adding random noises. İnan et al. [16] developed an improved data-perturbation method to perform privacy-preserving clustering over horizontally partitioned data, in which the communication cost is high due to the frequent interactions between distributed entities. To improve the efficiency of PPC, Haar wavelet transform and scaling data perturbation [13] were used to protect the underlying numerical attribute values subjected to clustering analysis. Oliveira et al. [29] further focused on balancing the trade-off between privacy and clustering quality, and proposed a geometric transformation-based method, which distorts data and enables the partition and hierarchical clustering. However, randomization-based PPC methods usually fail to guarantee the clustering accuracy [3], since clustering is carried out on the tailored data. Meanwhile, secure multi-party computation and homomorphic encryption [32] were utilized to protect users' privacy during data clustering. Secure multi-party computation technique was used to address the arbitrarily-partitioned PPC problem under partially truthful two-party and multi-party models [6,19,24]. The computational cost is high in most of these methods, hence they are only applicable to small datasets. In order to facilitate the clustering of a large dataset, researchers built many privacy-preserving and outsourced mechanisms on clouds [33,37–39] by combining homomorphic encryption technique and k-means clustering algorithm. Since interactive computations are carried out on the user side in the process of k-means clustering, most of these schemes [20,33,37] need online users. PPHOCFS [41] removes iteration calculation and improves clustering quality by using DP clustering, but it also fails to support offline users, as users should encrypt and decrypt the intermediate values on their terminal devices.

Different from the existing works, our preliminary work [14] proposed a Privacy-preserving rIde Clustering (PIC) scheme, which removes iterative calculation and allows users to stay offline after submitting their trip requests. In this paper, we further propose Privbus to provide a privacy-enhanced CBS. Privbus enhances the performance of implementing privacy-preserving DP clustering without sacrificing clustering quality and hampering the functions of PIC due to the data locality property of fog nodes. Besides, Privbus optimizes routes on the premise of satisfying the user demand for the purpose of increasing the profit and the coverage of CBS.

## 6. Conclusions

In this paper, we proposed Privbus, a privacy-enhanced crowdsourced bus service, to improve the performance of clustering calculation and maintain its clustering quality without exposing the individual travel plans of users. Privbus can perform the fog-assisted DP clustering efficiently with privacy preservation, while reducing computation cost on users' smart devices to save online traffic for users. In addition to the customized bus routes, Privbus can further optimize routes to maximize CBS provider's profit on the premise of satisfying the user demands, including the users' acceptable walking distance to bus stops, acceptable waiting time, expected arrival time and fast bus trips. We demonstrated that Privbus reaches the desirable security and privacy goals, and shown the low computational and communication overhead of Privbus. For the future work, we will make the privacy-preserving approaches for CBS work in dynamic scenarios.

[10] https://github.com/skgrush/GMP-Paillier-Crypto

## Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.jpdc.2019.09.007.

## Acknowledgments

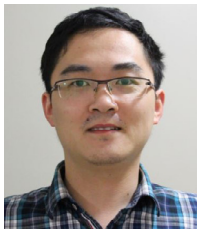
This work is supported by the grants from the National Key Research and Development Program of China (2017YFB0802203), the National Natural Science Foundation of China (61672515, U1836203) and the Youth Innovation Promotion Association, Chinese Academy of Sciences, China(2018196).

## References

[1] R. Agrawal, R. Srikant, Privacy-preserving data mining, in: Proc. of ACM SIGMOD, 2000, pp. 439–450.
[2] A. Alabdulatif, I. Khalil, M. Reynolds, H. Kumarage, Privacy-preserving data clustering in cloud computing based on fully homomorphic encryption, in: Proc. of PACIS, 2017, pp. 289–301.
[3] V. Baby, N. Chanda, Privacy-preserving distributed data mining techniques: A survey, Int. J. Comput. Appl. 143 (10) (2016) 37–41.
[4] D. Boneh, E.-J. Goh, K. Nissim, Evaluating 2-DNF formulas on ciphertexts, in: Proc. of the Second International Conference on Theory of Cryptography, 2005, pp. 325–341.
[5] E. Bresson, D. Catalano, D. Pointcheval, A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications., in: Proc. of ASIACRYPT, 2003, pp. 37–54.
[6] P. Bunn, R. Ostrovsky, Secure two-party k-means clustering, in: Proc. of ACM CCS, 2007, pp. 486–497.
[7] Z. Chen, H.T. Shen, X. Zhou, Discovering popular routes from trajectories, in: Proc. of IEEE ICDE, 2011, pp. 900–911.
[8] Y. Chen, P. Zhao, P. Li, K. Zhang, J. Zhang, Finding communities by their centers, Sci. Rep. 6 (2016) 24017:1–8.
[9] R. Cramer, V. Shoup, Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption, in: Proc. of ACM EUROCRYPT, 2002, pp. 45–64.
[10] Z. Cui, B. Sun, G. Wang, Y. Xue, J. Chen, A novel oriented cuckoo search algorithm to improve dv-hop performance for cyber–physical systems, J. Parallel Distrib. Comput. 103 (2017) 42–52.
[11] P.A. Fouque, D. Pointcheval, Threshold cryptosystems secure against chosen-ciphertext attacks, in: Proc. of ACM ASIACRYPT, 2001, pp. 351–368.
[12] E. Groff, D. Weisburd, N.A. Morris, Where the action is at places: examining spatio-temporal patterns of juvenile crime at places using trajectory analysis and GIS, in: D. Weisburd, W. Bernasco, G.J. Bruinsma (Eds.), Putting Crime in Its Place: Units of Analysis in Geographic Criminology, Springer New York, New York, NY, 2009, pp. 61–86.
[13] S. Hajian, M.A. Azgomi, A privacy preserving clustering technique using Haar wavelet transform and scaling data perturbation, in: Proc. of IEEE IIT, 2008, pp. 218–222.
[14] Y. He, J. Ni, B. Niu, F. Li, X. Shen, Privacy-preserving ride clustering for customized-bus sharing: A fog-assisted approach, in: Proc. of WiOpt, 2018, pp. 1–8.
[15] Z. Hong, Y. Chen, H.S. Mahmassani, S. Xu, Commuter ride-sharing using topology-based vehicle trajectory clustering: Methodology, application and impact evaluation, Transp. Res. C 85 (2017) 573–590.
[16] A. İnan, Y. Saygyn, E. Savas, A. Hintoglu, A. Levi, Privacy preserving clustering on horizontally partitioned data, Data Knowl. Eng. 63 (3) (2007) 646–666.
[17] J.G. Lee, J. Han, K.Y. Whang, Trajectory clustering:a partition-and-group framework, in: Proc. of ACM SIGMOD, 2007, pp. 593–604.

[18] H. Li, Q. Chen, Z. Haojin, D. Ma, W. Hong, X. Shen, Privacy leakage via de-anonymization and aggregation in heterogeneous social networks, IEEE Trans. Dependable Secure Comput. (2019) 1–12.
[19] F. Li, Y. He, B. Niu, H. Li, Small-world: secure friend matching over physical world and social networks, Inform. Sci. 387 (2017) 205–220.
[20] K.P. Lin, Privacy-preserving kernel k -means clustering outsourcing with random transformation, Knowl. Inf. Syst. 49 (3) (2016) 885–908.
[21] X. Liu, K.K.R. Choo, R.H. Deng, R. Lu, J. Weng, Efficient and privacy-preserving outsourced calculation of rational numbers, IEEE Trans. Dependable Secure Comput. 15 (1) (2018) 27–39.
[22] X. Liu, R.H. Deng, K.K.R. Choo, J. Weng, An efficient privacy-preserving outsourced calculation toolkit with multiple keys, IEEE Trans. Inf. Forensics Secur. 11 (11) (2016) 2401–2414.
[23] T.H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, L. Sun, Fog computing: Focusing on mobile users at the edge, Comput. Sci. (2015) 1–7.
[24] H. Miyajima, N. Shigei, H. Miyajima, Y. Miyanishi, S. Kitagami, N. Shiratori, New privacy preserving clustering methods for secure multiparty computation, Artif. Intell. Res. 6 (1) (2016) 27–36.
[25] J. Ni, X. Lin, X. Shen, Towards privacy-preserving valet parking in autonomous driving era, IEEE Trans. Veh. Technol. 68 (2) (2019) 2893–2905.
[26] J. Ni, K. Zhang, X. Lin, X. Shen, Securing fog computing for Internet of Things applications: challenges and solutions, IEEE Commun. Surv. Tutor. 20 (1) (2017) 601–628.
[27] J. Ni, K. Zhang, Q. Xia, X. Lin, X. Shen, Enabling strong privacy preservation and accurate task allocation for mobile crowdsensing, IEEE Trans. Mob. Comput. (2019) 1–16.
[28] J. Ni, K. Zhang, Y. Yu, X. Lin, X. Shen, Providing task allocation and secure deduplication for mobile crowdsensing via fog computing, IEEE Trans. Dependable Secure Comput. (2019) 1–13.
[29] S.R.M. Oliveira, O.R. Zaane, E.I. Agropecuaria, Privacy preserving clustering by data transformation, J. Inf. Data Manag. 1 (1) (2010) 37–51.
[30] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: Proc. of EUROCRYPT, 1999, pp. 223–238.
[31] A. Peter, E. Tews, S. Katzenbeisser, Efficiently outsourcing multiparty computation under multiple keys, IEEE Trans. Inf. Forensics Secur. 8 (12) (2013) 2046–2058.
[32] B. Pinkas, Cryptographic techniques for privacy-preserving data mining, ACM SIGKDD 4 (2) (2002) 12–19.
[33] F.Y. Rao, B.K. Samanthula, E. Bertino, X. Yi, D. Liu, Privacy-preserving and outsourced multi-user K-Means clustering, in: Proc. of Collaboration and Internet Computing, 2016, pp. 80–89.
[34] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[35] Y. Tian, M.M. Kaleemullah, M.A. Rodhaan, B. Song, A. Al-Dhelaan, T. Ma, A privacy preserving location service for cloud-of-things system, J. Parallel Distrib. Comput. 123 (2019) 215–222.
[36] M. Tiwary, D. Puthal, K.S. Sahoo, B. Sahoo, L.T. Yang, Response time optimization for cloudlets in Mobile Edge Computing, J. Parallel Distrib. Comput. 119 (2018) 81–91.
[37] K. Xing, C. Hu, J. Yu, X. Cheng, F. Zhang, Mutual privacy preserving k-means clustering in social participatory sensing, IEEE Trans. Ind. Inf. 13 (4) (2017) 2066–2076.
[38] Y. Zhang, R. Deng, X. Liu, Z. Dong, Outsourcing service fair payment based on blockchain and its applications in cloud computing, IEEE Trans. Serv. Comput. PP (2018) 1–18.
[39] Y. Zhang, R. Deng, D. Zheng, J. Li, J. Cao, Efficient and robust certificateless signature for data crowdsensing in cloud-assisted industrial IoT, IEEE Trans. Ind. Inf. PP (2019) 1–9.
[40] K. Zhang, X. Liang, R. Lu, K. Yang, X. Shen, Exploiting mobile social behaviors for Sybil detection, in: Proc. of IEEE INFOCOM, 2015, pp. 271–279.
[41] Q. Zhang, H. Zhong, L.T. Yang, Z. Chen, F. Bu, PPHOCFS: Privacy preserving high-order CFS algorithm on the cloud for clustering multimedia data, ACM Trans. Multimed. Comput. Commun. Appl. 12 (4s) (2016) 66:1–15.

**Yuanyuan He** is currently an assistant professor at the School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan, China. She received the Ph.D. degree in Computer System Architecture from Institute of Information Engineering, Chinese Academy of Sciencesthe, Beijing, China, in 2019, and received the B.S. degree and the M.S. degree in Computational Mathmatics from Chongqing University, Chongqing, China, in 2006 and 2009, respectively. She a visiting scholar in University of Waterloo, Waterloo, ON, Canada, from 2016-2018. Her current research interests include wireless network security, privacy computing, fog computing and Internet of Things.



**Jianbing Ni** is currently an assistant professor in the Department of Electrical and Computer Engineering, Queen's University, Kingston, Canada. He received his Ph.D. degree in Electrical and Computer Engineering from University of Waterloo, Waterloo, Canada, in 2018, and received the B.E. degree and the M.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2011 and 2014, respectively. His research interests are applied cryptography and network security, with current focus on cloud computing, smart grid, mobile crowdsensing and Internet of Things.

**Ben Niu** received his B.S. degree in Information Security, M.S. and Ph.D. degrees in Cryptography from Xidian University in 2006, 2010 and 2014 respectively. Currently, he is working as associate professor in State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences. He was a visiting scholar in Pennsylvania State University from 2011 to 2013. His current research interests include wireless network security and privacy preservation.

**Fenghua Li** received his B.S. degree in Computer Software, M.S. and Ph.D. degrees in Computer Systems Architecture from Xidian University in 1987, 1990 and 2009 respectively. Currently, he is working as professor and doctoral supervisor in Institute of Information Engineering, Chinese Academy of Sciences. He is also a doctoral supervisor in Xidian University, and University of Science and Technology of China. His current research interests include network security, system security, privacy preservation and trusted computing.

**Xuemin (Sherman) Shen** received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research focuses on resource management in interconnected wireless/wired networks, wireless network security, social networks, smart grid, and vehicular ad hoc and sensor networks. He is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society. Dr. Shen received the R.A. Fessenden Award in 2019 from IEEE, Canada, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015 and the Education Award in 2017 from the IEEE Communications Society. He has also received the Excellent Graduate Supervision Award in 2006 and the Outstanding Performance Award 5 times from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee Chair/Co-Chair for the IEEE Globecom'16, the IEEE Infocom'14, the IEEE VTC'10 Fall, the IEEE Globecom'07, the Symposia Chair for the IEEE ICC'10, the Tutorial Chair for the IEEE VTC'11 Spring, the Chair for the IEEE Communications Society Technical Committee on Wireless Communications, and P2P Communications and Networking. He is the Editor-in-Chief of the IEEE INTERNET OF THINGS JOURNAL and the Vice President on Publications of the IEEE Communications Society.