Received April 3, 2018, accepted May 15, 2018, date of publication May 28, 2018, date of current version June 20, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2840726

Differential Private POI Queries via Johnson-Lindenstrauss Transform

MENGMENG YANG[1], TIANQING ZHU[2,3], BO LIU[4], YANG XIANG[5], (Senior Member, IEEE), AND WANLEI ZHOU[3,6], (Senior Member, IEEE)

[1]School of Information Technology, Deakin University, Melbourne, VIC 3125, Australia
[2]School of Mathematics and Computer Science, Wuhan Polytechnic University, Wuhan 430023, China
[3]School of Software, University of Technology Sydney, Ultimo, NSW 2007, Australia
[4]Department of Engineering, La Trobe University, Melbourne, VIC 3086, Australia
[5]Digital Research and Innovation Capability Platform, Swinburne University of Technology, Melbourne, VIC 3122, Australia
[6]College of Information Science and Technology, Jinan University, Guangzhou 510632, China

Corresponding author: Tianqing Zhu (tianqing.zhu@uts.edu.au)

This work was supported in part by the National Natural Science Foundation of China under Grant 61502362 and in part by the Australia Research Council Linkage under Grant LP170100123.

ABSTRACT The growing popularity of location-based services is giving untrusted servers relatively free reign to collect huge amounts of location information from mobile users. This information can reveal far more than just a user's locations but other sensitive information, such as the user's interests or daily routines, which raises strong privacy concerns. Differential privacy is a well-acknowledged privacy notion that has become an important standard for the preservation of privacy. Unfortunately, existing privacy preservation methods based on differential privacy protect user location privacy at the cost of utility, aspects of which have to be sacrificed to ensure that privacy is maintained. To solve this problem, we present a new privacy framework that includes a semi-trusted third party. Under our privacy framework, both the server and the third party only hold a part of the user's location information. Neither the server nor the third party knows the exact location of the user. In addition, the proposed perturbation method based on the Johnson Lindenstrauss transform satisfies the differential privacy. Two popular point of interest queries, k-NN and Range, are used to evaluate the method on two real-world data sets. Extensive comparisons against two representative differential privacy-based methods show that the proposed method not only provides a strict privacy guarantee but also significantly improves performance.

INDEX TERMS Differential privacy, Johnson Lindenstrauss transform, location privacy, LBS.

I. INTRODUCTION

The pervasive diffusion of GPS-enabled devices has provided tremendous opportunities for the development of location-based services (LBSs). A typical example is providing recommendations about nearby points of interest (POIs). A user queries an LBS with their current location, and the LBS returns the corresponding POIs. Even though LBSs provide great benefits, they come at the cost of exposing a user's location. Where a person is, is sensitive information. It can easily be linked to highly confidential details, such as their home address, and their religious practices. Therefore, devising a solution that allows users to benefit from LBSs while guaranteeing the privacy of their location is highly desirable.

Numerous location privacy protection methods have been proposed in the last decade. Most solutions proposed in the literature are based on location obfuscation. The basic idea is to transform the user's exact location into a region large enough to thwart attacks, also known as cloaking region [1]. Unfortunately, most location obfuscation techniques proposed rely on syntactic approaches such as k-anonymity, which cannot provide rigorous privacy [2]. In addition, Dewri [3] highlighted the inadequacy of cloaking regions in preventing location privacy breaches when the adversary grasps some approximate location knowledge about the user. Another class of technique is Private Information Retrieval (PIR), which uses cryptography to protect the user's location information [4]. This technique allows a user to query POIs without revealing any information about the query. However, while LBS queries based on PIR provides strong cryptographic guarantees, they are often computationally and communicationally expensive and not practical in addition to requiring different query plans to be designed for different query types [2].

Differential privacy, a powerful privacy model, is widely accepted for providing rigorous privacy guarantees for aggregate data analysis [5]. It ensures that one individual cannot significantly affect the output of a query. Differential privacy is normally achieved by injecting random noise to the result of the query. Applying differential privacy for location protection is still at its early stage [2]. Dewri [3] proposed a differential location perturbation method that added Laplace noise to the x and y coordinate separately. Bordenabe *et al.* [6] introduced a generalized privacy notation geo-indistinguishability to formalize the problem of location privacy preserving. Palia and Tandon [7] further considered the impact of prior information about POIs on the utility. All of them need to add significant noise to hide the user's exact location in the safe region, which reduces the accuracy of returned POIs.

In this paper, we propose a new Johnson Lindenstrauss transform based location privacy protection method. The Johnson Lindenstrauss lemma states that a small set of points in a high dimensional space can be embedded into a much lower dimension that the Euclidean distances between the points can be nearly preserved [8]. Therefore, the basic idea of the proposed method is that transfer the user's exact location together with the map into another dimension in such a way that the adversary has no idea the user's exact location but the relative distances between POIs are maintained that helps to find nearby POIs.

There are several challenges in applying Johnson Lindenstrauss transform in the location privacy protection. *First, how to evaluate the privacy level protected by the Johnson Lindenstrauss transform?* Blocki *et al.* [9] proved that the Johnson-Lindenstrauss transform preserves edge differential privacy in graph sanitization. Based on this finding, we show that it also guarantees differential privacy in a location dataset. *Second, how to return the accurate POIs without disclosing it to the service provider?* We solve this challenge by introducing a semi-trusted third party who in charge of transferring the map and return anonymized encrypted POIs. In such way, the service provider only access the transformed dataset, while the third party access the original POI information, the exact queried POI information can be returned to the user by third party and service provider intersection without disclosing location privacy to either party.

Overall, our contributions are summarized as follows:

1) We propose a new privacy framework that introduces a semi-trusted third party to protect user locations regardless of adversaries background knowledge.
2) We present a location perturbation method based on the Johnson-Lindenstrauss transform that satisfies differential privacy. The proposed method not only guarantees rigorous privacy preservation but also allows LBS providers to provide accurate services.
3) We systematically analyze how the proposed method can defend against various background knowledge attacks while providing high-quality services.

The rest of the paper is organized as follows. Section II introduces the preliminaries. We propose our privacy preservation framework and apply it to two basic POI queries in Section III and Section IV respectively. Section VI details the results of the experiments. Section VII discusses the related work and Section VIII concludes the paper.

II. PRELIMINARIES

A. DIFFERENTIAL PRIVACY

Differential privacy is a provable privacy notation that has emerged as an important standard for preserving privacy in a variety of areas [5].

Definition 1 (ϵ-Differential Privacy): A randomized algorithm $\mathcal{M}$ gives ϵ-differential privacy for any pair of *neighbouring datasets* D and D', and for every set of outcomes Ω, $\mathcal{M}$ satisfies:

$$Pr[\mathcal{M}(D) \in \Omega] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(D') \in \Omega]. \quad (1)$$

Datasets D and D' are neighbouring datasets, which only differ in one individual record.

B. THE JOHNSON-LINDENSTRAUSS TRANSFORM

The Johnson-Lindenstrauss transform, also known as the famous Johnson-Lindenstrauss Lemma, is a random projection that projects points in a high dimension to a lower dimensional space while preserving the Euclidean distances between any pair of points [10]. Its formal definition is presented as follows:

Lemma 1 (Johnson-Lindenstrauss Lemma [10]): For any set S of n points in $\mathbb{R}^d$, given $0 < \delta < 1/2$, $m = \Omega(log(n)/\delta^2)$, there is a matrix $X \in \mathbb{R}^{m\times d}$, for all $u, v \in S$, there is a map $f : \mathbb{R}^d \to \mathbb{R}^m$, such that

$$(1-\delta)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\delta)\|u-v\|^2. \quad (2)$$

in which, $f(u) = Xu, f(v) = Xv$.

The Johnson-Lindenstrauss Lemma states that there exist a random matrix X can project the dataset from d dimension to m dimension while maintaining the Euclidean distance between records. Several constructions for X have been proposed [11]. A Gaussian distribution is used to generate a random matrix X in this paper.

III. LOCATION PRIVACY PRESERVATION FRAMEWORK BASED ON A THIRD PARTY

A. NOTATIONS

Let $\mathbf{U} = \{u_1, u_2, \ldots, u_n\}$ be a set of users. Each user has a true location and a perturbed location. The true location coordinates are denoted as $l_t(x, y)$, the perturbed location is denoted as an m dimension location vector $l_p(c_1, c_2, \ldots, c_m)$. M is the map matrix, whose records are POI location coordinates, Assume there are t POIs in the map, then $M \in \mathbb{R}^{t\times 2}$. $\hat{M} \in \mathbb{R}^{t\times m}$ is the perturbed map matrix. Important symbols

TABLE 1. Notations.

Parameters	Description	Parameters	Description
l_t	User's true location	l_p	Perturbed location
X	Transition matrix	m	Dimension of X
M	Map's location matrix	$\hat{M}$	Perturbed map matrix
$POIs$	Set of POIs	POI_{s_a}	Anonymized POI set
r	Radius	$\hat{r}$	Perturbed radius
k	Number of POI	$\hat{M}_a$	Anonymized $\hat{M}$

used in this section and following parts of the paper are listed in Table 1 for reference.

B. PROBLEM DEFINITION AND ASSUMPTION

Problem Definition. In this paper, we consider the location privacy problem in the popular location-based service where the user queries the LBS server for nearby POIs. However, the service provider may be untrusted and attacked by outside attackers. Therefore, we define the problem as follows:

Problem 1: Given a user who has a location $l_t(x, y)$, $P\{p_1, p_2, \ldots, p_k\}$ is the nearby POIs of the user. Design a location privacy protection method, through which, the service provider has no idea the user's exact location, while the returned POI set P' should have the following character: the value of $|P \triangle P'|$ should as small as possible.

Assumptions: To make the problem clear, we can make a few reasonable assumptions.

1) *The third party will hold the map information.* This is reasonable as many LBSs use public maps, e.g., Google Maps. The third party could even be a map provider.
2) *The third party is semi-trusted.* We assume the third party in our privacy framework is semi-trusted with following characters:
   - It curious about the user's location information;
   - It follows the process of POI queries;
   - It does not collude with the service provider.

C. FRAMEWORK

The proposed location privacy preservation framework contains three components as illustrated in Fig. 1.

1) USERS

Users are people with GPS-enabled devices who ask for location-based services. They prefer to mask their exact locations. Therefore, location information will be perturbed before sending from the device. User devices are assumed to be trusted. Any malicious software would not be able to access the position sensor [12].

2) SERVICE PROVIDER

Service providers are application platforms that provide location-based services, such as *Google Maps*, *Foursquare*, and *Yelp*. These providers require a user's location to deliver high-quality services to an individual. Service providers may be untrusted. In our proposed framework, a service provider

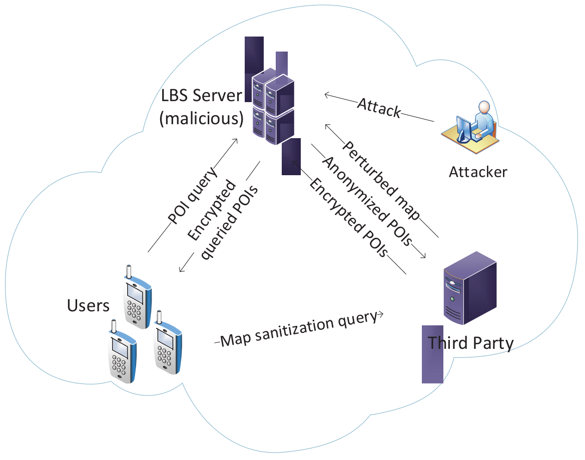


FIGURE 1. Location privacy framework.

receives a query directly from a user and returns the encrypted POIs received from a third party.

3) THIRD PARTY

The third party is semi-trusted and does not collude with the service provider. It holds a portion of the location information and acts as a bridge between the user and the service provider. The third party may acquire the map used by the service provider to provide location-based services, and it also has the essential tasks of perturbing the map and helping the service provider return highly accurate POIs.

As a basic outline of the entire process: (**a**) a user perturbs their location locally using the transition matrix X; (**b**) the user query the service provider for nearby POI and query the third party for map sanitization; (**c**) the third party perturbs the map accordingly; (**d**) the service provider searches for anonymized POIs, asks the third party for exact POI information and forwards them back to the user.

D. PRIVACY PROTECTION SCHEME BASED ON JOHNSON-LINDENSTRAUSS TRANSFORM

In this section, we introduce a perturbation method based on the Johnson-Lindenstrauss transform that operates within the proposed privacy framework. Blocki *et al.* [9] show that the Johnson-Lindenstrauss transform allows us to publish a sanitized graph that preserves edge differential privacy. Based on that, we prove that the Johnson-Lindenstrass transform preserves differential privacy for location dataset as well. According to the Johnson-Lindenstrauss lemma, the transformation can keep the relative distances between points. Therefore, it works well for answering POI queries, which is based on Euclidean distance. The details of the procedure are explained in the following parts.

a. A user has access to their location coordinates through their GPS-enabled devices, and these are perturbed by a linear transition matrix $X \in \mathbb{R}^m$. The details are shown in Algorithm 1.

Algorithm 1 Location Perturbation

```
Require: user $u_i$'s location $l_t(x_i, y_i)$, projected dimension $m$.
Ensure: perturbed location $l_p$, transformation matrix $X \in \mathbb{R}^{2\times m}$.
1: for $i = 1$ to 2 do
2:     for $j = 1$ to $m$ do
3:         Sample $X[i, j]$ from Gaussian distribution $N(0, 1)$;
4:     end for
5: end for
6: $l_p \leftarrow l_t X$;
7: return $l_p, X$.
```

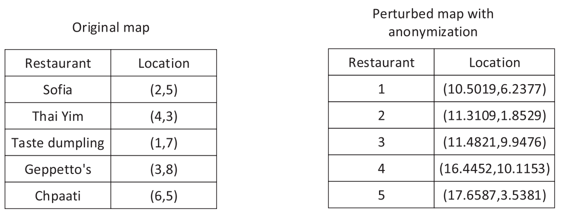
Algorithm 1 shows the proposed location perturbation method. First, a random transition matrix X is generated, which is constructed in Steps 1 to 5. X is a $2 \times m$ matrix and the entries for X are sampled from a 0 mean Gaussian distribution. We perturb the user's location by multiplying the original location coordinate with X at Step 6. Step 6 is the process of Johnson Lindenstrauss transform. After transformation, the user's location information is changed from a location coordinate (x, y) to a meaningless location vector $(c_1, c_2, \ldots c_m)$. We prove that the transformation satisfies ϵ- differential privacy in Section. V-A.

b. The user chooses one region R that he/she feels comfortable with and generates a key pair (sk, pk). Then, he/she queries the LBS nearby POIs using the perturbed location, and query the third party for map sanitization by sending the transition matrix X, region R, public key, and queries type to the third party. As the user's location coordinates are perturbed by matrix X, to guarantee the utility, the map should also be perturbed by the same matrix, after which the relative distances between user and POIs can be preserved. Both the service provider and the third party hold partial location information about the user. None of them can obtain the location of the user with only partial information.

c. Once the third party receives the map sanitization query, it transforms the region R's map with queried POI type using X. The relative distances between the records and user can be maintained because they have been transferred using the same transition matrix. The perturbed map is anonymized, after which it can be sent to the service provider. Fig. 2 shows an example of the map sanitization results.

Assume the user queries the nearby restaurants. The table on the left shows the sampled original map. Each restaurant has a geographic location coordinate. Assume the transition matrix is 2-dimensional, denoted by $X = \begin{bmatrix} 1.7892 & -0.6749 \\ 1.3847 & 1.5175 \end{bmatrix}$, resulting in perturbed location vectors shown in the right-hand side table. The location of the restaurants are totally changed and the names are replace by the meaningless number.

d. The service provider searches for POIs based on the perturbed locations and send them to the third party together with another $K - 1$ sets of POIs within safe region, where


Original map

Restaurant	Location
Sofia	(2,5)
Thai Yim	(4,3)
Taste dumpling	(1,7)
Geppetto's	(3,8)
Chpaati	(6,5)

Perturbed map with anonymization

Restaurant	Location
1	(10.5019,6.2377)
2	(11.3109,1.8529)
3	(11.4821,9.9476)
4	(16.4452,10.1153)
5	(17.6587,3.5381)

FIGURE 2. Map sanitization.

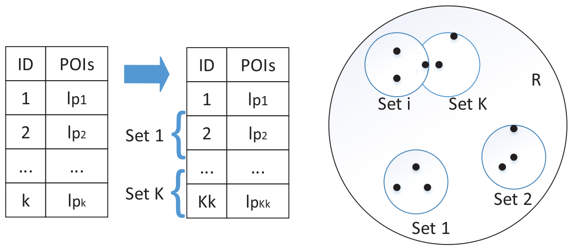


FIGURE 3. The process of k-anonymization.

K is the number of POI set. As the POIs found by the service provider are the meaningless number with the perturbed location, the third party will need to find the corresponding POIs by searching a mapping list. To avoid the third party gets the queried POIs by the user, the POIs calculated by the service provider are protected by k-anonymity technology shown in Fig. 3. The left hand side table is the searched POIs, denote as set 1. the right hand side table is the K-anonymized POI sets. There are totally K sets. The right hand side figure shows that the $K - 1$ sets of POIs are randomly chosen from the safe region. Therefore, the third party has no idea which sets are the queried POIs. The value of K is determined by the user. The probability of identify the user queried POIs is $1/K$. A bigger value of K means harder to identify the returned POIs for the third party. Users can choose it according to their own privacy consideration. The upper bound of K value is controlled under 20, which is big enough to prevent the POIs being identified. K sets of real POIs information will be returned to the service provider. In order to prevent the service provider from obtaining the accurate POIs information, the returned K sets of POIs are encrypted using user's public key. And the queried POIs are filtered according to the POI ID and returned to the user by the service provider.

The service provider is "blind" during the whole POIs query process. As shown in Table. 2, it knows nothing about the user's query. The service provider only receives a location vector. Therefore, it has no idea the user's exact location. During the searching POIs process, the service provider just accesses to the anonymized meaningless records. Therefore, it has no idea what the type of these POIs and which POIs these records refer to. For the third party, it needs to transform the map according to the user's requirement. Therefore, it knows the user's query type, such as restaurants and cinema. However, the third party has no idea the user's location and

TABLE 2. Information access.

	Queried POI type	User's location	Returned POIs
Service provider	×	×	×
Third party	√	×	×
User	√	√	√

returned POIs as well, because the third party never receives any location information of the user and only access the k anonymized POIs. Besides, using this privacy framework, the service provider can return very accurate POIs due to the Johnson-Lindenstrauss transform's ability to maintain relative Euclidean distances between records.

In the proposed privacy preservation method, the third party needs to encrypt the received POIs. Assume the number of queried POI is k, the third party needs to compute Kk encryptions, where K is the number of sets of POIs. In addition, the user needs to compute 1 decryption. Therefore, the total computation complexity for encryption is $O(Kk + 1)$ exp. As the number k is small, the computation complexity caused by the encryption part is acceptable. The proposed privacy framework supports the two most popular spatial queries: k-nearest POIs queries and range queries. In Section IV, we present the details of how these queries are processed in our privacy model.

IV. PRIVACY PROTECTION ALGORITHMS FOR TWO BASIC POI QUERIES

A. K-NEAREST POI QUERIES

Consider an application that the user wants to query the k-nearest POIs around his location. The query can be *'where is the k nearest restaurants'*.

1) k-NEAREST POIs QUERY GENERATION

It has two main operations for user: location perturbation and query generation. The specific steps are shown in Algorithm 2.

Algorithm 2 k-Nearest POIs Query (User)

```
Require: user ID, $u_i$'s location $l_t$, POI number $k$, POI set number $K$, dimension $m$.
Ensure: k-nearest POIs $POIs$.
1: Perturb user's location information using location perturbation method, get the perturbed location $l_p$.
2: Query LBS provider POIs with perturbed location $l_p$.
3: Generate a key pair $sk$ and $pk$ using RSA algorithm.
4: Choose a safe region $R$.
5: Query the third party map sanitization with user ID, transformation matrix $X$, safe region, queried POI type, and public key.
6: return Obtain $POIs$ from the service provider
```

- *Location perturbation.* The user's location l_t can be transferred into a meaningless location vector l_p using a Johnson transformation matrix (Step 1).

$$l_p \leftarrow f(l_t, m), \tag{3}$$

where m is the dimension of the new location vector, which is specified by users. f is the function of the transformation.

- *Query generation.* The query sent to the service provider is in the following form:

$$query \leftarrow \langle userID, k, l_p, K \rangle, \tag{4}$$

where k is the number of queried POIs, and K refers to the number of POIs sets.
User query k-nearest POIs using perturbed location l_p (Step 2). As l_p is a $1 \times m$ vector, the service provider cannot find any relationship between this vector and the user's location without the transition matrix X.
The user generates a key pair in Step 3, which is used to encrypt the POI information. A safe region R is chosen by the user in Step 4. The safe region means the user does not mind other people knows that he/she is in region R. While defining a safe region helps the third party transformed less data, which increase the algorithm efficiency. The query sent to the third party is in the following form:

$$query \leftarrow \langle userID, X, R, POI - type, pk \rangle . \tag{5}$$

The public key together with the transition matrix, safe region, and POI type are sent to the third party for map sanitization in Step 5. The user ID is also included in the query, such that the service provider can find the corresponding perturbed map for each specific query. At the end, the user can get the queried POIs from service provider shown in Step 6.

To guarantee utility, the map should be perturbed by the same transition matrix as the user's location. However, if the service provider knows the perturbed location vector and the transition matrix at the same time, the true location coordinates of the user would be disclosed. Therefore, this task is assigned to the third party.

2) MAP SANITIZATION

After the third party gets a query from the user, it starts to sanitize the map. Algorithm 3 shows the process.

First, the third party samples a small map according to the received user's safe region R and the queried POI type in Step 1 and perturbs it using received transition matrix X in Step 2. To avoid the service provider inferring the real POIs information, the transformed POIs are anonymized by replacing the identifiers by meaningless numbers in Step 3. The sanitized map is sent to the service provider for k-nearest POIs calculation in Step 4. After that, the third party will get K sets of k nearest POIs from the service provider. In step 5, the service provider finds the POIs real information according to the mapping function f, and encrypt it using encrypt algorithm in Step 6. The encrypted POIs are in the

```
Algorithm 3 Map Sanitization (Third Party)
Require: map $\mathbb{M}$, user ID, transformation matrix $X$, queried
    POI type, safe region $R$
Ensure: $POIs$.
 1: Sample a small map $M$ with same POI type required in
    the query in region $R$ from the map $\mathbb{M}$.
 2: Perturb the map $M . f : \hat{M} \overset{X}{\leftarrow} MX$,
 3: Anoymize the perturbed map $\hat{M} . \hat{M_a} \leftarrow \hat{M}$.
 4: Send the map $\hat{M_a}$ to LBS provider with the user ID.

    After getting the k anonymized POI sets $POIs_a$ from
    service provider :

 5: Find the real POIs information according to the mapping
    function $f$.
 6: Encrypt the POIs information: $Enc(POIs) \leftarrow POIs$, and
    send the encrypted POIs back to the service provider.
```

format of $< ID, Enc(POIs) >$. At the end, the encrypted real POIs information is sent back to the service provider.

3) SEARCH NEARBY POIs

After receives the user's POI query and the sanitized map, the service provider searches the nearby POIs blindly.

```
Algorithm 4 Response POI Queries (LBS Server)
Require: User ID, $u_i$'s perturbed location $l_p$, user' query $Q$
    anonymized perturbed map $\hat{M_a}$.
Ensure: $POIs_a$.
 1: Find the corresponding map $\hat{M_a}$ according to the user ID.

 2: for $i = 1$ to length($\hat{M_a}$) do
 3:     $dis(i) \leftarrow$ Euclidean distance between $l_p$ and $M_a(i)$;
 4: end for
 5: $POIs_a \leftarrow$ find the POIs corresponding to $k$ smallest $dis$;
 6: Send $POIs_a$ to the third party;
 7: $POIs_a \leftarrow$ find other $K$ sets of $k$ closest POIs randomly
    within the map.
 8: Send $POIs_a$ to the third party.

    After getting the encrypted accurate POI information
    from the third party

 9: Filter out the queried POI information according to the
    ID and return them back to the user.
10: return $POIs_a$
```

Algorithm 4 shows how to find the k-nearest POIs based on the perturbed information. First, the perturbed map $\hat{M_a}$ is searched according to the user's ID if there are multiple query users at the same time. Then, the distances between the perturbed locations of the POIs and the user's perturbed location are calculated in Steps 2 to Step 4. The k-closest POIs and another K sets of k closest POIs are chosen in Step 5 and 6 separately. All of them are sent to the third party in such way, the third party has no idea the returned POIs to the user and cannot infer the user's exact location. After getting the encrypted accurate POI information, the LBS server will filter out the records corresponding to the POIs obtained in Step 5 and return them to the user. The user can get the accurate POI information by decrypting it using the private key.

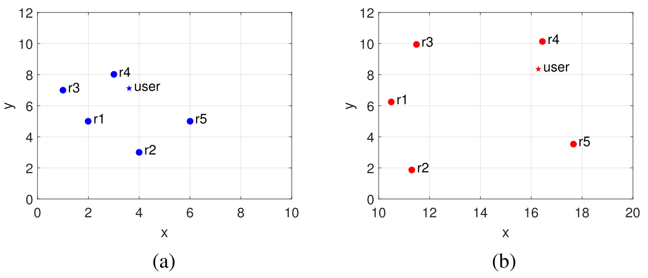


FIGURE 4. JL transform for k-NN queries. (a) Original locations. (b) Perturbed locations.

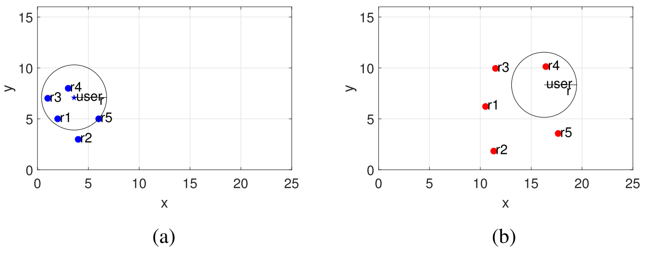


FIGURE 5. JL transform for range queries with radius r. (a) Original locations. (b) Perturbed locations.

Although the server has no idea where the user is, the k-nearest POIs can still be found accurately. Fig. 4 provides an example. The blue dots are the true locations of restaurants around user u. The red dots are the perturbed restaurant locations. The different restaurants are denoted as r_i for simplicity. Although the geographic positions are changed after the transformation, the relative distances are essentially unchanged. For example, in the original map, the two nearest restaurants are r_3 and r_4, while in the perturbed map, we reach the same conclusion.

B. RANGE QUERIES

An example of a range query is *'List all restaurants within 100m of the user.'* In this section, we present how the proposed privacy framework to process range query.

1) RANGE QUERY GENERATION

A privacy-preserving range query process is similar to k-NN query. However, the queried radius r cannot be used on the perturbed map directly because even though the relative distance is essentially the same after the transformation, the actual distance between any two locations has changed. Therefore, directly using the radius r would introduce a large error. Fig. 5 shows an example.

Assume a user queries restaurants within a radius r. In the original dataset, there are four restaurants $\langle r_1, r_3, r_4, r_5 \rangle$.

Algorithm 5 Range Queries (User)

```
Require: user ID, $u_i$'s location $l_t(x, y)$, range $r$, dimension $m$, transformation matrix $X$, queried POI type, safe region $R$, $K$.
Ensure: POIs in range $r$ $POI_a$.
1: $l_p \leftarrow f(l_t, m)$;
2: $s_i = (x + xcos\theta_i, y + ysin\theta_i)$;
3: $\hat{s}_i \leftarrow f(s_i, m)$;
4: $\hat{r} = \frac{\sum d_{s_i,l_p}}{|s_i|}$;
5: $Query_1 \leftarrow \langle userID, \hat{r}, l_p, K \rangle$;
6: Generate a key pair $sk$ and $pk$;
7: $Query_2 \leftarrow \langle userID, X, R, POI - type, pk \rangle$
8: $LBS\ Provider \leftarrow Query_1$;
9: $Third\ Party \leftarrow Query_2$;
```

If the server calculates the POIs using r on the perturbed dataset, we can see only r_4 would be returned to the user, which is wildly inaccurate. Therefore we need to find the mapping $f : r \rightarrow \hat{r}$, to make the relationship between $\hat{r}$ and the perturbed locations relatively consistent with the relationship between r and the original locations.

Algorithm 5 shows the details. First, similar to the k-nearest POI queries, the user's location coordinates $l_t(x, y)$ are perturbed to a location vector $l_p(c_1, c_2, \ldots, c_m)$ using location perturbation method in Step 1. To construct the effective radius $\hat{r}$ for the perturbed location, few points are chosen randomly from the edge of the queried range in Step 2. Where $\theta \in [0, 2\pi]$, to make the result more accurate, we choose more than 4 points. Identically, we get the perturbed value set $\hat{s}$ by applying location perturbation algorithm again in Step 3. In Step 4 calculates the average distance between the perturbed location l_p and the points in set $\hat{s}$, which is the perturbed radius $\hat{r}$. The first query is generated in Step 5, which includes the user's ID, perturbed radius $\hat{r}$, and perturbed location. The user sends the range query with this perturbed radius $\hat{r}$ to the service provider in Step 6. Step 6 generates a key pair for POI information encryption and the second query is generated in Steps 7. The second query is a map sanitization query, which includes user ID, transition matrix, safe region, POI type, public key. These two queries are sent to the service provider and the third party separately in Step 8 and Step 9 respectively.

2) MAP SANITIZATION

The third party perturbs the map using the same method as in Algorithm 3 for the k-nearest POI queries. A smaller map is sampled according to the user's safe region and query type. Then, perturb it using received transition matrix and send it back to the service provider for POI searching. The specific steps are shown in Section. IV-A2.

3) SEARCH NEARBY POIs

The service provider processes the range query with Algorithm 6.

Algorithm 6 Response Range Query (LBS Server)

```
Require: User ID, $u_i$'s perturbed location $l_p$, radius $\hat{r}$ anonymized perturbed map $\hat{M}_a$.
Ensure: anonymized POI $POIs_a$.
1: for $i = 1$ to length($\hat{M}_a$) do
2:     $dis(i) \leftarrow$ Euclidean distance between $l_p$ and $\hat{M_a}(i)$;
3:     if $d(i) < \hat{r}$ then
4:         $POI_{s_a} = POI_{s_a} \cup ID(\hat{M}_a(i))$;
5:     end if
6: end for
7: $P_s = \{P_1, P_2, \ldots, P_K\}$, where $P_i \in \hat{M}_a$;
8: Repeat step 1 to 5 for each $P_i$;
9: Third Party $\leftarrow POIs_a$;
10: return $POIs_a$
```

As shown in Algorithm 6, the distances between the user and all the POIs are calculated in Steps 1 to 2. As long as the distance $d(i) \leqslant \hat{r}$, the location is added to the set $POIs_a$ in Step 4. Another $K - 1$ POIs are randomly selected within perturbed map $\hat{M}_a(i)$ in Step 7, and the corresponding nearby POIs are calculated in Step 8. All of the POIs are sent to the third party. After that, the user can get the queried POIs by the same way for k - nearest queries. Fig. 6 shows an example of utility maintenance. After perturbation, the radius $\hat{r}$ accurately includes the POIs queried by the user.

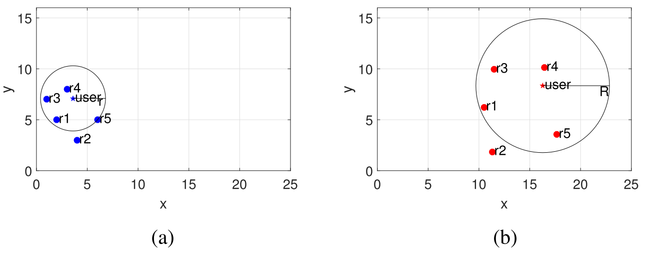


FIGURE 6. JL transform for range queries with radius $\hat{r}$. (a) Original locations. (b) Perturbed locations.

V. PRIVACY AND UTILITY

A. PRIVACY ANALYSIS

Theorem 1: Algorithm 1 guarantees ϵ-differential privacy.

Proof: We prove that the output of the perturbation is indistinguishable regardless of the input location. That is

$$Pr[\mathcal{M}(L) \in \Omega] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(L') \in \Omega]. \quad (6)$$

We observe that the perturbed location vector L_p is composed of m identically distributed variable. Each variable is created by multiplying the true location coordinate L_t with a vector $M_c \in \mathbb{R}^2$. Therefore, we proof theorem 1 by showing that each variable of the output satisfies 6.

As the entries of vector M_c are sampled from iid Gaussian distribution $N(0, \sigma^2)$, vector M_c follows the multi-dimensional Gaussian distribution $N(0, \Sigma)$, where $\Sigma = \begin{bmatrix} \sigma^2 & \\ & \sigma^2 \end{bmatrix}$. According to the linear combination property

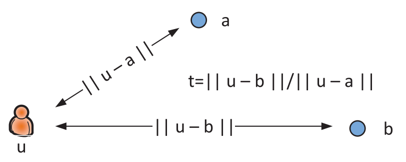


FIGURE 7. **Location placement.**

of multi-dimensional Gaussian distribution, the transformed variable $L_{p_c} \backsim N(0, L_t\Sigma L_t^{\mathrm{T}})$. Let $L_t = (x, y)$, therefore, $L_t\Sigma L_t^{\mathrm{T}} = (x^2+y^2)\sigma^2$. Denoted by $\lambda^2 = L_t\Sigma L_t^{\mathrm{T}}$, then, $L_{p_c} \backsim N(0,\lambda^2)$. Let $L_t'(x', y')$ be the neighbouring dataset, $L_{p_c}' \backsim N(0,\lambda'^2)$ We have

$$\frac{Pr[\mathcal{M}(L)\in\Omega]}{Pr[\mathcal{M}(L')\in\Omega]} = \frac{PDF_L(x)}{PDF_{L'}(x)} = \frac{\frac{1}{\sqrt{2\pi}\lambda}exp(-\frac{x^2}{2\lambda^2})}{\frac{1}{\sqrt{2\pi}\lambda'}exp(-\frac{x^2}{2\lambda'^2})} = \frac{\lambda'}{\lambda}exp(\frac{x^2}{2}(\frac{1}{\lambda'^2}-\frac{1}{\lambda^2})). \tag{7}$$

When variance $\lambda^2 > \lambda'^2$,

$$\frac{Pr[\mathcal{M}(L)\in\Omega]}{Pr[\mathcal{M}(L')\in\Omega]} \geq \frac{\lambda'}{\lambda} = \frac{\lambda-\Delta}{\lambda} = 1-\frac{\Delta}{\lambda}. \tag{8}$$

$\epsilon_0 = -ln(1-\frac{\Delta}{\lambda})$.
When variance $\lambda^2 < \lambda'^2$,

$$\frac{Pr[\mathcal{M}(L)\in\Omega]}{Pr[\mathcal{M}(L')\in\Omega]} \geq \frac{\lambda'}{\lambda} = \frac{\lambda+\Delta}{\lambda} = 1+\frac{\Delta}{\lambda}.$$
$$\epsilon_1 = ln(1+\frac{\Delta}{\lambda}). \tag{9}$$

As

$$\epsilon_1-\epsilon_0 = ln(1+\frac{\Delta}{\lambda})+ln(1-\frac{\Delta}{\lambda}) = ln(\frac{\lambda^2-\Delta^2}{\lambda^2}) < 0. \tag{10}$$

$\epsilon_0 > \epsilon_1$, Therefore,

$$e^{-\epsilon_0} \leq \frac{Pr[\mathcal{M}(L)\in\Omega]}{Pr[\mathcal{M}(L')\in\Omega]} \leq e^{\epsilon_0}. \tag{11}$$

Let $\epsilon = \epsilon_0$, therefore, the proposed method satisfies ϵ-differential privacy, where $\epsilon = -ln(1-\frac{\Delta}{\lambda})$. □

B. UTILITY ANALYSIS

For POI queries, the returned POIs are based on the relative distance between the users and the nearby POIs. Therefore, the utility is evaluated by comparing whether the relative distances between the users and the POIs have changed.

As shown in Fig. 7, given a user u, and two POIs a and b around u, assume $\| u-b \| = t \| u-a \|$ and $t > 1$, where $\| u-a \|$ is the Euclidean distance $d_{u,a}$ between u and a. After the Johnson Lindenstrauss transform, the distances between a and u and b and u are $\| f(u)-f(a) \|$ and $\| f(u)-f(b) \|$ separately. Then the probability of causing an error is $Pr[error] = Pr[\| f(u)-f(a) \| \geq \| f(u)-f(b) \|]$.

According to Lemma 1,

$$\begin{aligned} &Pr[\| f(u)-f(a) \| \\ &\quad \geq \| f(u)-f(b) \|] \\ &\quad \leq Pr[\| u-a \| \sqrt{1+\delta} \geq \| u-b \| \sqrt{1-\delta}] \\ &\quad = Pr[\frac{\| u-b \|}{\| u-a \|} \leq \frac{\sqrt{1+\delta}}{\sqrt{1-\delta}}] \\ &\quad = Pr[t \leq \frac{\sqrt{1+\delta}}{\sqrt{1-\delta}}] \qquad (12)\\ &Pr[\delta \geq \frac{t^2-1}{t^2+1}] \qquad (13)\end{aligned}$$

As $0 < \delta < \frac{1}{2}$, therefore, $1 < \frac{\sqrt{1+\delta}}{\sqrt{1-\delta}} < \sqrt{3}$. Variable δ is defined by the user, therefore, the value of δ can be chosen uniformly from $(1, \sqrt{3})$. So, we assume the variable δ obeys uniform distribution.

Case 1: For a given δ, $1 < \frac{\sqrt{1+\delta}}{\sqrt{1-\delta}} < \sqrt{3}$. As $t > 1$, when $t \geq \sqrt{3}$, $Pr[t \leq \frac{\sqrt{1+\delta}}{\sqrt{1-\delta}}] = 0$. When $t < \sqrt{3}$, we assume t obey uniform distribution in $(1, \sqrt{3})$, which is reasonable, because t is the ratio of distances between two POIs to the user. It can be an arbitrary value in the range of $(1, \sqrt{3})$ according to different POI locations selection. Therefore, $Pr[t \leq \frac{\sqrt{1+\delta}}{\sqrt{1-\delta}}] = \frac{\frac{\sqrt{1+\delta}}{\sqrt{1-\delta}}-1}{\sqrt{3}}$. Equation 12 can be written as

$$Pr[error] \leq \begin{cases} 0 & t \geq \sqrt{3} \\ \dfrac{\sqrt{1+\delta}}{\sqrt{3}\sqrt{1-\delta}} - \dfrac{\sqrt{3}}{3} & t < \sqrt{3} \end{cases}$$

As

$$\frac{\partial(\frac{\sqrt{1+\delta}}{\sqrt{3}\sqrt{1-\delta}}-\frac{\sqrt{3}}{3})}{\partial\delta} = \frac{\sqrt{3}}{6}\frac{\sqrt{1+\delta}+\sqrt{1-\delta}}{1-\delta} > 0, \tag{14}$$

Therefore, for a given δ, when $t \geq \sqrt{3}$, the probability of cause an error is 0; when $t < \sqrt{3}$, the greater the δ is, the higher the error probability is. According to the Johnson Lindenstrauss Lemma, $m = \Omega(log(n)/\delta^2)$. We can infer that for a fixed n, $m \sim \frac{1}{\delta^2}$. Therefore, we can draw the conclusion that greater value of m helps to reduce the error probability.

Case 2: For a given t, where $t > 1$. Equation 15 can be written as

$$\begin{aligned} Pr[\| f(u)-f(a) \| &\geq \| f(u)-f(b) \|] \\ &\leq Pr[t \leq \frac{\sqrt{1+\delta}}{\sqrt{1-\delta}}] \\ &= Pr[\delta \geq \frac{t^2-1}{t^2+1}]. \end{aligned} \tag{15}$$

As $0 < \delta < \frac{1}{2}$, when $\frac{t^2-1}{t^2+1} \geq \frac{1}{2}$, that is $t \geq \sqrt{3}$, $Pr[\delta \geq \frac{t^2-1}{t^2+1}] = 0$. When $1 < t < \sqrt{3}$, $Pr[\delta \geq \frac{t^2-1}{t^2+1}] = \frac{3-t^2}{t^2+1}$.

Equation 15 can be written as

$$Pr[error] \leq \begin{cases} 0 & t \geq \sqrt{3} \\ \dfrac{3-t^2}{t^2+1} & t < \sqrt{3} \end{cases}$$

As $\frac{\partial \frac{3-t^2}{t^2+1}}{\partial t} = -\frac{8t}{(t^2+1)^2} < 0$, when $t < \sqrt{3}$, the error probability is monotonically decreasing with the increasing of t. Which means the larger the proportional distance between two POIs, the higher the accuracy achieved.

Overall, when $t \geq \sqrt{3}$, the Johnson-Lindenstrauss transform cause no error. When $t \leq \sqrt{3}$, the greater the m and the greater the t, the smaller the error rate.

VI. EVALUATION AND DISCUSSION

We evaluated the performance of our privacy framework through an extensive set of experiments. First, we present the experiment settings, and then discuss the experimental results.

Datasets: We used two real-world datasets. SimpleGeo Places dataset [13] and Yelp business dataset [14]. We extracted 8275 business entries in the area of Sydney from SimpleGeo dataset, and 22830 business entries in Las Vegas from Yelp dataset.

Metrics: The effectiveness of the proposed method was evaluated by comparing the similarities between the result sets. We evaluated the accuracy of our method in terms of displacement, resemblance [15] and recall.

1) *Resemblance.* Let $P = \{p_1, p_2, \dots p_k\}$ be the POI set retrieved by the POI query, relative to the true location l_t of the user u, and $P' = \{p'_1, p'_2, \dots p'_k\}$ be the retrieved POI set based on the perturbed location. Resemblance measures the fraction of POIs in the actual result set that is included in the approximated result set.

$$Resemblance = \frac{|P \cap P'|}{|P|}, \tag{16}$$

where $|P|$ is the size of the set P.

2) *Displacement.* Displacement measures how closely P is measured by P' on average. It shows the average difference between the real POIs distance across the mismatched POIs on $k-NN$ query.

$$Displacement = \frac{\Sigma_{i=1}^{k}\|l_t - p_i\| - \Sigma_{i=1}^{k}\|l_t - p'_i\|}{|P|}, \tag{17}$$

where $\|\cdot\|$ is the Euclidean distance between the true location of the user and the location of the POI.

3) *Recall.* We use *recall* to evaluate the proportion of relevant POIs retrieved on range query.

$$Recall = Resemblance = \frac{|P \cap P'|}{|P'|}, \tag{18}$$

where the length of P can be different from P'.

Experimental Setup: We used 1000 users to retrieve the k-nearest POIs and POIs in a region with an r radius corresponding to the true and perturbed locations. For the *SimpleGeo* dataset, we choose *Restaurant* and *Shopping* locations as interesting POIs and 1000 users were chosen randomly from the *Professionals* attribute. For the *Yelp* dataset, we choose *Nightlife* and *Beauty & Spas* as interesting POIs and 1000 users were chosen randomly from *Restaurants*. All of these query strings reflect different POI densities. Given the proposed method is based on a random mapping, we repeat each experiment 20 times and used the average to ensure accuracy. All algorithms were implemented in Matlab on a PC with 2.7 GHz Intel Core i5 Processor and 8 GB Memory. Table 3 shows the parameters used.

TABLE 3. Parameter settings.

Parameter	Description	Value range	Default
m	dimension of transition matrix	$2 \sim 50$	10
k	number of queried POIs	$1 \sim 50$	20
r	radius of range query	$100 \sim 1000$	400
ϵ	privacy budget	–	0.5
R	privacy protection radius	–	2km

We compared our method to two other differential privacy-based methods with a client-server structure, as no other similar technique applying a semi-trusted third party.

1) *The Geo-indistinguishability method.* This concept was proposed by Andrés *et al.* [16] in 2013. It states that for any x and x' within radius r, the distance $d(K(x), K(x'))$between the corresponding distributions should at most be l, where the mechanism K is a probabilistic function for selecting a reported value, and $l = \epsilon r$. One idea proposed by Chatzikokolakis *et al.* [17] for achieving geo-indistinguishability is that whenever the actual location is x_0, instead, a point x is randomly generated instead, according to the planar Laplace noise function.

2) *A simple Laplace method.* In this approach, the user's location is perturbed by adding Laplace noise to the x and y coordinates independently. The noise is determined by the privacy parameter ϵ and sensitivity s, where the sensitivity equals the radius r, so that the user cannot be distinguished with from other users within r.

To make these methods comparable, we assumed the neighbor location $l'(x', y')$ in our method was within the radius r of true location $l(x, y)$. Then, $\Delta = |\sqrt{x'^2 + y'^2} - \sqrt{x^2 + y^2}| \ll \lambda$. Therefore, $\epsilon \ll 1$. We made $\epsilon = 0.5$ for all methods. The maximum radius calculated in paper [17] is $2km$; therefore, $r = 2km$ in the simple differential privacy method. Our method and the two other methods are denoted as JL, GEO, and DP, respectively.

A. PERFORMANCE OF THE PROPOSED METHOD

1) k-NN QUERY

We examined the performance of the proposed method in relation to the number of queried POIs k for k-NN queries

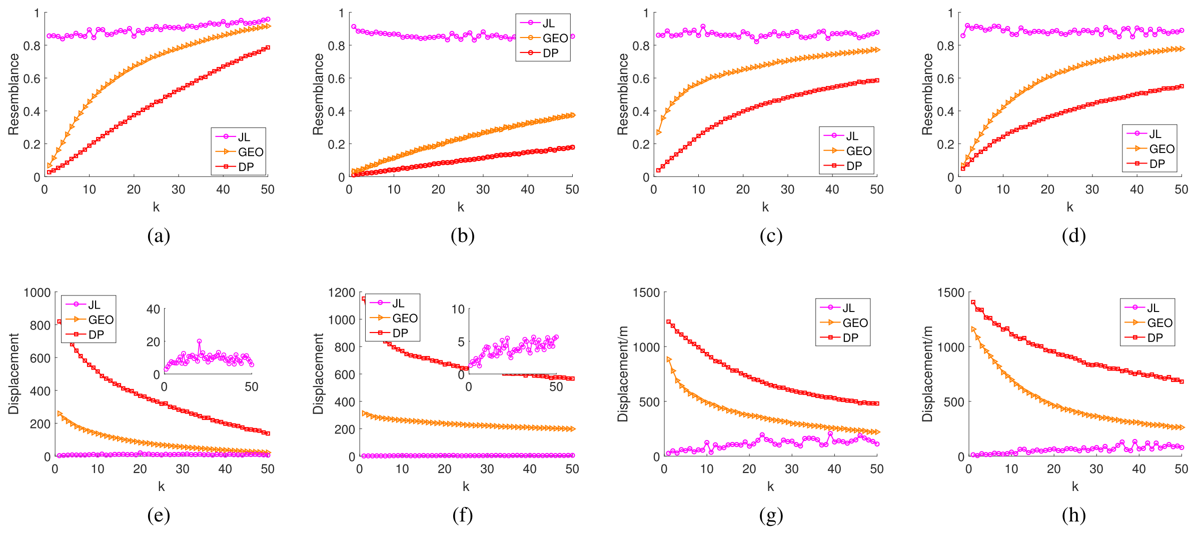


FIGURE 8. ***k*-NN query performance. (a) Restaurant (SimpleGeo). (b) Shopping(SimpleGeo). (c) Nightlife (Yelp). (d) Beauty & Spas (Yelp). (e) Restaurant (SimpleGeo). (f) Shopping(SimpleGeo). (g) Nightlife (Yelp). (h) Beauty & Spas (Yelp).**

in terms of resemblance and displacement. We varied the number of queried POIs between 1 and 50 in Step 1 on both datasets.

The resemblance values corresponding to different values of k for both the SimpleGeo and Yelp datasets are shown in Figs. 8a, 8b, 8c, and 8d. It is clear that JL significantly outperformed GEO and DP in all configurations. As shown in Fig. 8a, when $k = 5$ and nearby Restaurants were queried from the SimpleGeo dataset, GEO achieved a resemblance of 0.2560, DP achieved a resemblance of 0.0850, and JL achieved a resemblance of 0.8582, which outperformed GEO by 60% and DP by 77%. A similar result was also observed when measuring resemblance on the Yelp dataset. Obviously, GEO and DP performed very badly when $k < 10$ on both datasets, but they were greatly improved by increasing the value of k. However, the number of queried POIs had little effect on JL in terms of the resemblance metric. The JL method always performed well, even when k was small, because its utility is related to the dimensions m and t, as shown in Section V-B. For a fixed m, a larger t helped maintain a higher accuracy and was not affected by the size of k.

The displacement values corresponding to different values of k on both the SimpleGeo and Yelp datasets are shown in Fig. 8e, 8f, 8g, and 8h. We observed that JL had a much lower displacement value than both GEO and DP when querying all attributes on the two datasets. Additionally, we observed that the size of k did not affect the performance of JL significantly in terms of the displacement metric, while GEO and DP increased faster when k decreased from 20 to 1. For example, in Fig. 10a, the displacement JL achieved across all values of k was under $20m$. However, when $k = 5$, GEO and DP achieved displacements of 181.4295 and 642.0554, respectively, much larger than the displacement values of $83.3879m$ for GEO and $367.0430m$ for DP when $k = 20$. A similar observation was found in Fig. 8f, 8g, and 8h. This indicates that a smaller k means a larger difference between the returned POIs based on the true location and the false location for both the GEO and DP methods. JL had no such problem. The same observation was made when considering performance in terms of the resemblance metric.

The proposed JL method has such excellent performance because it simultaneously perturbs the user's location and the POI locations by mapping them to different dimensions instead of reporting false locations. And the relative distances between records are nearly preserved. However, any method that reports a false location will have a large margin of error when querying only a few nearby POIs, especially when the distribution of the queried POIs is compact. It is easy to deduce that the closest POI to the true location is not the closest one to the false location.

2) RANGE QUERY

We further examined the performance of the proposed method in relation to the radius r for range queries in terms of resemblance and recall. We varied the radius r between $100m$ and $1000m$ in steps of 100 on both datasets.

The resemblance results of the three methods on the SimpleGeo and Yelp datasets are shown in Fig. 9a, 9b, 9c, and 9d. Obviously, JL had a higher resemblance value than the other two methods-irrespective of the value of r. For the JL method, the resemblance metric was stable and close to 1 even with an increasing r, which indicates the dataset's utility was not overly sacrificed, and the accuracy of the returned POIs was mostly enhanced. However, as shown in Fig. 9a, when $r < 600$, neither GEO nor DP maintained good utility. Specifically, when $r = 800$, GEO and DP achieved resemblances of 0.8291 and 0.2712, respectively. JL achieved a resemblance of 0.9569, an improvement of 13% and 68%,

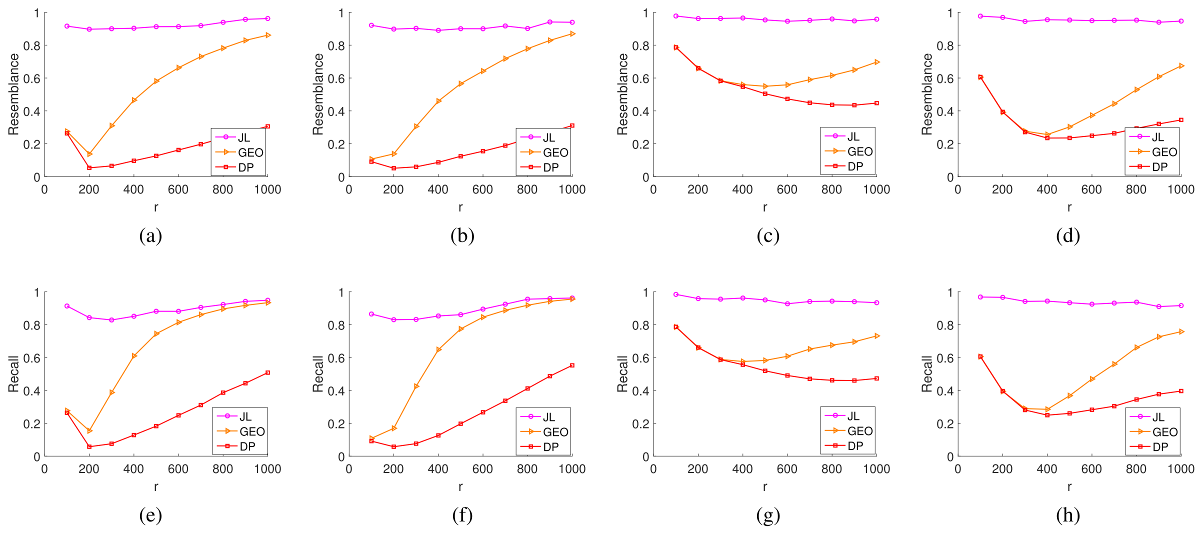


FIGURE 9. Range query performance. (a) Restaurant (SimpleGeo). (b) Shopping(SimpleGeo). (c) Nightlife (Yelp). (d) Beauty & Spas (Yelp). (e) Restaurant (SimpleGeo). (f) Shopping(SimpleGeo). (g) Nightlife (Yelp). (h) Beauty & Spas (Yelp).

respectively. When $r = 300$, JL achieved a resemblance of 0.9001, outperforming GEO by 60% and DP by 84%. These improvements by JL were observed on the Yelp dataset. As shown in Fig. 9d, when $r = 400$, GEO and DP achieved resemblances of 0.2560 and 0.2347, respectively. JL outperformed them by 70% and 72% with a resemblance of 0.9548. When $r = 800$, JL achieved a resemblance of 0.9521, which is an improvement of 43% and 66% over GEO and DP's resemblances of 0.5291 and 0.2935, respectively. Figs. 9e, 9f, 9c, and 9d show the results for the SimpleGeo dataset in terms of recall. We observed that the performance of the three methods was similar to the results of the resemblance metric. JL outperformed GEO and DP across all r values. JL was relatively stable compared to GEO and DP - they changed significantly when varying the radius r.

We attribute the reason JL performed so well to the properties of the Johnson-Lindenstrauss transform. The distances between the locations were almost fully preserved after the transformation, regardless of the queried radius r. However, in the GEO and DP methods, if the queried range is very small, it is possible for there to be no overlap between the true POIs and the perturbed POIs. So, the JL method outperforms GEO and DP significantly when r is small.

B. THE IMPACT OF POI DENSITY

To examine the impact of POI density on the three methods for both k-NN queries and range queries. We choose three attributes that reflect different POI densities in the SimpleGeo and Yelp datasets.

1) k-NN QUERIES

The resemblance and displacement values corresponding to the three methods in two datasets by querying three different k-NN POIs when $k = 20$ are shown in Figs. 10a - 10d. The POI density had little effect on JL; however, it affected the GEO and DP significantly. As shown in Fig. 10a, JL's resemblance value increased slightly when increasing the POI density, while the resemblance value decreased significantly for the GEO and DP methods. Specifically, in Fig. 10a, JL achieved approximately 0.83 resemblance for all POI queries on the SimpleGeo dataset. While, when querying Restaurant, the resemblance achieved by GEO was 0.6730. When querying nearby Shopping and Health Service, GEO achieved resemblances of 0.1960 and 0.1159, respectively, a decrease of 48% and 8%. The resemblance achieved by DP decreased by approximately 30% and 5%. The resemblance results on the Yelp dataset closely resembled those on SimpleGeo, shown in Fig. 10c. Accordingly, the displacement decreased slightly for JL and increased significantly for both GEO and DP. As shown in Fig. 10b, JL achieved displacements of 20.0431, 6.0760, and 3.9532 for Restaurant, Shopping, and Health Service, respectively. While GEO achieved 83.3879, 199.6278, and 236.7437, an increase of around 116 and 27 when increasing the POI density. DP achieved 367.0430, 670.1029, and 689.0549, an increase of 303 and 19.

Because for GEO and DP methods, when the density increased, more mismatched POIs reduced the accuracy. However, for JL method, the k nearest POIs were much closer to the user as the density increased. Therefore, the proportion of the distances became slightly larger compared to low-density distribution. According to our utility analysis, accuracy increases as the distance proportionally increases between two POIs and the user.

2) RANGE QUERY

The effects of the POI density on the range query are shown in Figs. 10e - 10h. It is clear that the POI density had no obvious impacts on JL for both datasets. For example, JL achieved

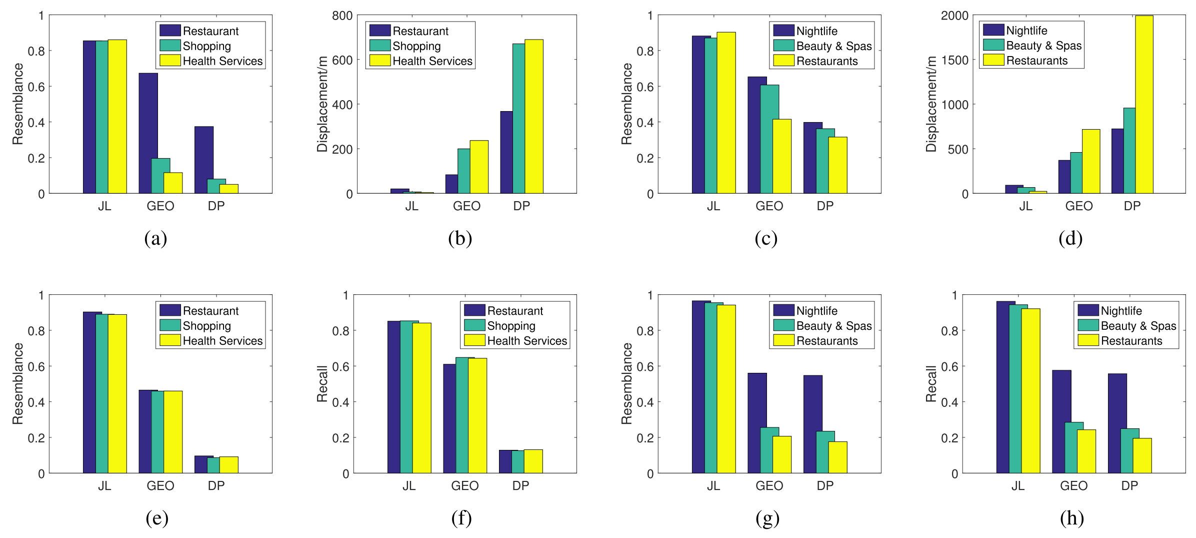


FIGURE 10. The effect of density on the *k*-NN and range queries. (a) SimpleGeo/*k*-NN. (b) SimpleGeo/*k*-NN. (c) Yelp/*k*-NN. (d) Yelp/*k*-NN. (e) SimpleGeo/range. (f) SimpleGeo/range. (g) Yelp/range. (h) Yelp/range.

a resemblance of approximately 0.89 for all three different POI queries in Fig. 10e on SimpleGeo dataset, and achieved a recall of approximately 0.86 in Fig. 10f. Similar results can be observed in Fig. 10g and Fig. 10h on Yelp dataset. However, there was a different result for GEO and DP. Both resemblance and recall slight changed on SimpleGeo dataset, but had a significant reduction in the performance when increasing the POI density of the queries on Yelp dataset. For instance, the GEO method achieved a resemblance of 0.5600 when querying nearby Nightlife; however and reduced to 0.2560 when querying Beauty & and Spas. The performance was much worse when querying Restaurants, which has the largest density. The DP method had similar results. Fig. 10h shows a similar trend in terms of recall.

As we mentioned, for a given m, the performance of JL is only related to the ratio of distances between the user and POIs t. The change of POI density cannot affect the JL performance significantly for range query.

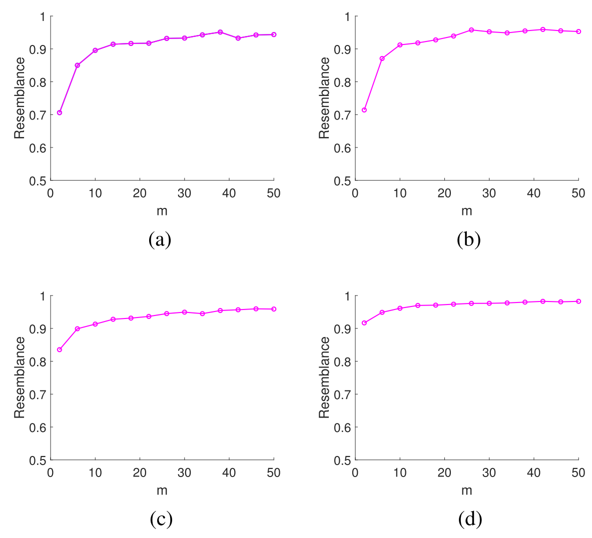


FIGURE 11. The effect of *m* on *k*-NN and range queries. (a) SimpleGeo/*k*-NN. (b) Yelp/*k*-NN. (c) SimpleGeo/range. (d) Yelp/range.

C. IMPACT OF TRANSITION MATRIX M

In the proposed method, m is an important parameter that determines the length of the perturbed location vector. To determine how m contributed to the results, we changed the value of m from 2 to 50 in steps of 4 on both the SimpleGeo and Yelp datasets.

The results on SimpleGeo and Yelp for the k-NN queries are shown in Fig. 11. JL's performance was greatly improved by increasing the value of m. When the dimension of the transition matrix increased, resemblance showed an upward trend on both datasets. As shown in Fig. 11a, when $m = 2$ for SimpleGeo querying nearby POIs resulted in a resemblance of 0.7. When m is increased, the resemblance increases rapidly. When $m = 14$, the resemblance $= 0.9$. Performance started to level off with an m higher than 14. A similar result is observed when querying nearby POIs on the Yelp dataset. As shown in Fig. 11b, when m is in the range of [2, 14], the changes in resemblance are significant. The query result is much more accurate when the dimensions of the transition matrix are much higher. When it reaches a threshold, performance is not expected to improve dramatically. Fig. 11d and Fig. 11d show the results of the range queries on two datasets. We can observe that the increasing parameter m has a positive effect on performance. For instance, in Fig. 11c, when $m = 2$, resemblance is around 0.82. When $m = 10$, a resemblance of 0.91 is achieved.

The relationship between the two fixed POIs did not change, which means t is invariable. According to the analysis in Section V-B, when t is fixed, the probability of an error

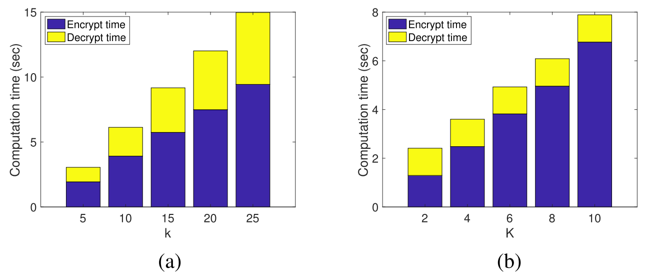


FIGURE 12. **Computational cost of the proposed method. (a) Computational cost by varying k. (b) Computational cost by varying K.**

decreases with the increase in the value of m. Therefore, when m becomes larger, the error probability is reduced, and better performance is achieved.

D. OVERHEAD ANALYSES

In this section, we analyse the computational cost, query process delay and transmission overhead induced by the proposed privacy framework. There are three processes may cause the delay: map sanitization, POI encryption, and POI decryption. Because in the map sanitization process, only the queried POIs in the safe region are transformed by the transition matrix, the time cost is very short that can be ignored. Therefore, we tested the computational cost of POI encryption and POI decryption under the proposed privacy framework by varying the number of queried POIs and number of dummy POI sets.

Fig. 12 shows the results. The dark blue regions in the bars indicate the computational cost at the third party for encrypting the POI information. The yellow regions in the bars indicate the computation burden at the user side for decrypting the received POI information. Therefore, the whole bars indicates the overall query response time, that is the delay induced by the proposed privacy framework.

Fig. 12a hows the computation time cost by varying the number of queried POIs, where $K = 3$. We can observe that the time cost is increasing with the increase of the number k, no matter the encrypt time or decrypt time. This is because when the number of queried POIs increases, the number of POIs needed to be encrypted and decrypted increases. Therefore, both time costs are increasing. Fig. 12b shows the computation time cost by varying the number of dummy POI sets, where $k = 5$. We observe that the encrypt time increases with the increase of number K, however, the decrypted time does not change. This is because the user only needs to decrypt the queried POIs. When the number of queried POIs does not change, the time cost caused by decryption does not change.

It's clear that the whole query response time is very short that was controlled within few seconds. The delay induced by the proposed privacy framework is negligible compared with tradition encryption-based method. This is because in the proposed method, the encryption technology is only used to hide the POI information. It was never used to calculate or search the POI records. In addition, only few POI records are encrypted instead of the whole region. Also, there is no big transmission overhead, because only the queried POIs in a safe region and limited encrypted POIs are transmitted between the service provider and the third party.

VII. RELATED WORKS

Numerous techniques have been provided to protect a user's location privacy [7], [18]–[22]. The current main techniques for preserving privacy in LBSs are described in the following review.

Most proposed methods are designed to operate in client-server structures. In dummy location methods, multiple false locations along with the user's true location are sent to the LBS server such that the true location cannot be distinguished from the false locations [18]. Obfuscation techniques try to protect a user's location by reducing the precision of the position information, reporting a region to the LBS server instead of the precise user location [23]. Beyond the cloaking method, Gutscher [24] proposed a coordinate transformation method. The user performs some basic geometric operations over their positions, such as shifting and rotating, before sending it to the LBS server. Differential privacy-based perturbation methods have also been proposed. Dewri [3] proposed a method that perturbs the user's location by adding Laplace noise to the x and y coordinates independently. Andrés *et al.* [16] introduced the notion of geo-indistinguishability to formalize the problem of protecting a single user's position. It achieves privacy by adding controlled noise to the user's location making the user's location indistinguishable within a radius r [17]. Private information retrieval-based protocols [25] were also proposed for POI queries. This technique allows a user to retrieve a record from a database server without revealing any information about the query. However, such cryptography-based approaches rely on heavy cryptographic mechanisms, which are often computationally and communicatively expensive.

A few privacy-preserving techniques have attempted to user trusted third parties (TTP) for location-based services. The most commonly used TTP approaches rely on an anonymizer to create a spatial region that includes at least $k - 1$ other users to hide the true location [19]. Because k-anonymity is achieved, an adversary can only identify a user's true location with probability no higher than $1/k$. Papadopoulos *et al.* [26] employed trusted hardware to perform PIR for LBS queries. Their hardware-aided PIR technique relies on a trusted third party to set a secret key and permutate the database. However, the third party is aware of the precise user positions and is, therefore, vulnerable to misbehavior by the fully-trusted third party. Recently, Schlegel *et al.* [27] proposed a dynamic grid system to provide privacy-preserving continuous LBS. They introduced a semi-trusted third party, responsible for simple matching operations. However, their method is based on complicated cryptographic functions, which is, as previously mentioned, carries a heavy computational burden.

VIII. CONCLUSION

In this paper, we have proposed a new privacy preservation framework that includes three elements: a user, a third party, and the LBS server. The third party is a semi-trusted entity based on a weak assumption that it does not collude with the LBS server. In addition, we propose a novel privacy protection scheme, based on the Johnson-Lindenstrauss transform. Moreover, the user's exact location is perturbed by Johnson-Lindenstrauss which satisfies the definition of differential privacy, and maintains high level application utility at the same time. The combination of privacy preservation method and client-bridge-server structure ensure that none of the party is aware of the user's exact location. Our method supports two very popular queries: k-nearest-neighbour queries and range queries. Performance is evaluated through extensive experiments, and our proposed method is further compared to two representative differential privacy-based methods. The results demonstrate that our framework provides better privacy guarantees and is more efficient in terms of resemblance, displacement, and recall.

REFERENCES

[1] F. Li, S. Wan, B. Niu, H. Li, and Y. He, "Time obfuscation-based privacy-preserving scheme for location-based services," in *Proc. WCNC*, Doha, Qatar, Apr. 2016, pp. 1–6.

[2] Y. Xiao and L. Xiong, "Protecting locations with differential privacy under temporal correlations," in *Proc. SIGSAC*, Denver, CO, USA, 2015, pp. 1298–1309.

[3] R. Dewri, "Local differential perturbations: Location privacy under approximate knowledge attackers," *IEEE Trans. Mobile Comput.*, vol. 12, no. 12, pp. 2360–2372, Dec. 2013, doi: 10.1109/TMC.2012.208.

[4] W. Eltarjaman, R. Dewri, and R. Thurimella, "Private retrieval of POI details in top-K queries," *IEEE Trans. Mobile Comput.*, vol. 16, no. 9, pp. 2611–2624, Sep. 2017, doi: 10.1109/TMC.2016.2625256.

[5] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. TCC*, New York, NY, USA, 2006, pp. 265–284.

[6] N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Optimal geo-indistinguishable mechanisms for location privacy," in *Proc. SIGSAC*, Scottsdale, AZ, USA, 2014, pp. 251–262.

[7] A. Palia and R. Tandon. (May 2017). "Optimizing noise level for perturbing geo-location data." [Online]. Available: https://arxiv.org/abs/1705.02108

[8] S. Dasgupta and A. Gupta, "An elementary proof of the Johnson-Lindenstrauss lemma," Int. Comput. Sci. Inst., Berkeley, CA, USA, Tech. Rep. 99-006, 1999.

[9] J. Blocki, A. Blum, A. Datta, and O. Sheffet, "The johnson-lindenstrauss transform itself preserves differential privacy," in *Proc. FOCS*, New Brunswick, NJ, USA, Oct. 2012, pp. 410–419.

[10] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," *Contemp. Math.*, vol. 26, no. 1, pp. 189–206, 1984.

[11] D. Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *J. Comput. Syst. Sci.*, vol. 66, no. 4, pp. 671–687, Jun. 2003, doi: 10.1016/S0022-0000(03)00025-4.

[12] P. Gilbert, L. P. Cox, J. Jung, and D. Wetherall, "Toward trustworthy mobile sensing," in *Proc. Workshop Mobile Comput. Syst. Appl.*, Annapolis, MD, USA, 2010, pp. 31–36.

[13] *SimpleGeo Public Spaces CC0 Collection*. Accessed: 2011. [Online]. Available: http://archive.org/details/2011-08-SimpleGeo-CC0-Public_Spaces

[14] *Yelp Dataset Challenge*. Accessed: 2017. [Online]. Available: https://www.yelp.com/dataset_challenge

[15] A. Khoshgozaran and C. Shahabi, "Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy," in *Proc. SSTD*, Boston, MA, USA, 2007, pp. 239–257.

[16] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *Proc. CCS*, Berlin, Germany, 2013, pp. 901–914.

[17] K. Chatzikokolakis, C. Palamidessi, and M. Stronati, "Geo-indistinguishability: A principled approach to location privacy," in *Proc. ICDCIT*, Bhubaneswar, Odisha, 2015, pp. 49–72.

[18] T. Hara, A. Suzuki, M. Iwata, Y. Arase, and X. Xie, "Dummy-based user location anonymization under real-world constraints," *IEEE Access*, vol. 4, pp. 673–687, 2016, doi: 10.1109/ACCESS.2016.2526060.

[19] F. Fei, S. Li, H. Dai, C. Hu, W. Dou, and Q. Ni, "A K-anonymity based schema for location privacy preservation," *IEEE Trans. Sustain. Comput.*, to be published, doi: 10.1109/TSUSC.2017.2733018.

[20] Y. Qu, S. Yu, L. Gao, and J. Niu, "Big data set privacy preserving through sensitive attribute-based grouping," in *Proc. ICC*, Paris, France, May 2017, pp. 1–6.

[21] B. Liu, W. Zhou, T. Zhu, L. Gao, T. H. Luan, and H. Zhou, "Silence is golden: Enhancing privacy of location-based services by content broadcasting and active caching in wireless vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9942–9953, Dec. 2016, doi: 10.1109/TVT.2016.2531185.

[22] Z. Wang *et al.*, "Heterogeneous incentive mechanism for time-sensitive and location-dependent crowdsensing networks with random arrivals," *Comput. Netw.*, vol. 131, pp. 96–109, Feb. 2018, doi: 10.1016/j.comnet.2017.12.010.

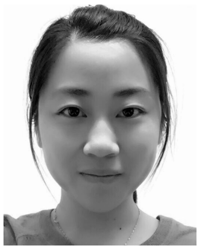
[23] M. Xue, P. Kalnis, and H. K. Pung, "Location diversity: Enhanced privacy protection in location based services," in *Proc. LoCA*, Tokyo, Japan, 2009, pp. 70–87.

[24] A. Gutscher, "Coordinate transformation—A solution for the privacy problem of location based services?" in *Proc. IPDPS*, Rhodes Island, Greece, Apr. 2006, pp. 1–8.

[25] X. Yi, R. Paulet, E. Bertino, and V. Varadharajan, "Practical approximate k nearest neighbor queries with location and query privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1546–1559, Jun. 2016, doi: 10.1109/TKDE.2016.2520473.

[26] S. Papadopoulos, S. Bakiras, and D. Papadias, "Nearest neighbor search with strong location privacy," *Proc. VLDB Endowment*, vol. 3, nos. 1–2, pp. 619–629, Sep. 2010.

[27] R. Schlegel, C. Y. Chow, Q. Huang, and D. S. Wong, "User-defined privacy grid system for continuous location-based services," *IEEE Trans. Mobile Comput.*, vol. 14, no. 10, pp. 2158–2172, Oct. 2015, doi: 10.1109/TMC.2015.2388488.

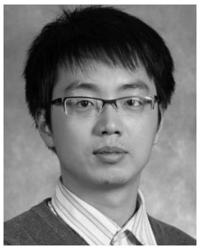


MENGMENG YANG received the B.Eng. degree from Qingdao Agricultural University, China, in 2011, and the M.Eng. degree from Shenyang Normal University, China, in 2014.

She is currently pursuing the Ph.D. degree with the School of Information Technology, Deakin University, Australia. Her research interests include privacy preserving, machine learning, and network security.

TIANQING ZHU received the B.Eng. and M.Eng. degrees from Wuhan University, China, in 2000 and 2004, respectively, and the Ph.D. degree in computer science from Deakin University, Australia, in 2014.

From 2014 to 2018, she was a Lecturer with the School of Information Technology, Deakin University, Australia. She is currently a Senior Lecturer with the School of Software, University of Technology Sydney, Australia. Her research interests include privacy preserving, data mining, and network security.



BO LIU received the B.Sc. degree from the Department of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing, China, in 2004, and the M.Eng. and Ph.D. degrees from the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2007 and 2010 respectively.

From 2010 and 2014, he was an Assistant Research Professor with the Department of Electronic Engineering, Shanghai Jiao Tong University. From 2014 to 2017, he was a Post-Doctoral Research Fellow with Deakin University, Australia,. Since 2017, he has been a Lecturer with the Department of Engineering, La Trobe University. His research interests include wireless communications and networking and security and privacy issues in wireless networks.

YANG XIANG (SM'06) received the Ph.D. degree in computer science from Deakin University, Australia.

He is the Dean of Digital Research & Innovation Capability Platform, Swinburne University of Technology, Australia. He is currently leading his team developing active defense systems against large-scale distributed network attacks. He has published over 200 research papers in many international journals and conferences. His research interests include cyber security, which covers network and system security, data analytics, distributed systems, and networking.

He has been the PC member for over 80 international conferences in distributed systems, networking, and security. He has served as the Program/General Chair for many international conferences. Two of his papers were selected as the featured articles in the April 2009 and the July 2013 issues of the IEEE Transactions on Parallel and Distributed Systems. Two of his papers were selected as the featured articles in the July/August 2014 and the November/December 2014 issues of the IEEE Transactions on Dependable and Secure Computing. He is the Chief Investigator of several projects in network and system security, funded by the Australian Research Council.

WANLEI ZHOU (SM'91) received the B.Eng. and M.Eng. degrees in computer science and engineering from the Harbin Institute of Technology, Harbin, China, in 1982 and 1984, respectively, the Ph.D. degree in computer science and engineering from The Australian National University, Canberra, ACT, Australia, in 1991, and the D.Sc. degree from Deakin University in 2002.

He is currently the Head of School of Software, University of Technology Sydney, Australia. He was an Alfred Deakin Professor and the Chair of information technology with Deakin University. He has published over 300 papers in refereed international journals and refereed international conferences proceedings. His research interests include distributed systems, network security, and privacy preserving.

Dr. Zhou has chaired many international conferences and has been invited to deliver keynote address in many international conferences.