EPGQ: Efficient and Private Feature-based Group Nearest Neighbor Query over Road Networks

Yunguo Guan, Rongxing Lu, *Fellow, IEEE,* Yandong Zheng, Songnian Zhang, Jun Shao, *Senior Member, IEEE,* and Guiyi Wei

***Abstract*—The rapidly growing location-based services enable service providers to accumulate plentiful descriptions on points of interest (POI), which can be used to support expressive POI queries. In this paper, we study a type of POI query, named feature-based group k nearest neighbor query over road networks, in which a user has a feature set and several locations and wishes to find k closest POIs that have similar sets of features to the query. As the POI datasets grow, service providers tend to outsource their datasets to a powerful yet not-fully-trusted cloud, which calls for privacy preservation on datasets and user queries. Although many schemes have been proposed for privacy-preserving POI queries, none of them can simultaneously support privacy-preserving set similarity and road network distance comparison. To address this challenge, we propose an efficient and private feature-based group nearest neighbor query scheme. In our scheme, we achieve privacy-preserving distance comparison by employing the road network hypercube embedding technique, and design an encrypted index based on B+-tree for privacy-preserving set similarity range queries. Security analysis shows our proposed scheme can preserve the privacy of the dataset and queries, and performance evaluation also demonstrates it is computationally efficient.**

***Index Terms*—Privacy-preserving, Location-based services, Group nearest neighbor query, Feature similarity, Road network.**

I. Introduction

THE proliferation of smartphones, positioning and wireless communication technologies has stimulated the wider use of location-based services (LBS). Meanwhile, as LBS becoming part of our daily life, the service providers accumulate a huge volume of descriptions (e.g., check-in data and tags) over points of interest (POI). These data can be used to support more expressive POI queries, such as skyline queries [1]–[3] and group k nearest neighbor (GkNN) queries [4], [5]. In this paper, we consider feature-based GkNN queries motivated by the scenario where a user wants to find the k POIs that are located closely to a set of referenced locations and have sets of features similar to query features. For instance, when a user visits a city in his/her first time, the trip may consist of multiple locations in the city. The user can employ the GkNN query to find the k-nearest hotels to the locations which have sets of features similar to the user's preference, e.g., whether they provide breakfast and are beachfront or not. Specifically, we employ the road network distance as the distance metric to better capture distances between locations in the city, and the similarity between the feature sets is described by Jaccard similarity. As the volume of the dataset grows, it may exceed the capability of the service providers. Hence, they tend to outsource the dataset and the feature-based GkNN query services to a powerful yet not-fully-trusted cloud, similar to many other services [6]–[8].

While enjoying the reliable quality of service, the service provider and query users still have their privacy concerns. As the dataset is the service provider's property, it prefers to outsource the services while preserving the data privacy against the cloud. For the query users, the query requests contain their preferences and planned locations, and the users want to enjoy the services without leaking the information to other entities in the system. An intuitive approach is to use privacy preservation techniques, but it will obstruct the cloud from querying the protected data. As detailed in Section VII, many schemes [9]–[17] have been proposed for privacy-preserving kNN queries over road networks or GkNN queries. For the privacy-preserving kNN schemes over road networks [9]–[13], they cannot be efficiently adapted to support GkNN queries. While for the GkNN schemes [14]–[17], they either focus on Euclidean distance or do not consider dataset privacy against. Furthermore, these schemes cannot support queries with feature similarity criteria. Therefore, the privacy-preserving feature-based GkNN query is still challenging.

Aiming at the above challenges, this paper propose an efficient and private feature-based group nearest neighbor query (EPGQ) scheme that supports road network distance. Specifically, the contributions of this paper are three-fold.

- First, we design a privacy-preserving distance comparison (PDC) technique based on the road network hypercube embedding technique, which can compare the sums of road network distances between two records and a set of reference locations without revealing the locations and the difference of the sums.
- Second, we build an encrypted index to organize the encrypted dataset and support feature-based GkNN queries. With the encrypted index, an encrypted feature-based GkNN query can be conducted in a filtration-and-verification manner, and both the prefix and length of the feature sets are considered during the filtration step.
- Third, we conduct rigid security analysis to show that our PDC technique is selectively secure, and our proposed EPGQ scheme can well preserve the privacy of the dataset, query requests, and the corresponding query results. Performance

Y. Guan, R. Lu, Y. Zheng, and S. Zhang are with the Faculty of Computer Science, University of New Brunswick, Fredericton, Canada E3B5A3. E-mail: yguan4@unb.ca, rlu1@unb.ca, yzheng8@unb.ca, szhang17@unb.ca.

J. Shao and G. Wei are with Zhejiang Gongshang University, Hangzhou, China 310018. E-mail: chn.junshao@gmail.com, weigy@zjgsu.edu.cn.

analysis also demonstrates that our proposed scheme is indeed efficient for both query users and the cloud server.

The remainder of this paper is organized as follows. In Section II, we present the formal definition of the feature-based GkNN query and the road network hypercube embedding (RNHE) as our preliminaries. Then, we formalize the system model and security model, and identify our design goal in Section III. In Section IV, we present our EPGQ scheme, followed by its security and performance analysis in Section V and Section VI, respectively. Finally, we review some related works in Section VII and conclude this work in Section VIII.

II. Preliminaries

In this section, we formalize our feature-based group kNN query and recall the road network hypercube embedding technique as our preliminaries.

A. Feature-Based Group kNN Query

Given a set of POI (points of interests) records $\mathcal{P}$, where each POI $p \in \mathcal{P}$ has a set of features $F_p \subseteq \mathcal{F}$ and a location l_p, and $\mathcal{F}$ is the space of all possible features in the system, a feature-based group kNN query can be defined as follows.

Definition 1 (Feature-Based Group kNN Query). Given a set of features $F_q \subseteq \mathcal{F}$, a similarity threshold τ, and a set of reference points $Q = \{l_{q_i} \mid i = 1, 2, \cdots\}$, a feature-based group kNN query $\mathtt{q} = \{F_q, \tau, Q\}$ is to obtain a set of k POI records $\mathcal{P}_q$ from $\mathcal{P}$ satisfying the following two conditions.

i) $\mathcal{P}_q \subseteq \mathcal{P}_{F_q,\tau} = \{p \mid p \in \mathcal{P}, \mathtt{sim}(F_p, F_q) \geq \tau\}$, where $\mathtt{sim}(\cdot,\cdot)$ is the similarity function;

ii) $\sum_{l_{q_i} \in Q} \mathtt{dist}(l_p, l_{q_i}) \leq \sum_{l_{q_i} \in Q} \mathtt{dist}(l_{p'}, l_{q_i})$, for all $p \in \mathcal{P}_q$ and $p' \in \mathcal{P}_{F_q,\tau} \setminus \mathcal{P}_q$, where $\mathtt{dist}(\cdot,\cdot)$ represents the distance function.

Here, we choose the Jaccard similarity as the similarity metric. Specifically, given two sets F_p and F_q, their Jaccard similarity is defined as $\mathtt{sim}(F_p, F_q) = \frac{|F_p \cap F_q|}{|F_p \cup F_q|}$.

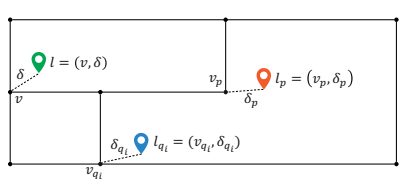


Fig. 1: An example of a road network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W(\cdot))$ where each location l is described as a tuple $l = (v, \delta)$, i.e., the nearest vertex $v \in \mathcal{V}$ to l and the Euclidean distance δ between l and v.

Furthermore, in this work, we consider all POI's locations together with all reference points Q as the locations in the road network $\mathcal{G}$. Similar to many previously-reported works on road networks [12], we define the road network to be a 3-tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W(\cdot))$, where $\mathcal{V}$ and $\mathcal{E}$ respectively represent the sets of vertices (junctions) and edges (road segments), and each edge $e \in \mathcal{E}$ has a positive integer weight $W(e)$ (non-integer weights can be easily converted [18]). Then, as shown in Fig. 1, a location l in $\mathcal{G}$, which may not always locate at a vertex, can be represented as a tuple $l = (v \in \mathcal{V}, \delta)$, i.e., the nearest vertex to l and the Euclidean distance between l and v. In this way, the distance between two locations $l_p = (v_p, \delta_p)$ and $l_{q_i} = (v_{q_i}, \delta_{q_i})$ in this paper will be defined as $\mathtt{dist}(l_p, l_{q_i}) = d_{SP}(v_p, v_{q_i}) + \delta_p + \delta_{q_i}$, i.e., the sum of the shortest path distance between v_p and v_{q_i}, denoted by $d_{SP}(v_p, v_{q_i})$, and the distances from l_p and l_{q_i} to their closest vertices v_p and v_{q_i}.

B. Road Network Hypercube Embedding

Road Network Hypercube Embedding (RNHE) [19] is a technique that maps a road network into a high-dimensional hypercube space where each vertex is linked to a label, such that the shortest path distance between two vertices can be computed from the Hamming distance between their labels.

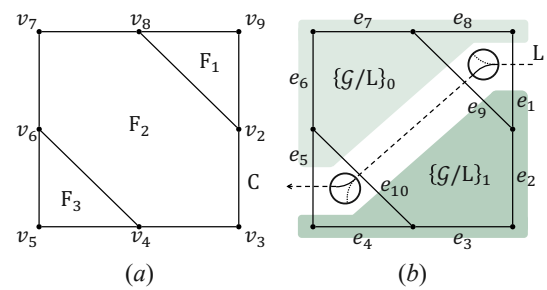


Fig. 2: Given a road network $\mathcal{G}$ as shown in (a), it has one outer face $\mathtt{C}$ and three interior faces $\{\mathtt{F}_1, \mathtt{F}_2, \mathtt{F}_3\}$, where $\mathtt{F}_1$ and $\mathtt{F}_3$ are odd faces, and $\mathtt{F}_2$ is an even face. As shown in (b), $\mathtt{L} = \{e_1, e_9, e_{10}, e_5\}$ is an alternating cut, which partitions $\mathcal{G}$ into two subgraphs, $\{\mathcal{G}/\mathtt{L}\}_0$ and $\{\mathcal{G}/\mathtt{L}\}_1$. Specifically, beginning from e_1, $\mathtt{L}$ cuts through two odd faces. In the first odd face $\mathtt{F}_1$, e_1 has two opposite edges e_9 and e_8, and e_9 (the left one) is chosen. Therefore, in the second odd face $\mathtt{F}_3$, $\mathtt{L}$ chooses e_5 (the right one) instead of e_4.

Before delving into the details of RNHE, we first introduce several related notions. Given a road network $\mathcal{G}$, an *interior face* is a cycle of $\mathcal{G}$ that bounds a connected region. Correspondingly, the *outer face* $\mathtt{C}$ of $\mathcal{G}$ is the unbounded face, as shown in Fig. 2(a). In a face $\mathtt{F}$ of $\mathcal{G}$, two edges $e = (v_i, v_j)$ and $e' = (v'_i, v'_j)$ are opposite to each other if $d_{SP}(v_i, v'_i) = d_{SP}(v_j, v'_j) = dia_{\mathtt{F}}$, where $dia_{\mathtt{F}}$ represents the diameter of $\mathtt{F}$. Furthermore, based on the number of edges forming a *face*, faces can be categorized into two types, namely, *odd faces* and *even faces*. Then, we can deduce that, in an *even face* (resp., *odd face*), each edge e has one unique (resp., two) opposite edge(s). In addition, a cut $\mathtt{L}$ is a sequence of edges $\{e_1, e_2, \cdots, e_{|\mathtt{L}|}\}$ satisfying that i) $e_1 = e_{|\mathtt{L}|}$ or e_1 and $e_{|\mathtt{L}|}$ are edges of the outer face $\mathtt{C}$; ii) every two adjacent edges e_i and e_{i+1} are included in the same interior face; and iii) e_{i+1} is an opposite edge of e_i. By removing the edges in a cut $\mathtt{L}$, $\mathcal{G}$ can be partitioned into to two subgraphs $\{\mathcal{G}/\mathtt{L}\}_0$ and $\{\mathcal{G}/\mathtt{L}\}_1$. While cutting through $\mathcal{G}$, a cut $\mathtt{L}$ may encounter several odd faces, where an edge has two opposite edges. Then, we define an *alternating cut* to be a cut $\mathtt{L}$ that alternates over odd faces. That is, if $\mathtt{L}$ chooses the left (resp. right) opposite edge in an odd face, then it chooses the right (resp. left) opposite edge in the next odd face, as shown in Fig. 2(b).

Based on the above notions, the labels for all vertices in $\mathcal{G}$ can be generated in three steps. First, the label of each vertex $v \in \mathcal{V}$ will be initialized as an empty string. Second, we

enumerate all alternating cuts in $\mathcal{G}$. Third, for each alternating cut $\mathtt{L}$, we append a bit 0 to the label of each vertex in $\{\mathcal{G}/\mathtt{L}\}_0$ and append 1 to those of vertices in $\{\mathcal{G}/\mathtt{L}\}_1$. In addition, all alternating cuts that have not cut through an odd face will be duplicated. Finally, each vertex in $\mathcal{G}$ will be assigned a binary string L_v as its label (see [16] for further details). In Fig. 3, we present all alternative cuts and the labels of all vertices in the road network $\mathcal{G}$ presented in Fig. 2. Based on the labels of v_1 and v_6, we can easily compute their shortest path distance

$$d_{SP}(v_1, v_6) = \frac{1}{2} d_H((11111111)_2, (01000010)_2) = 3,$$

where $d_H(\cdot,\cdot)$ represents the Hamming distance between two binary strings.

Cut	Edges
$\mathtt{L}_1$	$\{e_1, e_8\}$
$\mathtt{L}_2$	$\{e_1, e_9, e_{10}, e_5\}$
$\mathtt{L}_3$	$\{e_2, e_6\}$
$\mathtt{L}_4$	$\{e_2, e_6\}$
$\mathtt{L}_5$	$\{e_3, e_7\}$
$\mathtt{L}_6$	$\{e_3, e_7\}$
$\mathtt{L}_7$	$\{e_4, e_5\}$
$\mathtt{L}_8$	$\{e_4, e_{10}, e_9, e_8\}$

(a)

Vertex	Label
v_1	11111111
v_2	00111111
v_3	00001111
v_4	00000011
v_5	00000000
v_6	01000010
v_7	01110010
v_8	01111110

(b)

Fig. 3: All alternative cuts in $\mathcal{G}$ presented in Fig. 2, and the corresponding labels generated for the vertices in $\mathcal{G}$. Note that $\mathtt{L}_3$ and $\mathtt{L}_5$ never cut through an odd face, so we duplicate them as $\mathtt{L}_4$ and $\mathtt{L}_6$, respectively.

III. Models and Design Goal

In this section, we formalize our system model, security model, and identify our design goal.

A. System Model

In our system model, we consider a privacy-preserving feature-based group kNN query scenario, which contains three types of entities, namely, a service provider SP, a cloud server CS, and a set of query users, as shown in Fig. 4.

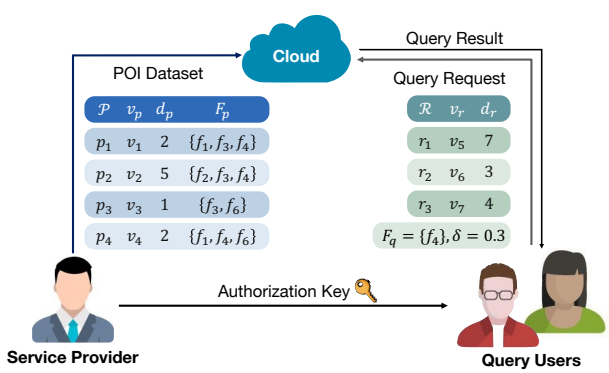


Fig. 4: System model under consideration.

• **Service provider:** In our system model, the service provider SP has a set of POIs $\mathcal{P}$, where each POI record $p \in \mathcal{P}$ is linked to a set of features $F_p \subseteq \mathcal{F}$. Furthermore, each POI p has a location $l_p = (v_p \in \mathcal{V}, \delta_p)$ in the road network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W(\cdot))$. Based on the POI dataset $\mathcal{P}$, SP wants to offer the feature-based group kNN query services to authorized users, but he/she might be weak in terms of computational capacity. Hence, SP is willing to outsource $\mathcal{P}$ and the services to a powerful cloud server. Before being outsourced to the cloud, $\mathcal{P}$ will be encrypted to preserve its data privacy.

• **Cloud Server:** The cloud server CS is equipped with powerful computational resources and abundant storage space, so that it can be appointed to store the dataset and deliver the feature-based group kNN query services. In specific, after being initialized by the service provider, CS can respond to feature-based group kNN queries, and each of these queries consists of three parts, namely i) a feature set $F_q \subseteq \mathcal{F}$, ii) a similarity threshold τ, and iii) a set of reference points $Q = \{l_{q_i} \mid i = 1, 2, \cdots\}$, where each point $l_{q_i} = (v_{q_i}, \delta_{q_i})$ represents a location in $\mathcal{G}$. Upon receiving a query request $\mathtt{q} = \{F_q, \tau, Q\}$ from an authorized query user, the cloud server returns k POIs whose sums of distances to the reference points are minimum, and their similarities to F_q are greater than or equal to the threshold τ.

• **Query Users:** In our system model, a query user has to be authorized by the service provider before he/she can query the feature-based group kNN query services. As shown in Fig. 4, the service provider authorizes a query user by assigning the authorization key. After that, the query user can enjoy the feature-based group kNN query services offered by the cloud server. That is, they can query the cloud server to find the k POIs that satisfy the similarity range constraints (F_q, τ) and are the closest to the reference points Q in terms of the sum of distances.

B. Security Model

In our security model, we consider the service provider SP is *trusted*, since he/she owns the POI dataset and the services, and has no motivation to deviate from the services. For the query users, we consider they are *honest-but-curious*. Specifically, they might be curious about the plaintext of other users' query requests and the corresponding results, but they will submit correct queries. Furthermore, the cloud server is *honest-but-curious* as well. That is, it will faithfully conduct feature-based group kNN queries over the encrypted dataset, but it might be curious about the plaintext of not only the POI dataset but also the query requests and results. Note that, external adversaries may also launch other active attacks, e.g., Denial-of-Service (DoS) attacks. As this paper focuses on privacy preservation in feature-based group kNN queries, those attacks are out of scope of this paper and will be discussed in our future work.

C. Design Goal

Based on the above system model and security model, the design goal of this work is to design an efficient and privacy-preserving feature-based group kNN query scheme. Specifically, the following properties should be satisfied.

• *The proposed scheme should be privacy-preserving.* As detailed in our system model, the POI dataset are private to the service provider, and the query requests and results will also reveal some privacy of the query users. Hence, the primary

requirement of our scheme is to protect the plaintext of the POI datasets, query requests, and the corresponding results against other query users.

• *The proposed scheme should be efficient.* While preserving the data privacy in the system, some cryptographic techniques need to be employed, which will unavoidably introduce extra computational costs. However, to make the proposed scheme more practical, its efficiency should also be considered.

IV. Our Proposed Scheme

In this section, we present our proposed efficient and private feature-based group k nearest neighbor query (EPGQ) scheme. Before delving into the details of our scheme, we first introduce our privacy-preserving distance comparison (PDC) technique as our building block, followed by the main idea of our EPGQ scheme.

A. Privacy-Preserving Distance Comparison

Given locations $l = (v, \delta)$, $l' = (v', \delta')$, and $l_q = (v_q, \delta_q)$ in the road network $\mathcal{G}$, our PDC technique returns whether $\mathtt{dist}(l, l_q) \geq \mathtt{dist}(l', l_q)$ while without revealing l, l' and l_q. We introduce our idea to compress RNHE and describe our PDC construction.

As detailed in [19], the length of an original RNHE label is $|L_v| = |\mathtt{C}|$, i.e., the length of the outer face. However, many alternative cuts may partition the graph into the same pair of subgraphs in real-world road networks, e.g., cuts $\{\mathtt{L}_3, \mathtt{L}_4\}$ and $\{\mathtt{L}_5, \mathtt{L}_6\}$ in Fig. 3 (a). Based on this observation, we convert $\mathbb{L} = \{L_v\}_{v \in \mathcal{V}}$ into $\mathbb{L}^* = (\{L_v^* \mid v \in \mathcal{V}\}, \mathtt{W} = \{w_i\}_{i=1}^{|L_v|})$, where each bit $b_i \in L_v^*$ links to a weight w_i representing the number of alternative cuts related to it. For instance, $L_{v_1} = 11\underline{111}11$ and $L_{v_6} = 01\underline{0000}10$ are respectively mapped into $L_{v_1}^* = 11\underline{1}1\underline{1}1$ and $L_{v_6}^* = 01\underline{0}0\underline{1}0$, and the weight array $\mathtt{W} = \{1, 1, 2, 2, 1, 1\}$. Then, the distance between v_1 and v_6 can be computed by

$$d_{SP}(v_1, v_6) = \frac{1}{2} \sum_{i=1}^{|L_v^*|} w_i \cdot (b_{v_1,i} \oplus b_{v_6,i}) = 3.$$

For the simplicity of notation, hereinafter, we use $\mathbb{L}$ as the collection of compressed labels and $L_v \in \mathbb{L}$ is one of such labels. Concretely, we describe our PDC construction $\Pi_{\mathsf{PDC}} = (\mathsf{DKeyGen}, \mathsf{DEnc}, \mathsf{DTokenGen}, \mathsf{DCmp})$ as follows.

• $\mathsf{DKeyGen}(\mathcal{G})$: Given the road network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W(\cdot))$, the algorithm generates the compressed RNHE labels $\mathbb{L}$. Let $\mathtt{d} = |L_v|$. Then, the algorithm generates two invertible real matrices M_1 and $M_2 \in \mathbb{R}^{(\mathtt{d}+3)\times(\mathtt{d}+3)}$ as the secret key sk_D.

• $\mathsf{DEnc}(\mathbb{L}, sk_D = (M_1, M_2), l = (v, \delta))$: Given the secret key sk_D and a location $l = (v, \delta)$, the algorithm first retrieves the $L_v = \{b_i\}_{i=1}$ from $\mathbb{L}$. Then, the algorithm encrypts l into two vectors, namely, $E_D(l)_1$ and $E_D(l)_2$, where

$$\begin{aligned} E_D(l)_1 =& (r_1, r_1 L_v, r_1\delta, r_2) M_1, \\ E_D(l)_2 =& (r_1', -r_1' L_v, -r_1'\delta, r_2') M_2^T, \end{aligned}$$

and $r_1 > r_2 > 0$ and $r_1' > r_2' > 0$ are random real numbers.

• $\mathsf{DTokenGen}(\mathbb{L}, sk_D, l_q)$: Given $\mathbb{L}$, sk_D and a location $l_q = (v_q, \delta_q)$, the algorithm first retrieves $L_{v_q} = \{b_{q,i}\}_{i=1}^{\mathtt{d}}$ from $\mathbb{L}$. Then, the algorithm encrypts l_q into a matrix

$$TK_D(l_q) = M_1^{-1} \begin{bmatrix} 0 & r_1'' \widetilde{L}_{v_q} & 0 \\ r_1'' \widetilde{L}_{v_q}^T & \mathbf{0} & \mathbf{0} \\ 0 & \mathbf{0} & r_2'' \end{bmatrix} M_2^{-1},$$

where $\widetilde{L}_{v_q} = (\{w_i(1 - 2b_{q,i})\}_{i=1}^{\mathtt{d}}, 1)$ is a vector, and $r_1'' > r_2'' > 0$ are random numbers.

• $\mathsf{DCmp}(E_D(l)_1, E_D(l')_2, TK_D(l_q))$: Given two ciphertexts generated by DEnc and an encrypted token generated by $\mathsf{DTokenGen}$, the algorithm can determine whether $\mathtt{dist}(l, l_q) \geq \mathtt{dist}(l', l_q)$ or not by comparing

$$E_D(l)_1 TK_D(l_q) E_D(l')_2^T > 0. \tag{1}$$

Correctness. The difference between two hamming distances equals to the sum of difference on each bit, i.e.,

$$\begin{aligned} & d_H(L_v, L_{v_q}) - d_H(L_{v'}, L_{v_q}) \\ =& \sum\nolimits_{i=1}^{\mathtt{d}} w_i (b_i \oplus b_{q,i} - b_i' \oplus b_{q,i}) \\ =& \sum\nolimits_{i=1}^{\mathtt{d}} w_i \left((b_i + b_{q,i} - 2b_i b_{q,i}) - (b_i' + b_{q,i} - 2b_i' b_{q,i}) \right) \\ =& \sum\nolimits_{i=1}^{\mathtt{d}} w_i (b_i - b_i')(1 - 2b_{q,i}). \end{aligned}$$

Then, we can simplify the left-hand side of Eq. (1) to be

$$\begin{aligned} & E_D(l)_1 TK_D(l_q) E_D(l')_2^T \\ =& \begin{bmatrix} r_1 \\ r_1 L_v \\ r_1\delta \\ r_2 \end{bmatrix}^T M_1 M_1^{-1} \begin{bmatrix} 0 & r_1'' \widetilde{L}_q & 0 \\ r_1'' \widetilde{L}_q^T & \mathbf{0} & \mathbf{0} \\ 0 & \mathbf{0} & r_2'' \end{bmatrix} M_2^{-1} M_2 \begin{bmatrix} r_1' \\ -r_1' L_v' \\ -r_1'\delta' \\ r_2' \end{bmatrix} \\ =& r_1 r_1' r_1'' (L_v \circ \widetilde{L}_q - L_v' \circ \widetilde{L}_q + \delta - \delta') + r_2 r_2' r_2'' \\ =& r_1 r_1' r_1'' (\sum_{i=1}^{\mathtt{d}} w_i (b_i - b_i')(1 - 2b_{q,i}) + \delta - \delta') + r_2 r_2' r_2'' \\ =& r_1 r_1' r_1'' (d_H(L_v, L_q) - d_H(L_v', L_q) + \delta - \delta') + r_2 r_2' r_2''. \end{aligned}$$

Since $r_1 > r_2 > 0$, $r_1' > r_2' > 0$, and $r_1'' > r_2'' > 0$, we have $r_1 r_1' r_1'' > r_2 r_2' r_2'' > 0$, and thereby,

$$E_D(l)_1 TK_D(l_q) E_D(l')_2^T > 0 \Leftrightarrow \mathtt{dist}(l, l_q) \geq \mathtt{dist}(l', l_q).$$

As a result, the correctness of our PDC holds.

Note that, this technique can be easily adapted to answer queries like $\sum_{l_{q_i} \in Q} d(l, l_{q_i}) \geq \sum_{l_{q_i} \in Q} d(l', l_{q_i})$ by using $TK_D(Q) = \sum_{l_{q_i} \in Q} TK_D(l_{q_i})$. We denote this modification of $\mathsf{DTokenGen}$ as $\mathsf{DTokenGen}(\mathbb{L}, sk_D, Q)$ and analyze its correctness as follows.

Since $TK_D(Q) = \sum_{l_{q_i} \in Q} TK_D(l_{q_i})$, we have

$$\begin{aligned} & E_D(l)_1 TK_D(Q) E_D(l')_2^T \\ =& r_1 r_1' r_1^* \sum_{q_i \in Q} (d_H(L_v, L_{q_i}) - d_H(L_v', L_{q_i}) + \delta - \delta') + r_2 r_2' r_2^* \\ =& r_1 r_1' r_1^* \sum_{q_i \in Q} (\mathtt{dist}(l, l_{q_i}) - \mathtt{dist}(l', l_{q_i})) + r_2 r_2' r_2^*, \end{aligned}$$

where $r_1^* = \sum_{l_{q_i} \in Q} r_1''$ and $r_2^* = \sum_{l_{q_i} \in Q} r_2''$, and $r_1 r_1' r_1^* > r_2 r_2' r_2^* > 0$ still holds. Therefore, we have

$E_D(l)_1 TK_D(Q)E_D(l')_2^T > 0 \Leftrightarrow \sum_{l_{q_i}\in Q} \texttt{dist}(l, l_{q_i}) \geq \sum_{l_{q_i}\in Q} \texttt{dist}(l', l_{q_i})$, i.e., the correctness of PDC holds with $\mathsf{DTokenGen}(\mathbb{L}, sk_D, Q)$.

B. Main Idea of Our EPGQ Scheme

The main idea of our EPGQ scheme consists of two parts, namely, extracting $\mathcal{P}_{F_q,\tau} = \{p \mid \texttt{sim}(F_p, F_q) \geq \tau, p \in \mathcal{P}\}$ and finding the k nearest records in $\mathcal{P}_{F_q,\tau}$, where the second part can be achieved by comparing their distances to the reference locations Q through our PDC technique. For the first part, we employ a filtration-and-verification approach, where the filtration is achieved through the two following lemmas.

Lemma 1 (Prefix-Filter [20]). Given two sets s and s', whose elements are sorted in global order (e.g., frequency-based order), and a threshold τ, if $\texttt{sim}(s, s') \geq \tau$, we have $\psi(s) \cap \psi(s') \neq \varnothing$, where $\psi(s)$ is the collection of the first $(\lfloor(1-\tau)|s|\rfloor + 1)$ elements in s.

Lemma 2 (Length-Filter [21]). Given two sets s and s' and a threshold τ, if $\texttt{sim}(s, s') \geq \tau$, we have $\tau \cdot |s| \leq |s'| \leq \frac{|s|}{\tau}$.

Based on these two lemmas, we can easily deduce that

$$\texttt{sim}(F_p, F_q) \geq \tau \Rightarrow \begin{cases} |F_p| \in [\tau \cdot |F_q|, \frac{|F_q|}{\tau}], \\ F_p \cap \psi(F_q) \neq \varnothing. \end{cases} \tag{2}$$

As the first condition in Eq. (2) can be regarded as a range query over the lengths of records' feature sets $|F_p|$ for $p \in \mathcal{P}$, we design an extended B+-tree where each record p is indexed by $|F_p|$. Then, for each inner node $\texttt{node}$ in $\mathcal{T}_0$, we extend it with a set $F_{\texttt{node}}$ containing all features in its descendants, and we denote the extended B+-tree as $\mathcal{T}_1$. To improve the effectiveness of Prefix-Filter, we sort all features $f_i \in \mathcal{F}$ by their frequencies in $\mathcal{P}$ in ascending order, i.e., f_1 is the least frequently appeared element, while $f_{|\mathcal{F}|}$ is the most frequent element. Thereby, $F_{\texttt{node}}$ can be represented as a binary vector $\mathcal{B}_{\texttt{node}} = \{b_i\}_{i=1}^{|\mathcal{F}|}$, where $b_i = 1$ if $f_i \in F_{\texttt{node}}$, and $b_i = 0$ otherwise; and $\texttt{node}$ can be represented as $\texttt{node} = \{\mathcal{B}_{\texttt{node}}, [x_{\texttt{node},1}, x_{\texttt{node},2}]\}$, where $[x_{\texttt{node},1}, x_{\texttt{node},2}]$ denotes the range of $|F_p|$ covered by $\texttt{node}$. In the following steps, we construct two vectors $\mathbf{x}_{\texttt{node}}$ and $\mathbf{q}_{\mathtt{f}}$ from $\texttt{node}$ and query $\mathtt{q} = \{F_q, \tau, Q\}$, such that $\mathbf{x}_{\texttt{node}} \circ \mathbf{q}_{\mathtt{f}} \leq 0 \Rightarrow \texttt{sim}(F_p, F_q) < \tau$ for all p covered by $\texttt{node}$.

Step 1: For Prefix-Filter, we build a vector $\mathcal{B}_{q,1} = \{\tilde{b}_{q,i}\}_{i=1}^{|\mathcal{F}|}$ from the query $\mathtt{q}$, where $\tilde{b}_{q,i} = 1$ if $f_i \in \psi(F_q)$, and $\tilde{b}_{q,i} = 0$ otherwise. Then, $\mathcal{B}_{\texttt{node}} \circ \mathcal{B}_{q,1} > 0$ if $F_{\texttt{node}} \cap \psi(F_q) \neq \varnothing$, and $\mathcal{B}_{\texttt{node}} \circ \mathcal{B}_{q,1} = 0$ otherwise.

Step 2: For Length-Filter, we have

$$[x_{\texttt{node},1}, x_{\texttt{node},2}] \cap \left[\tau|F_q|, \frac{|F_q|}{\tau}\right] \neq \varnothing \Leftrightarrow \begin{cases} x_{\texttt{node},1} \leq \frac{|F_q|}{\tau}, \\ x_{\texttt{node},2} \geq \tau|F_q|. \end{cases}$$

Hence, we respectively convert $x_{\texttt{node},1}$ and $x_{\texttt{node},2}$ into sets $\textsc{VEnc}(x_{\texttt{node},1})$ and $\textsc{VEnc}(x_{\texttt{node},2})$, where $\textsc{VEnc}(\cdot)$ encodes a value into $\Delta+1$ prefixes, where $\Delta = \lceil \log_2(\max_{F_p\in\mathcal{P}} |F_p|) \rceil$. For instance, $\textsc{VEnc}(2) = \{*, 0*, 00*, 001*, 0010\}$. Correspondingly, we map the ranges $R_1 = [0, \frac{|F_q|}{\tau}]$ and $R_2 = [\tau|F_q|, 2^{\Delta}]$ into two sets $\textsc{REnc}(R_1)$ and $\textsc{REnc}(R_2)$, where $\textsc{REnc}(\cdot)$ encodes a range into a compact set of prefixes that can cover it. For instance, when $\Delta = 4$, we have $\textsc{REnc}([0, 5]) = \{00*, 010*\}$. Please see [22] for more details on $\textsc{VEnc}(\cdot)$ and $\textsc{REnc}(\cdot)$. We can observe that, when a value x is included in a range R, we have $|\textsc{VEnc}(x) \cap \textsc{REnc}(R)| = 1$, and $|\textsc{VEnc}(x) \cap \textsc{REnc}(R)| = 0$ otherwise. Then, we can deduce that

$$\begin{aligned} &[x_{\texttt{node},1}, x_{\texttt{node},2}] \cap \left[\tau|F_q|, \frac{|F_q|}{\tau}\right] \neq \varnothing \\ \Leftrightarrow\ &|\textsc{VEnc}(x_{\texttt{node},\ell}) \cap \textsc{REnc}(R_\ell)| = 1, \text{ for } \ell = 1, 2. \end{aligned}$$

Step 3: For $\ell = 1, 2$, we respectively build two Bloom filters $\Psi_{\mathsf{val}}(x_{\texttt{node},\ell})$ and $\Psi_{\mathsf{range}}(R_\ell)$ from $\textsc{VEnc}(x_{\texttt{node},\ell})$ and $\textsc{REnc}(R_\ell)$, where the lengths of Bloom filters are $\mathtt{d}_2$, and they are initialized by the same set of h hash functions. Next, we respectively constructs two vectors

$$\begin{cases} \Psi_{\texttt{node}} = \{b'_j\}_{j=1}^{2\mathtt{d}_2+1} = (\Psi_{\mathsf{val}}(x_{\texttt{node},1}), \Psi_{\mathsf{val}}(x_{\texttt{node},2}), 1), \\ \Psi_{\mathtt{q}} \quad = \{\tilde{b}'_j\}_{j=1}^{2\mathtt{d}_2+1} = (\Psi_{\mathsf{range}}(R_1), \Psi_{\mathsf{range}}(R_2), -2h), \end{cases}$$

and $\Psi_{\texttt{node}} \circ \Psi_{\mathtt{q}} = \sum_{\ell=1,2} |\Psi_{\mathsf{val}}(x_{\texttt{node},\ell}) \circ \Psi_{\mathsf{range}}(R_\ell)| - 2h < 0 \Rightarrow [x_{\texttt{node},1}, x_{\texttt{node},2}] \cap \left[\tau|F_q|, \frac{|F_q|}{\tau}\right] = \varnothing$.

Step 4: By combining $\mathcal{B}_{\texttt{node}}$ and $\Psi_{\texttt{node}}$, we obtain a vector

$$\begin{aligned} \mathbf{x}_{\texttt{node}} = (&r_1 b_1 b'_1, \cdots, r_1 b_i b'_j, \cdots, r_1 b_{|\mathcal{F}|} b'_{2\mathtt{d}_2+1}, \\ &r_2 b_1, \cdots, r_2 b_{|\mathcal{F}|}, -r_3), \end{aligned}$$

where $r_1 > r_2 > r_3 > 0$ are random numbers. Correspondingly, we combine $\mathcal{B}_{q,1}$ and $\Psi_{\mathtt{q}}$ to obtain

$$\begin{aligned} \mathbf{q}_{\mathtt{f}} = (&r'_1 \tilde{b}_1 \tilde{b}'_1, \cdots, r'_1 \tilde{b}_i \tilde{b}'_j, \cdots, r'_1 \tilde{b}_{|\mathcal{F}|} \tilde{b}'_{2\mathtt{b}_2+1}, \\ &r'_2 \tilde{b}_1, \cdots, r'_2 \tilde{b}_{|\mathcal{F}|}, r'_3), \end{aligned}$$

where $r'_1 > r'_2 > r'_3 > 0$ are random numbers. Then, we have

$$\begin{aligned} \mathbf{x}_{\texttt{node}} \circ \mathbf{q}_{\mathtt{f}} =& \sum_{i=1}^{|\mathcal{F}|} b_i \tilde{b}_i \cdot \sum_{j=1}^{2\mathtt{d}_2+1} (r_1 r'_1 b'_j \tilde{b}'_j + r_2 r'_2) - r_3 r'_3 \\ =& \mathcal{B}_{\texttt{node}} \circ \mathcal{B}_{q,1} \cdot (r_1 r'_1 \cdot \Psi_{\texttt{node}} \circ \Psi_{\mathtt{q}} + r_2 r'_2) - r_3 r'_3, \end{aligned}$$

and $\mathbf{x}_{\texttt{node}} \circ \mathbf{q}_{\mathtt{f}} \leq 0 \Leftrightarrow \mathcal{B}_{\texttt{node}} \circ \mathcal{B}_{q,1} = 0 \ \vee\ \Psi_{\texttt{node}} \circ \Psi_{\mathtt{q}} < 0 \Rightarrow [x_{\texttt{node},1}, x_{\texttt{node},2}] \cap \left[\tau|F_q|, \frac{|F_q|}{\tau}\right] \vee F_{\texttt{node}} \cap \psi(F_q) = \varnothing$. That is, $\mathbf{x}_{\texttt{node}} \circ \mathbf{q}_{\mathtt{f}} \leq 0 \Rightarrow \texttt{sim}(F_p, F_q) < \tau$ for all p covered by $\texttt{node}$.

To verify whether $p \in \mathcal{P}_{F_q,\tau}$ through a vector inner-product, from F_p and $\mathtt{q}$, we can construct two vectors

$$\begin{cases} \mathbf{x}_{F_p} = (r_1 \mathcal{B}_{F_p}, -r_1|F_p|, -r_1), \\ \mathbf{q}_{\mathtt{v}} = (\bar{r}'_1(1+\tau)\mathcal{B}_{q,2}, \bar{r}'_1\tau, \bar{r}'_1\tau|F_q|), \end{cases}$$

where $\mathcal{B}_{q,2} = \{\bar{b}_{q,i}\}_{i=1}^{|\mathcal{F}|}$, $\bar{b}_{q,i} = 1$ if $f_i \in F_q$, and $\bar{b}_{q,i} = 0$ otherwise, and $r_1 > 0$ and $\bar{r}'_1 > 0$ are random numbers. Then, we have

$$\begin{aligned} &\mathbf{x}_{F_p} \circ \mathbf{q}_v \geq 0 \\ \Leftrightarrow\ &r_1 \bar{r}'_1((1+\tau)\mathcal{B}_{F_p} \circ \mathcal{B}_{q,2} - \tau(|F_p| + |F_q|)) \geq 0 \\ \Leftrightarrow\ &(1+\tau)\mathcal{B}_{F_p} \circ \mathcal{B}_{q,2} - \tau(|F_p| + |F_q|) \geq 0 \\ \Leftrightarrow\ &\frac{\mathcal{B}_{F_p} \circ \mathcal{B}_{q,2}}{|F_p| + |F_q| - \mathcal{B}_{F_p} \circ \mathcal{B}_{q,2}} \geq \tau. \end{aligned}$$

Thus, $\mathbf{x}_{F_p} \circ \mathbf{q}_v \geq 0 \Leftrightarrow \texttt{sim}(F_p, F_q) \geq \tau$, i.e., $p \in \mathcal{P}_{F_q,\tau}$. We denote the resulting B+-tree index as $\mathcal{T}$.

C. Description of Our EPGQ Scheme

Based on the main idea, we construct our EPGQ scheme, which consists of four algorithms, i.e., *System Initialization*, *Data Outsourcing*, *Token Generation*, and *Query Conduction*.

1) System Initialization: In this algorithm, the service provider SP generates secret keys for each party in the system in the following steps.

Step 1: SP runs $\mathsf{DKeyGen}(\mathcal{G})$ to obtain $(\mathbb{L}, sk_D = (M_1, M_2))$. Let $\mathtt{d}_1 = |\mathcal{F}|$. SP selects an integer $\mathtt{d}_2$ which will be the length of Bloom filters. Then, SP generates two random invertible real matrices $M_3 \in \mathbb{R}^{(\mathtt{d}_1(2\mathtt{d}_2+1)+2)\times(\mathtt{d}_1(2\mathtt{d}_2+1)+2)}$ and $M_4 \in \mathbb{R}^{(\mathtt{d}_1+2)\times(\mathtt{d}_1+2)}$.

Step 2: SP selects a hash function $H(\cdot)$ and a symmetric key encryption scheme $SE(\cdot)$, e.g., Advanced Encryption Standard (AES). After that, SP selects a secret key K_1 for $SE(\cdot)$ and a random string K_2.

Step 3: SP securely sends K_2 to the cloud server CS. For each user U with $\mathtt{UID}_u$, he/she is authorized by SP with $(\{M_i\}_{i=1}^4, K_1, K_u)$, where $K_u = H(K_2 \| \mathtt{UID}_u)$. In addition, SP publishes $\mathbb{L}$, i.e., the encoding of all vertices in $\mathcal{G}$.

2) Data Outsourcing: In this algorithm, SP builds the index $\mathcal{T}$ and outsources it to CS. Specifically, after constructing $\mathcal{T}$ following the main idea, SP first encrypts each nodes in $\mathcal{T}$. For each inner node $\mathtt{node} = \{F_{\mathtt{node}}, [x_{\mathtt{node},1}, x_{\mathtt{node},2}]\}$, SP encrypts it into $\hat{\mathbf{x}}_{\mathtt{node}} = \mathbf{x}_{\mathtt{node}} M_3$. While for each record $p = (F_p, l_p) \in \mathcal{P}$, it is assigned to a leaf node in $\mathcal{T}$ and is encrypted into three parts, namely, i) $\hat{\mathbf{x}}_{F_p} = \mathbf{x}_{F_p} M_4$ encrypted from F_p, ii) $(E_D(l_p)_1, E_D(l_p)_2) = \mathsf{DEnc}(\mathbb{L}, sk_D, l_q)$, and iii) $SE(K_1, p)$. We denote each encrypted record $E(p)$ as $E(p) = \{\hat{\mathbf{x}}_{F_p}, E_D(l_p)_1, E_D(l_p)_2, SE(K_1, p)\}$. After encrypting each nodes, SP randomly permutes nodes in the encrypted tree $E(\mathcal{T})$. Finally, SP uploads $E(\mathcal{T})$ to CS.

3) Token Generation: Given $(\{M_i\}_{i=1}^4, K_1, K_u)$, an authorized user U with ID $\mathtt{UID}_u$ can encrypt his/her query requests into tokens. Specifically, given a query request $\mathtt{q} = (F_q, \tau, Q)$, U first builds the two vectors $\mathbf{q}_{\mathtt{f}}$ and $\mathbf{q}_{\mathtt{v}}$ following the main idea. Then, U runs $\mathsf{DTokenGen}(\mathbb{L}, sk_D, Q)$ to obtain $TK_D(Q)$. After that, U computes $\hat{\mathbf{q}}_{\mathtt{f}} = (M_3^{-1})\mathbf{q}_{\mathtt{f}}^T, \hat{\mathbf{q}}_{\mathtt{v}} = (M_4^{-1})\mathbf{q}_{\mathtt{v}}^T$, and $K_{u,ts} = H(K_u \| ts)$, where ts is the current timestamp. Finally, U queries CS with

$$E(\mathtt{token}) = \{SE(K_{u,ts}, (\hat{\mathbf{q}}_{\mathtt{f}}, \hat{\mathbf{q}}_{\mathtt{v}}, TK_D(Q))), ts, \mathtt{UID}_u\}.$$

4) Query Conduction: Upon receiving an encrypted query token $E(\mathtt{token})$, the cloud server CS can run the following steps to conduct the query over $E(\mathcal{T})$.

Step 1: CS extracts ts and $\mathtt{UID}_u$ from $E(\mathtt{token})$ and rejects the query if ts is not fresh. Otherwise, it computes $K_{u,ts} = H(H(K_2 \| \mathtt{UID}_u) \| ts)$ to recover $\hat{\mathbf{q}}_{\mathtt{f}}, \hat{\mathbf{q}}_{\mathtt{v}}, TK_D(Q)$ from $E(\mathtt{token})$.

Step 2: CS traverses the encrypted index $\mathcal{T}$. For each $\mathtt{node} \in E(\mathcal{T})$, CS verifies whether $\hat{\mathbf{x}}_{\mathtt{node}} \circ \hat{\mathbf{q}}_{\mathtt{f}} > 0$, and prunes it if $\hat{\mathbf{x}}_{\mathtt{node}} \circ \hat{\mathbf{q}}_{\mathtt{f}} \leq 0$. After running this step, CS collects all records from nodes that have not been pruned and gets a set of candidate records $\mathcal{C}$.

Step 3: For each $E(p) \in \mathcal{C}$, CS checks whether $\hat{\mathbf{x}}_{F_p} \circ \hat{\mathbf{q}}_{\mathtt{v}} \geq 0$, and it removes $E(p)$ from $\mathcal{C}$ if $\hat{\mathbf{x}}_{F_p} \circ \hat{\mathbf{q}}_{\mathtt{v}} < 0$. We denote the resulting set as $\mathcal{P}_{F_q,\tau} = \{E(p) \mid \forall p \in \mathcal{C}, \mathtt{sim}(E_p, E_q) \geq \tau\}$.

Step 4: CS compares all encrypted records in $\mathcal{P}_{F_q,\tau}$ to obtain the k nearest records, denoted as $\mathcal{P}_q$. Specifically, for each two POI records $E(p_i)$ and $E(p_j) \in E(\mathcal{T})$, CS compares their distance to Q through $\mathsf{DCmp}(E_D(p_i)_1, E_D(q_j)_2, TK_D(Q))$.

Step 5: CS returns $SE(K_{u,ts}, \{SE(K_1, p) \mid p \in \mathcal{P}_q\})$ as the query result. Upon receiving the ciphertext from CS, U can decrypt it with $K_{u,ts}$ and K_1 to recover the plaintexts of the query result.

Correctness. Based on the correctness of our PDC technique, the correctness of extracting kNN from $\mathcal{P}_{F_q,\tau}$, i.e., **Step 4**, can be easily verified. As for **Step 2** and **Step 3**, its correctness can be verified through our main idea presented in Section IV-B, since $\hat{\mathbf{x}}_{\mathtt{node}} \circ \hat{\mathbf{q}}_{\mathtt{f}} = \mathbf{x}_{\mathtt{node}} M_3 M_3^{-1} \mathbf{q}_{\mathtt{f}}^T = \mathbf{x}_{\mathtt{node}} \circ \mathbf{q}_{\mathtt{f}}$ and $\hat{\mathbf{x}}_{F_p} \circ \hat{\mathbf{q}}_{\mathtt{v}} = \mathbf{x}_{F_p} M_4 M_4^{-1} \mathbf{q}_{\mathtt{v}}^T = \mathbf{x}_{F_p} \circ \mathbf{q}_{\mathtt{v}}$. Therefore, the correctness of our EPGQ construction holds.

V. Security Analysis

In this section, we analyze the security of the proposed EPGQ scheme. Before that, we first analyze the security of our building block PDC technique.

A. Security Analysis of Our PDC Technique

Here we first show that our PDC technique is selectively secure in the real/ideal simulation setting. Before formally analyzing the security, we first define the leakage in PDC. Given two locations $\{l, l'\}$ and a query location l_q, the leakage is $\mathcal{L} = \{\mathbb{L}, \mathtt{sign}(\mathtt{dist}(l, l_q) - \mathtt{dist}(l', l_q))\}$, i.e., the compressed RNHE labels for all vertices in the road network $\mathcal{G}$ and whether $\mathtt{dist}(l, l_q)$ is larger/smaller than or equal to $\mathtt{dist}(l', l_q)$. Based on the leakage, we respectively define the real and ideal experiments.

Real Experiment: The real experiment involves a challenger and a PPT (probabilistic polynomial-time) adversary $\mathcal{A}$, and they interact as follows.

- *Setup:* In the setup algorithm, $\mathcal{A}$ chooses two random locations l and l', and sends them to the challenger. Then, the challenger runs $\mathsf{DKeyGen}(\mathcal{G})$ to generate a secret key $sk_D = (M_1, M_2)$ and publishes $\mathbb{L}$.
- *Query phase 1:* $\mathcal{A}$ adaptively chooses α query locations $\{l_{q_i} \mid 1 \leq i \leq \alpha\}$ and sends them to the challenger, where α is a polynomial number. Then, the challenger generates a token for each location l_{q_i} as $TK_D(l_{q_i}) = \mathsf{DTokenGen}(\mathbb{L}, sk_D, l_{q_i})$ and returns $\{TK_D(l_{q_i})\}_{i=1}^{\alpha}$ to $\mathcal{A}$.
- *Challenge phase:* The challenger encrypts the two locations $\{l, l'\}$ as $(E_D(l)_1, E_D(l)_2) = \mathsf{DEnc}(\mathbb{L}, sk_D, l)$ and $(E_D(l')_1, E_D(l')_2) = \mathsf{DEnc}(\mathbb{L}, sk_D, l')$, and it sends the ciphertexts to $\mathcal{A}$.
- *Query phase 2:* Similar to *Query phase 1*, $\mathcal{A}$ submits another $(\beta - \alpha)$ query locations $\{l_{q_i} \mid \alpha < i \leq \beta\}$ and gets the query tokens $\{TK_D(l_{q_i}) \mid \alpha < i \leq \beta\}$ from the challenger, where β is a polynomial number.

Ideal Experiment: The ideal experiment involves a simulator with leakage $\mathcal{L}$ and a PPT adversary $\mathcal{A}$, and they interact with each other as follows.

- *Setup:* In the setup algorithm, $\mathcal{A}$ first computes and publishes $\mathbb{L}$, i.e., the RNHE labels for all vertices in $\mathcal{G}$. Then, it chooses two random locations l and l', and it sends l and l'

to the simulator. After that, the simulator generates two pairs random vectors $(E_D(l)_1, E_D(l)_2)$ and $(E_D(l')_1, E_D(l')_2)$ as the ciphertexts of the two locations.

• *Query phase 1:* $\mathcal{A}$ adaptively chooses α query locations $\{l_{q_i} \mid 1 \leq i \leq \alpha\}$ and sends them to the simulator, where α is a polynomial number. For each l_{q_i}, the simulator has $\mathcal{L} = \texttt{sign}(\texttt{dist}(l, l_{q_i}) - \texttt{dist}(l', l_{q_i}))$, and it generates a random matrix $TK_D(l_{q_i})$ with rank $\rho(TK_D(l_{q_i})) = 3$ satisfying

$$\begin{cases} \mathcal{L}_1 > 0 \text{ and } \mathcal{L}_2 < 0, & \text{if } \mathcal{L} = 1; \\ \mathcal{L}_1 > 0 \text{ and } \mathcal{L}_2 > 0, & \text{if } \mathcal{L} = 0; \\ \mathcal{L}_1 < 0 \text{ and } \mathcal{L}_2 > 0, & \text{if } \mathcal{L} = -1, \end{cases}$$

where $\mathcal{L}_1$ and $\mathcal{L}_2$ are respectively $E_D(l)_1 TK_D(l_{q_i}) E_D(l')_2$ and $E_D(l')_1 TK_D(l_{q_i}) E_D(l)_2$. Then, the simulator returns these query tokens to $\mathcal{A}$.

• *Challenge phase:* The simulator sends $(E_D(l)_1, E_D(l)_2)$ and $(E_D(l')_1, E_D(l')_2)$ to $\mathcal{A}$.

• *Query phase 2:* Similar to *Query phase 1*, $\mathcal{A}$ adaptively chooses $(\beta - \alpha)$ query locations $\{l_{q_i} \mid \alpha < i \leq \beta\}$ and submits them to the simulator, where β is a polynomial number. The simulator returns the tokens $\{TK_D(l_{q_i}) \mid \alpha < i \leq \beta\}$.

Definition 2 (Security of the PDC technique). Our PDC technique is selectively secure with the leakage $\mathcal{L}$ *iff* for all PPT adversaries issuing polynomial numbers of query tokens, there exists a PPT simulator such that the probability for a PPT distinguisher $\mathcal{D}$ to distinguish the real and ideal experiments is negligible, i.e., $|\Pr[\mathcal{D}(\mathsf{View}_{\mathcal{A},\text{Real}} = 1)] - \Pr[\mathcal{D}(\mathsf{View}_{\mathcal{A},\text{Ideal},\mathcal{L}} = 1)]|$ is negligible.

In our real/ideal experiments, the view of $\mathcal{D}$ is $\{(E_D(l)_1, E_D(l)_2), (E_D(l')_1, E_D(l')_2), \{TK_D(l_{q_i})\}_{i=1}^{\beta}\}$. Then, we show that $\mathcal{D}$ cannot distinguish the two experiments as follows. First, in the real experiment, $(E_D(l)_1, E_D(l)_2)$, $(E_D(l')_1, E_D(l')_2)$ are ciphertexts generated by DEnc, and $TK_D(l_{q_i})$ is a query token generated by DTokenGen. During generating each ciphertext $(E_D(l)_1, E_D(l)_2)$ and each query token $TK_D(l_{q_i})$, random numbers $\{r_1, r_2, r_1', r_2'\}$ and $\{r_1'', r_2''\}$ are involved, respectively. These random numbers make the ciphertexts and query tokens look like random vectors/matrices. Second, in the ideal experiment, the ciphertexts and query tokens are randomly chosen based on the leakage, so they are also random vectors/matrices. Thus, the view of $\mathcal{D}$ is random ciphertexts and query tokens in both experiments, and the probability that $\mathcal{D}$ can distinguish them is negligible. Therefore, the PDC technique is selectively secure.

B. Security Analysis of Our EPGQ Scheme

Next, we analyze the security of our EPGQ scheme. Specifically, we show that the EPGQ scheme can preserve the privacy of the dataset, query requests, and query results as follows.

• *The dataset in our EPGQ scheme is privacy-preserving.* The cloud server is considered to be *honest-but-curious* as detailed in our security model. That is, it may attempt to recover the plaintext of the outsourced dataset, which is represented by an encrypted B+-tree $E(\mathcal{T})$. Each node in $E(\mathcal{T})$ is encrypted as a vector $\hat{\mathbf{x}}_{\texttt{node}}$, and each POI record p is encrypted as $E(p) = ((E_D(l_p)_1, E_D(l_p)_2), \hat{\mathbf{x}}_{F_p}, SE(K_1, p))$, where $(E_D(l_p)_1, E_D(l_p)_2)$ and $SE(K_1, p)$ are generated by DEnc and SE. Thus, CS cannot obtain useful information from $(E_D(l_p)_1, E_D(l_p)_2)$ and $SE(K_1, p)$, based on the security of our PDC scheme and SE. As for $\hat{\mathbf{x}}_{\texttt{node}}$ and $\hat{\mathbf{x}}_{F_p}$, without knowing M_3, M_4, and random numbers $\{r_1, r_2, r_3\}$ that are independently chosen for each node/record, CS cannot recover useful information. While processing group kNN queries, CS can only know whether a node satisfies Eq. (2) and whether a record p satisfies $\texttt{sim}(F_p, F_q) \geq \tau$, but it cannot tell whether a node is pruned by *Prefix-Filter* or *Length-Filter*. Thus, the privacy of the dataset is preserved against the cloud server.

• *The query requests in our EPGQ scheme are privacy-preserving.* As the query requests contain information related to the query users' preferences and location privacy, they need to be protected against the cloud server and other query users. In our EPGQ scheme, a query request (F_q, τ, Q) is encrypted as $E(\texttt{token}) = (SE(K_{u,ts}, (\hat{\mathbf{q}}_{\mathbf{f}}, \hat{\mathbf{q}}_{\mathbf{v}}, TK_D(Q))), ts, \texttt{UID})$. A query user cannot derive $K_{u,ts}$ of other query users without knowing K_2. Therefore, the query requests are protected against other query users, and only CS can extract $(\hat{\mathbf{q}}_{\mathbf{f}}, \hat{\mathbf{q}}_{\mathbf{v}}, TK_D(Q))$ from $E(\texttt{token})$. Based on the security of our PDC technique, CS cannot obtain any useful information related to the set of query locations Q from $TK_D(Q)$. As for $\hat{\mathbf{q}}_{\mathbf{f}}$ and $\hat{\mathbf{q}}_{\mathbf{v}}$, CS cannot recover the plaintext of F_q and τ without knowing the matrices $\{M_3, M_4\}$ and the random numbers used during encryption. Furthermore, while conducting the queries, CS can only obtain $\{\hat{\mathbf{x}}_{\texttt{node}} \circ \hat{\mathbf{q}}_{\mathbf{f}}\}$ and $\{\hat{\mathbf{x}}_{F_q} \circ \hat{\mathbf{q}}_{\mathbf{v}}\}$, which can only reveal whether a node satisfies Eq. (2) and whether a record p satisfies $\texttt{sim}(F_p, F_q) \geq \tau$. However, CS cannot recover the plaintext of F_q. Therefore, the query requests is privacy-preserved against CS and query users.

• *The query results in our EPGQ scheme are privacy-preserving.* The query result is a set of POI records that satisfy the query request, and query users may be curious about other users' query results as they contain the information related to the users' preferences and query locations. However, in our EPGQ scheme, the query result is encrypted by $K_{u,ts}$ as $SE(K_{u,ts}, \{SE(K_1, p) \mid p \in \mathcal{P}_q\})$. Therefore, based on the security of SE, a query user cannot recover others' query results without knowing the corresponding $K_{u,ts}$. That is, the privacy of the query results in our EPGQ scheme is preserved against other query users.

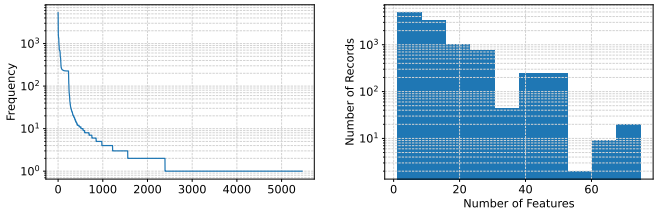
VI. Performance Analysis

In this section, we study the performance of our proposed EPGQ scheme. Specifically, we evaluate the computational costs for the service provider, query users, and the cloud server in terms of time consumption during *Data Outsourcing*, *Token Generation*, and *Query Conduction*, respectively.

A. Evaluation Setup

We implement our EPGQ scheme in Rust, and we build our dataset upon the New York road network data [23] and a set of photos taken in New York extracted through Flickr public API [24]. Specifically, each photo links to a location in New York and a set of keywords (features), and it is regarded as a POI in our dataset. The dataset contains 10,490 records and

5,461 features, and as shown in Fig. 5, 1,000 features in the dataset link to 5 records, while a majority of records link to less than 30 features. To demonstrate how the dataset's size affects the performance of our scheme, we extract vertices from the original network to form the testing datasets such that their lengths of the compressed RNHE encoding $\mathtt{d}$ range from 100 to 400. Furthermore, to show the effect of $\mathtt{d}_1$ on the performance of our EPGQ scheme, we extract the feature spaces of different sizes $\mathtt{d}_1$ from the original dataset, and records that do not contain any selected features will be removed from the dataset.



(a) # of records containing a feature. (b) # of features related to a record.

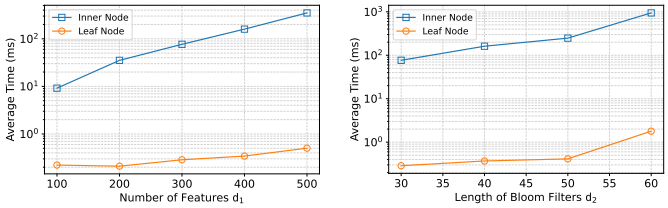
Fig. 5: Distribution of the features' frequencies in the dataset and the distribution of the number of features for each record.

In the following, we conduct the evaluations with different parameters on a platform equipped with an Intel(R) Core(TM) i5-1038NG7 CPU @2.00GHz, 16GB RAM and macOS 11.4 operating system, and we set the fanout of B+-tree to be 32. In addition, we set the number of hash functions used in Bloom filters as $h = 2$ and their bit length to be 100. Note that, as the Bloom filters will only be used for filtration, their false positive rate will not affect the accuracy of the proposed scheme. For each case, we run the evaluations for 10 times, and the average performance results are reported.

B. Evaluation Results

Here we analyze the computational cost of *Data Outsourcing*, *Token Generation*, and *Query Conduction*, respectively.

• *Data Outsourcing:* In the data outsourcing phase, the service provider SP builds the index $\mathcal{T}$ and encrypts the resulting index into $E(\mathcal{T})$. Specifically, to encrypt an inner node in $\mathcal{T}$, SP computes one outer product of order $\mathtt{d}_1 \times (2\mathtt{d}_2 + 1)$ and a vector-matrix multiplication of order $(\mathtt{d}_1 \times (2\mathtt{d}_2+1)+2) \times (\mathtt{d}_1 \times (2\mathtt{d}_2+1)+2)$. As shown in Fig. 6, the computational cost for encrypting an inner node increases with the number of features $\mathtt{d}_1$ and the size of Bloom filters $\mathtt{d}_2$. On the other hand, to encrypt a leaf node in $\mathcal{T}$, SP computes three vector-matrix multiplications, where the square matrix in the first multiplication is of order $(\mathtt{d}_1 + 2)$, and those for the second and third multiplications are of order $(\mathtt{d}+3)$. According to Fig. 6, the computational cost for SP to encrypt a leaf node increases with the length of the compressed RNHE labels $\mathtt{d}$, the number of features $\mathtt{d}_1$ and the length of Bloom filters $\mathtt{d}_2$. Specifically, as reported by the figure, when $\mathtt{d}_1 = 300$ and $\mathtt{d}_2 = 60$, the average time consumption for encrypting an inner node is less than 1s, and that for encrypting a leaf node is less than 2ms. Hereinafter, we fix the length of Bloom filters to be $\mathtt{d}_2 = 30$.



(a) Avg. time for encrypting a node when $\mathtt{d}_2 = 30$ with different $\mathtt{d}_1$. (b) Avg. time for encrypting a node when $\mathtt{d}_1 = 300$ with different $\mathtt{d}_2$.

$\mathtt{d}$	100	200	300	400
Time (μs)	430.21	432.49	544.51	604.48

(c) Avg. time for encrypting a leaf node vs $\mathtt{d}$.

Fig. 6: Average time consumption for encrypting a inner node or leaf node in the index $\mathcal{T}$ with different $\mathtt{d}$, $\mathtt{d}_1$ and $\mathtt{d}_2$.

• *Token Generation:* In the token generation phase, a query user encrypts his/her query token into two vectors and a matrix by conducting two vector-matrix multiplications with square matrices of orders $(\mathtt{d}_1 \times (2\mathtt{d}_2 + 1) + 2)$ and $(\mathtt{d}_1 + 2)$, and two matrix multiplications between square matrices of order $(\mathtt{d}+3)$. As shown in Fig. 7, the computational cost for the user to encrypt a query token increases with the number of features $\mathtt{d}_1$. As for the length of the compressed RNHE labels $\mathtt{d}$, its effect on the computational cost is less significant according to the experimental data. Furthermore, the number of locations in the query token, i.e., $|Q|$, has negligible affect on the user's computational cost. This is mainly because that, as detailed in Section IV, the increase of $|Q|$ will only result in more vector additions but will not increase the matrix size or the number of matrix multiplications.

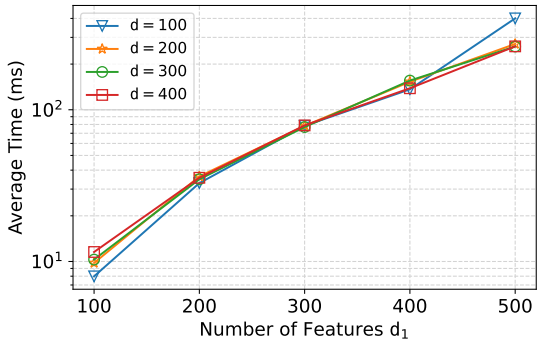
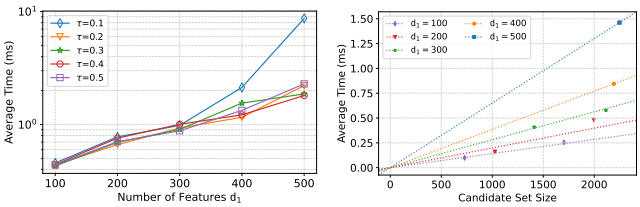


Fig. 7: Average time for encrypting a query token with different numbers of features $\mathtt{d}_1 = |\mathcal{F}|$ and different lengths of compressed RNHE labels $\mathtt{d}$.

• *Query Conduction:* In the query conduction phase, the cloud server CS traverses the encrypted index $E(\mathcal{T})$. As detailed in Section IV-C, CS conducts a query request by i) running Breadth-First Search on $E(\mathcal{T})$ to obtain a set of candidates, ii) refining the candidate set such that all elements in the set satisfy the Jaccard similarity range criteria, and iii) comparing the distances between the records in the resulting set to extract the k nearest neighbors. Then, we respectively analyze the computational cost of these three steps as follows.

Step 1: In the filtration step, CS traverses $E(\mathcal{T})$ and verifies whether each node can be pruned by conducting an inner product of vectors of order $(\mathtt{d}_1(\mathtt{d}_2 + 1) + 2)$. That is, the

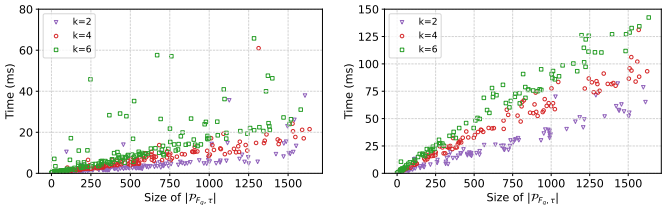
computational cost for CS to process one inner node in $E(\mathcal{T})$ increases with the number of features $\mathtt{d}_1 = |\mathcal{F}|$, as demonstrated in Fig. 8 (a). Since the number of inner nodes visited in this step is affected by the distribution of the dataset and the query request, the overall computational cost for this step varies for different queries. As shown in the figure, when $\tau = 0.1$, the number of visited inner nodes is much higher than that of $\tau = 0.2, \cdots, 0.5$ and leads to a higher time consumption. Specifically, when $\tau = 0.1$ and $\mathtt{d}_1 = 500$, the average time consumption for processing a node in $E(\mathcal{T})$ during *Step 1* is less than 10ms.



(a) Avg. time for *Step 1* (b) Avg. time for *Step 2*

Fig. 8: Avg. time consumption for *Step 1* with different numbers of features $\mathtt{d}_1$ and that for *Step 2* with different sizes of candidate set.

Step 2: Based on the candidate set generated in the filtration step, CS verified whether a record p satisfies that $\mathtt{sim}(F_p, F_q) \geq \tau$, where F_q is the set of query features. Specifically, for each F_p, CS conducts an inner product between vectors of order $(\mathtt{d}_1 + 2)$, i.e., the computational cost for each p increases with $\mathtt{d}_1$, i.e., the slope of the line increases with $\mathtt{d}_1$ as shown in Fig. 8 (b).



(a) $\mathtt{d} = 100$ (b) $\mathtt{d} = 400$

Fig. 9: Average time consumption for *Step 3* with different $\mathtt{d}$ and k.

Step 3: Based on the resulting set from *Step 2*, CS extracts the kNN records by comparing $(|\mathcal{P}_{F_q,\tau}| \cdot k - k(k-1)/2)$ times of comparison, where each comparison is achieved by one vector-matrix-vector multiplication of order $(\mathtt{d} + 3)$. As shown in Fig. 9, the time consumption of *Step 3* increases with the size of $|\mathcal{F}_{F_q,\tau}|$ and k, and larger $\mathtt{d}$ leads to higher time consumption. Specifically, when $\mathtt{d} = 400$, extracting $k = 6$ records from 1,500 records takes roughly 170ms.

VII. Related Work

In this section, we review some works that are closely related to our proposed scheme in terms of privacy-preserving k nearest neighbor (kNN) over road networks and privacy-preserving group k nearest neighbor (GkNN) queries. Furthermore, we present a comparison between our proposed scheme and these related works in Table I.

TABLE I: Comparison with Existing Works

	GkNN	Enc. Dataset	Road Network Dist.	Feature-based
[9]	✗	✓	✗	✗
[10]	✗	✓	✓	✗
[11]	✗	✓	✓	✗
[12]	✗	✓	✓	✗
[13]	✗	✓	✗	✗
[14]	✓	✗	✗	✗
[15]	✓	✗	✓	✗
[16]	✓	✗	✓	✗
[17]	✓	✓	✗	✗
Ours	✓	✓	✓	✓

Several schemes [9]–[13] focused on privacy-preserving NN or kNN queries have been proposed. Yang et al. [9] built their scheme based on the Paillier publickey encryption (PKE) and 2-hop labeling index [25], and Voronoi graph are employed to achieve road network distance computation and kNN extraction. However, as the Voronoi graph is built upon pre-computed distances [26], it is hard to be adapted for GkNN queries where the distances depend on the combination of query locations. Some works [10]–[13] were proposed for privacy-preserving ride hailing over road networks, which can also be regarded as nearest neighbor queries. However, both Luo et al. [10] and Yu et al. [11]–[13] built their schemes with homomorphic encryption schemes under a two-server setting. Therefore, extending these works to support feature-based GkNN queries will invoke heavy overheads in terms of both computation and communication.

Some schemes [14]–[17] were proposed to support privacy-preserving GkNN queries. Hashem et al. [14] proposed a scheme by employing an R-tree based index to organize the dataset, and users' locations are obfuscated into regions to preserve the location privacy. Wu et al. [15], [16] built their schemes based on Paillier PKE [27], and dummies are employed to hide the real requests. However, these two schemes are designed for a different scenario from ours. Yu et al. [17] built their scheme upon somewhat homomorphic encryption scheme [28] and the distances between a location and the query locations are captured by Euclidean distance, which can only achieve approximate road network GkNN queries [29]. In addition, as demonstrated in Table I, none of the above mentioned works can support GkNN queries over encrypted datasets under road network distance. Therefore, these schemes cannot apply to our scenario.

As demonstrated in Table I, different from the above mentioned schemes, our proposed scheme can simultaneously support feature-based criteria and accurate GkNN queries over road network while preserving the privacy of the dataset, query requests, and query results.

VIII. Conclusion

This paper proposes an efficient and private feature-based group k nearest neighbor query scheme over road networks, called EPGQ. Specifically, we first built a privacy-preserving distance comparison (PDC) technique that can compare the sums of distances from two records to a set of locations. We constructed our EPGQ scheme based on PDC, which handles GkNN queries through filtration and verification. Security

analysis shows that our PDC technique is selectively secure and the EPGQ scheme can well preserve the privacy of the dataset, query requests and results. Performance analysis further demonstrates that our EPGQ scheme is efficient for both query users and the cloud server. In our future work, we will evaluate the performance of our EPGQ scheme under real-world settings.

Acknowledgments

This research was supported in part by NSERC Discovery Grants (04009, RGPIN-2022-03244).

References

[1] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*, pp. 467–478.

[2] M. Sharifzadeh and C. Shahabi, "The spatial skyline queries," in *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*. ACM, 2006, pp. 751–762.

[3] S. Zhang, S. Ray, R. Lu, Y. Zheng, Y. Guan, and J. Shao, "Achieving efficient and privacy-preserving dynamic skyline query in online medical diagnosis," *IEEE Internet Things J.*, pp. 1–1, 2021.

[4] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis, "Group nearest neighbor queries," in *Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, 30 March - 2 April 2004, Boston, MA, USA*. IEEE Computer Society, 2004, pp. 301–312.

[5] F. Guo, Y. Yuan, G. Wang, L. Chen, X. Lian, and Z. Wang, "Cohesive group nearest neighbor queries over road-social networks," in *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE, 2019, pp. 434–445.

[6] S. Zhang, S. Ray, R. Lu, Y. Zheng, and J. Shao, "Preserving location privacy for outsourced most-frequent item query in mobile crowdsensing," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 9139–9150, 2021.

[7] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Efficient and privacy-preserving similarity range query over encrypted time series data," *IEEE Trans. Dependable Secur. Comput.*, pp. 1–1, 2021.

[8] ——, "Towards practical and privacy-preserving multi-dimensional range query over cloud," *IEEE Trans. Dependable Secur. Comput.*, pp. 1–1, 2021.

[9] S. Yang, S. Tang, and X. Zhang, "Privacy-preserving k nearest neighbor query with authentication on road networks," *J. Parallel Distributed Comput.*, vol. 134, pp. 25–36, 2019.

[10] Y. Luo, X. Jia, S. Fu, and M. Xu, "pride: Privacy-preserving ride matching over road networks for online ride-hailing service," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 7, pp. 1791–1802, 2019.

[11] H. Yu, J. Shu, X. Jia, H. Zhang, and X. Yu, "lpride: Lightweight and privacy-preserving ride matching over road networks in online ride hailing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 10 418–10 428, 2019.

[12] H. Yu, X. Jia, H. Zhang, and J. Shu, "Efficient and privacy-preserving ride matching using exact road distance in online ride hailing services," *IEEE Trans. Serv. Comput.*, pp. 1–1, 2020.

[13] H. Yu, X. Jia, H. Zhang, X. Yu, and J. Shu, "Psride: Privacy-preserving shared ride matching for online ride hailing systems," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 3, pp. 1425–1440, 2021.

[14] T. Hashem, L. Kulik, and R. Zhang, "Privacy preserving group nearest neighbor queries," in *EDBT 2010, 13th International Conference on Extending Database Technology, Lausanne, Switzerland, March 22-26, 2010, Proceedings*, ser. ACM International Conference Proceeding Series, vol. 426. ACM, 2010, pp. 489–500.

[15] Y. Wu, K. Wang, Z. Zhang, W. Lin, H. Chen, and C. Li, "Privacy preserving group nearest neighbor search," in *Proceedings of the 21st International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018*. OpenProceedings.org, 2018, pp. 277–288.

[16] Y. Wu, K. Wang, R. Guo, Z. Zhang, D. Zhao, H. Chen, and C. Li, "Enhanced privacy preserving group nearest neighbor search," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 2, pp. 459–473, 2021.

[17] H. Yu, H. Zhang, X. Yu, X. Du, and M. Guizani, "Pgride: Privacy-preserving group ridesharing matching in online ride hailing services," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5722–5735, 2021.

[18] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*. The Internet Society, 2015.

[19] S. Gupta, S. Kopparty, and C. V. Ravishankar, "Roads, codes and spatiotemporal queries," in *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 14-16, 2004, Paris, France*. ACM, 2004, pp. 115–124.

[20] S. Chaudhuri, V. Ganti, and R. Kaushik, "A primitive operator for similarity joins in data cleaning," in *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*. IEEE Computer Society, 2006, p. 5.

[21] R. J. Bayardo, Y. Ma, and R. Srikant, "Scaling up all pairs similarity search," in *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. ACM, 2007, pp. 131–140.

[22] A. X. Liu and F. Chen, "Collaborative enforcement of firewall policies in virtual private networks," in *Proceedings of the Twenty-Seventh Annual ACM Symposium on Principles of Distributed Computing, PODC 2008, Toronto, Canada, August 18-21, 2008*. ACM, 2008, pp. 95–104.

[23] C. Demetrescu, "9th dimacs implementation challenge - shortest paths," http://www.diag.uniroma1.it//challenge9/download.shtml, Jun. 2010, (Accessed on Jul. 2021).

[24] "Flickr services," https://www.flickr.com/services/api/, (Accessed on Jul. 2021).

[25] T. Akiba, Y. Iwata, and Y. Yoshida, "Fast exact shortest-path distance queries on large networks by pruned landmark labeling," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*. ACM, 2013, pp. 349–360.

[26] M. R. Kolahdouzan and C. Shahabi, "Voronoi-based K nearest neighbor search for spatial network databases," in *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, Toronto, Canada, August 31 - September 3 2004*, M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, Eds. Morgan Kaufmann, 2004, pp. 840–851.

[27] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques*, 1999, pp. 223–238.

[28] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptol. ePrint Arch.*, vol. 2012, p. 144, 2012.

[29] H. Hua, H. Xie, and E. Tanin, "Is euclidean distance really that bad with road networks?" in *Proceedings of the 11th ACM SIGSPATIAL International Workshop on Computational Transportation Science, IWCTS@SIGSPATIAL 2018, Seattle, WA, USA, November 6, 2018*. ACM, 2018, pp. 11–20.

Yunguo Guan is a PhD student of the Faculty of Computer Science, University of New Brunswick, Canada. His research interests include applied cryptography and game theory.

Rongxing Lu (Fellow, IEEE) is Mastercard IoT Research Chair, a University Research Scholar, an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Before that, he worked as an assistant professor at the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore from April 2013 to August 2016. Rongxing Lu worked as a Postdoctoral Fellow at the University of Waterloo from May 2012 to April 2013. He was awarded the most prestigious "Governor General's Gold Medal", when he received his PhD degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and won the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013. Dr. Lu is an IEEE Fellow. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He has published extensively in his areas of expertise, and was the recipient of 9 best (student) paper awards from some reputable journals and conferences. Currently, Dr. Lu serves as the Chair of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee), and the founding Co-chair of IEEE TEMS Blockchain and Distributed Ledgers Technologies Technical Committee (BDLT-TC). Dr. Lu is the Winner of 2016-17 Excellence in Teaching Award, FCS, UNB.

Yandong Zheng received her M.S. degree from the Department of Computer Science, Beihang University, China, in 2017 and she is currently pursuing her Ph.D. degree in the Faculty of Computer Science, University of New Brunswick, Canada. Her research interest includes cloud computing security, big data privacy and applied privacy.

Songnian Zhang received his M.S. degree from Xidian University, China, in 2016 and he is currently pursuing his Ph.D. degree in the Faculty of Computer Science, University of New Brunswick, Canada. His research interest includes cloud computing security, big data query and query privacy.

Jun Shao (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2008. He was a Post-Doctoral Fellow with the School of Information Sciences and Technology, Pennsylvania State University, Pennsylvania, PA, USA, from 2008 to 2010. He is currently a Professor with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China. His current research interests include network security and applied cryptography.

Guiyi Wei is a professor of the School of Computer and Information Engineering at Zhejiang Gongshang University. He obtained his Ph.D. in Dec 2006 from Zhejiang University, where he was advised by Cheung Kong chair professor Yao Zheng. His research interests include wireless networks, mobile computing, cloud computing, social networks and network security.