This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2022.3174184, IEEE Internet of Things Journal

PPAQ: Privacy-Preserving Aggregate Queries for Optimal Location Selection in Road Networks

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

***Abstract*—Aggregate Nearest Neighbor (ANN) query, which can find an optimal location with the smallest aggregate distance to a group of query users' locations, has received considerable attention and been practically useful in many real-world location-based applications. Nevertheless, query users still hesitate to use these applications due to privacy concerns, as there is a worrisome that the location-based service (LBS) providers may abuse their locations after collecting them. In this paper, to tackle this issue, we propose a novel privacy-preserving aggregate query (PPAQ) scheme to select an optimal location for query users in road networks. Specifically, we first analyze the problem of the ANN query in road networks and identify two basic operations, i.e., addition and comparison, in the query. Then, we carefully design efficient addition and comparison circuits to securely add and compare two bit-based inputs, respectively. With these two secure circuits, we propose our PPAQ scheme, which can simultaneously protect the users' locations, query results, and access patterns from leaking. Detailed security analysis shows that our proposed scheme is indeed privacy-preserving. In addition, extensive performance evaluations are conducted, and the results indicate that our proposed scheme has an acceptable efficiency for non-real-time applications.**

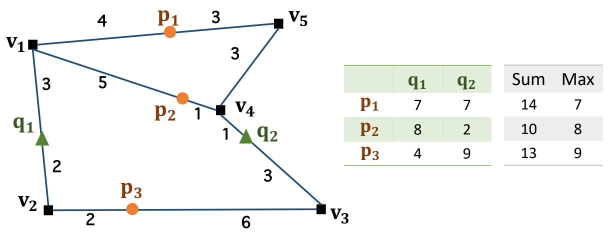
***Index Terms*—Aggregate queries, Optimal location, Privacy preservation, Road networks, Location-based service (LBS)**

I. Introduction

Aggregate nearest neighbor (ANN) query is one of the classic problems due to its quite wide range of applications [1]–[8]. Especially in road networks [5]–[8], ANN query has huge practical significance and demand for LBS providers to select the optimal point in road networks for a group of users. A representative example of ANN query is illustrated as follows.

Example 1 (Motivation). An LBS provider, e.g., Yelp, who owns the road network information, e.g., locations of junctions and points of interest (POI), offers the ANN query services for users. Fig. 1 shows a road network with five junctions $\{\mathrm{v}_1, \mathrm{v}_2, \cdots, \mathrm{v}_5\}$ *and three POIs* $\{\mathrm{p}_1, \mathrm{p}_2, \mathrm{p}_3\}$. *Assume a group of colleagues want to find a restaurant, which should have the minimum aggregated distance to all group members, to have dinner on the weekend. In Fig. 1, we take two query users as an example* $\{\mathrm{q}_1, \mathrm{q}_2\}$, *and they can launch the ANN query by separately sending their own locations to the service provider. After receiving locations, the service provider performs the ANN query to select the optimal restaurant and distribute it to all colleagues in the group.*

In the above example, if the group of colleagues want to retrieve a restaurant with the minimum total distance, the ANN query will return p_2 as the optimal point since it has the minimum sum distance. While if they hope to find a restaurant minimizing the maximum distance for every member, p_1 will be selected. See Section III-A for the formal definition of the ANN queries in a road network. Hereinafter, we will use "location" and "point" interchangeably.



	q_1	q_2	Sum	Max
p_1	7	7	14	7
p_2	8	2	10	8
p_3	4	9	13	9

Fig. 1. An example of ANN query in a road network

However, since the query users need to provide their locations to the service providers when enjoying the ANN query services, they may hesitate to use such services due to privacy concerns. As reported in [9], the disclosure of locations indeed incurs serious personal safety threats. Therefore, it is imperative to preserve the location privacy of users when populating the ANN query services in road networks. Unfortunately, many previously reported works [5]–[8] on ANN queries in road networks focused on improving the query performance over plaintexts and did not consider the privacy issues. Though the works in [10]–[12] exploited the problem of privacy-preserving ANN queries, there still exist privacy issues in these techniques. In [10], a cloak-based technique is adopted to protect users' locations, and the service providers return a super-set of the query result to the query users. It is clear that the approach cannot fully protect the locations of users and query results since service providers know their approximate locations. Regarding [11], the query users can perform group kNN queries by using the secure multiparty computation technique. Although it can protect users' locations, this approach leaks the query result to service providers. Recently, the work in [12] proposed a privacy-preserving ANN query scheme by combining the dummy technique and Paillier encryption. However, when a query user launches multiple queries in a fixed location, e.g., at home, the service provider may infer or narrow down the range of the user's location by intersecting

S. Zhang, S. Ray, R. Lu, Y. Zheng and Y. Guan are with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: szhang17@unb.ca, sray@unb.ca, rlu1@unb.ca, yzheng8@unb.ca, yguan4@unb.ca).

J. Shao is with School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, 310018, China (e-mail: chn.junshao@gmail.com).

the location lists that include the dummy location and user's actual location.

In this paper, we aim to propose a fully privacy-preserving ANN query (PPAQ) scheme in road networks, which can simultaneously preserve the privacy of users' locations, the query result, and access patterns [13] while returning the exact query result. Note that the privacy of query results and access patterns are also crucial since leaking them is equivalent to leaking users' locations if the service providers know the selected location or which POI is selected as the optimal location. In addition, our proposed scheme should guarantee the privacy of the user's location no matter how many times the user queries at the same location. To achieve them, we employ a lightweight symmetric homomorphic encryption (SHE) [14] scheme as the cryptographic primitive to encrypt users' locations. It is worth noting that SHE will lower the data utility, especially for the operations of comparison and equality tests. Although a two-server model can easily achieve these operations over encrypted data [15]–[17], it is still challenging to practically implement them over a single-server model. To tackle it, we carefully devise a bit-based secure comparison scheme to obtain the order relation of two values. Meanwhile, a novel matching approach is also proposed to select the equal value in a given set with the encrypted value without leaking the selected value. Furthermore, our proposed scheme can hide access patterns in a single server, which prevents the service provider from knowing the selected POI. Specifically, the main contribution of this paper is three-fold as follows.

- First, we analyze the problem of ANN queries in road networks and identify two basic operations: addition and comparison. Correspondingly, we propose secure addition and secure comparison circuits to respectively perform these two operations over ciphertexts in a single-server model. Note that, when designing the secure comparison circuit, we also extend the SHE scheme to support dividing a value by 2 over encrypted data, which significantly makes SHE more practical.
- Second, based on the above secure circuits, we propose our privacy-preserving aggregate query (PPAQ) scheme to find the optimal location in road networks while preserving the privacy of users' locations, query results, and access patterns simultaneously. To the best of our knowledge, we are the first to consider the fully privacy-preserving ANN queries in road networks to find the optimal location.
- Third, we formally analyze the security of our proposed scheme and demonstrate that our PPAQ scheme can indeed achieve our privacy requirements. Furthermore, we conduct extensive experiments to evaluate the performance of our proposed secure circuits and PPAQ scheme, and the results show that our PPAQ scheme has an acceptable efficiency for non-real-time applications.

The rest of this paper is organized as follows. In Section II, we introduce our system model, security model and design goal. Then, we review our preliminaries in Section III. After that, we present our proposed scheme in Section IV, followed by security analysis and performance evaluation in Section V and Section VI, respectively. Finally, we discuss some related works in Section VII and draw our conclusion in Section VIII.

II. 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 typical and practical client-server model, which mainly consists of two types of entities: a location-based services provider $\mathcal{LSP}$, a group of users $\mathcal{U} = \{u_1, u_2, \cdots, u_m\}$, as shown in Fig. 2.

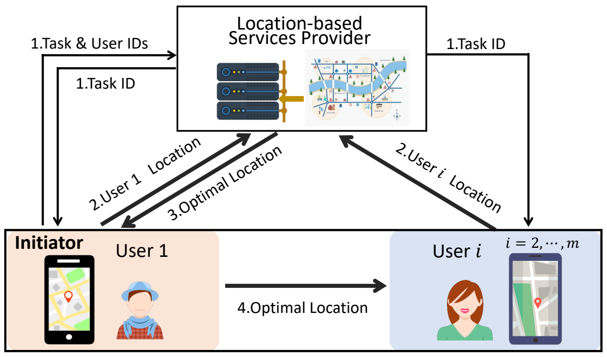


Fig. 2. System model under consideration

Location-based Services Provider $\mathcal{LSP}$: In our system model, $\mathcal{LSP}$ is a real-world service provider company, e.g., Yelp, who can provide location-based services to users. $\mathcal{LSP}$ deploys servers to interact with its client apps that have been installed in users' mobile phones. Meanwhile, $\mathcal{LSP}$ is equipped with enriched spatial databases, including road networks and business location information, for example, the locations of restaurants.

Users $\mathcal{U} = \{u_1, u_2, \cdots, u_m\}$: In our system, some users at different locations, who want to obtain the common optimal location of interest, initially form a virtual group. In Fig. 2, we assume there are m users, i.e., $m \geq 2$, in a group. The users in the group can provide their locations to $\mathcal{LSP}$ through the installed app (here, we assume all users have installed $\mathcal{LSP}$'s app and have registered themselves). With the received locations, $\mathcal{LSP}$ can perform the ANN queries in road networks [5] to find the optimal location and return it to the group of users. For the ease of description, hereafter, we denote u_1 as an initiator in the group, who is responsible for sending a task request to $\mathcal{LSP}$, receiving the selected optimal location from $\mathcal{LSP}$, and distributing the location to other users in the group. Specifically, u_1 first sends a task generation request to $\mathcal{LSP}$, including the identity information (id) of other users. Upon receiving the query, $\mathcal{LSP}$ generates a unique task id *tid* for the request and disseminates it to all users in the group. Each user then responds his/her location together with the received *tid* to $\mathcal{LSP}$. After receiving all users' responses, $\mathcal{LSP}$ chooses an optional meeting location and forwards it to the initiator u_1. After receiving the optimal meeting location, u_i forwards it to other group users.

B. Security Model

In our security model, we consider all users $\mathcal{U}$ to be *honest*, i.e., they will honestly report location information and task id to $\mathcal{LSP}$. For the initiator u_1, he/she honestly distributes the received optimal location to other users. This is reasonable since all users expect to correctly obtain an optimal location and have no motivation to deviate from the purpose. However, due to the sensitive nature of personal data, users do not want $\mathcal{LSP}$ to know their locations or trajectories. For $\mathcal{LSP}$, we assume it is *semi-honest* [18], i.e., $\mathcal{LSP}$ will faithfully follow the protocol to select the optimal location (performing aggregate nearest neighbor queries) for users but may be curious to learn the location information of users. From our system model, we know that $\mathcal{LSP}$ receives the report locations from users and finds the optimal location, which reveals the potential location of users. As a result, in this work, the privacy threats are from semi-honest $\mathcal{LSP}$, who may attempt to obtain: i) query users' location information; ii) the selected optimal location, in the process of performing ANN queries. Note that, since this work mainly focuses on secure computation techniques, other active attacks, e.g., data poisoning and modification, are beyond the scope of this paper and will be discussed in our future work.

C. Design Goal

Under the above system model and security model, we aim to design a privacy-preserving aggregate nearest neighbor query scheme for selecting optimal location in road networks. In particular, the following objectives should be attained.

- Preserving Users' Location Privacy: The basic requirement of our proposed scheme is to preserve the location privacy of users, i.e., the location information of users should be always kept secret from $\mathcal{LSP}$.
- Preserving Optimal Location Privacy: The selected optimal location should also be kept secret from $\mathcal{LSP}$, which implies that we need to hide access patterns in the process of finding optimal location, i.e., $\mathcal{LSP}$ does not know which location has been selected as the optimal one.

III. Preliminaries

In this section, we first state the problem of the aggregate nearest neighbor queries in road networks. Then, we recall a symmetric homomorphic encryption (SHE) scheme, which serves as the building block of our proposed scheme.

A. Aggregate Nearest Neighbor Queries in Road Networks

The aggregate nearest neighbor (ANN) query in road networks is a classic problem and has been used to find the optimal location or point of interest in road networks, which minimizes an aggregate distance function with respect to a set of query points [5]. Before delving into the ANN query, we first define an aggregate network distance function.

Definition 1 (Aggregate Network Distance Function)**.** *Given a set of query points $\mathcal{Q} = \{\mathtt{q}_1, \mathtt{q}_2, \cdots, \mathtt{q}_m\}$ and a point $\mathtt{p}$ in a road network, the aggregate network distance function $\mathcal{F}$ is monotonically increasing and can compute the aggregate distance from $\mathtt{p}$ to $\mathcal{Q}$:*

$$\mathcal{F}(\mathtt{p}, \mathcal{Q}) = \mathcal{F}(d(\mathtt{q}_1, \mathtt{p}), d(\mathtt{q}_2, \mathtt{p}), \cdots, d(\mathtt{q}_m, \mathtt{p})), \quad (1)$$

where $d(\mathtt{q}_t, \mathtt{p}), t \in [1, m]$, is the shortest distance between $\mathtt{p}$ and $\mathtt{q}_t$ along a given road network.

Generally, $\mathcal{F}$ is comprised of two basic functions *sum* and *max* [5], [7], in which *sum* indicates the total distance from $\mathtt{p}$ to all query points $\mathcal{Q}$, while *max* means the maximum distance between $\mathtt{p}$ and $\mathcal{Q}$.

Definition 2 (Aggregate Nearest Neighbor Queries in Road Networks)**.** *Given $\mathcal{Q}$ and a set of data points $\mathcal{P} = \{\mathtt{p}_1, \mathtt{p}_2, \cdots, \mathtt{p}_n\}$ in a road network, the aggregate nearest neighbor queries can return an optimal point $\mathtt{p}_o \in \mathcal{P}$ that has a minimal network distance, i.e., there does not exist $\mathtt{p}_i \in \mathcal{P}/\{\mathtt{p}_o\}$ such that $\mathcal{F}(\mathtt{p}_i, \mathcal{Q}) < \mathcal{F}(\mathtt{p}_o, \mathcal{Q})$.*

B. Symmetric Homomorphic Encryption (SHE)

SHE is a lightweight symmetric homomorphic encryption that can support homomorphic addition and multiplication. It was first proposed in [19] and proved to be IND-CPA secure in [14]. Concretely, SHE consists of three algorithms, namely i) key generation $\mathsf{KeyGen}()$; ii) encryption $\mathsf{Enc}()$; and iii) decryption $\mathsf{Dec}()$, as follows.

- $\mathsf{KeyGen}(k_0, k_1, k_2)$: Given three security parameters $\{k_0, k_1, k_2\}$ satisfying $k_1 \ll k_2 < k_0$, the algorithm first chooses two large prime numbers p, q with $|p| = |q| = k_0$ and sets $\mathcal{N} = pq$. Then, it generates the secret key $sk = (p, \mathcal{L})$, where $\mathcal{L}$ is a random number with $|\mathcal{L}| = k_2$, and the public parameter $pp = (k_0, k_1, k_2, \mathcal{N})$. Besides, the algorithm sets the basic message space $\mathcal{M} = [-2^{k_1-1}, 2^{k_1-1})$.
- $\mathsf{Enc}(sk, m)$: On input of a secret key sk and a message $m \in \mathcal{M}$, the encryption algorithm outputs the ciphertext $\mathsf{E}(m) = (r\mathcal{L} + m)(1 + r'p) \mod \mathcal{N}$, where $r \in \{0,1\}^{k_2}$ and $r' \in \{0,1\}^{k_0}$ are random numbers.
- $\mathsf{Dec}(sk, \mathsf{E}(m))$: Taking the secret key sk and a ciphertext $\mathsf{E}(m)$ as inputs, the decryption algorithm recovers a message $m' = (\mathsf{E}(m) \mod p) \mod \mathcal{L} = (r\mathcal{L} + m) \mod \mathcal{L}$. If $m' < \frac{\mathcal{L}}{2}$, it indicates $m \geq 0$ and $m = m'$. Otherwise, $m < 0$ and $m = m' - \mathcal{L}$.

SHE satisfies the homomorphic addition and multiplication properties as follows: i) Homomorphic addition-I: $\mathsf{E}(m_1) + \mathsf{E}(m_2) \mod \mathcal{N} \rightarrow \mathsf{E}(m_1 + m_2)$; ii) Homomorphic multiplication-I: $\mathsf{E}(m_1) \cdot \mathsf{E}(m_2) \mod \mathcal{N} \rightarrow \mathsf{E}(m_1 \cdot m_2)$; iii) Homomorphic addition-II: $\mathsf{E}(m_1) + m_2 \mod \mathcal{N} \rightarrow \mathsf{E}(m_1 + m_2)$; iv) Homomorphic multiplication-II: $\mathsf{E}(m_1) \cdot m_2 \mod \mathcal{N} \rightarrow \mathsf{E}(m_1 \cdot m_2)$ when $m_2 > 0$.

SHE encryption under public key setting. In order to realize the SHE encryption under public key setting, we take $sk = (p, \mathcal{L})$ as the private key, and use sk to generate two ciphertexts $\mathsf{E}(0)_1$, $\mathsf{E}(0)_2$ of 0 with different random numbers, and set the public key $pk = \{\mathsf{E}(0)_1, \mathsf{E}(0)_2, pp\}$. In such a way, one can use the above homomorphic properties to encrypt a message $m \in \mathcal{M}$ by the following

$$\mathsf{E}(m) = m + r_1 \cdot \mathsf{E}(0)_1 + r_2 \cdot \mathsf{E}(0)_2 \mod \mathcal{N} \quad (2)$$

where r_1, $r_2 \in \{0,1\}^{k_2}$ are two random numbers. Note that the encryption in Eq. (2) was proven as IND-CPA secure [20].

IV. Our Proposed PPAQ Scheme

In this section, we first analyze the problem of ANN query over plaintexts and extract ANN query's two basic operations. Then, we design secure circuits as building blocks to deal with the two basic operations over ciphertexts. After that, based on these building blocks, we present our PPAQ scheme.

A. Analyzing ANN Queries in Road Network

A typical road network is usually abstracted as road segments (edges) and junctions (vertices) [5], as shown in Fig. 1. Assume that there are k vertices in a road network, denoted as $\mathcal{V} = \{\mathtt{v}_1, \mathtt{v}_2, \cdots, \mathtt{v}_k\}$, and n points of interest $\mathcal{P} = \{\mathtt{p}_1, \mathtt{p}_2, \cdots, \mathtt{p}_n\}$ that lie on edges. Given m query points $\mathcal{Q} = \{\mathtt{q}_1, \mathtt{q}_2, \cdots, \mathtt{q}_m\}$ that also lie on edges, we have the following network distances: for each $\mathtt{p}_i$, $i = 1, 2, \cdots, n$,

$$\mathcal{F}(\mathtt{p}_i, \mathcal{Q}) = \mathcal{F}\big(d(\mathtt{q}_t, \mathtt{p}_i) = d(\mathtt{q}_t, \hat{\mathtt{v}}_t) + d(\hat{\mathtt{v}}_t, \mathtt{p}_i)\ |_{t=1}^{m}\big) \quad (3)$$

where $\hat{\mathtt{v}}_t \in \mathcal{V}$ is one of the vertices lying on the same edge as the query point $\mathtt{q}_t \in \mathcal{Q}$. For the simplicity sake, we assume that $\hat{\mathtt{v}}_t \in \mathcal{V}$ is the closest neighbor vertex to $\mathtt{q}_t$ in the road network. If we let $\mathcal{F}$ be the *sum* function, the ANN query will return the point in $\mathcal{P}$ that has the minimum total distance:

$$\begin{aligned}\arg\min_{i\in[1,n]}(\mathcal{F}(\mathtt{p}_i, \mathcal{Q})) &= \arg\min_{i\in[1,n]}\left(\sum_{t=1}^{m}(d(\mathtt{q}_t, \hat{\mathtt{v}}_t) + d(\hat{\mathtt{v}}_t, \mathtt{p}_i))\right)\\ &= \arg\min_{i\in[1,n]}\left(\sum_{t=1}^{m} d(\mathtt{q}_t, \hat{\mathtt{v}}_t) + \sum_{t=1}^{m} d(\hat{\mathtt{v}}_t, \mathtt{p}_i)\right)\\ &\Leftrightarrow \arg\min_{i\in[1,n]}\left(\sum_{t=1}^{m} d(\hat{\mathtt{v}}_t, \mathtt{p}_i)\right).\end{aligned} \quad (4)$$

Meanwhile, since $\mathcal{LSP}$ usually holds the road network, i.e., the location of nodes $\mathcal{V}$ and data points $\mathcal{P}$, $\mathcal{LSP}$ can pre-compute $\sum_{t=1}^{m} d(\hat{\mathtt{v}}_t, \mathtt{p}_i)$ and quickly obtain the optimal point if it knows the neighbor node of the query points.

On the other hand, if $\mathcal{F}$ is the *max* function, the ANN query can return the point with the following distance:

$$\begin{aligned}&\arg\min_{i\in[1,n]}(\mathcal{F}(\mathtt{p}_i, \mathcal{Q}))\\ &= \arg\min_{i\in[1,n]}\left(\arg\max_{t\in[1,m]}\big(d(\mathtt{q}_t, \hat{\mathtt{v}}_t) + d(\hat{\mathtt{v}}_t, \mathtt{p}_i)\big)\right)\end{aligned} \quad (5)$$

Similarly, $d(\hat{\mathtt{v}}_t, \mathtt{p}_i)$ can be pre-computed by $\mathcal{LSP}$. If users send $d(\mathtt{q}_t, \hat{\mathtt{v}}_t)$ and $\hat{\mathtt{v}}_t$ to $\mathcal{LSP}$, it is simple for $\mathcal{LSP}$ to obtain the maximum distance by matching vertex $\hat{\mathtt{v}}_t$.

From the above analysis, we can simplify the problem of ANN queries in road networks by converting the distance calculation between $\mathcal{P}$ and $\mathcal{Q}$ into the distance between $\mathcal{V}$ and $\mathcal{P}$, which can be pre-computed by $\mathcal{LSP}$. Furthermore, from Eq. (4) and Eq. (5), we can see that the ANN queries essentially have two basic operations: addition and comparison. Accordingly, we need to perform these two operations over ciphertexts if we apply SHE to encrypt users' locations. Note that, although it is easy to have the addition operation over ciphertexts by using the homomorphic addition property, it is not easy to compare values over their ciphertexts in a single-server model. This is because the operator is expected to obtain the comparison result without knowing any information about the underlying plaintexts. Although the arithmetic operations (multiplication and addition) can be performed on ciphertexts due to the homomorphic properties of SHE, the operator cannot directly use them to compare two ciphertexts and get the comparison result in a single server. As a result, to tackle this challenge, we carefully devise two novel bit-based secure circuit schemes to perform addition and comparison operations over ciphertexts.

a	0	0	1	1	0	0	1	1
b	0	1	0	1	0	1	0	1
Carry	0	0	0	0	1	1	1	1
o	0	1	1	0	1	0	0	1
$\hat{c}$	0	0	0	1	0	1	1	1

Fig. 3. Truth table of addition, in which **Carry** is the current carry bit, o is the output of $(a + b + \textbf{Carry} \mod 2)$, and $\hat{c}$ is the new carry bit.

B. Secure Addition and Comparison Circuits

In order to securely compare two integers without leaking them and the result, we design a secure comparison scheme based on the encrypted bit sequences. Since we employ SHE to encrypt each bit, and only the addition and multiplication (homomorphic) operations are used in our comparison scheme, we denote it as secure comparison circuit. Correspondingly, to keep consistency, we also design a secure addition circuit to achieve the addition operation over encrypted bit sequences. In the following, we first introduce the secure addition circuit, and then present the secure comparison circuit.

1) Secure Addition Circuit: Assume there are two non-negative integers $\{x, y\}$ of the same bit length, and the corresponding encrypted bit sequences are $\mathsf{E}(\vec{x}) = (\mathsf{E}(x^j)|_{j=l}^{0})$ and $\mathsf{E}(\vec{y}) = (\mathsf{E}(y^j)|_{j=l}^{0})$, where $x^j, y^j \in \{0, 1\}$, $j = l, \cdots, 1, 0$, and l is the most significant bit position of inputs. The secure addition circuit is to compute $\mathsf{E}(\vec{z}) = (\mathsf{E}(z^j)|_{j=l}^{0})$ by operating $\mathsf{E}(\vec{x})$ and $\mathsf{E}(\vec{y})$ to satisfy $z = x + y$, where z is an integer and $\mathsf{E}(\vec{z})$ is its corresponding encrypted bit sequence. For example, if we set $x = 5, y = 7$, $\mathsf{E}(\vec{5}) = (\mathsf{E}(0), \mathsf{E}(1), \mathsf{E}(0), \mathsf{E}(1))$ and $\mathsf{E}(\vec{7}) = (\mathsf{E}(0), \mathsf{E}(1), \mathsf{E}(1), \mathsf{E}(1))$, then we can compute $z = 5 + 7 = 12$ and $\mathsf{E}(\vec{12}) = (\mathsf{E}(1), \mathsf{E}(1), \mathsf{E}(0), \mathsf{E}(0))$.

Fig. 3 illustrates the truth table of the addition operation, in which o is the output of current bit and $\hat{c}$ is the new carry bit. It is easy to obtain the logical expressions of o and $\hat{c}$ as follows:

$$\begin{aligned}o &= (^{\neg}a \wedge ^{\neg} b \wedge c) \vee (^{\neg}a \wedge b \wedge ^{\neg} c)\\ &\quad \vee (a \wedge ^{\neg} b \wedge ^{\neg} c) \vee (a \wedge b \wedge c)\\ &= a + b + c - 2(ab + ac + bc) + 4abc\end{aligned}$$

$$\begin{aligned}\hat{c} &= (^{\neg}a \wedge b \wedge c) \vee (a \wedge ^{\neg} b \wedge c)\\ &\quad \vee (a \wedge b \wedge ^{\neg} c) \vee (a \wedge b \wedge c)\\ &= ab + ac + bc - 2abc\end{aligned}$$

According to the above logical expressions, we have $\mathsf{E}(\vec{z}) =$

$(\mathsf{E}(z^l), \cdots, \mathsf{E}(z^j), \cdots, \mathsf{E}(z^1), \mathsf{E}(z^0))$, for each $j = l, \cdots, 1, 0$

$$\begin{aligned} \mathsf{E}(z^j) &= \mathsf{E}(x^j + y^j + \hat{c}^j) \\ &\quad + \mathsf{E}(-1) \cdot \mathsf{E}(2) \cdot \mathsf{E}(x^j y^j + x^j \hat{c}^j + y^j \hat{c}^j) \\ &\quad + \mathsf{E}(4) \cdot \mathsf{E}(x^j y^j \hat{c}^j), \end{aligned} \tag{6}$$

where:

$$\mathsf{E}(\hat{c}^j) = \begin{cases} \mathsf{E}(x^{j-1}y^{j-1} + x^{j-1}\hat{c}^{j-1} + y^{j-1}\hat{c}^{j-1}) & \\ \quad + \mathsf{E}(-1) \cdot \mathsf{E}(2) \cdot \mathsf{E}(x^{j-1}y^{j-1}\hat{c}^{j-1}) & j > 0 \\ \mathsf{E}(0) & j = 0. \end{cases} \tag{7}$$

Next, we depict the details about the process of $\mathsf{E}(\vec{12}) = \mathsf{E}(\vec{5}) + \mathsf{E}(\vec{7})$ with our secure addition circuit. First of all, since $\hat{c}^0 = 0$, we have $z^0 = x^0 + y^0 - 2x^0y^0 = 0$. Then, for $j = 1$, we can obtain:

$$\begin{aligned} \hat{c}^1 &= x^0y^0 + x^0\hat{c}^0 + y^0\hat{c}^0 - 2x^0y^0\hat{c}^0 = 1; \\ z^1 &= x^1 + y^1 + \hat{c}^1 - 2(x^1y^1 + x^1\hat{c}^1 + y^1\hat{c}^1) + 4x^1y^1\hat{c}^1 \\ &= 2 - 2 = 0; \end{aligned}$$

Repeatedly, with Eq. (7) and Eq. (6), we can calculate $\hat{c}^2 = 1$, $z^2 = 1$ and $\hat{c}^3 = 1$, $z^3 = 1$. Finally, we have $(\mathsf{E}(z^j)|_{j=3}^{0}) = (\mathsf{E}(1), \mathsf{E}(1), \mathsf{E}(0), \mathsf{E}(0)) = \mathsf{E}(\vec{12})$.

2) Secure Comparison Circuit: Given $\mathsf{E}(\vec{x}) = (\mathsf{E}(x^j)|_{j=l}^{0})$ and $\mathsf{E}(\vec{y}) = (\mathsf{E}(y^j)|_{j=l}^{0})\}$, the secure comparison circuit is used to determine whether the corresponding integers x, y satisfy $x > y$ without leaking x, y, and the comparison result. If yes, it outputs $\mathsf{E}(\delta) = \mathsf{E}(1)$, otherwise $\mathsf{E}(\delta) = \mathsf{E}(0)$.

The main idea over the plaintexts is to check whether there exists the most significant differing bit position j such that $x^j > y^j$ and $x^{j'} = y^{j'}$ for all $j' = l, \cdots, j+1$. If yes, then $x > y$; otherwise $x \leq y$. For $j = l, \cdots, 1, 0$, we can compute $d^j = x^j - y^j$, then there are three cases for d^j, i.e., $d^j = 1, 0, -1$. When $d^j = 1$, or -1, we can directly obtain the output from the difference of the j-th bit. While when $d^j = 0$, we have to recursively use the $(j-1)$-th bit to determine the order relation until $j = 0$. When the operations work on the ciphertexts, since we cannot determine whether $\mathsf{E}(d^j)$ is $\mathsf{E}(d^j) = \mathsf{E}(1), \mathsf{E}(0)$, or $\mathsf{E}(-1)$, we have to go through all $j = l, \cdots, 1, 0$.

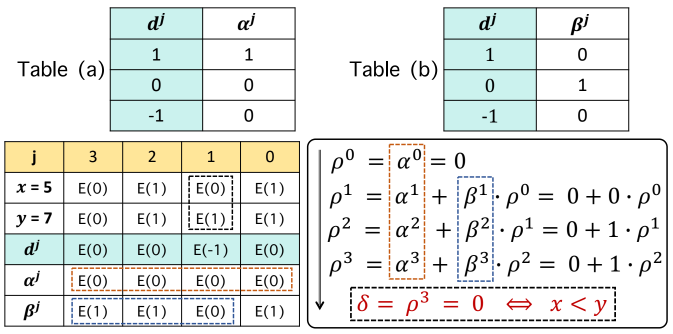


Table (a)

d^j	α^j
1	1
0	0
-1	0

Table (b)

d^j	β^j
1	0
0	1
-1	0

j	3	2	1	0
x = 5	E(0)	E(1)	E(0)	E(1)
y = 7	E(0)	E(1)	E(1)	E(1)
d^j	E(0)	E(0)	E(-1)	E(0)
α^j	E(0)	E(0)	E(0)	E(0)
β^j	E(1)	E(1)	E(0)	E(1)

$$\begin{aligned} \rho^0 &= \alpha^0 = 0 \\ \rho^1 &= \alpha^1 + \beta^1 \cdot \rho^0 = 0 + 0 \cdot \rho^0 \\ \rho^2 &= \alpha^2 + \beta^2 \cdot \rho^1 = 0 + 1 \cdot \rho^1 \\ \rho^3 &= \alpha^3 + \beta^3 \cdot \rho^2 = 0 + 1 \cdot \rho^2 \end{aligned}$$

$$\boxed{\delta = \rho^3 = 0 \iff x < y}$$

Fig. 4. Relations of α^j, β^j and d^j with an example of $x = 5, y = 7$.

For the ease of description, we denote the current j-th bit value as α^j and the transitive bit value from j to $j-1$ as β^j. Since we consider $x > y$, we have the relation of α^j, β^j and d^j shown in Fig. 4, where

$$\begin{cases} \alpha^j = (d^j \cdot (d^j + 1))/2 \\ \beta^j = 1 - d^j \cdot d^j \end{cases} \tag{8}$$

Then, over the ciphertexts $\mathsf{E}(\vec{x}) = (\mathsf{E}(x^j)|_{j=l}^{0})$ and $\mathsf{E}(\vec{y}) = (\mathsf{E}(y^j)|_{j=l}^{0})\}$, we have the output of each bit ρ^j as follows.

$$\mathsf{E}(\rho^j) = \begin{cases} \mathsf{E}(\alpha^j) + \mathsf{E}(\beta^j) \cdot \mathsf{E}(\rho^{j-1}) & j > 0 \\ \mathsf{E}(\alpha^j) & j = 0, \end{cases} \tag{9}$$

where $j = l, \cdots, 1, 0$. Finally, the secure comparison circuit produces $\mathsf{E}(\delta) = \mathsf{E}(\rho^l)$ as the output.

Correctness. We say our secure comparison circuit is correct if it outputs $\mathsf{E}(\delta) = \mathsf{E}(\rho^l) = \mathsf{E}(1)$ when $x > y$, and outputs $\mathsf{E}(\delta) = \mathsf{E}(\rho^l) = \mathsf{E}(0)$ otherwise.

Proof. When $x > y$, it means $\exists j \in [0, l], d^j = 1$ and $d^{j'} = 0$, for all $j' = l, \cdots, j+1$. In this case, $\mathsf{E}(\rho^j) = \mathsf{E}(1) + \mathsf{E}(0) \cdot \mathsf{E}(\rho^{j-1}) = \mathsf{E}(1)$. Since $d^{j'} = 0$ for $j' = l, \cdots, j+1$, we have $\mathsf{E}(\alpha^{j'}) = \mathsf{E}(0)$ and $\mathsf{E}(\beta^{j'}) = \mathsf{E}(1)$, leading to $\mathsf{E}(\rho^{j'}) = \mathsf{E}(\rho^{j'-1})$. As a result, $\mathsf{E}(\rho^l) = \mathsf{E}(\rho^j) = \mathsf{E}(1)$. Similarly, when $x < y$, we have $\mathsf{E}(\rho^l) = \mathsf{E}(\rho^j) = \mathsf{E}(0)$. When $x = y$, it means $\forall j \in [0, l]$, $d^j = 0$, leading to $\mathsf{E}(\alpha^j) = \mathsf{E}(0)$ and $\mathsf{E}(\beta^j) = \mathsf{E}(1)$. Therefore, $\mathsf{E}(\rho^l) = \mathsf{E}(\rho^0) = \mathsf{E}(\alpha^0) = \mathsf{E}(0)$. □

Nevertheless, for obtaining $\mathsf{E}(\alpha^j)$, we still need to further compute $\mathsf{E}(d^j \cdot (d^j + 1)) / \mathsf{E}(2)$. To tackle the challenge, we can choose an odd random number $\mathcal{L}$ in SHE, then it is easy for us to find an inverse element $\mathsf{E}(\Delta)$ satisfying $\mathsf{E}(\Delta) \cdot \mathsf{E}(2) = \mathsf{E}(1)$.

Theorem 1. *In the* $\mathsf{KeyGen}(k_0, k_1, k_2)$ *algorithm of SHE, when an odd random number* $\mathcal{L}$ *is chosen, i.e.,* $\gcd(2, \mathcal{L}) = 1$, *we can set* $\Delta = \frac{1+\mathcal{L}}{2}$ *to satisfy* $\mathsf{E}(\Delta) \cdot \mathsf{E}(2) = \mathsf{E}(1)$.

Proof. Since "$\bmod \mathcal{L}$" is applied in the SHE decryption algorithm, it is easy to see $\mathsf{E}(\Delta) \cdot \mathsf{E}(2) = \mathsf{E}(\Delta \cdot 2) = \mathsf{E}(\frac{1+\mathcal{L}}{2} \cdot 2) = \mathsf{E}(1 + \mathcal{L}) = \mathsf{E}(1)$. The correctness follows. □

According to Theorem 1, we can obtain $\mathsf{E}(\alpha^j)$ by computing $\mathsf{E}(\alpha^j) = \mathsf{E}(d^j \cdot (d^j + 1)) \cdot \mathsf{E}(\Delta)$.

In Fig. 4, we also demonstrate an example of comparing $x = 5$ and $y = 7$. Since $x < y$, the result $\delta = \rho^3 = 0$. Note that, in the above analysis, we only show that our secure comparison circuit can determine whether $x > y$. In fact, we can also determine any order relations of two encrypted bit sequences by constructing a α-function:

$$\alpha^j = f_\alpha(d^j) = \lambda_1 \frac{d^j(d^j+1)}{2} + \lambda_2 \frac{(d^j-1)(d^j+1)}{-1} + \lambda_3 \frac{d^j(d^j-1)}{2}. \tag{10}$$

The correctness of α-function can be easily verified. In Fig. 4, if the i-th row of Table (a) is 1, we can set the i-th term of $f_\alpha(d^j)$ is 1, and others are 0. For example, if we want to determine $x \geq y$, we should make $\lambda_1 = \lambda_2 = 1$ and $\lambda_3 = 0$. For determining whether $x > y$, we can set $\lambda_1 = 1$ and $\lambda_2 = \lambda_3 = 0$. Thus, $\alpha^j = f_\alpha(d^j) = \frac{d^j(d^j+1)}{2}$.

C. Description of PPAQ

In this subsection, we present our PPAQ scheme, which is comprised of four phases: 1) system initialization; 2) location report; 3) optimal location selection; and 4) location recovery.

Before delving into the details, we first assume $\mathcal{LSP}$ holds a road network with k vertices $\mathcal{V} = \{\mathtt{v}_1, \mathtt{v}_2, \cdots, \mathtt{v}_k\}$ and n points of interest $\mathcal{P} = \{\mathtt{p}_1, \mathtt{p}_2, \cdots, \mathtt{p}_n\}$, and there are m users in the group, where $k, n, m \geq 2$. We say $\mathtt{v}_s \in \mathcal{V}$ and $\mathtt{p}_i \in \mathcal{P}$, where $s \in [1, k]$ and $i \in [1, n]$, are unique integers encoded by $\mathcal{LSP}$. The corresponding binary formats are denoted as $\vec{\mathtt{v}}_s$ and $\vec{\mathtt{p}}_i$. Besides, we say $\mathsf{E}(\vec{\mathtt{v}}_s) = (\mathsf{E}(\mathtt{v}_s^j)|_{j=l}^0)$ and $\mathsf{E}(\vec{\mathtt{p}}_i) = (\mathsf{E}(\mathtt{p}_i^j)|_{j=l}^0)$, where $j = l, \cdots, 1, 0$, indicate that the binary is encrypted bit by bit, and l is the common maximum bit length.

1) System Initialization: In our scheme, the initiator u_1 initializes a privacy-preserving aggregate query. First, u_1 obtains the user ids of other users in the group and sends a task generation request with these ids to $\mathcal{LSP}$. Then, $\mathcal{LSP}$ randomly generates a task id *tid* and disseminates it to users $\mathcal{U}$ according to the received user ids. Next, given security parameters (k_0, k_1, k_2), u_1 generates the $sk = (p, \mathcal{L})$, where $\mathcal{L}$ is odd, and $pp = (k_0, k_1, k_2, \mathcal{N})$. Afterward, u_1 calls $\mathsf{Enc}(sk, m)$ to generate $\{\mathsf{E}(0)_1, \mathsf{E}(0)_2, \mathsf{E}(-1), \mathsf{E}(\Delta)\}$, where $\Delta = \frac{1+\mathcal{L}}{2}$. Finally, u_1 sets the public key $pk = \{\mathsf{E}(0)_1, \mathsf{E}(0)_2, \mathsf{E}(-1), \mathsf{E}(\Delta), pp\}$.

2) Location Report: Assume that each user $u_t, t \in [1, m]$, has installed the client ($\mathcal{LSP}$'s application) in his/her mobile phone. First, the client senses the location of u_t, denoted as $\mathtt{q}_t$, and calculates the distance from $\mathtt{q}_t$ to its closest neighbor vertex $\hat{\mathtt{v}}_t$. Since $\mathcal{V}$ is embedded into clients by $\mathcal{LSP}$ in advance, the client can easily compute $\theta_t = d(\mathtt{q}_t, \hat{\mathtt{v}}_t)$ with the sensed $\mathtt{q}_t$. After that, θ_t is encoded into a bit sequence and encrypted into $\mathsf{E}(\vec{\theta}_t) = (\mathsf{E}(\theta_t^j)|_{j=l}^0)$ using Eq. (2). With the same encryption approach, $\hat{\mathtt{v}}_t$ is encrypted into $\mathsf{E}(\vec{\mathtt{v}}_t) = (\mathsf{E}(\hat{\mathtt{v}}_t^j)|_{j=l}^0)$. Finally, u_t sends $\langle tid, \mathsf{E}(\vec{\theta}_t), \mathsf{E}(\vec{\mathtt{v}}_t)\rangle$ to $\mathcal{LSP}$.

3) Optimal Location Selection: Assume there are m users in the group. $\mathcal{LSP}$ will receive m tuples $\langle tid, \mathsf{E}(\vec{\theta}_t), \mathsf{E}(\vec{\mathtt{v}}_t)\rangle$, where $t \in [1, m]$. Since $\mathcal{LSP}$ holds spatial databases, it can pre-compute the shortest distances $\{\sigma_{si} = d(\mathtt{v}_s, \mathtt{p}_i) \mid s \in [1, k], i \in [1, n]\}$ over plaintexts using existing shortest path algorithms, such as Dijkstra [21] and HiTi [22], and encode $\{\sigma_{si}, \mathtt{v}_s\}$ into bit sequences $\{\vec{\sigma}_{si}, \vec{\mathtt{v}}_s\}$, respectively. Afterward, $\mathcal{LSP}$ can construct tuples $\langle \vec{\mathtt{v}}_s, \vec{\sigma}_{si}, \mathtt{p}_i\rangle$, where $s \in [1, k], i \in [1, n]$ in advance. It is worth noting that both calculating shortest path and encoding bit sequences can be achieved offline. With the constructed and received tuples, $\mathcal{LSP}$ can perform the following steps to find the optimal location.

Step 1: Matching. For each $\mathtt{p}_i$, if all vertices can reach it, there will be k tuples related to $\mathtt{p}_i$: $\langle \vec{\mathtt{v}}_s, \vec{\sigma}_{si}, \mathtt{p}_i\rangle$ for $s = 1, \cdots, k$. Consequently, there must exist a tuple in the k tuples that has a joint vertex with the received $\vec{\mathtt{v}}_t$. That is, $\exists s \in [1, k]$ such that $\hat{\mathtt{v}}_t(\text{a.k.a } \vec{\mathtt{v}}_t) = \mathtt{v}_s(\text{a.k.a } \vec{\mathtt{v}}_s)$. It means $\mathtt{q}_t$ can reach $\mathtt{p}_i$ with the shortest distance $\theta_t + \sigma_{si} = d(\mathtt{q}_t, \hat{\mathtt{v}}_t) + d(\mathtt{v}_s, \mathtt{p}_i)$ passing the vertex $\mathtt{v}_s$ that is also the neighbor vertex $\hat{\mathtt{v}}_t$ of $\mathtt{q}_t$. As a result, the problem is converted into: $\mathcal{LSP}$ needs to find $\vec{\mathtt{v}}_s$ in the k tuples that satisfies $\vec{\mathtt{v}}_s = \vec{\mathtt{v}}_t$ with encrypted $\mathsf{E}(\vec{\mathtt{v}}_t)$, and has no idea about which vertex in $\mathcal{V}$ is the same as $\vec{\mathtt{v}}_t$. To address it, we adopt the following matching approach.

- First, $\mathcal{LSP}$ calculates a flag $\mathtt{f}_s$ for each $\mathtt{v}_s$ as follows.

$$\mathsf{E}(\mathtt{f}_s) = \vec{\mathtt{v}}_s \odot \mathsf{E}(\vec{\mathtt{v}}_t) = \prod_{j=0}^{l} \mathtt{v}_s^j \odot \mathsf{E}(\hat{\mathtt{v}}_t^j), \tag{11}$$

where the operation $\mathtt{v}_s^j \odot \mathsf{E}(\hat{\mathtt{v}}_t^j)$ is defined as

$$\mathtt{v}_s^j \odot \mathsf{E}(\hat{\mathtt{v}}_t^j) = \begin{cases} \mathsf{E}(1 - \hat{\mathtt{v}}_t^j) & \mathtt{v}_s^j = 0 \\ \mathsf{E}(\hat{\mathtt{v}}_t^j) & \mathtt{v}_s^j = 1. \end{cases} \tag{12}$$

It means that only if $\vec{\mathtt{v}}_s$ is the same as $\vec{\mathtt{v}}_t$, i.e., for each bit $\mathtt{v}_s^j = \hat{\mathtt{v}}_t^j$, the corresponding flag $\mathtt{f}_s$ is 1, otherwise it is 0.

- Then, with the k tuples ($\langle \vec{\mathtt{v}}_s, \vec{\sigma}_{si}, \mathtt{p}_i\rangle$ for $s = 1, \cdots, k$) and the corresponding $\mathtt{f}_s$, $\mathcal{LSP}$ computes $\mathsf{E}(\vec{\sigma}_{ti}) = (\mathsf{E}(\sigma_{ti}^j)|_{j=l}^0)$:

$$\mathsf{E}(\sigma_{ti}^j) = \sum_{s=1}^{k} \mathsf{E}(\mathtt{f}_s) \cdot \mathsf{E}(\sigma_{si}^j) = \mathsf{E}(\sum_{s=1}^{k} (\mathtt{f}_s \cdot \sigma_{si}^j)), \tag{13}$$

in which $\mathcal{LSP}$ can first encrypt σ_{si}^j into $\mathsf{E}(\sigma_{si}^j)$ with Eq. (2). Next, $\mathcal{LSP}$ can construct a two-element tuple $\langle \mathsf{E}(\vec{\sigma}_{ti}), \mathtt{p}_i\rangle$ that indicates $\mathtt{q}_t$ can reach $\mathtt{p}_i$ via its neighbor $\hat{\mathtt{v}}_t = \mathtt{v}_s$ with the shortest distance σ_{ti}. Now, $\mathcal{LSP}$ totally holds $m \times n$ tuples: $\langle \mathsf{E}(\vec{\sigma}_{ti}), \mathtt{p}_i\rangle$, where $t \in [1, m]$ and $i \in [1, n]$.

Step 2: Calculating Aggregate Values. If $\mathcal{F}$ is *sum*, $\mathcal{LSP}$ computes the total distance from $\mathcal{Q}$ to $\mathtt{p}_i$. According to Eq. (4), $\mathcal{LSP}$ can only add up all users' shortest distances from their neighbor vertices $\hat{\mathtt{v}}_t$ to $\mathtt{p}_i$, i.e., $\mathsf{E}(\sigma_i) = \sum_{t=1}^{m} \mathsf{E}(\sigma_{ti}) = \mathsf{E}(\sum_{t=1}^{m} \sigma_{ti})$. It can be achieved by our secure addition circuit (Section IV-B1) with inputs $\mathsf{E}(\vec{\sigma}_{ti})$. Note that it is simple for $\mathcal{LSP}$ to set a proper l to guarantee that there is no overflow when using the secure addition circuit.

If $\mathcal{F}$ is *max*, $\mathcal{LSP}$ computes the maximum distance from $\mathcal{Q}$ to $\mathtt{p}_i$. First, $\mathcal{LSP}$ adds up $\mathsf{E}(\theta_t)$ and $\mathsf{E}(\sigma_{ti})$ using the secure addition circuit, i.e., $\mathsf{E}(\tau_{ti}) = \mathsf{E}(\theta_t) + \mathsf{E}(\sigma_{ti})$, in which the inputs of the secure addition circuit are $\mathsf{E}(\vec{\theta}_t)$ and $\mathsf{E}(\vec{\sigma}_{ti})$, and the output is $\mathsf{E}(\vec{\tau}_{ti})$. Then, $\mathcal{LSP}$ determines the maximum value among $\mathsf{E}(\tau_{ti})$ for $t = 1, 2, \cdots, m$. Take comparing $\mathsf{E}(\tau_{1i})$ and $\mathsf{E}(\tau_{2i})$ as an example. $\mathcal{LSP}$ employs our secure comparison circuit (Section IV-B2) to determine whether $\tau_{1i} > \tau_{2i}$ with inputs $\mathsf{E}(\vec{\tau}_{1i})$ and $\mathsf{E}(\vec{\tau}_{2i})$. If yes, our secure comparison circuit outputs $\mathsf{E}(\delta) = \mathsf{E}(1)$, otherwise $\mathsf{E}(\delta) = \mathsf{E}(0)$. Using the following equation, $\mathcal{LSP}$ can calculate the maximum value, denoted as $\mathsf{E}(\vec{\tau}_{\max i})$:

$$\mathsf{E}(\tau_{\max i}^j) = \mathsf{E}(\delta) \cdot (\mathsf{E}(\tau_{1i}^j) - \mathsf{E}(\tau_{2i}^j)) + \mathsf{E}(\tau_{2i}^j). \tag{14}$$

If δ is 1, $\mathsf{E}(\tau_{\max i}) = \mathsf{E}(\tau_{1i})$. Otherwise, $\mathsf{E}(\tau_{\max i}) = \mathsf{E}(\tau_{2i})$. It means $\tau_{\max i}$ is always the larger one. After that, $\mathcal{LSP}$ can continue to compare $\mathsf{E}(\tau_{\max i})$ and $\mathsf{E}(\tau_{3i})$ with the secure comparison circuit. Repeating $m-1$ times, $\mathcal{LSP}$ can obtain the maximum distance from $\mathcal{Q}$ to $\mathtt{p}_i$: $\mathsf{E}(\sigma_i) = \arg\max_{t\in[1,m]}(\mathsf{E}(\tau_{ti}))$.

After calculating aggregating values, $\mathcal{LSP}$ constructs n tuples: $\langle \mathsf{E}(\vec{\sigma}_i), \mathtt{p}_i\rangle$ for $i = 1, 2, \cdots, n$. Since $\mathsf{E}(\sigma_i)$ is computed using the secure addition circuit (*sum*) or the secure comparison circuit (*max*), it is in binary format $\mathsf{E}(\vec{\sigma}_i)$ here.

Step 3: Calculating Optimal Location. After obtaining a set of distances $\langle \mathsf{E}(\vec{\sigma}_i), \mathtt{p}_i\rangle, i \in [1, n]$, $\mathcal{LSP}$ can determine a POI that has the minimum distance among these n distances, i.e., $\arg\min_{i\in[1,n]}(\mathsf{E}(\sigma_i))$. Specifically, $\mathcal{LSP}$ computes the minimum value among $\mathsf{E}(\sigma_i)$ for $i = 1, 2, \cdots, n$. Take comparing $\mathsf{E}(\sigma_1)$ and $\mathsf{E}(\sigma_2)$ as an example. $\mathcal{LSP}$ employs our secure comparison circuit to determine whether $\sigma_1 > \sigma_2$ with inputs $\mathsf{E}(\vec{\sigma}_1)$ and $\mathsf{E}(\vec{\sigma}_2)$. If yes, our secure comparison circuit outputs $\mathsf{E}(\delta) = \mathsf{E}(1)$, otherwise $\mathsf{E}(\delta) = \mathsf{E}(0)$. Then,

$\mathcal{LSP}$ calculates the minimum distance and the corresponding POI, denoted as $\mathsf{E}(\vec{\sigma}_{\min})$ and $\mathsf{E}(\mathtt{p}_{\min})$, respectively,

$$\begin{cases}\mathsf{E}(\sigma^j_{\min}) = \mathsf{E}(\delta)\cdot(\mathsf{E}(\sigma^j_2) - \mathsf{E}(\sigma^j_1)) + \mathsf{E}(\sigma^j_1),\\ \mathsf{E}(\mathtt{p}_{\min}) = \mathsf{E}(\delta)\cdot(\mathsf{E}(\mathtt{p}_2) - \mathsf{E}(\mathtt{p}_1)) + \mathsf{E}(\mathtt{p}_1),\end{cases} \quad (15)$$

where $\mathsf{E}(\mathtt{p}_1)$ and $\mathsf{E}(\mathtt{p}_2)$ are encrypted by $\mathcal{LSP}$ using Eq. (2). If δ is 1, $\mathsf{E}(\sigma_{\min}) = \mathsf{E}(\sigma_2)$ and $\mathsf{E}(\mathtt{p}_{\min}) = \mathsf{E}(\mathtt{p}_2)$. Otherwise, $\mathsf{E}(\sigma_{\min}) = \mathsf{E}(\sigma_1)$ and $\mathsf{E}(\mathtt{p}_{\min}) = \mathsf{E}(\mathtt{p}_1)$. It means that $\mathtt{p}_{\min}$ always has the minimum distance $\sigma_{\min}$. After that, $\mathcal{LSP}$ can continue to compare $\mathsf{E}(\sigma_{\min})$ and $\mathsf{E}(\sigma_3)$ with the secure comparison circuit. Repeating $n-1$ times, $\mathcal{LSP}$ can obtain the optimal location $\mathsf{E}(\mathtt{p}_o)$ that has minimum distance cost $\mathsf{E}(\vec{\sigma}_o)$ from $\mathcal{Q}$ to $\mathcal{P}$, i.e., $\mathsf{E}(\sigma_o) = \arg\min_{i\in[1,n]}(\mathsf{E}(\sigma_i))$. Notably, the parallelization technique can be used here to improve performance. Finally, $\mathcal{LSP}$ returns $\mathsf{E}(\mathtt{p}_o)$ to the initiator u_1. See Algorithm 1 for the complete process of optimal location selection, in which *Sadd* and *Scom* indicate our secure addition circuit and secure comparison circuit, respectively.

4) Location Recovery: Upon receiving $\mathsf{E}(\mathtt{p}_o)$, the initiator u_1 recovers the plaintext $\mathtt{p}_o$ with the secret key sk. Then, u_1 distributes $\mathtt{p}_o$ to other group users via a secure channel.

Correctness. We say our PPAQ scheme is correct if $\mathcal{LSP}$ obtains the optimal location $\mathsf{E}(\mathtt{p}_o)$ with regard to $\mathcal{F}$.

Proof. First of all, we prove that our PPAQ scheme is correct when $\mathcal{F}$ is the *sum* function. In the matching step, according to Eq. (12), we know that $\mathtt{f}_s = 1$ *iff* $\vec{\mathtt{v}}_s = \vec{\mathtt{v}}_t$. As a result, $\mathsf{E}(\sigma_{ti}) = \sum_{s=1}^{k}\mathsf{E}(\mathtt{f}_s)\cdot\mathsf{E}(\sigma_{si})$ must be the shortest distance from $\mathtt{q}_t$ to $\mathtt{p}_i$ since only the matched vertex has $\mathtt{f}_s = 1$, while others are 0. Then, in the aggregating step, since $\mathcal{F}$ is *sum*, we need to obtain the minimum total distances from all query locations to one POI location $\mathtt{p}_i$. According to Eq. (4), we can achieve it by adding up all shortest distances from the matched vertices to $\mathtt{p}_i$, i.e., $\mathsf{E}(\sigma_i) = \sum_{t=1}^{m}\mathsf{E}(\sigma_{ti})$. Next, in the last step, by repeatedly using our secure comparison circuit and Eq. (15), we can obtain $\mathsf{E}(\mathtt{p}_o)$, where $\mathtt{p}_o$ has the minimum total distances. The correctness of our secure comparison circuit (see details in Section IV-B) and Eq. (15) ensures that $\mathtt{p}_o$ is the optimal location when $\mathcal{F}$ is *sum*. From Algorithm 1, we know that the only difference between the sum function and the max function is the aggregating step, in which the maximum distance is guaranteed by the correctness of our secure comparison circuit. Similarly, we can prove that our PPAQ scheme is correct when $\mathcal{F}$ is *max*. □

V. Security Analysis

In this section, we analyze the security of our proposed scheme PPAQ. Following our design goals, we will prove that our PPAQ scheme can preserve the privacy of users' locations and the query result, i.e., the selected optimal location. Since $\mathcal{LSP}$ holds the plaintext information of road networks, it is essential for our PPAQ scheme to hide access patterns.

First, we briefly review the security model for securely realizing an ideal functionality in the presence of the static semi-honest adversary [23]. Since only $\mathcal{LSP}$ is semi-honest in our security model, our proof will focus on the service provider $\mathcal{LSP}$.

Algorithm 1 Optimal Location Selection

```
Input: Encrypted locations of query users, $\{\langle tid, \mathsf{E}(\vec{\theta}_t), \mathsf{E}(\vec{\mathtt{v}}_t)\rangle \mid t \in [1,m]\}$.
       Spatial database, $\{\langle \vec{\mathtt{v}}_s, \vec{\sigma}_{si}, \mathtt{p}_i\rangle \mid s \in [1,k], i \in [1,n]\}$.
Output: The selected optimal location, $\mathsf{E}(\mathtt{p}_o)$.
 1: for $s \leftarrow 1$ to $k$ do
 2:     $\mathsf{E}(\mathtt{f}_s) \leftarrow \vec{\mathtt{v}}_s \odot \mathsf{E}(\vec{\mathtt{v}}_t) = \prod_{j=0}^{l} \mathtt{v}_s^j \odot \mathsf{E}(\hat{\mathtt{v}}_t^j)$;
 3: $\mathsf{E}(\sigma_{ti}) \leftarrow \sum_{s=1}^{k}\mathsf{E}(\mathtt{f}_s)\cdot\mathsf{E}(\sigma_{si}) = \mathsf{E}(\sum_{s=1}^{k}(\mathtt{f}_s\cdot\sigma_{si}))$;
 4: if $\mathcal{F}$ is sum then
 5:     $\mathsf{E}(\sigma_i) \leftarrow \sum_{t=1}^{m}\mathsf{E}(\sigma_{ti}) = \mathsf{E}(\sum_{t=1}^{m}\sigma_{ti})$;        ▷ Sadd
 6: else if $\mathcal{F}$ is max then
 7:     $\mathsf{E}(\tau_{ti}) \leftarrow \mathsf{E}(\theta_t) + \mathsf{E}(\sigma_{ti})$;        ▷ Sadd
 8:     $\mathsf{E}(\sigma_i) \leftarrow \arg\max_{t\in[1,m]}(\mathsf{E}(\tau_{ti}))$;        ▷ Scom and Eq. (14)
 9: $\mathsf{E}(\mathtt{p}_o) \leftarrow \mathsf{E}(\mathtt{p}_1)$; $\mathsf{E}(\sigma_o) \leftarrow \mathsf{E}(\sigma_1)$;
10: for $i \leftarrow 2$ to $n$ do
11:     $\mathsf{E}(\delta_i) \leftarrow Scom(\mathsf{E}(\sigma_o), \mathsf{E}(\sigma_i))$;        ▷ Inputs are bit sequences
12:     $\mathsf{E}(\sigma_o) \leftarrow \mathsf{E}(\delta_i)\cdot(\mathsf{E}(\sigma_o) - \mathsf{E}(\sigma_i)) + \mathsf{E}(\sigma_i)$;
13:     $\mathsf{E}(\mathtt{p}_o) \leftarrow \mathsf{E}(\delta_i)\cdot(\mathsf{E}(\mathtt{p}_o) - \mathsf{E}(\mathtt{p}_i)) + \mathsf{E}(\mathtt{p}_i)$;
14: return $\mathsf{E}(\mathtt{p}_o)$;
```

Real world model: The real world execution of a scheme Π takes place in $\mathcal{LSP}$ and adversary $\mathcal{A}$, who corrupts $\mathcal{LSP}$. Assume that x indicates the input of users' locations, y represents the input held by $\mathcal{LSP}$, and z is auxiliary input, e.g., the length of ciphertexts and bit sequence. With the inputs of x, y, and z, the execution of Π under $\mathcal{A}$ in the real world mode is defined as:

$\mathrm{REAL}_{\Pi,\mathcal{A},z}(x,y) \overset{def}{=} \{\mathrm{Output}^{\Pi}(x,y), \mathrm{View}^{\Pi}(x,y), z\}$,
in which $\mathrm{Output}^{\Pi}(x,y)$ is the output of $\mathcal{LSP}$ after an execution of Π on (x,y), and $\mathrm{View}^{\Pi}(x,y)$ is the view of $\mathcal{LSP}$ during an execution of Π on (x,y).

Ideal world model: In the ideal world execution, there is an ideal functionality $\mathcal{F}$ for a function f, and $\mathcal{LSP}$ interacts only with $\mathcal{F}$. Here, the execution of f under simulator $\mathtt{Sim}$ in the ideal world model on input pair (x,y) and auxiliary input z is defined as:

$\mathrm{IDEAL}_{\mathcal{F},\mathtt{Sim},z}(x,y) \overset{def}{=} \{f(x,y), \mathtt{Sim}(x, f(x,y)), z\}$.

Definition 3 (Security against semi-honest adversary)**.** *Let $\mathcal{F}$ be a deterministic functionality and Π be a scheme in $\mathcal{LSP}$. We say that Π securely realizes $\mathcal{F}$ if there exists $\mathtt{Sim}$ of PPT (Probabilistic Polynomial Time) transformations (where $\mathtt{Sim} = \mathtt{Sim}(\mathcal{A})$) such that for semi-honest PPT adversary $\mathcal{A}$, for x, y and z, for $\mathcal{LSP}$ holds:*

$$REAL_{\Pi,\mathcal{A},z}(x,y) \overset{c}{\approx} IDEAL_{\mathcal{F},\mathtt{Sim},z}(x,y)$$

where $\overset{c}{\approx}$ compactly denotes computational indistinguishability.

A. The PPAQ scheme is privacy-preserving

First, we use Definition 3 to demonstrate that our PPAQ scheme can preserve the privacy of users' locations and the optimal location.

Theorem 2. *The PPAQ scheme securely selects the optimal location in the presence of semi-honest adversary $\mathcal{A}$.*

Proof. Here, we show how to construct the simulator. $\mathtt{Sim}$ randomly chooses m tuples: $\langle tid', \vec{\theta}'_t, \vec{\mathtt{v}}'_t\rangle, t \in [1,m]$ and $\mathtt{p}'_o$, where θ'^j_t, $\mathtt{v}'^j_t \in \{0,1\}$, and $j = l, \cdots, 1, 0$. Then, $\mathtt{Sim}$ simulates $\mathcal{A}$ as follows: i) it generates ciphertexts $\langle tid', \mathsf{E}(\vec{\theta}'_t), \mathsf{E}(\vec{\mathtt{v}}'_t)\rangle$ and $\mathsf{E}(\mathtt{p}'_o)$ by the encryption algorithm of SHE; ii) it outputs $\mathcal{A}$'s view. In the real execution, $\mathcal{A}$ receives m tuples: $\langle tid, \mathsf{E}(\vec{\theta}_t), \mathsf{E}(\vec{\mathtt{v}}_t)\rangle$ and obtains $\mathsf{E}(\mathtt{p}_o)$, whereas $\mathtt{Sim}$

gives $\langle tid', \mathsf{E}(\vec{\theta}'_t), \mathsf{E}(\vec{\mathrm{v}}'_t)\rangle$ and $\mathsf{E}(\mathrm{p}'_o)$ in the ideal execution. The semantic security of SHE [14] guarantees that the views of $\mathcal{A}$ in the real and the ideal worlds are indistinguishable. Here, we ignore *tid* and *tid*$'$ since both of them are randomly chosen.

Since all operations are performed over ciphertexts, it is impossible to break the intermediate results during the scheme without the secret key. Therefore, there is no advantage for $\mathcal{A}$ to distinguish the real world and the ideal world. □

From the above proof, we can see that our PPAQ scheme can ensure the privacy of users' locations $\{\vec{\theta}_t, \vec{\mathrm{v}}_t \mid t \in [1, m]\}$ and the optimal location p_o. As a result, our PPAQ scheme is privacy-preserving.

B. The PPAQ scheme hides access patterns

Next, we show that our PPAQ scheme can preserve the privacy of access patterns.

Theorem 3. *The PPAQ scheme can hide the information about which POI in $\mathcal{P}$ is selected as the optimal location.*

Proof. We prove Theorem 3 by demonstrating $\mathcal{LSP}$ does not know which point in $\mathcal{P}$ is the optimal location p_o though $\mathcal{LSP}$ holds the plaintext information of the road network: $\mathcal{V} = \{\mathrm{v}_1, \mathrm{v}_2, \cdots, \mathrm{v}_k\}$ and $\mathcal{P} = \{\mathrm{p}_1, \mathrm{p}_2, \cdots, \mathrm{p}_n\}$.

First of all, $\mathcal{LSP}$ receives m tuples $\langle tid, \mathsf{E}(\vec{\theta}_t), \mathsf{E}(\vec{\mathrm{v}}_t)\rangle, t \in [1, m]$ and then calculates k flags $\mathsf{E}(\mathtt{f}_s), s \in [1, k]$ for each user with the plaintexts v_s by Eq. (11). However, from Eq. (12), we can see that $\mathsf{E}(\mathtt{f}_s)$ are calculated from $\mathsf{E}(\vec{\mathrm{v}}_t)$ that are kept secret from $\mathcal{LSP}$. As a result, $\mathcal{LSP}$ has no idea about $\mathsf{E}(\mathtt{f}_s)$ and thus does not know $\mathsf{E}(\vec{\sigma}_{ti})$ even though $\vec{\sigma}_{si}$ (calculated by Eq. (13)) are in plaintext. Note that $k \geq 2$. Next, with $\mathsf{E}(\vec{\sigma}_{ti})$, $\mathcal{LSP}$ calculates $\mathsf{E}(\vec{\sigma}_i)$ under the aggregate distance function $\mathcal{F}$. Here, we take $\mathcal{F} = sum$ as an example, in which $\mathsf{E}(\sigma_i) = \sum_{t=1}^{m} \mathsf{E}(\sigma_{ti}) = \mathsf{E}(\sum_{t=1}^{m} \sigma_{ti})$. Since it is achieved by our secure addition circuit that operates on SHE ciphertexts bit by bit, $\mathsf{E}(\vec{\sigma}_i)$ are kept secret from $\mathcal{LSP}$ due to the security of $\mathsf{E}(\vec{\sigma}_{ti})$. After that, $\mathcal{LSP}$ uses Eq. (15) to calculate $\mathsf{E}(\mathrm{p}_{\min})$ with the minimum distance cost. Since all calculations in our secure comparison circuit are performed on encrypted bits, it ensures that $\mathcal{LSP}$ learns nothing about the inputs $\mathsf{E}(\vec{\sigma}_i)$ and the output $\mathtt{E}(\delta)$. Therefore, $\mathcal{LSP}$ does not know which of the two locations to be compared is selected as $\mathsf{E}(\mathrm{p}_{\min})$. Thus, $\mathcal{LSP}$ cannot know which location in $\mathcal{P}$ is selected as the optimal location $\mathsf{E}(\mathrm{p}_o)$. □

VI. Performance Evaluation

In this section, we will first evaluate the performance of our proposed secure addition and comparison circuits and then evaluate our PPAQ scheme. Since the optimal location selection phase is the core part of our PPAQ scheme, and other phases have negligible computational costs compared to the phase, we will focus on the performance of the optimal location selection phase in this section. Notably, the communication cost in our PPAQ scheme is evident and trivial. As a result, we do not provide such evaluations.

Experimental setting: We implemented our scheme in Java and executed it on a machine with 16 GB memory, 3.4 GHz Intel(R) Core(TM) i7-3770 processors, and Ubuntu 16.04 OS. In our experiments, we adopt a synthetic dataset to evaluate our secure addition and secure comparison circuits and two real-world datasets: **Sequoia** [24] used in [12], [25] and **DIMACS** [26] used in [8], to evaluate our PPAQ scheme. In real-world scenarios, since query users usually obtain the optimal location of a specific type of POI, e.g., restaurant, we evaluate the POI of "bar" as the target points in **Sequoia** and "cafe" in **DIMACS**, in which there are 150 bars and 100 cafes, respectively. Since the location space in the **Sequoia** dataset has been normalized into a square space, we randomly choose 200 vertices in the dataset. For the **DIMACS** dataset, we select 200 real locations of junctions in the New York City as vertices with the help of OpenStreetMap [27]. As the shortest distances between vertices and POIs are calculated by $\mathcal{LSP}$ over plaintexts in advance, and how to calculate them is not the focus of this work, we employ the Euclidean distance between vertices and POIs in our experiments.

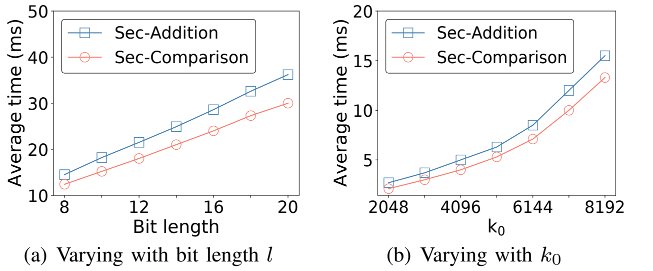


Fig. 5. Average execution time of secure addition and secure comparison circuits. (a) Varying with bit length l, k_0=8192; (b) Varying with k_0, $l = 8$.

A. Performance of Secure Addition and Comparison Circuits

In this subsection, we evaluate the execution time of our proposed secure circuits over the synthetic dataset, in which we randomly generate 2,000 integers for each bit length from 8 to 20, i.e., $l \in [8, 20]$. Recalling Section IV-B, the performance of our secure addition and secure comparison circuits is related to the inputs' bit length l and the key size k_0 (see details in the KeyGen() algorithm of SHE). As a result, we explore the impact of these two parameters in Fig. 5, where the secure addition circuit and secure comparison circuit are denoted as Sec-Addition and Sec-Comparison, respectively. Fig. 5(a) depicts the average execution time of Sec-Addition and Sec-Comparison varying with the bit length l from 8 to 20. We can see that the secure addition circuit is always more expensive than the secure comparison circuit in execution time. By comparing Eq. (6) and Eq. (9), it is easy to see that our devised secure comparison circuit is simpler than the secure addition circuit due to the fewer homomorphic operations. In Fig. 5(b), we explore the impact of key size k_0 from 2048 to 8192. Similarly, the secure addition circuit has higher execution time for the same reason. Although our proposed secure circuits adopt the bit-based homomorphic operations to achieve addition and comparison, both Fig. 5(a) and Fig. 5(b) show their efficiency because they run at the millisecond level and have no communication with other entities.

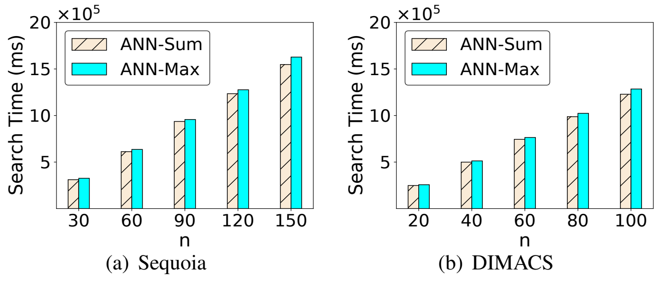


Fig. 6. Search time of our PPAQ scheme varying with the number of POIs n. (a) Over the Sequoia dataset; (b) Over the DIMACS dataset.

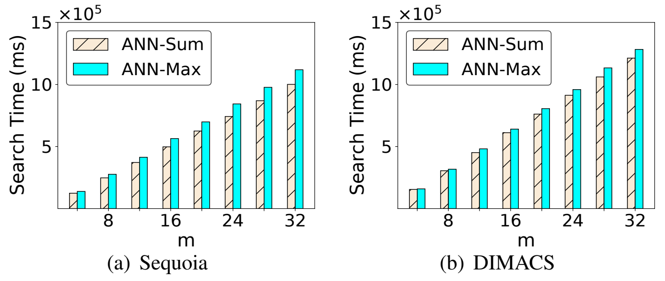


Fig. 7. Search time of our PPAQ scheme varying with the number of query users m. (a) Over the Sequoia dataset; (b) Over the DIMACS dataset.

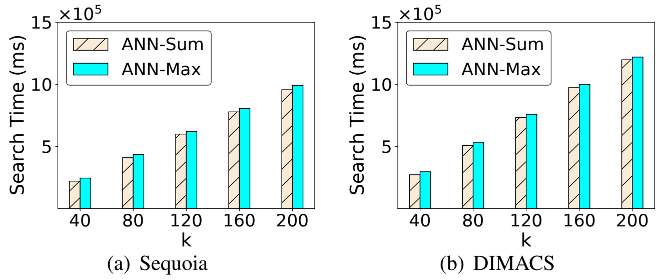


Fig. 8. Search time of our PPAQ scheme varying with the number of vertices k. (a) Over the Sequoia dataset; (b) Over the DIMACS dataset.

B. Performance of Our PPAQ Scheme

In this subsection, we evaluate the search time of our PPAQ scheme over **Sequoia** and **DIMACS** datasets. Since the total distance in both datasets is less than 2^{16}, we set $l = 16$ in the following experiments. In addition, for the security parameters of SHE, we let $k_0 = 8192, k_1 = 20$, and $k_2 = 80$. Note that, since SHE is a leveled homomorphic encryption scheme, and its maximum homomorphic multiplication depth is related to k_0, we can either adopt a bootstrapping protocol [14] between $\mathcal{LSP}$ and u_1 to refresh ciphertexts or enlarge k_0 to support more homomorphic multiplication operations. For simplicity sake, here we set $k_0 = 8192$ and adopt the bootstrapping protocol to refresh ciphertexts.

From Section IV-C, we know that the search time of our PPAQ scheme is related to the number of points of interest POIs n, the number of query users m, and the number of vertices k. Accordingly, Fig. 6, Fig. 7, and Fig. 8 plot the search time of our PPAQ scheme varying with n, m, and k, respectively. In particular, we compare the *sum* function and *max* function of the ANN queries, which are denoted as ANN-Sum and ANN-Max in the following evaluations.

- *Impact of the number of POIs n.* Since there are 150 bars in the **Sequoia** dataset and 100 cafes in the **DIMACS** dataset, we vary n from 30 to 150 in Fig. 6(a) and from 20 to 100 in Fig. 6(b). In both figures, we set $k = 100$ and $m = 16$. We can see that the search time linearly increases with the number of POIs in both figures. The reason is clear since our PPAQ scheme is to select an optimal location from n POIs, and there is no other impacts when we make m and k be fixed. In addition, we can see that the *sum* function requires slightly less search time than the *max* function in our PPAQ scheme. That is because the *max* function needs to calculate sum value $\mathsf{E}(\vec{\tau}_{ti})$ by using the secure addition circuit before obtaining the maximum distance, which entails the extra computational costs compared to the *sum* function.
- *Impact of the number of query users m.* Fig. 7(a) and Fig. 7(b) illustrate the search time varying with m over the **Sequoia** and **DIMACS** datasets, respectively. We randomly generate the locations of query users in the spaces of these two datasets, set the ranges from 4 to 32, and fix $k = 100, n = 50$. Similar to the impact of n, both the figures show a linearly increasing tread. However, the difference of Fig. 7(a) is larger than that of Fig. 7(b). That is because the *max* function needs to add each user's uploaded distance $\mathsf{E}(\theta_t)$ to the shortest path distance $\mathsf{E}(\sigma_{ti})$, i.e., $\mathsf{E}(\tau_{ti}) = \mathsf{E}(\theta_t) + \mathsf{E}(\sigma_{ti})$. Therefore, as m increases, so does the difference of the search time between the *sum* and the *max* functions.
- *Impact of the number of vertices k.* Fig. 8 plots the search time of our PPAQ scheme varying with the number of vertices k from 40 to 200, in which Fig. 8(a) shows the search time over the **Sequoia** dataset, while Fig. 8(b) is for the **DIMACS** dataset. Very differently, the differences of the search time between the *sum* and the *max* functions remain stable in Fig. 8(a) and Fig. 8(b). It is due to that the number of vertices only affects the search time of the matching step (Step 1). When performing the calculating aggregate values step (Step 2), it has nothing to do with the number of vertices.

From Fig. 6, Fig. 7, and Fig. 8, we can see that the search time linearly increases with the corresponding parameter in all figures. In real-world scenarios, the numbers of POIs and vertices are usually fixed. As a result, our PPAQ scheme can achieve linear complexity. Note that our PPAQ scheme is not designed for real-time applications but for plan-ahead systems, in which the group of users launch the ANN queries in advance. Therefore, it is acceptable for an application to select the optimal location with several minutes for ensuring the fully privacy preservation.

Besides, we have observed that the matching step (Step 1) consumes the most of execution time in our PPAQ scheme, and it is significantly affected by the number of vertices. To illustrate it, we list the *percentage* of execution time of Step 1 in Table I, where *percentage* = $\frac{\text{execution time of Step 1}}{\text{execution time of (Step 1 + Step 2 + Step 3)}}$.

TABLE I
PERCENTAGE OF EXECUTION TIME OF STEP 1

Varying k	40	80	120	160	200
ANN-Sum	88.9 %	94.5%	96.2%	97.1%	97.6%
ANN-Max	89.8 %	94.2%	95.9%	96.9%	97.4%

It is worth noting that the number of POIs n and query users m has little impact on the percentage of the execution time of Step 1, i.e., their ratios remain stable (around 95%) when varying n and m. In order to improve the efficiency, $\mathcal{LSP}$ can have the following approaches to optimize Step 1: i) adopt multiple threads; ii) select fewer vertices to calculate shortest path distances. The second approach may affect the accuracy of the selected optimal location, and how to select vertices is an interesting topic. We leave the research on these optimization approaches as future work.

VII. Related work

The ANN query has attracted considerable attention due to its wide range of applications [1]–[8]. Among them, the works in [1]–[4] focus on the Euclidean space, while the others [5]–[8] pay their attention to road networks. Papadias et al. [1], [2] proposed several algorithms to efficiently perform the ANN queries by assuming that the users' locations fit in memory and POIs are indexed by an R-tree. Lian et al. [3] considered the ANN queries over the uncertain database and integrated effective pruning methods to facilitate reducing the search space of probabilistic ANN queries. In 2010, Li et al. [4] designed efficient algorithms for the group enclosing queries, which actually is a variant of the ANN query by calculating the maximum distance instead of minimum distance after applying aggregate functions. In the road network scenario, Yiu et al. [5] first studied the ANN queries in road networks and presented three algorithms to explore the network around the query points until the desired results are discovered. Zhu et al. [6] presented a voronoi-based approach to solve the ANN problem in road networks. The advantage of this work is that it does not need to retrieve the network from disk and just utilizes the look-up tables of the voronoi diagram generated in advance. Yan et al. [7] adopted the ANN query to find the optimal meeting point in road networks, which has the same scenario as our work. Recently, Yao et al. [8] studied the flexible ANN queries in road networks and proposed a series of universal methods to solve this problem, e.g., Dijkstra-based algorithm. However, all of the above works aimed to improve ANN query efficiency and did not consider the privacy issues.

Existing privacy-preserving ANN query schemes mainly concentrated on the Euclidean space [10]–[12]. Although their privacy-preserving techniques can be used in the road network scenario, there are privacy problems with these techniques. In [10], Hashem et al. converted each user's location into a region, and the service provider returns a super-set of the exact query result. This approach not only increases the communication cost but also has privacy issues in terms of users' locations and query results since they are not fully protected. Ashouri et al. [11] adopted a secure multiparty computation technique to make each user participate in the calculation of the query results. However, this approach cannot protect the privacy of query results against the service provider. Recently, Wu et al. [12] presented a novel privacy-preserving scheme for ANN queries by integrating the dummy technique and Paillier encryption. However, this approach has a privacy issue in terms of the users' real locations when a query user launches multiple queries in a fixed location. Note that although Yilmaz et al. [28] also proposed a privacy-preserving optimal location selection scheme, their work is not for the ANN query of a group considered in this paper. Moreover, it mainly focused on the private set intersection (PSI) technique and did not protect the location privacy of the client.

VIII. Conclusion

In this paper, we have proposed a privacy-preserving aggregate query (PPAQ) scheme in road networks, which can preserve the privacy of users' locations, query results, and access patterns. Specifically, we first analyzed the problem of the ANN queries in road networks over plaintexts. Then, we designed secure addition and secure comparison circuits to securely perform addition and comparison operations in a single-server model without leaking operands and results. Based on these secure circuits, we proposed our PPAQ scheme in road networks by designing a secure matching approach. Security analysis indicated that our PPAQ scheme is indeed privacy-preserving, and extensive performance evaluations validated its efficiency. In our future work, we will further exploit its efficiency and practically implement our proposed scheme, such as developing the mobile application for users and ANN query services for service providers.

Acknowledgements

This research was supported in part by NSERC Discovery Grants (04009, 03787, RGPIN-2022-03244).

References

[1] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis, "Group nearest neighbor queries," in *Proceedings. 20th International Conference on Data Engineering*. IEEE, 2004, pp. 301–312.

[2] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui, "Aggregate nearest neighbor queries in spatial databases," *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 2, pp. 529–576, 2005.

[3] X. Lian and L. Chen, "Probabilistic group nearest neighbor queries in uncertain databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 6, pp. 809–824, 2008.

[4] F. Li, B. Yao, and P. Kumar, "Group enclosing queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 10, pp. 1526–1540, 2010.

[5] M. L. Yiu, N. Mamoulis, and D. Papadias, "Aggregate nearest neighbor queries in road networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 820–833, 2005.

[6] L. Zhu, Y. Jing, W. Sun, D. Mao, and P. Liu, "Voronoi-based aggregate nearest neighbor query processing in road networks," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2010, pp. 518–521.

[7] D. Yan, Z. Zhao, and W. Ng, "Efficient algorithms for finding optimal meeting point on road networks," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 968–979, 2011.

[8] B. Yao, Z. Chen, X. Gao, S. Shang, S. Ma, and M. Guo, "Flexible aggregate nearest neighbor queries in road networks," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 761–772.

[9] Q. Zhao, C. Zuo, G. Pellegrino, and L. Zhiqiang, "Geo-locating drivers: A study of sensitive data leakage in ride-hailing services." in *Annual Network and Distributed System Security symposium, February 2019 (NDSS 2019)*, 2019.

[10] T. Hashem, L. Kulik, and R. Zhang, "Privacy preserving group nearest neighbor queries," in *Proceedings of the 13th International Conference on Extending Database Technology*, 2010, pp. 489–500.

[11] M. Ashouri-Talouki, A. Baraani-Dastjerdi, and A. A. Selçuk, "Glp: A cryptographic approach for group location privacy," *Computer Communications*, vol. 35, no. 12, pp. 1527–1533, 2012.

[12] Y. Wu, K. Wang, R. Guo, Z. Zhang, D. Zhao, H. Chen, and C. Li, "Enhanced privacy preserving group nearest neighbor search," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[13] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: ramification, attack and mitigation." in *NDSS*. Citeseer, 2012.

[14] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Efficient and privacy-preserving similarity range query over encrypted time series data," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[15] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *2014 IEEE 30th International Conference on Data Engineering*. IEEE, 2014, pp. 664–675.

[16] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure skyline queries on cloud platform," in *2017 IEEE 33rd international conference on data engineering (ICDE)*. IEEE, 2017, pp. 633–644.

[17] J. Chen, L. Liu, R. Chen, W. Peng, and X. Huang, "Secrec: A privacy-preserving method for the context-aware recommendation system," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[18] J. Brickell and V. Shmatikov, "Privacy-preserving graph algorithms in the semi-honest model," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2005, pp. 236–252.

[19] H. Mahdikhani, R. Lu, Y. Zheng, J. Shao, and A. A. Ghorbani, "Achieving o ($\log^3$n) communication-efficient privacy-preserving range query in fog-based iot," *IEEE Internet of Things Journal*, pp. 5220–5232, 2020.

[20] Y. Guan, R. Lu, Y. Zheng, S. Zhang, J. Shao, and G. Wei, "Toward privacy-preserving cybertwin-based spatio-temporal keyword query for its in 6g era," *IEEE Internet of Things Journal*, 2021.

[21] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[22] S. Jung and S. Pramanik, "An efficient path computation model for hierarchically structured topographical road maps," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 5, pp. 1029–1046, 2002.

[23] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[24] "Sequoia: locations of california," *http://chorochronos.datastories.org/?q=node/58*, 2012.

[25] R. Paulet, M. G. Kaosar, X. Yi, and E. Bertino, "Privacy-preserving and content-protecting location based queries," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 5, pp. 1200–1210, 2013.

[26] "Dimacs: locations of new york city," *http://www.diag.uniroma1.it/challenge9/download.shtml*, 2006.

[27] "Openstreetmap," *https://www.openstreetmap.org*.

[28] E. Yilmaz, H. Ferhatosmanoglu, E. Ayday, and R. C. Aksoy, "Privacy-preserving aggregate queries for optimal location selection," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 2, pp. 329–343, 2017.

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.

Suprio Ray (Member, IEEE) is an Associate Professor with the Faculty of Computer Science, University of New Brunswick, Fredericton, Canada. He received a Ph.D. degree from the Department of Computer Science, University of Toronto, Canada, in 2015. His research interests include big data and database management systems, run-time systems for scalable data science, provenance and privacy issues in big data and query processing on modern hardware. E-mail: sray@unb.ca.

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.

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.

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.