Article

Efficient Aggregate Queries on Location Data with Confidentiality

Da Feng, Fucai Zhou *, Qiang Wang, Qiyu Wu and Bao Li

Software College, Northeastern University, Shenyang 110000, China; dfengneu@gmail.com (D.F.); wangq3635@126.com (Q.W.); kathywuq@gmail.com (Q.W.); libaoneu@gmail.com (B.L.)

* Correspondence: fczhou@mail.neu.edu.cn

Abstract: Location data have great value for facility location selection. Due to the privacy issues of both location data and user identities, a location service provider cannot hand over the private location data to a business or a third party for analysis or reveal the location data for jointly running data analysis with a business. In this paper, we propose a newly constructed PSI filter that can help the two parties privately find the data corresponding to the items in the intersection without any computations and, subsequently, we give the PSI filter generation protocol. We utilize it to construct three types of aggregate protocols for facility location selection with confidentiality. Then we propose a ciphertext matrix compressing method, making one block of cipher contain lots of plaintext data while keeping the homomorphic property valid. This method can efficiently further reduce the computation/communication cost of the query process—the improved query protocol utilizing the ciphertext matrix compressing method is given followed. We show the correctness and privacy of the proposed query protocols. The theoretical analysis of computation/communication overhead shows that our proposed query protocols are efficient both in computation and communication and the experimental results of the efficiency tests show the practicality of the protocols.

Keywords: aggregate query; confidentiality; location data; private set intersection; Paillier cryptosystem

1. Introduction

With the widespread popularity of electronic devices with GPS functions and thanks to advances in information and communication technologies, location-based services (LBS) have developed significantly, and location service providers (LSP), such as Google Maps and FourSquare, already own a very large amount of user location data, which are of great commercial value in facility location selection. LSPs seek to find ways to provide location data aggregate analysis services to other businesses. Businesses would also like to pay for this kind of service for location selection because an appropriate location can bring much more economic interest than an ordinary location. Specifically, for example, a bank plans to open a branch, and it has several candidate addresses. “Which is the best location” is a puzzle for the bank because it does not have the location data of its users, so the bank seeks the help of the LSPs. Even in some public sectors (communal facilities) where economic interests are not the core issue, for businesses that define “value” as being used by more people or making something convenient for more people, a proper location makes the public facilities more valuable. The desire of the LSP and the business to cooperate raises a problem that the LSP can not simply send location data to the business or a third party for analyzing or reveal the location data for jointly running a data analysis query with the business due to privacy issues in both terms of law and ethics. From the above points of view, how to run a two-party privacy-preserving query between an LSP and a business is a meaningful topic.

The selection of a facility location among alternative locations has been widely studied, which is regarded as a multi-criteria decision-making problem, including both quantitative
and qualitative criteria [1]. The location selection query is to choose a location with maximum influence for a new facility [2]. We mainly focus on three types of objectives [3] for location selection—(1) maximizing the number of users attracted (Maximum is not appropriate for some special businesses such as express. Instead, they may pursue sharing workload equally for each facility. For this objective, the RNNC query is still applicable.); (2) minimizing the average distance from all users to their nearest facilities; (3) minimizing the maximum distance from all users to their nearest facilities. The above selection indicators correspond to three types of aggregate queries, respectively, as follows:

1. **Reverse Nearest Neighbor Cardinality (RNNC) query**. The RN query is a very common query for location selection. In our two-party privacy-preserving query scenario, for a (public) set of existing facilities and given query locations, the RN query computes all the users that are closer to each given query location than to any other facilities. Considering the first objective above, our RN query only needs to return the set of the cardinalities of the RN.

2. **Average Distance (AVGD) query**. The average distance represents the mean value of the distances between all users and their nearest facilities. The average distance is valuable information for the client to minimize is to maximize user benefit. The AVGD query result measures the impact of a candidate location of the new facility on all users, which can help a business such as a supermarket to decide where to put a new branch store in order to maximize the benefit to its customers. In short, the minimized average distance corresponds to the most valuable location.

3. **Maximum Distance (MAXD) query**. The maximum distance represents the maximum value of the distances between all users and their nearest facilities. It takes the worst case as a selection indicator, i.e., after running several MAXD queries returning the maximum distances corresponding to different candidate locations, the business can compare those and choose the location which has the smallest one to maximize the benefits to the most “inconvenient” user.

In practice, for multiple alternative locations, a business can run one of these three types of queries according to the chosen criterion for each location, then decide which location is more beneficial to it.

In this paper, we use “server (S)” to replace the LSP that provides services of aggregate queries and use “client (C)” to replace the business that needs location data analyses. Consistently, we use Us/Sc to represent S/C’s user list. We continue to focus on the practical settings described in [3]: (1) We use I = Us ∩ Uc, the intersection of the user lists of S and C, as a substitute for C’s user list Uc because the scale of Uc is so big that the number of items (user identities) in Uc that are out of the intersection is relatively small enough to be ignored; (2) These two user lists Us and Uc are both private, each of which should not be revealed to the other party; (3) The location data of users in the hands of S are private from C.

**OUR CONTRIBUTIONS.** We proposed two-party privacy-preserving aggregate query protocols for three types of queries—RNN Cardinality query, Average Distance query and Maximum Distance query for facility location selection. Compared with the state-of-the-art scheme [3], our protocols are one step closer to practical application in terms of efficiency. Specifically, our main contributions are as follows:

1. We propose a new construction that we call a PSI filter, which can be used by two parties to filter irrelevant data corresponding to items out of the intersection without any computations and without revealing any information of items in their private sets to each other. Specifically, we extract the PSI process of [4] and modify it to a PSI filter generation protocol. The generated filter \( \mathcal{F} \) consists of two parts, in both of which the items are indistinguishable (looks random). These two parts are held by these two parties (S and C), respectively, and they can help to find the valid data in our aggregate query protocols.
2. We propose privacy-preserving aggregate query protocols for three types of queries—
RNN Cardinality query, Average Distance query and Maximum Distance query
by using our PSI filter to find the valid data (corresponding to the items in the
intersection) and using a Paillier cryptosystem to achieve the processing on the
ciphertext. Compared with the state-of-the-art schemes [3], the use of a PSI filter
rather than a superset (for hiding user identities and serving as an index) of user
identity space can bring a new advantage: no need to increase the communication
costs for hiding the user identity lists from each other. Further, it can efficiently reduce
the times of encryption from \( n \) to \( n_s \) by removing many invalid computations where
\( n (\gg n_s) \) is the size of the superset and \( n_s \) is the size of LSP’s user set.

3. We propose a ciphertext matrix compressing (CMC) method that can further reduce
the communication cost (and the computation cost incidentally) of the RNN Car-
dinality query protocol then we give the optimized version of the RNNC query
protocol utilizing CMC. Although our original three types of query processing are
similar to some extent, the communication cost of the RNN cardinality query protocol
is \( O(n_s \cdot k) \times \) that of the other two protocols, where \( k \) is the number of facilities.
Our ciphertext matrix compressing method can reduce the communication cost to
\( O(n_s \cdot [k/\eta]) \), where \( \eta \) is relevant to the security parameter \( \lambda \) and a threshold value \( \alpha \)
(Section 6). What is striking is that our compressing method has a broader scope of
application; more specifically, it has great potential in small-value plaintext scenarios
for compressing the scale of ciphertext while maintaining the homomorphic property.

4. We give the security analysis and performance evaluation of the aggregate query
protocols. First, we prove the security of a PSI filter generation protocol, based on
which we illustrate that the aggregate query protocols are privacy-preserving for
the location data owned by the LSP and user lists of both the LSP and the business.
Finally, we give the theoretical analysis and experimental results of performance.

The remainder of this paper is organized as follows. After reviewing the related
research contents and giving the preliminaries in Section 2, we introduce our system model
and problem definitions in Section 3. In Section 4, we give the definition of a PSI filter and
the generation protocol that we utilized to construct the three types of query protocols
in Section 5. We present a ciphertext matrix compressing method in Section 6 to further
improve the efficiency of the RNNC query protocol. In Section 7, we give the proofs of
correctness and privacy. In Section 8, we give the theoretical analysis and experimental
results for performance evaluation. Finally, the concluding remarks are given in Section 9.

2. Related Work and Preliminaries

2.1. Related Work

2.1.1. Aggregate Query on Location Data

The nearest neighbor (NN) query has received vast attention in the spatial database
research community since its introduction in [5]. We say point A is the nearest neighbor
of B when A is closer to B than any other points under consideration, and naturally, B is
one of the reverse nearest neighbors of A. The RNN query was firstly considered in [6].

In monochromatic RNN [7,8], all points are regarded as in the same class; in contrast, the
dichromatic RNN [2,6,9,10] considers two different classes of points—objects and sites. The
average distance (for \( L_1 \) distance) query was first considered in min-dist optimal location
query[11]. Then, [12] gave a scheme for the \( L_2 \) distance version. Recently, for the TIPS
problem (Trajectory-aware Inconvenience-minimizing Placement of Services), [13] showed
both the MAX-TIPS and AVG-TIPS are NP-hard. Then, a novel query called the reverse
nearest neighborhood (RNH) query was proposed in [14]. Unlike an RNN query, an RNH
query emphasizes a group of users instead of an individual user.

Most of the schemes of aggregate queries on location data assume that data is public
or allowed to be sent to another party. Today, when people attach great importance to data
privacy, these schemes do not meet our requirements.
2.1.2. Privacy-Preserving Aggregate Query

The privacy-preserving database query was first defined in [15]; Ref. [16] then gave a solution. Ref. [17] showed how to execute SQL aggregations over encrypted data. The authors developed an enhanced encrypted data storage model and introduced formal query implementation techniques to translate original aggregation queries to a form that can be directly executed over the encrypted data. Ref. [18] proposed a privacy-preserving range query protocol. Ref. [19] proposed a privacy-aware query processing framework Casper for privacy-preserving NN query. Then, ref. [20] gave efficient protocols for privacy-preserving k-NN searches by using secure multi-party computation technologies. Ref. [21] proposed a simple privacy-preserving protocol PDAS for computing and verifying queries in outsourced databases against malicious adversaries. For the same problem in [20], ref. [22] gave a solution using a Paillier cryptosystem. Recently, ref. [3] proposed two types of solutions for privacy-preserving aggregate query protocols. The main difference between these two types is who (the server or the client) does most of the encryptions and decryptions. No matter who does, the protocols need a superset of size $\tilde{n}$ in order to conceal the identities of users and to serve as an index—the positions of any users are fixed where both the two parties can operate directly on the corresponding positions.

How to cancel the use of a superset and filtrate data without an “index” for aggregate queries on location data while the two parties both have their own private user sets is not solved yet. In general, we use homomorphic encryption and the idea of PSI to design our protocols. Compared with the traditional PSI protocols, our newly constructed PSI filter generation protocol does not find the intersection directly due to privacy issues. What the two parties get are the sets containing computationally indistinguishable labels that were generated in the protocol. In comparison with [3], we cancel the use of a superset through the use of our new primitive PSI filter and further improve the efficiency by compressing multiple data into one block of ciphertext.

2.2. Preliminaries

2.2.1. Private Set Intersection

A private set intersection (PSI) protocol enables two parties, the Sender and the Receiver, holding private sets $X$ and $Y$, respectively, to jointly compute the intersection $X \cap Y$ without revealing any additional information about their respective sets (the conditions may be weakened or changed in different circumstances). In a one-way version of PSI, only the Receiver can learn the intersection $X \cap Y$, while the Sender learns nothing. In a concrete setting called labeled PSI [23] for applications, there is a label $l_i$ for each item $x_i \in X$, and the protocol enables the Receiver to learn the labels corresponding to the items in the intersection, i.e., the Receiver should learn the set $\{(x_i, l_i) : x_i \in Y\}$. A labeled PSI can be seen as a variant of one-way PSI.

We construct our protocols using a PSI filter (Section 4), which is generated by a modified one-way PSI. The generation process is particularly similar to a labeled PSI protocol, and the difference is that the labels do not exist before our PSI processing. In other words, the labels corresponding to the items will be generated in the intermediate process. Moreover, the Receiver does not know the specific correspondence of any pair $(x_i, l_i)$. Note that for reducing the communication/computation cost, the (generated) corresponding labels are important for filtrating data in queries. For consistency, we will use the titles Server ($S$)/Client ($C$) instead of Sender/Receiver, and accordingly, use $U_S/U_C$ instead of $X/Y$, afterward.

2.2.2. Decisional Diffie–Hellman Assumption

The decisional Diffie–Hellman (DDH) assumption is a computational hardness assumption in cyclic groups. It indicates that no efficient algorithm can distinguish between these two distributions $(g, g^a, g^b, g^{ab})$ and $(g, g^a, g^b, g^c)$, which enables one to construct efficient cryptographic systems with strong security properties.
Formally, let $G(\lambda)$ be a cyclic group of order $q$ with secure parameter $\lambda$. For a probabilistic adversary (algorithm) $A$, we define the advantage of $A$ as

$$\text{Adv}[A, G] = |\Pr[A(g, g^a, g^b, g^{ab}) = 1] - \Pr[A(g, g^a, g^b, g^c) = 1]|$$

(1)

where the probability is over a random generator $g$ of $G$, and random $a, b, c \in G$. We say that the decisional Diffie–Hellman assumption holds in group $G$ if for any probabilistic adversary $A$, the advantage $\text{Adv}[A, G]$ is negligible.

2.2.3. Homomorphic Encryption

Homomorphic encryption is a cryptographic primitive that allows computation on ciphertexts without decryption. The homomorphism can be expressed as private key

$$\text{Enc}(m) = g^m \mod n^2$$

where $x \in \mathbb{Z}_n^*$ and can be computed from its own view of the execution. We denote $\text{Enc}(\lambda)$ as $\text{Enc}(\lambda)$, more strictly, $\text{Enc}(\lambda)$ be a probabilistic polynomial-time functionality and let $\text{Enc}(\lambda)$ be a two-party protocol during an execution of $\pi$ on the $i$th party.

We choose a Paillier cryptosystem [24] to achieve the additive homomorphism, i.e., $\text{Enc}(m_1) \cdot \text{Enc}(m_2) = \text{Enc}(m_1 \cdot m_2)$. After selecting the public key $pk = (n, g)$ and the private key $sk = (\lambda, \mu)$, for message $m \in M$, the encryption algorithm computes ciphertext $c = g^m \cdot r^\mu \mod n^2$ where $r$ is randomly selected from $\mathbb{Z}^*_n$. The decryption algorithm recovers the message by computing $m = L(c^\lambda \mod n^2) \cdot \mu \mod n$ where $L(x) = \frac{x - 1}{n}$. The Paillier cryptosystem provides semantic security against chosen-plaintext attacks (IND-CPA) under the decisional composite residuosity assumption (DCRA). In other words, the encryptions of different messages are computationally indistinguishable without knowledge of the private key.

2.2.4. Security Definition

We consider two-party privacy-preserving aggregate query protocols in a semi-honest model, i.e., the adversaries are honest but curious, where the main security issue is to protect the privacy of the two parties. We formally define the security of a two-party protocol by using a simulation paradigm [25,26], which gives us an effective method to prove the security.

Let $f = (f_1, f_2)$ be a probabilistic polynomial-time functionality and let $\pi$ be a two-party protocol for computing $f$. The view of the $i$th party $(i \in \{1, 2\})$ during the execution of $\pi$ on $(x, y)$ and security parameter $\lambda$ is denoted by $\text{view}^\pi_i(x, y, \lambda)$ and equals $(w, r; m_1^i, \ldots, m_l^i)$, where $w \in \{x, y\}$ (depends on the value of $i$), $r^i$ equals the contents of the $i$th party’s internal random tape and $m_l^i$ represents the $i$th message that it received. The output of the $i$th party during an execution of $\pi$ on $(x, y)$ and security parameter $\lambda$ is denoted by $\text{output}^\pi_i(x, y, \lambda)$ and can be computed from its own view of the execution. We denote the joint output of both parties by $\text{output}^\pi(x, y, \lambda) = (\text{output}^\pi_1(x, y, \lambda), \text{output}^\pi_2(x, y, \lambda))$.

**Definition 1.** (Security for Semi-honest Adversaries) Let $f = (f_1, f_2)$ be a functionality. We say that $\pi$ securely computes $f$ in the presence of static semi-honest adversaries if there exist probabilistic polynomial-time algorithms $\text{SIM}_1$ and $\text{SIM}_2$ such that

$$\begin{align*}
\{\text{SIM}_1(1^\lambda, x, f_1(x, y)), f(x, y)\}_{x, y, \lambda} &\equiv \{\text{view}^\pi_1(x, y, \lambda), \text{output}^\pi_1(x, y, \lambda)\}_{x, y, \lambda}, \text{ and} \\
\{\text{SIM}_2(1^\lambda, x, f_2(x, y)), f(x, y)\}_{x, y, \lambda} &\equiv \{\text{view}^\pi_2(x, y, \lambda), \text{output}^\pi_2(x, y, \lambda)\}_{x, y, \lambda},
\end{align*}$$

where $x, y \in \{0, 1\}^*$ such that $|x| = |y|$, and $n \in \mathbb{N}$.

The simulation paradigm is one of the most important paradigms in the definition and design of cryptographic primitives. In a two-party setting, it can ensure one has not
learned anything about the other’s secret by showing that the one could have simulated the entire interaction by itself. In other words, one can gain no further knowledge as the result of interacting with the other beyond what it could have discovered by itself. It gives a formal way to prove the security; the details of the proof are shown in Section 7.

3. Models and Problem Definitions

3.1. System Model

The system model is composed of two parties, as shown in Figure 1: the server $S$ provides location-based services and the client $C$ that is interested in some aggregated data (query results) that are helpful for selecting the optimal facility location.

$S$ has a private user set $U_s = \{v_1, \ldots, v_{n_s}\}$, where $v_i$ is the unique ID of the $i$th user of $S$. Further, $S$ has the knowledge of all the users’ locations $L = \{l_1, \ldots, l_{n_s}\}$, where $l_i$ is the location of $u_i$. ($S$’s own data privacy—how it prevents getting compromised—is not in our consideration. Some techniques can be found in related fields, such as hardware security and database security. We focus on the privacy-preserving interactive protocols between $S$ and $C$.) Actually, $l_i$ can be the expression of a point with respect to any coordinate system, for example, $l_i = (x_i, y_i)$ in Cartesian coordinate system and $l_i = (\text{lng}_i, \text{lat}_i)$ in geographic coordinate system. $C$ has a private user set $U_c = \{v_1, \ldots, v_{n_c}\}$, where $v_i$ is the unique ID of the $i$th user of $C$. In addition, $C$ has $F = \{f_1, \ldots, f_k\}$, a list of locations of $(k - 1)$ existing facilities and one of the candidate locations, which can be public to others.

As described in Figure 1, there are two phases of interactions: the setup phase and the query phase. In the setup phase, $C$ and $S$ interact to do some pre-computations where a PSI filter will be generated on the basis of the two user sets $U_c$ and $U_s$, in order to facilitate the query phase. After finishing the setup phase, when $C$ wants to do a query, it can send a query request with a specific query type to $S$. After receiving the query request, $S$ interacts with $C$ to jointly compute an aggregate query result that will be obtained by $C$ while protecting the privacy of both $C$ and $S$. We discuss the details of privacy requirements and list all involving sensitive information in Section 3.2. There is a good property that a fixed $C$ collaborating with $S$ can process an aggregate query many times with just one setup phase, so the computation and communication costs of the setup phase can be amortized over a number of queries.

3.2. Threat Model

Both $C$ and $S$ are considered as semi-honest (also called honest-but-curious) adversaries [27], as described in Section 2.2.4. That is, they honestly act according to their
prescribed actions in the designed protocols but may try to learn additional information from the encrypted data and all the intermediate messages obtained by themselves, and there is evidently no collusion between \( C \) and \( S \). We say a two-party protocol \( \pi \) is secure in the semi-honest model if no party of the two gains information about the other’s private inputs, other than what can be deduced from the result of the protocol.

In our system model, there is some private information that needs to be hidden as follows.

- \( U_L \): these user identities should not be revealed to \( C \);
- \( U_C \): these user identities should not be revealed to \( S \);
- \( L \): the user location data is private to \( C \);
- \( Q \): the query result is private to \( S \).

Note that the values of \( n_u, n_c \) and \( n_f \) are actually not sensitive information. We do not hide them in our protocol, but just in case, we give a simple way to hide \( n_u \) and \( n_c \) to some degree. Moreover, the location that was finally selected is not private information, while the query result \( Q \) is private to \( S \) because the final location is visible to anyone.

### 3.3. Problem Definitions

Different sorts of aggregate queries can employ the PSI filter proposed in Section 4 to build privacy-preserving query protocols. In our system model, a privacy-preserving aggregate query protocol is a secure two-party computation protocol for functionality \( Q : (U_L, F) \rightarrow (Q, \bot) \), where \((U_L, F)\) is \( C \)'s input and \((U_c, L)\) is \( S \)'s input. After executing the protocol, \( C \) receives the query result \( Q \) corresponding to a specific query type while \( S \) gains nothing. Note that it is a sketchy representation of the functionality \( Q \) ignoring some inputs such as secure parameters and keys, and the input of functionality \( Q \) has not changed from the previous state-of-the-art scheme. For popularity (usefulness and usage frequency) and ease of comparison with the previous literature, we consider three types of queries, the same as in [3]—RNN Cardinality (RNNC) query, Average Distance (AVGD) query, and Maximum Distance (MAXD) query. The formal definitions of the three types of queries are given as follows:

**Definition 2.** (RNNC Query) Given a set of locations of facilities \( F = \{f_1, \ldots, f_k\} \), RNNC query intends to find

\[
\text{RNNC}(F) = \{s_i = |\{u_j | f_i \text{ is NN of } u_j\}| \}_{i \in [k]} \quad \text{where } |\cdot| \text{ means the cardinality of a set. In other words, it intends to find the cardinality of RNN for each facility } f_i. 
\]

For the RNNC query, the query result \( Q = \text{RNNC}(F) \) is a \( k \)-vector (ordered set) of RNN cardinalities.

**Definition 3.** (AVGD Query) Given a set of locations of facilities \( F = \{f_1, \ldots, f_k\} \), AVGD query intends to find

\[
\text{AVGD}(F) = \frac{\sum_{i=1}^{n_c} d(u_i, f_j)}{n_c} \tag{2}
\]

where \( f_j \) is the NN of \( u_i \), and \( d(u_i, f_j) \) means the distance between \( u_i \) and \( f_j \). In other words, it intends to find the average distance of all users and their nearest facilities.

**Definition 4.** (MAXD Query) Given a set of locations of facilities \( F = \{f_1, \ldots, f_k\} \), MAXD query intends to find

\[
\text{MAXD}(F) = \max(d(u_i, f_j)), \quad \text{where } f_j \text{ is the NN of } u_i.\]

In other words, it intends to find the maximum distance between a user \( u_i \in I \) and its nearest facility \( f_j \in F \).

Note that we consider Manhattan distance (\( L_1 \) distance) instead of Euclidean distance (\( L_2 \) distance), because \( L_1 \) distance is more accurate for representing the driving distance in a city road network [28].
3.4. Design Objectives

To meet the anticipated requirements under the aforementioned system model and the threat model, the proposed protocols aim to simultaneously achieve the following objectives:

- **Correctness.** The client should correctly obtain the desired query result if the two parties execute the protocol honestly.
- **Privacy.** The private information \( U_s, U_c, L \) and \( Q \) should be hidden throughout the whole protocol. Although we use \( l \) as a substitute for \( U_c \), the client does not gain any knowledge of \( U_c \); in other words, the client has no idea if an item in \( U_c \) also exactly belongs to \( U_s \), and similarly, the server gains nothing about \( U_c \).
- **Efficiency.** Our protocol should be efficient for both \( S \) and \( C \). That is, both the computation cost and the communication cost should be low to support the aggregate queries with large-scale location datasets and user lists.

4. PSI Filter

**Definition 5.** (PSI filter) For a two-party query protocol \( \pi \), and two datasets \( D_1 \) and \( D_2 \) held by \( S \) and \( C \), respectively, let \( F = (\Theta_1, \Theta_2) \) be a tuple of two formally analogous sets that are held by \( S \) and \( C \), respectively, where \( \Theta_1 = \{\text{label}^S_1, \ldots, \text{label}^S_{|D_1|}\} \), \( \Theta_2 = \{\text{label}^C_1, \ldots, \text{label}^C_{|D_1 \cap D_2|}\} \). \( F \) is said to be a PSI filter if it does not contain any sensitive information and the protocol \( \pi \) can utilize \( F \) to filtrate out irrelevant data corresponding to items out of the intersection \( D_1 \cap D_2 \) with no extra computation.

Our newly constructed PSI filter is a pair of special sets of labels designed to filtrate data, which means helping \( C \) to find and retain the relevant data corresponding to the items (users) in the intersection and discard other irrelevant data without revealing the private identities of users to either \( C \) or \( S \). As described in Section 2.2 the process of generating the PSI filter is, in fact, a modified one-way PSI that is similar to a labeled PSI. Concretely, \( C \) and \( S \) hold private sets \( U_c = \{u_i\}_{i \in [n_c]}, U_s = \{v_i\}_{i \in [n_s]} \), respectively. An originally non-existent label \( \text{label}_i \) that looks random will be generated for each item \( v_i \in U_c \) in the intermediate process, and \( C \) will learn \( \{\text{label}_i : v_i \in U_c\} \), the set of labels corresponding to the items in the intersection as a result. It should be noted that in the PSI filter generating process, \( C \) does not know the specific correspondence of any pair \( (v_i, \text{label}_i) \), and \( S \) does not know which labels \( C \) will retain, so the identities are hidden for both \( C \) and \( S \).

We give the basic idea of the PSI filter generation protocol design in Figure 2. \( C \) and \( S \) hold their private keys \( k_c \) and \( k_s \), respectively. For each item, \( v_i \in U_c \), \( S \) computes \( H(v_i)^{k_s} \) as the distinguishable labels forming the set \( \Theta_1 \) and specifying all the correspondence \( < v_i, H(v_i)^{k_s} > \). To determine if \( u_i = v_j \), \( C \) and \( S \) jointly compute the double-encryption results \( H(u_i)^{k_c k_s} \) and \( H(v_j)^{k_c k_s} \), which is similar to Diffie–Hellman key exchange process. We have that if \( u_i = v_j \) then \( H(u_i)^{k_c k_s} = H(v_j)^{k_c k_s} \), and the converse proposition holds with an overwhelming probability close to 1 while the collision resistance holds. After the intersection operation on the sets \( \{H(u_i)^{k_c k_s}\}_{i \in [n_c]} \) and \( \{H(v_j)^{k_c k_s}\}_{j \in [n_s]} \), while guaranteeing the privacy of identities \( u_i \) and \( v_j \) due to the DDH assumption, \( C \) could get the subset \( \Theta_2 \) of the distinguishable labels in which the labels are corresponding to the items in \( U_c \) one by one. The details of the PSI filter generation protocol PFGen are given in Figure 3.
Common input: a secure parameter $\lambda$, a group $G$ in which the DDH assumption holds, a user identifier space $U$, and a cryptographic hash function $H(x): U \mapsto G$.

$S$’s private input: $k_S \leftarrow \mathcal{S} G$

$C$’s private input: $k_C \leftarrow \mathcal{S} G$

1. For each $H(u_i)^{k_C}$ received from $C$, compute $H(u_i)^{k_S}$, that is, map the user identities to the group $G$ then encrypt it.
2. Shuffle the set $\{H(u_i)^{k_C}\}_{i \in [n]}$ to $\text{Shuf}((H(u_i)^{k_C})_{i \in [n]})$.

3. For each $H(u_i)^{k_C}$ received from $C$, encrypt $label_i$ by computing $\text{label}_i = H(u_i)^{k_S}$.
4. Upon input of $\{H(u_i)^{k_C}\}_{i \in [n]}$ and $\{\{\text{label}_i, H(u_i)^{k_S}\}\}_{i \in [n]}$, perform a intersection operation on $\{H(u_i)^{k_C}\}_{i \in [n]}$ and $\{\text{label}_i\}_{i \in [n]}$.
5. From $j = 1$ to $n$, and integer $i$ increasing from 1, set $\text{label}_i = \text{label}_i$, when $\{\text{label}_i\}_{i \in [n]}$: $\{H(u_i)^{k_C}\}_{i \in [n]}$.
6. Hold $\Theta_2 = \{\text{label}_i\}_{i \in [n]}$.

Thus far, the PSI filter $\mathcal{F} = (\Theta_1, \Theta_2)$ has been generated and $\Theta_1, \Theta_2$ are held separately by $S$ and $C$. Note that $\Theta_1$ is required to be an ordered set while $\Theta_2$ is an ordinary set. The generation protocol PFGen will be used as a subprotocol in the setup phase of aggregate query protocols. We will formally state and prove the security of the PFGen protocol in Section 7, and we just simply indicate here that the PFGen protocol is secure against semi-honest adversaries, so it can be used to construct privacy-preserving query protocols. Note that the difference from a traditional PSI protocol is that both the two parties have no idea if an item is exactly in the intersection $I$. All they can get are the labels that look random.
5. Aggregate Query Protocols

In this section, we use the PFGen protocol proposed in Section 4 to build privacy-preserving query protocols for different types of aggregate queries. In these protocols, the client \( C \) and the server \( S \) execute the PFGen protocol to generate a PSI filter in the setup phase. In the query phase, \( C \) finds out the encrypted data related to the users in the intersection from all candidate structured data received from \( S \) by comparing the labels in \( \Theta_1 \) with those in \( \Theta_2 \).

5.1. RNNC Query

Setup:

Step 1: \( S \) and \( C \) execute the PFGen protocol with secure parameter \( \lambda \). The output is the PSI filter \( F = (\Theta_1, \Theta_2) \), where \( \Theta_1 \) and \( \Theta_2 \) are held by \( S \) and \( C \), respectively.

Step 2: \( S \) takes the same secure parameter \( \lambda \) to generate the key pair \((sk, pk)\) of the Paillier cryptosystem, then sends \( pk \) to \( C \).

Query:

Step 1: \( C \) sends the query request \( F = \{f_1, \ldots, f_k\} \) with the type of query “RNNC” to \( S \).

Step 2: After receiving the facility locations, \( S \) computes \( d(f_i, l_j) \) for \( i \in [k] \) and \( j \in [n_s] \), which means the distances between \( f_i \in F \) and \( u_j \in U_i \). \( S \) determines each user’s nearest neighbor.

Step 3: \( S \) initializes a \( n_s \times k \) zero matrix \( M \), then sets \( m_{ij} = 1 \) for all \( i, j \) satisfying \( f_j \) is \( u_i \)'s nearest neighbor. \( S \) computes \( M' \), the ciphertext of \( M \) under the public key \( pk \), i.e., computes each \( m'_{ij} = \text{Enc}(pk, m_{ij}) \).

Step 4: \( S \) augments the ciphertext matrix \( M' \) with \( \Theta_1 \), and gets a \( n_s \times (k + 1) \) matrix \( N' = \Theta_1[M'] \) as a result (treating the ordered set \( \Theta_1 \) as a column vector, the same below), then sends \( N' \) to \( C \).

Step 5: For each row of \( N' \), \( C \) retains the row if the first element of the row is also in \( \Theta_2 \). Otherwise, it discards the row. Then, \( C \) discards the first column to obtain a \( n_1 \times k \) matrix \( Y' \).

Step 6: \( C \) multiplies the elements in each column of \( Y' \) and gets the vector \( s' = [s_{1}', \ldots, s_{k}'] \). \( C \) picks a random vector \( r = [r_1, \ldots, r_k] \) where \( r_i \in \mathbb{Z}_n \), computes \( r' = [r_{1}', \ldots, r_{k}'] \) where \( r_{i}' = \text{Enc}(r_i) \), then computes \( t' = [t_{1}', \ldots, t_{k}'] \) where \( t_{i}' = s_{i}'r_{i}' \). \( C \) sends \( t' \) to \( S \).

Step 7: \( S \) decrypts \( t' \) by computing \( t_i = \text{Dec}(sk, t_{i}') \), then returns \( t = [t_1, \ldots, t_k] \) to \( C \).

Step 8: \( C \) computes \( s = t - r \) and lets \( q_i = s_i \) to obtain \( Q = [q_1, \ldots, q_k] \) as the query result.

The key parts are (1) \( C / S \) hides its user set from the other one by using the PSI filter, which also reduces the computational cost and communication cost, (2) \( S \) hides the user’s location information by using additively homomorphic encryption, (3) \( C \) picks a random vector \( r \) to mask \( s \) (on ciphertext) to hide the query result from \( S \). Clearly, \( Q = [q_1, \ldots, q_k] \) is the desired query result where \( q_i \) is equal to the RNN cardinality of \( f_i \). We will give the other two aggregate query protocols with similar ideas, and a detailed explanation of correctness and privacy are given in Section 7.

In practice, it is not necessary to generate a key pair of Paillier cryptosystems every time a client comes in. The server can set several key pairs \( \{sk_i, pk_i\} \) corresponding to different security levels (parameters \( \lambda_i \)) and make all \( \{\lambda_i, pk_i\} \) public. When a new client comes in and needs to set up the server, it can pick an appropriate public key and just tell the server which one it chose. The same goes for the next two query protocols in Sections 5.2 and 5.3.

5.2. AVGD Query

Setup:

The setup phase is the same as in Section 5.1.

Query:

Step 1: \( C \) sends the query request \( F = \{f_1, \ldots, f_k\} \) with the type of query “AVGD” to \( S \).
Step 2: After receiving the facility locations, $S$ computes $d(f_i, l_j)$ for $i \in [k]$ and $j \in [n_s]$, which means the distances between $f_i \in F$ and $u_j \in U_s$. $S$ determines each user’s nearest neighbor.

Step 3: $S$ initializes a column vector $D = [d_1, \ldots, d_{n_s}]^T$ where $d_i$ is the distance between $u_i$ and its nearest neighbor, then computes $D'$, the ciphertext of $D$ under the public key $pk$, i.e., computes each $d'_i = \text{Enc}(pk, d_i)$.

Step 4: $S$ augments ciphertext matrix $D'$ with $\Theta_1$, gets $n_s \times 2$ matrix $N' = \Theta_1[D']$, then sends $N'$ to $C$.

Step 5: For each row of $N'$, $C$ retains the row if the first element of the row is also in $\Theta_2$, otherwise, it discards the row. Then, $C$ discards the first column to obtain a $n_1$-dimensional column vector $Y' = [y'_{1}, \ldots, y'_{n_1}]^T$ and saves $n_1$.

Step 6: $C$ picks a random number $r \in \mathbb{Z}_n$ and lets $r' = \text{Enc}(r)$. $C$ computes $s' = \Gamma_{i=1}^{n_1} y'_i$ and $t' = s' r'$, then sends $t'$ to $S$.

Step 7: $S$ decrypts $t'$ by computing $t = \text{Dec}(sk, t')$, then returns $t$ to $C$.

Step 8: $C$ computes $s = t - r$ and $Q = s / n_1$ to obtain the query result $Q$.

5.3. MAXD Query

Setup:

The setup phase is the same as in Section 5.1.

Query:

Step 1: $C$ sends the query request $F = \{f_1, \ldots, f_k\}$ with the type of query “MAXD” to $S$.

Step 2: After receiving the facility locations, $S$ computes $d(f_i, l_j)$ for $i \in [k]$ and $j \in [n_s]$, which means the distances between $f_i \in F$ and $u_j \in U_s$. $S$ determines each user’s nearest neighbor.

Step 3: $S$ initializes a column vector $D = [d_1, \ldots, d_{n_s}]^T$ where $d_i$ is the distance between $u_i$ and its nearest neighbor, sorts all $d_i \in D$ to obtain $\hat{D} = [d_1, \ldots, d_{n_s}]^T$ where $d_{\tau(1)} \geq d_{\tau(i)} \geq \ldots \geq d_{\tau(n_s)}$, then sets $\Theta_1 = [\text{label}^S_{\tau(1)}, \ldots, \text{label}^S_{\tau(n_s)}]^T$.

Step 4: $S$ computes $\hat{D'} = [d'_{\tau(1)}, \ldots, d'_{\tau(n_s)}]^T$ where $d'_{\tau(i)} = \text{Enc}(pk, d_{\tau(i)})$, then sends $N' = \Theta_1[\hat{D'}]$ to $C$.

Step 5: From the top to bottom of $N'$, $C$ finds out the first row in which the $\text{label}^S_{\tau(i)}$ is in $\Theta_2$ (suppose it is the $j$th row). $C$ holds $d'_{\tau(j)}$ and discards all other elements.

Step 6: $C$ picks a random number $r \in \mathbb{Z}_n$, computes $r' = \text{Enc}(r)$, then computes $t' = d'_{\tau(j)} r'$. $C$ sends $t'$ to $S$.

Step 7: $S$ decrypts $t'$ by computing $t = \text{Dec}(sk, t')$, then returns $t$ to $C$.

Step 8: $C$ computes $Q = t - r$ (in fact $Q = d_{\tau(j)}$) as the query result $Q$.

5.4. Application Examples

In order to make it more clear how to apply our privacy-preserving aggregate query protocols to the actual scenes, we give the application examples for each of the three types of queries as follows.

- A bank wants to open a branch.
  The bank may hope that the new branch serves more people, so it can carry out the RNNNC query by interacting with the LSP. Then the bank can choose the best one from several alternative locations to open a branch.

- An express (delivery) company wants to build a new distribution station.
  The company probably hopes to minimize the total delivery distance (count the distance of all distribution stations), so it can carry out the AVGD query by interacting with the LSP. Then the company can choose the best one from several alternative locations to build a new distribution station.

- Choosing a site for a non-profit hospital to be established.
Choosing a site for a hospital may consider the worst case—it hopes that the furthest one of the all potential patients can arrive at the hospital in a shorter time, i.e., the goal is to minimize the maximum distance between all potential patients and his/her nearest hospital, so it can carry out the MAXD query by interacting with the LSP. Then the hospital can be established in the best location chosen from several alternative locations.

6. Ciphertext Matrix Compressing

Although the processes of the three query protocols in Section 5 have similar design ideas, the communication costs are not exactly of the same magnitude in different query types. Concretely, the communication costs of AVGD query and MAXD query are both roughly $O(n_k)$, but that of RNNC query is roughly $O(n_k k)$. The essential cause is that the dimension of matrix $M$ initialized in Step 3 of the RNNC query must be sufficient to contain the relationship information between all $n_k$ users and all $k$ facilities when each row of the $n_k$-dimension vector $D$ in Step 3 of AVGD/MAXD query corresponds to only one (the nearest) facility. When the scale of the facility set $F$ is large, the linear relationship between communication cost and the number of facilities will be a challenge. To address this issue, we compress the ciphertext for this particular scenario (the plaintext space actually used is much less than $Z_n$). Properly speaking, under the same security parameter, we make the same size ciphertext contain more plaintext information, which will lead to the reduction of the ciphertext matrix dimension.

In homomorphic cryptosystems satisfying CPA-security, the size of the ciphertext is bigger than that of plaintext because there must be enough randomization to achieve a “non-determination” property. In a Paillier cryptosystem, the encryption function is essentially a map $f : Z_n \times Z_n^* \rightarrow Z_{n^2}$, where $Z_n$ is the plaintext space, $Z_{n^2}$ is the ciphertext space, and the selected random number $r \in Z_n^*$ ensures the “non-determination” property. Clearly, the size of the ciphertext is equal to $2 \times$ the size of the plaintext. It seems unrealistic to directly compress the ciphertext space under the premise of ensuring security: (1) It is impossible to compress $Z_{n^2}$ while keeping $Z_n$ and $Z_{n^2}^*$ fixed because $f$ is bijective, (2) Ensuring a constant security level means that $Z_{n^2}^*$ can not be compressed, (3) It is hard to compress both $Z_n$ and $Z_{n^2}^*$ simultaneously because such behavior will easily break the ingenious bijection structure—the underlying theoretical basis of the Paillier cryptosystem. The generalized Paillier cryptosystem [29] can efficiently reduce the ciphertext expansion factor to $\frac{4\lambda s}{s^k - 1}$ where $s$ is proportional to the block size of the ciphertext, but under the same secure parameter $\lambda$, picking a large $s$ means concurrently expanding both plaintext space and ciphertext space. Concisely, in a generalized Paillier cryptosystem, a smaller ciphertext expansion factor leads to a larger ciphertext block size, which is not suitable for our scenario. A method [30] similar to ours is an efficient algorithm for fast batch summation using homomorphism on generalized Paillier cryptosystems, but we do not want to use the big ciphertext block because the number of facilities $k$ may be relatively too small to waste a lot of space and bring unnecessary computational overhead.

We observe that even encrypting a small number needs a complete block in a Paillier cryptosystem. For a small number $m \ll n$, the plaintext space $Z_n$ is far from being efficiently used, i.e., $\log_2 \frac{n}{m}$ bits are wasted. In the RNNC query protocol, the elements in matrix $M'$ are either $\text{Enc}(0)$ or $\text{Enc}(1)$; even in steps 6 and 8 for each $s_i' = \text{Enc}(s_i)$, $s_i$ is relatively very small compared with $n$.

Our solution is putting multiple data (small numbers) in one plaintext while keeping the ciphertext able to be calculated using the homomorphism property from the vertical direction. Supporting the plaintext space is $Z_n$ and each single data in plaintext throughout the process (considering the intermediate calculation products after decrypting) is no more than $\alpha$ bits, i.e., no overflows of plaintext occur after the sum operations by the homomorphic property in the intermediate process. The process of ciphertext matrix compressing is given in Figure 4.
Suppose we have already generated the parameters of the Paillier cryptosystem over the security parameter $\lambda$. First, we choose the parameter $\alpha$ such that no overflows occur for plaintext throughout the whole query protocol and naturally let $\eta = \log_2 \frac{n}{\alpha}$, where $\eta$ means that any block can contain up to $\eta$ data ($m_{i1}, \ldots, m_{i\eta}$).

Then, we perform the following process.

1. For $1 \leq i \leq \mu$, put $\eta$ data in one plaintext block: for $1 \leq j \leq \eta$, pad each $m_{ij}$ = 1 or 0 to $\alpha$ bits with “0” then attach them end to end—a plaintext block has been constructed.

2. For $1 \leq i \leq \mu$, compute $c_i = \text{Enc}(pk, \text{block}_i)$.

3. Compute $P = \prod_{i=1}^{\mu} c_i$.

4. Compute $\hat{s} = \text{Dec}(sk, P)$, then donate $\hat{s}$ as $s_1 || s_2 || \ldots || s_\eta$ where $s_i$ is a $\alpha$-bit binary string.

Thus far, we have encrypted $\mu \cdot \eta$ data using just $\mu$ plaintext blocks. Meanwhile, we have gotten $s_1, \ldots, s_\eta$ by running $(\mu - 1)$ rather than $(\mu - 1) \cdot \eta$ multiplications on the ciphertext. In order to express more clearly, we give a toy example here and interpret why the homomorphic property can still be used.

A toy example. Let $\lambda = 1024$, $\mu = 3$, $\alpha = 256$ and $\eta = \frac{\lambda}{\alpha} = 4$. Take the matrix as input

$$M = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$  

After padding and attaching procedures, we get three blocks as follows.

$$\begin{align*}
\text{block}_1: & \quad 0\ldots 00|0\ldots 00|0\ldots 01|0\ldots 01 = 2^{1a} + 2^{0a} \\
\text{block}_2: & \quad 0\ldots 00|0\ldots 01|0\ldots 00|0\ldots 01 = 2^{2a} + 2^{0a} \\
\text{block}_3: & \quad 0\ldots 01|0\ldots 01|0\ldots 01|0\ldots 01 = 2^{3a} + 2^{2a} + 2^{1a} + 2^{0a}
\end{align*} \quad (3)$$

After (2)–(4), it is easy to draw that

$$P = \prod_{i=1}^{\mu} c_i = \text{Enc}(3 \sum_{i=1}^{3} \text{block}_i) = 1 \times 2^{3a} + 2 \times 2^{2a} + 2 \times 2^{1a} + 3 \times 2^{0a} = 0\ldots 01|0\ldots 01|0\ldots 01|0\ldots 01.$$
Therefore, we get $s_1 = 1, s_2 = 2, s_3 = 2$ and $s_4 = 3$. It is easy to verify that $s_j = \sum_{i=1}^{3} m_{ij}$ for any $j \in \{1, 2, 3, 4\}$.

For our column-wise summation problem of a small-value $\mu \times \eta$ matrix, the proposed ciphertext matrix compressing combines (pads with 0 then links) multiple plaintexts in a row before encrypting. The number of ciphertext blocks can be reduced from $\mu \times \eta$ to $\mu$, and the number of encryption/decryption operations is reduced from $\mu \times \eta$ to $\mu$. At the same time, the homomorphic property is kept valid. Therefore, we can reform the RNNC query protocol by utilizing the ciphertext matrix compressing method. Here is the variation of the steps of the query process.

**An Improved Protocol RNNC-CMC:**

**Step 3:** $S$ initializes an $n_s \times k$ zero matrix $M$, then sets $m_{ij} = 1$ for all $i, j$ satisfying $f_j$ is $u_i$’s nearest neighbor. $S$ compresses the matrix $M$ to $n_s \times \lceil k/\eta \rceil$ matrix $\tilde{M}$. (When $k$ is not an integer multiple of $\eta$, the way to deal with the extra space is simple: padding with “0” in Step 3.) $S$ computes $\tilde{M}'$, the ciphertext of $\tilde{M}$ under the public key $pk$, i.e., computes each $\tilde{m}_{ij}' = \text{Enc}(pk, \tilde{m}_{ij})$.

**Step 4:** $S$ augments the ciphertext matrix $\tilde{M}'$ with $\Theta_1$, and gets $n_s \times (\lceil k/\eta \rceil + 1)$ matrix $\tilde{N}' = \Theta_1 | \tilde{M}'$ as a result, then sends $\tilde{N}'$ to $C$.

**Step 5:** For each row of $\tilde{N}'$, $C$ retains the row if the first element of the row is also in $\Theta_2$. Otherwise, it discards the row. Then, $C$ discards the first column to obtain a $n_1 \times \lceil k/\eta \rceil$ matrix $\tilde{Y}'$.

**Step 6:** $C$ multiplies the elements in each column of $\tilde{Y}'$ and gets the vector $s' = [s'_1, \ldots, s'_{\lceil k/\eta \rceil}]$. $C$ picks a random vector $r = [r_1, \ldots, r_{\lceil k/\eta \rceil}]$, computes $r' = [r'_1, \ldots, r'_{\lceil k/\eta \rceil}]$ where $r'_i = \text{Enc}(r_i)$, then computes $\tilde{I}' = [I'_1, \ldots, I'_{\lceil k/\eta \rceil}]$ where $I'_i = s'_i r'_i$. $C$ sends $\tilde{I}'$ to $S$.

**Step 7:** $S$ decrypts $t'$ by computing $t_i = \text{Dec}(sk, I'_i)$, then returns $\tilde{I} = [I'_1, \ldots, I'_{\lceil k/\eta \rceil}]$ to $C$.

**Step 8:** $C$ computes $s = \tilde{I} - r$, then separates the vector $[s_1, \ldots, s_{\lceil k/\eta \rceil}]$ to $[s_1, \ldots, s_k]$—like (4) in CMC for each $\tilde{s}_i$ (ignoring separated $s_i$ for $i > k$). Finally, $C$ sets $q_i = s_i$ to obtain $Q = [q_1, \ldots, q_k]$ as the query result.

The changes of the reformed protocol RNNC-CMC are intuitive: (1) compressing the scale of matrix $M$ from $n_s \times k$ to $n_s \times \lceil k/\eta \rceil$, which can efficiently reduce the communication cost; (2) one homomorphic operation (multiplication) on ciphertext by column is equivalent to $\eta$ operations in the original RNNC query protocol. The details of the comparison are given in Section 8.

**7. Security Analysis**

In this section, we first derive the correctness of our aggregate protocols. Then, we consider the privacy of the PSI filter generation protocol PFGen, give an optional handling for stricter privacy protection as a supplement and finally discuss the privacy of our aggregate query protocols and two types of external attackers.

**7.1. Correctness**

We simply derive the correctness of three types of queries.

For a component $q_i \in Q$ in RNNC query, the following equation holds:

$$q_i = t_i - r_i = \text{Dec}(sk, t'_i) - r_i = \text{Dec}(sk, s'_i r'_i) - r_i = \text{Dec}(sk, \prod_{j=1}^{\eta_1} y_{ji} \cdot r_{ji}) - r_i = \sum_{j=1}^{\eta_1} y_{ji} r_{ji} - r_i = \sum_{u_i \in l} m_{ij}.$$  (5)
For the query result $Q$ in AVGD query,
\[
\frac{s}{n_1} = \frac{t - r}{n_1} = \text{Dec}(sk, t') - r = \text{Dec}(sk, s') - r
\]
\[
= \frac{\text{Dec}(sk, \prod_{i=1}^{n_1} y'_i) - r}{n_1} = \frac{\sum_{i=1}^{n_1} y_i + r - r}{n_1} = \frac{\sum_{i=1}^{n_1} d_i}{n_1}.
\]

For the query result $Q$ in MAXD query, due to the monotonicity of $\mathbf{D}$, i.e., $d_{\tau(i)} \geq d_{\tau(i+1)}$, and Step 5, we only need to demonstrate that $t - r = d_{\tau(j)}$ holds as follows:
\[
t - r = \text{Dec}(sk, t') - r = \text{Dec}(sk, d_{\tau(j)}) - r
\]
\[
= d_{\tau(j)} + r - r = d_{\tau(j)}.
\]

7.2. Privacy

As for the privacy of our aggregate query protocols, we first show our PSI filter generation protocol PFGen proposed in Section 4 would not reveal any information about the private data and give optional handling for stricter privacy protection (treating $n_s$ and $n_c$ as privacy). Based on this, we illustrate that the proposed aggregate query protocols are privacy-preserving.

The following lemma is given to illustrate the security of the PFGen protocol in a semi-honest model, and we give the proof in the simulation paradigm.

**Lemma 1.** There exist PPT simulators $\text{SIM}_1$ and $\text{SIM}_2$, s.t. for all security parameters $\lambda$ and inputs $\{u_i\}_{i \in [n]}, \{v_j\}_{j \in [n]}$,
\[
\text{REAL}^\lambda(\{u_i\}_{i \in [n]}, \{v_j\}_{j \in [n]}) \equiv \text{SIM}_1(1^\lambda, \{u_i\}_{i \in [n]}, n_s, n_l),
\]
and
\[
\text{REAL}^\lambda(\{u_i\}_{i \in [n]}, \{v_j\}_{j \in [n]}) \equiv \text{SIM}_2(1^\lambda, \{v_j\}_{j \in [n]}, n_c, n_l).
\]

**Proof.** We give the simulators $\text{SIM}_c$ and $\text{SIM}_s$ in Algorithms 1 and 2 as follows.

**Algorithm 1 SIM$_c$: The simulator for Client**

**Input:** $(1^\lambda, \{u_i\}_{i \in [n]}, n_s, |I|)$

**Output:** $\text{view}_c = \text{SIM}_c(1^\lambda, \{u_i\}_{i \in [n]}, n_s, |I|)$

1. Generate key $k_c \in \mathbb{G}$
2. Honestly generate and send $\text{Shuf}(\{H(u_i)k_c\}_{i \in [n]})$ as Client’s message in Step 2.
3. Create a dummy set $D_1 = \{g_i\}_{i \in [n]}$, where each $g_i$ is randomly selected from $\mathbb{G}$.
4. Compute then send $\text{Shuf}(\{g_i^k\}_{i \in [n]})$ as the message received from Server. (regard $g_i$ as a dummy $H(u_i)k_c$)
5. Create a dummy set $D_2 = \{h_i\}_{i \in [n]}$, where $h_i = g_i$ for $i \in [|I|]$ and each $h_i$ for $i \in [||I|+1, n_s]$ is randomly selected from $\mathbb{G}$.
6. Send $\text{Shuf}(\{h_i\}_{i \in [n]})$ as the message received from Server. (regard $h_i$ as a dummy label$_i = H(v_i)k_c$)
7. Honestly generate Client’s message in Step 3 using $D_2 = \{h_i\}_{i \in [n]}$.
8. Honestly execute Step 4 using $\{g_i^k\}_{i \in [n]}$ and $\{h_i, h_i^k\}_{i \in [n]}$
9. Output Client’s view $\text{view}_c$. 

**Algorithm 2 SIM$_s$: The simulator for Server**

**Input:** $(1^\lambda, \{v_j\}_{j \in [n]}, n_c, |I|)$

**Output:** $\text{view}_s = \text{SIM}_s(1^\lambda, \{v_j\}_{j \in [n]}, n_c, |I|)$

1. Generate key $k_s \in \mathbb{G}$
2. Honestly generate and send $\text{Shuf}(\{H(v_j)^k\}_{j \in [n]})$ as Server’s message in Step 2.
3. Create a dummy set $D_1 = \{g_i\}_{i \in [n]}$, where each $g_i$ is randomly selected from $\mathbb{G}$.
4. Compute then send $\text{Shuf}(\{g_i^k\}_{i \in [n]})$ as the message received from Client. (regard $g_i$ as a dummy $H(v_j)^k$)
5. Create a dummy set $D_2 = \{h_i\}_{i \in [n]}$, where $h_i = g_i$ for $i \in [|I|]$ and each $h_i$ for $i \in [||I|+1, n_c]$ is randomly selected from $\mathbb{G}$.
6. Send $\text{Shuf}(\{h_i\}_{i \in [n]})$ as the message received from Client. (regard $h_i$ as a dummy label$_i = H(v_i)^k$)
7. Honestly generate Server’s message in Step 3 using $D_2 = \{h_i\}_{i \in [n]}$.
8. Honestly execute Step 4 using $\{g_i^k\}_{i \in [n]}$ and $\{h_i, h_i^k\}_{i \in [n]}$
9. Output Server’s view $\text{view}_s$. 

**Proof.**
The simulation process of $SIM_s$ is the same as $C$’s Step 1–4 of the $PFGen$ protocol as long as we regard $\{g_i\}_{i \in [n_s]}$ and $\{h_i\}_{i \in [n_s]}$ as the dummy $\{H(u_i)^k\}_{i \in [n_s]}$ and the dummy $\{label_i = H(v_i)^k\}_{i \in [n_s]}$: Similarly, the simulation process of $SIM_s$ is the same as $S$’s Step 1–4 of the $PFGen$ protocol when we regard $\{g_i'\}_{i \in [n_c]}$ as the dummy $\{H(u_i)^k\}_{i \in [n_c]}$.

We can say that for all security parameters $\lambda$ and inputs $\{u_i\}_{i \in [n_s]} \in I$ and $\{v_j\}_{j \in [n_c]} \equiv SIM_s(1^\lambda, \{u_i\}_{i \in [n_s]}, n_s, n)$, and

\[
\text{REAL}^\lambda(\{u_i\}_{i \in [n_s]}, \{v_j\}_{j \in [n_c]}) \equiv SIM_s(1^\lambda, \{u_i\}_{i \in [n_s]}, n_s, n),
\]

from the DDH assumption. That is, $C$ could not distinguish $\{\{g_i^k\}_{i \in [n_s]}, \{h_i^k\}_{i \in [n_s]}\}$ and $\{\{H(u_i)^k\}_{i \in [n_s]}\}$; and $S$ could not distinguish $\{\{g_i^k\}_{i \in [n_c]}\}$ and $\{\{H(u_i)^k\}_{i \in [n_c]}\}$. □

From Lemma 1, we know that throughout the $PFGen$ protocol, $C$ can know nothing more than $n_s$ and $n_c$; $S$ can know nothing more than $n_c$ and $n_t$. These values are, in fact, not so private for $S$ and $C$. When $S$ and $C$ want to further enhance the privacy, i.e., to hide the cardinalities $n_s$ and $n_c$ from each other, $S/C$ can simply add some dummy users to $U_s/U_c$. This can efficiently conceal $n_s$ and $n_c$ to some extent, and when a new dummy user is added, after being hashed to group $G$, it may collide with a real user with a probability of $\frac{1}{|\mathbb{Z}_q|}$. It can be said that $n_t$ is not a privacy issue when $n_s$ and $n_c$ have been concealed.

In the query process, all the messages $C$ received are $\mathbf{N}'$ and $\mathbf{t}$ (or $\mathbf{t}'$) where $\mathbf{t}$ (or $\mathbf{t}'$) is not a private message to $C$; and the only message $S$ received is $\mathbf{t}'$ (or $\mathbf{t}$'). In the RNNC query protocol, $\mathbf{N}' = \Theta_1^\mathbf{M}'$ is a ciphertext matrix augmented with a vector of randomized labels from where $C$ can not learn any plaintext information without the private key $sk$, and $\mathbf{t}' = [\mathbf{t}_1', \ldots, \mathbf{t}_k']$ is a ciphertext vector where each component $\mathbf{t}_i' = s_i^\mathbf{t}_i' = \text{Enc}(pk, q_i + r_i)$—even a curious $S$ decrypts $\text{Enc}(pk, q_i + r_i)$ by using $sk$, a masked query result $q_i + r_i$ would not reveal any information about $q_i$. In the AVGD query protocol, $\mathbf{N}' = \Theta_1^\mathbf{D}'$ is also ciphertext from where $C$ can not learn anything, and $\mathbf{t}' = s'\mathbf{t}' = \text{Enc}(Q \cdot n_t + r)$ is a masking version of $\text{Enc}(Q \cdot n_t)$. Similarly, $\text{MAXD}$ query, $\text{N}'$ reveals nothing to $C$, and $C$ computes $\mathbf{t}' = d_{i(j)}^t \mathbf{t}'$ to ensure the privacy of the query result $Q = d_{i(j)}(Q \cdot n_t + r)$. Furthermore, our ciphertext matrix compressing method does not sacrifice or reduce the security level. Because of the indistinguishability of the ciphertext in the Paillier cryptosystem, for the same security parameter $\lambda$, the RNNC-CMC query protocol is strictly as secure as the original RNNC query protocol in Section 5.

In addition, traditional protocols with similar ideas to Diffie–Hellman key exchange may suffer from two attacks—statistical attack and man-in-the-middle attack. We give the discussions on these two types of attacks as follows.
Statistical attack. In the PFGen protocol, each user will obtain a fixed label, and the label corresponding to each user is always the same. Intuitively, it seems that a statistical attack can get the frequency of the users during the queries. In fact, when the client and the server have jointly executed the PFGen protocol, the generated PSI filter (two sets of fixed labels) has also fixed the user sets with unknown correspondences, which means the client with a modified user set $U'_c$ could not successfully execute the query protocols—even the client keeps the set $\text{Shuf}(\{\text{label}_i\}_{i \in [n]})$, the set $\Theta_2$ could not be modified appropriately based only on $U'_c$ without the correspondences.

Man-in-the-middle attack. There may be a malicious man intercepting in the middle between $S$ and $C$, sending and receiving data by using another key $k_e$. There is a simple way to deal with this attack: using the signature technique to ensure the authenticity of the source. Specifically, the Sender (whether $S$ or $C$) signs all messages as a whole during each interaction. Even without using the signature technique, it is difficult for the attacker to obtain useful information. When the attacker has jointly performed the PFGen protocol with $S$, the computationally indistinguishable labels are meaningless for the attacker, and they can not obtain any private information from $S$ during some forged queries, according to Lemma 1. When the attacker pretending to be a server has jointly performed the PFGen protocol with $C$, the difference from the previous is that $C$ may obtain a wrong result (non-optimal location) from the message received from the attacker during some queries. It can be said that the attacker may consume a lot of energy and can not obtain the information they want to know. Again, to prevent “man-in-the-middle attacker returning wrong query result”, the signature technique is sufficient.

8. Performance Evaluation
8.1. Theoretical Analysis
We first analyze the computational complexity for each party in the four proposed protocols and compare them with those of the state-of-the-art schemes in [3]—the server-based protocols RNNC/S, AVGD/S and MAXD/S, and the client-based protocols RNNC/C, AVGD/C and MAXD/C, as shown in Table 1. We focus on the relatively time-consuming operations, i.e., encryption, decryption, multiplication on ciphertext and exponentiation on ciphertext, and the time-saving but repetitive operation, i.e., distance calculation. Note that we ignore the constant permutation operations in Table 1. To be more intuitive, the setup phase and query phase are considered separately.

Setup phase. In the setup phase of all our four query protocols, $S$ and $C$ jointly run the PFGen protocol where both $S$ and $C$ need $(n_s + n_c)$ exponentiation operations modular $n = 2^{\lambda}$, which is much faster than the modular $n^2$. In addition, $S$ needs two permutations and $C$ needs one permutation, both of which can be ignored. It is a small cost compared to the client-based protocols RNNC/C, AVGD/C and MAXD/C, where the client $C$ needs to execute $\tilde{n}$ encryptions. Although the setup phase is very simple in server-based protocols RNNC/S, AVGD/S and MAXD/S, it will bring a great amount of computation cost to $C$ in the query phase.
Table 1. Computational Complexity.

| Comp. Complexity | Setup | Query Process |
|------------------|-------|---------------|
|                  | $S$   | $C$           |
| State-of-the-art Schemes | $n_s \cdot k \text{ dist}$ | $n_c \cdot k \text{ mult}$ |
|                   | $\tilde{n} \cdot k \text{ enc}$ | $k \text{ enc}$ |
|                   | $k \text{ dec}$ | |
| RNNC/S            | -     | -             |
|                   | $n_s \cdot k \text{ dist}$ | $2 \cdot n_c \text{ mult}$ |
|                   | $2 \cdot \tilde{n} \text{ enc}$ | $2 \text{ enc}$ |
|                   | $2 \text{ dec}$ | $1 \text{ div}$ |
| AVGD/S            | -     | -             |
|                   | $n_s \cdot k \text{ dist}$ | $w \cdot (n_c - 1) \text{ mult}$ |
|                   | $n \cdot w \text{ enc}$ | $w \text{ exp}$ |
|                   | $w \text{ dec}$ | |
| MAXD/S            | -     | -             |
|                   | $n_s \cdot k \text{ dist}$ | $k \text{ dec}$ |
|                   | $n_s + k \text{ mult}$ | |
| RNNC/C            | -     | $\tilde{n} \text{ enc}$ |
|                   | $n_s \cdot k \text{ dist}$ | $2 \text{ enc}$ |
|                   | $2 \cdot n_c \text{ mult}$ | $1 \text{ div}$ |
|                   | $n_s \text{ exp}$ | |
| AVGD/C            | -     | $\tilde{n} \text{ enc}$ |
|                   | $n_s \cdot k \text{ dist}$ | $w \cdot q + 1 \text{ dec}$ |
|                   | $n \cdot w \text{ enc}$ | $\leq w \text{ exp}$ |
|                   | $w \text{ exp}$ | |
| MAXD/C            | -     | $\tilde{n} \text{ enc}$ |
|                   | $n_s \cdot k \text{ dist}$ | $1 \text{ enc}$ |
|                   | $n \cdot w \text{ enc}$ | $1 \text{ mult}$ |

| Our Schemes |
|-------------|
| RNNC        | 1 PFGen  |
| AVGD        | 1 PFGen  |
| MAXD        | 1 PFGen  |
| RNNC-CMC    | 1 PFGen  |
|             | $n_s \cdot [k/\eta] \text{ enc}$ | $n_1 \cdot [k/\eta] \text{ mult}$ |
|             | $[k/\eta] \text{ dec}$ | $[k/\eta] \text{ enc}$ |

1. The $\tilde{n}$ is the size of the superset. Typically, in order to achieve enough privacy for both $U$ and $U_c$, $\tilde{n}$ should be far greater than $n_s$, or at least $\tilde{n} \gg n_s$ should be not less than a specified threshold. 2. $w$ is a large number satisfying $w > \text{max}$, where $\text{max}$ is the maximum distance between a user and their nearest facility. 3. When $S$ and $C$ jointly run a PFGen processing, both $S$ and $C$ need $(n_c + n_s)$ exponentiation (modular $n$) operations, while the “$\text{exp}$” represents the exponentiation operation modular $n^2$. 4. The parameter $\eta = \frac{\lambda}{(\log_2 \lambda)}$ is the amount of data concluded in one ciphertext block where $\lambda$ is the security parameter and $\alpha$ is the number of bits reserved for intermediate process with no overflows.

Query phase. In the query phase, for our RNNC, AVGD and MAXD query protocols, the computation processes of $S$ are similar—$n_s \cdot k$ distance calculations, $n_s$ encryptions and 1 decryption for both the AVGD query and MAXD query, and due to the dimension issue of matrix $M$, $n_s \cdot k$ distance calculations, $n_s \cdot k$ encryptions and $k$ decryptions for the RNNC query. The main operations of $C$ take place in Step 6—$n_1 \cdot k$ multiplications on the
cipher and \( k \) encryptions for RNNC, \( n_f \) multiplications on the cipher and one encryption for AVGD, one multiplication on cipher and one encryption for MAXD. In addition, for AVGD, \( C \) needs to execute one division in Step 8. For our RNNC-CMC protocol, we further reduce the computational complexity after the distance calculations to \( 1/k \) times of our original RNNC protocol for both \( S \) and \( C \).

From the analysis results, since we avoid the use of a superset with the very large cardinality \( n \) (no “\( n \) enc” operations in our schemes), we have completely avoided a large number of homomorphic encryption operations on both the server and client-sides and the corresponding computational complexity has been reduced. Further, we avoid a big number \( w > \max \) in the MAXD query protocol, resulting in a reduction of a large number of exponentiation and encryption operations. In addition, we further reduce the parameter \( k \) to \( \lceil k/\eta \rceil \) in the improved protocol RNNC-CMC.

For communication cost, we give the final result directly as follows, which is between those of the server-based protocols and the client-based protocols in [3].

1. RNNC: \( O(n_s \cdot k) \times \text{size of the block of cipher} \);
2. AVGD: \( O\left(\frac{1}{2}n_s + n_f\right) \times \text{size of the block of cipher} \);
3. MAXD: \( O\left(\frac{1}{2}n_s\right) \times \text{size of the block of cipher} \);
4. RNNC-CMC: \( O(n_s \cdot (\lceil k/\eta \rceil + \frac{1}{2})) \times \text{size of the block of cipher} \).

In summary, the proposed aggregate query protocols are comprehensively ahead of the server-based protocols and have advantages and disadvantages in some aspects compared with the client-based protocols.

### 8.2. Experimental Results

We implement all our protocols in Python 3.7 with a 64-bit Windows 10 system, 3.20 GHz Intel Core i5 processor and 16 GB RAM. We choose the security parameter \( \lambda = 1024 \), and set \( n_c = 20,000 \) and \( n = 1,000,000 \), the same values as in [3] for reasonable comparisons (We do not emphasize that we use the real-world dataset [31], because whether the location data are real or randomly selected has no effect on our experimental results, and without considering differential privacy, real-world datasets are meaningless for efficiency tests. In fact, the results of the experiments (the trends of the lines in Figures A1 and A2) on real and unreal location datasets are exactly the same). In addition, we suppose \( n_f = n_c \) in our experiment. First, for the three types of queries—RNNC, AVGD and MAXD, we fix the value \( n_f = 100,000 \) and test the computation costs of our protocols and those of the previous along with the change of \( k \). Then, we fix the value \( k = 50 \) and test the computation costs along with the change of \( n_f \). For our RNNC-CMC protocol, we set the threshold \( \alpha = 16 \), which is enough for the general RNNC query (each \( q_i \) is no more than \( 2^{16} - 1 = 65,535 \)), and accordingly, \( \eta = 64 \).

Since the server and the client may have different scales of computing power in a real environment, we test the cost of the server and the cost of the client separately and give their lines in figures. Our experimental results are given in Figures A1 and A2 in Appendix A.

There are two different types of exponentiation operations in our experiments. The first one is the exponentiation modular \( n^2 \) (2048 bits) executed on the client-side of MAXD/S and the server-side of AVGD/C and MAXD/C. The second one is the exponentiation modular \( n \) (1024 bits) executed in the setup phase (jointly running PFGen protocol) of our four protocols. Note that the time cost of modular exponentiation operation is very unstable when the exponent changes, and we know that computing \( a^x \) costs the equivalent of \( \log_2 x \) multiplications by using the fast exponential algorithm. Therefore, we use the mathematical expectation instead of the unstable value. Suppose an exponent is a \( b \)-bit number, the expectation of equivalent times of multiplications can be calculated as \( E(t) = \sum_{x=0}^{2^b-1} \frac{I(x)}{2^x} \approx \int_{b-1}^{b} \log_2 x \, dx \approx b - 1 \). Therefore, we use \((b-1) \times \text{the time of 1 multiplication instead of 1 exponentiation to draw the figures to clearly show the trend of change. Moreover,
according to our test, the time cost of the 1024-bit modular exponentiation operation is a little smaller than a quarter of that of the 2048-bit.

From the experimental results, we give what we see for reference as follows.

(1) For the RNNC query, our protocols (RNNC and RNNC-CMC) are more efficient than RNNC/S protocols on both the client-side and the server-side, and not as efficient as the RNNC/C protocol but acceptable (Our protocols avoid a lot of encryption operations in the setup phase compared to client-based protocols (RNNC/C, AVGD/C and MAXD/C). The price of the client-based protocols being faster than ours in some cases (on the client-side for RNNC and AVGD queries, and on the server-side for RNNC and MAXD queries) is that a lot of calculations are transferred to the setup phase. In addition, taking note of the multiples of the time cost comparisons in (1), (2) and (3), \( \frac{1}{30} \) and \( \frac{1}{25} \) are non-ignorable values, but 0.24 s (in our experimental environment) to the client and hundreds of seconds (in our experimental environment) to the server are actually very small values). Concretely, taking \( k = 50 \) and \( n_s = 100,000 \), on the client-side, our RNNC-CMC protocol is \( (0.24 \text{ s}) \times 50 \times \) faster than the RNNC/S protocol, and \( \frac{1}{30} \times \) faster than the RNNC/C protocol; on the server-side, our RNNC-CMC protocol (407.47 s) is \( 620 \times \) faster than the RNNC/S protocol, and \( \frac{1}{25} \times \) faster than the RNNC/C protocol.

(2) For the AVGD query, on the client-side, the time cost of our AVGD protocol is between that of the AVGD/S protocol and AVGD/C protocol. However, on the server-side, our protocol is much more efficient than the previous two, and it is valuable that our query protocol reduces the rate of increase in the time cost of the server when \( n_s \) increases (Figure A2-AVGD Query). Concretely, taking \( k = 50 \) and \( n_s = 100,000 \), on the client-side, our AVGD protocol is \( (0.24 \text{ s}) \times 2 \times \) faster than the AVGD/S protocol, and \( \frac{1}{25} \times \) faster than the AVGD/C protocol; on the server-side, our AVGD protocol is \( (508.46 \text{ s}) \times 20 \times \) faster than the AVGD/S protocol, and \( 5 \times \) faster than the AVGD/C protocol. In addition, when \( n_s \) increases to 500,000, our AVGD protocol is \( (2542.30 \text{ s}) \times 4 \times \) faster than the AVGD/S protocol, and \( 5 \times \) faster than the AVGD/C protocol on the server-side.

(3) For the MAXD query, our protocol is friendly to a resource-limited client compared with the two previous protocols. It is more than one magnitude faster than the client-based protocol MAXD/C. Moreover, on the server-side, the time cost of our MAXD protocol is between that of the MAXD/S protocol and MAXD/C protocol. Concretely, taking \( k = 50 \) and \( n_s = 100,000 \), on the client-side, our MAXD protocol is \( (0.0051 \text{ s}) \times 26,000 \times \) faster than the MAXD/S protocol, and \( 15 \times \) faster than the MAXD/C protocol; on the server-side, our MAXD protocol is \( (508.46 \text{ s}) \times 5000 \times \) faster than the MAXD/S protocol, and \( \frac{1}{25} \times \) faster than the MAXD/C protocol.

9. Conclusions

In this paper, we have studied the aggregate queries for facility location selection between an LSP and a business with confidentiality. We proposed the construction of a PSI filter to help two parties find the relevant data corresponding to the items in the intersection without revealing the privacy of the items, including those in the intersection. Then, for location data analysis, we utilize our PSI filter to construct three types of aggregate query protocols without setting a superset of a large size for concealing the user identities and serving as an index. Moreover, we propose a ciphertext matrix-compressing method that makes one ciphertext block contain multiple small plaintext data while keeping the homomorphic property valid, and, subsequently, we give the improved protocol by utilizing this method. It further raises the efficiency of both computation and communication on the basis of the original query process. Finally, we illustrate the correctness and privacy and give the performance evaluation through theoretical analysis and experiments that show the superiorities of our protocols.

Author Contributions: Conceptualization, D.F. and F.Z.; methodology, D.F.; software, D.F. and B.L.; validation, Q.W. (Qiang Wang), B.L. and Q.W. (Qiyu Wu); formal analysis, D.F.; investigation, Q.W. (Qiyu Wu) and B.L.; resources, F.Z. and Q.W. (Qiang Wang); data curation, Q.W. (Qiyu Wu); writing—original draft preparation, D.F. and Q.W. (Qiyu Wu); writing—review and editing, D.F. and
Notations

The following notations are used in this manuscript:

| Notation | Description |
|----------|-------------|
| \( C \) and \( S \) | client and server |
| \( k_c, k_s \) | \( C, S \)'s secret key |
| \( U_c = \{u_1, \ldots, u_n\} \) | \( C \)'s user set |
| \( U_s = \{v_1, \ldots, v_n\} \) | \( S \)'s user set |
| \( I \) | the intersection of \( U_c \) and \( U_s \) |
| \( F \) | facility location set |
| \( L \) | user location set |
| \( Q \) (or \( Q \)) | the desired query result |
| \( n_s, n_c, n_I \) and \( k \) | the cardinalities of the sets \( U_s, U_c, I \) and \( F \) |
| \( (sk, pk) \) | the key pair of Paillier cryptosystem |
| \( label_i \) | a label looks random |
| \( H() \) | a hash function |
| \( \tau() \) | a random permutation |

Appendix A. Figures of Experimental Results

We give the experimental results (of Section 8.2) in Figures A1 and A2.
Figure A1. Query processing time of the protocols (vs. $k$). (a) Time cost of the server $S$. (b) Time cost of the client $C$.

Figure A2. Query processing time of the protocols (vs. $n_s$). (a) Time cost of the server $S$. (b) Time cost of the client $C$. 

(a)

(b)
References

1. Kahraman, C.; Ruan, D.; Doğan, I. Fuzzy Group Decision-making for Facility Location Selection. Inf. Sci. 2003, 157, 135–153. [CrossRef]

2. Du, Y.; Zhang, D.; Xia, T. The optimal-location query. In Proceedings of the International Symposium on Spatial and Temporal Databases, Angra dos Reis, Brazil, 22–24 August 2005; pp. 163–180.

3. Yilmaz, E.; Ferhatosmanoglu, H.; Ayday, E.; Aksoy, R.C. Privacy-preserving aggregate queries for optimal location selection. IEEE Trans. Dependable Secur. Comput. 2017, 16, 329–343. [CrossRef]

4. Ion, M.; Kreuter, B.; Nergiz, E.; Patel, S.; Saxena, S.; Seth, K.; Shanahan, D.; Yung, M. Private intersection-sum protocol with applications to attributing aggregate ad conversions. IACR Cryptol. ePrint Arch. 2017, 738.

5. Roussopoulos, N.; Kelley, S.; Vincent, F. Nearest neighbor queries. In Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, USA, 22–25 May 1995; pp. 71–79.

6. Korn, F.; Muthukrishnan, S. Influence sets based on reverse nearest neighbor queries. ACM Sigmod Rec. 2000, 29, 201–212. [CrossRef]

7. Stanoi, I.; Agrewal, D.; Abbadi, A.E. Reverse nearest neighbor queries for dynamic databases. In Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, Dallas, TX, USA, 14 May 2000; pp. 44–53.

8. Tao, Y.; Papadias, D.; Lian, X. Reverse kNN search in arbitrary dimensionality. In Proceedings of the Very Large Data Bases Conference (VLDB), Tokyo, Japan, 31 August–3 September 2004; pp. 744–755.

9. Yang, C.; Lin, K.I. An index structure for efficient reverse nearest neighbor queries. In Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, 2–6 April 2001; pp. 485–492.

10. Stanoi, I.; Riedewald, M.; Agrawal, D.; Abbadi, A.E. Discovery of influence sets in frequently updated databases. In Proceedings of the 27th International Conference on Very Large Data Bases, 2001; pp. 99–108.

11. Zhang, D.; Du, Y.; Xia, T.; Tao, Y. Progressive computation of the min-dist optimal-location query. In Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, 12–15 September 2006; pp. 643–654.

12. Qi, J.; Zhang, R.; Wang, Y.; Xue, A.Y.; Yu, G.; Kulik, L. The min-dist location selection and facility replacement queries. World Wide Web 2014, 17, 1261–1293. [CrossRef]

13. Mitra, S.; Saraf, P.; Bhattacharya, A. Tips: Mining top-k locations to minimize user-inconvenience for trajectory-aware services. IEEE Trans. Knowl. Data Eng. 2019, 33, 1238–1250. [CrossRef]

14. Islam, M.S.; Shen, B.; Wang, C.; Taniar, D.; Wang, J. Efficient processing of reverse nearest neighborhood queries in spatial databases. Inf. Syst. 2020, 92, 101530. [CrossRef]

15. Du, W.; Atallah, M.J. Secure multi-party computation problems and their applications: A review and open problems. In Proceedings of the 2001 Workshop on New Security Paradigms, Cloudcroft, NM, USA, 10–13 September 2001; pp. 13–22.

16. Du, W.; Atallah, M.J. Protocols for secure remote database access with approximate matching. In E-Commerce Security and Privacy; Springer: Berlin/Heidelberg, Germany, 2001; pp. 87–111.

17. Hácsğım, H.; Iyer, B.; Mehrotra, S. Efficient execution of aggregation queries over encrypted databases. In Proceedings of the International Conference on Database Systems for Advanced Applications, Jeju Island, Korea, 17–19 March 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 125–136.

18. Cheng, R.; Zhang, Y.; Bertino, E.; Prabhakar, S. Preserving user location privacy in mobile data management infrastructures. In Privacy Enhancing Technologies; Springer: Berlin/Heidelberg, Germany, 2006; pp. 393–412.

19. Mokbel, M.F.; Chow, C.-Y.; Aref, W.G. The new casper: Query processing for location services without compromising privacy. In Proceedings of the 32nd International Conference on Very Large Data Bases, Berlin, Germany, 12–13 September 2006; pp. 763–774.

20. Qi, Y.; Atallah, M.J. Efficient privacy-preserving k-nearest neighbor search. In Proceedings of the 2008 the 28th International Conference on Distributed Computing Systems, Beijing, China, 17–20 June 2008; pp. 311–319.

21. Thompson, B.; Haber, S.; Tomas, H.; Yao, D. Privacy-preserving computation and verification of aggregate queries on outsourced databases. In International Symposium on Privacy Enhancing Technologies Symposium; Springer: Berlin/Heidelberg, Germany, 2009; pp. 185–201.

22. Yi, X.; Paulet, R.; Bertino, E.; Varaharajan, V. Practical k nearest neighbor queries with location privacy. In Proceedings of the 2014 IEEE 30th International Conference on Data Engineering, Chicago, IL, USA, 31 March–4 April 2014; pp. 640–651.

23. Chen, H.; Huang, Z.; Laine, K.; Rindal, P. Labeled PSI from fully homomorphic encryption with malicious security. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1223–1237.

24. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 223–238.

25. Goldreich, O. Foundations of Cryptography: Volume 2; Basic Applications; Cambridge University Press: Cambridge, UK, 2009.

26. Lindell, Y. How to simulate it—a tutorial on the simulation proof technique. In Tutorial on the Foundations of Cryptography; Springer: Cham, Switzerland, 2017; pp. 277–346.

27. Kissner, L.; Song, D. Privacy-preserving set operations. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 241–257.
28. Shekhar, S.; Kohli, A.; Coyle, M. Path Computation Algorithms for Advanced Traveller Information System (ATIS). In Proceedings of the IEEE 9th International Conference on Data Engineering, Vienna, Austria, 19–23 April 1993; pp. 31–39.

29. Damgård, I.; Jurik, M.; Nielsen, J.B. A generalization of Paillier’s public-key system with applications to electronic voting. *Int. J. Inf. Secur.* **2010**, *9*, 371–385. [CrossRef]

30. Ge, T.; Zdonik, S. Answering aggregation queries in a secure system model. In Proceedings of the 33rd International Conference on Very Large Data Bases, Vienna, Austria, 23–27 September 2007; pp. 519–530.

31. Yang, D.; Zhang, D.; Zheng, V.W.; Yu, Z. Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. *IEEE Trans. Syst. Man, Cybern. Syst.* **2014**, *45*, 129–142. [CrossRef]