Location Oblivious Privacy Protection for Group Nearest Neighbor Queries

A.K.M. Mustafizur Rahman Khan[1], Tanzima Hashem[2], Egemen Tanin[1], and Lars Kulik[1]

[1] University of Melbourne, Victoria, Australia
[2] Bangladesh University of Engineering & Technology, Dhaka, Bangladesh

Abstract. Finding a convenient meeting point for a group is a common problem. For example, a group of users may want to meet at a restaurant that minimizes the group's total travel distance. Such queries are called Group Nearest Neighbor (GNN) queries. Up to now, users have had to rely on an external party, typically a location service provider (LSP), for computing an optimal meeting point. This implies that users have to trust the LSP with their private locations. Existing techniques for private GNN queries either cannot resist sophisticated attacks or are computationally too expensive to be implemented on the popular platform of mobile phones. This paper proposes an algorithm to efficiently process private GNN queries. To achieve high efficiency we propose an approach that approximates a GNN with a high accuracy and is robust to attacks. Unlike methods based on obfuscation, our method does not require a user to provide an imprecise location and is in fact location oblivious. Our approach is based on a distributed secure sum protocol which requires only light weight computation. Our experimental results show that we provide a readily deployable solution for real life applications which can also be deployed for other geo-spatial queries and applications.

1 Introduction

Internet accessibility through smartphones has fueled the increased popularity of Location Based Services (LBS). Finding a convenient meeting point for a group is a common problem. For example, a group of friends may want to know the restaurant which minimizes the total travel distance for them. Such queries are called Group Nearest Neighbor (GNN) queries. In a GNN query, the users issue a query to find out the point of interest (POI) which minimizes aggregate distance (AD) for the group. The AD could be the total distance of all users to the POI. To access GNN queries, users have to disclose their locations to a location service provider (LSP). The users risk their privacy by exposing locations while receiving such services. An LSP might use a user's location as a clue to derive sensitive and private information about her habits, health, social relations, etc. An alternative approach would be to compute ADs by sharing a user's location with other group members. However, users may want to hide their locations not only from an LSP but also from other group members. Thus, the major challenge for processing a privacy preserving GNN query is to compute ADs for POIs, stored on the LSP's database, without revealing user locations to others.

Obfuscation based techniques such as spatial cloaking, i.e., providing imprecise location, have been widely used to protect location privacy while accessing LBSs [1–4].

But, in many cases, this direction of research of blurring a point to a region is not enough to provide location privacy. For example, if a user frequently reveals her imprecise location, her trajectory can be derived. Obfuscation cannot protect a user from absence disclosure attack as well [5].

Most of the existing techniques are designed to preserve location privacy of an individual and cannot preserve location privacy for a group of users. Hashem et al. [3] first proposed a technique to preserve privacy for a group of users. Talouki et al. [6] and Huang et al. [7] followed them. In Sec. 2.1, we show that these techniques are either vulnerable to sophisticated attacks or computationally expensive for mobile phones.

In this paper, we also identify a new attack type that no previous work related to GNN queries has addressed. We show that if attackers can learn a group's ADs to a sufficient number of POIs, they can compute the locations of all users in the group without knowing which location corresponds to which user. We call it *Multi-point Aggregate Distance (MPAD) attack*.

In a privacy preserving GNN query, a user's location must be kept secret from other users as well as the LSP. If a user's distances to three POIs are known, the user's location can be calculated by 2D trilateration. Therefore, the group is required to compute ADs of POIs without revealing the users' locations or individual distances to POIs. In order to resist the MPAD attack, the number of revealed ADs of POIs must be limited.

In any LBS, a user expects a certain quality of that service. In our scenario, the quality depends on at least three factors: (i) the level of guaranteed privacy, (ii) the cost to the user, i.e., the communication and computation cost for a mobile device, and (iii) the accuracy of the results returned by the LSP. As there is no known algorithm that fulfills all three criteria, in this paper, we worked on addressing the problem in a way that could be readily deployed in the real world and could be used by real users. We have developed two new algorithms. Our new algorithms are LOOP and H-LOOP. Location Oblivious Private (LOOP) algorithm produces an accurate answer with very little cost. ADs of all POIs are calculated in a distributed privacy preserving manner, and the POI with the smallest AD is declared as the GNN. LOOP is based on a distributed secure sum protocol, proposed by Sheikh et al. [8], which does not use cryptography and thus is computationally inexpensive. Although LOOP provides better location privacy than existing methods, it still has two limitations: first, it cannot resist the MPAD attack from malicious users, second, LOOP requires the LSP to reveal locations of all POIs to the users. However in many cases an LSP might be unwilling to reveal all POIs' locations.

To overcome these limitations, we need to reduce the number of POIs whose ADs are calculated. In this paper, we propose H-LOOP, an improved version of LOOP, which fulfills all the requirements, i.e., ensures both high level of privacy and high accuracy of the query answer, and incurs less processing overhead. H-LOOP takes a heuristic approach to drastically narrow down the search space and returns an approximate answer with an extremely high level of accuracy. H-LOOP predicts the location of the GNN in the part of the search space which has the smallest minimum AD. Existing methods cannot maintain a high level of privacy, efficiency and accuracy simultaneously. H-LOOP is the first method which provides high levels of privacy in an efficient way while rarely sacrificing accuracy at a small degree.

Unlike methods based on imprecise location, our methods do not require a user to provide any information about her location and thus is location oblivious. A user divides her distance from a point/square into random parts and sends those to other users. Her distance remains undisclosed unless all other users collude and recover all parts. As a result, a user's location cannot be recovered or inferred by an attacker.

1.1 Attack Model

In our scenario, neither the LSP nor the users are trustworthy. The LSP may collude with malicious group members to organize an attack. Therefore, an honest user trusts neither the LSP nor the fellow group members. In this study, we assume that both users and the LSP act according to the protocol. We consider only passive attacks, where an attacker or a group of colluding attackers try to recover the locations of participating users using information gained through query processing.

Existing works related to privacy preserving GNN queries considered 2D trilateration based attacks. In these attacks, an attacker or a group of colluding attackers compute distances of an individual user to multiple POIs using gained information during the distance aggregation process. Then these distances are used in 2D trilateration to recover the exact location of that user. We, too, consider these attacks.

We identify a new attack type, called Multi-Point Aggregate Distance (MPAD) attack in this paper. In an MPAD attack, an attacker can use aggregate total distances of multiple POIs to calculate users' locations. Let the coordinate of a POI P_i be (P_{xi},P_{yi}) and coordinate of a user U_j be (U_{xj},U_{yj}). Aggregate total distance of P_i from n users is computed as

$$D_{Pi} = \sum_{j=1}^{n} \sqrt{(P_{xi} - U_{xj})^2 + (P_{yi} - U_{yj})^2}$$

Since the locations of users are unknown, there are $2n$ unknown variables in the above mentioned equation. Hence, if an attacker can learn aggregate total distances to $2n$ POIs, she can solve the equations. However, the equations have multiple solutions. If the attacker knows more than $2n$ ADs, she can use the extra equations to discard false solutions. Aggregate total distance functions are symmetric functions. Therefore, the set of solutions is invariant under any permutation of the unknown variables. An attacker can target only the whole group collectively and cannot distinguish individuals in the group. The attacker may associate a single point to a user using background knowledge or revealed imprecise location of that user.

2 Background

2.1 Finding an Optimal Meeting Point in a Privacy Preserving Manner

Papadias et al. introduced the GNN query [9]. Hashem et al. [3] first proposed a privacy preserving GNN query processing technique. The users provide their imprecise locations as regions instead of exact points to an LSP and thus preserve their location privacy. The LSP returns a set of candidate POIs with respect to the provided regions. Users update AD for every POI with respect to their actual locations such that each user's distance to POIs remain hidden in an aggregate form. After all users' updates, the POI with the smallest AD becomes the GNN. Talouki et al. [6] have shown that Hashem et al's [3] technique is vulnerable to a Partial Collusion (PC) attack and a user's location can be recovered. Moreover, as we mentioned in Sec. 1, imprecise location does not provide a high level of location privacy.

Talouki et al. took a cryptographic approach to solve the privacy preserving GNN query problem and proposed two solutions. In [6], a group of users calculate their centroid by averaging their coordinates using an anonymous veto network and homomorphic encryption thus hiding their locations. The LSP returns the nearest point to that centroid as the GNN. Unfortunately, the nearest point to the centroid is not necessarily the GNN. Suppose user A and B are located in (0,0) and (8,0) respectively and POI p and q are located in (4,1) and (8,0) respectively. The centroid of the users' locations is (4,0). p is the nearest POI to the centroid and its aggregate total distance is $\sqrt{4^2+1^2}+\sqrt{4^2+1^2} = 8.24$. AD of q is $8+0 = 8$. Hence, p is not the GNN but q is. We can conclude that, their approach cannot guarantee a correct answer. In [10], Talouki et al. used imprecise location and calculated ADs of all candidate POIs using an anonymous veto network and homomorphic encryption. As mentioned in Sec. 1, we consider hiding location in a larger region is not enough to provide privacy.

Huang et al. also used a cryptographic technique to preserve privacy in GNN queries [7]. A single user creates a Garbled circuit, and another user evaluates it. All other users get their encrypted input bits from the circuit creator by Oblivious Transfers (OT) and send their input bits to circuit evaluator. Then the circuit evaluator evaluates the Garbled circuit to find out the GNN. Creating and evaluating a Garbled circuit when there were 10 users and 2000 POIs took 20 seconds in a centralized model and 4 seconds in a distributed model in their experiment. The number of OTs performed by a user is equal to the number of POIs. In [11], Huang et al. reported that Garbled circuit implementations on smartphones are about three orders of magnitude slower than desktop computers. Moreover, the circuit creator needs to send a circuit file of several GB to the circuit evaluator [12] and thus incur heavy communication cost. Hence, Garbled circuit based methods are not applicable in various real world mobile phone applications.

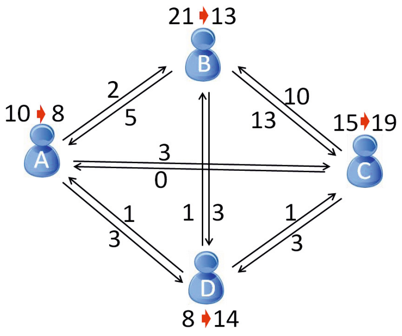
Existing privacy preserving techniques for GNN query [3, 7, 10] cannot bound the number of POIs whose ADs are calculated. As a result, they are open to MPAD attack as well. The number of candidate POIs in [3] and [10]'s technique depends on the size of cloaking rectangle. Larger rectangles lead to more candidate POIs. ADs to all candidate POIs are compared to find out the GNN. Huang et al.'s [7] technique requires calculation and comparison of ADs of all POIs.

2.2 Distributed Secure Sum

If a user's distances from three points are known, her position can be easily calculated by 2D trilateration. The group members have to calculate AD of a point, which is a function of all members' distances, without revealing their own distances. This problem is similar to secure multi-party computation problems, where a group of users compute a function which takes one input from each user. The function is public, but each user can keep her input secret. Sheikh et al. proposed a simple distributed secure sum protocol to compute the summation function [8]. We base our work on their technique in order to calculate aggregate total distance.

We illustrate their technique with a simple example. Suppose, there are four players A, B, C, and D and a coordinator. A, B, C, and D have 10, 21, 15, and 8 cards, respectively. They do not want to disclose their number of cards but want to calculate how many cards they have in total. The coordinator publishes the result and can help in computation if necessary. The solution is as follows.

Each player randomly divides her cards into four parts, keeps one part for herself and gives other three parts to the other three players. We assume that transactions between two players are not visible by others. Their transactions are shown in Fig. 1. Change of a player's stock is shown above/below her.



	Receives				Stock before transaction
	A	B	C	D	
A gives	2	2	3	3	10
B gives	5	0	13	3	21
C gives	0	10	2	3	15
D gives	1	1	1	5	8
Stock after transaction	8	13	19	14	54 (Total)

Fig. 1. Transactions between players

The players inform the coordinator about the number of cards they possess, i.e., 8, 13, 19, and 14. The coordinator adds these numbers and gets 54 as total. Since, the cards only change hands, the total numbers of cards before and after the transactions are equal. Even if the coordinator collude with dishonest players, they cannot calculate a player's initial stock if there is at least one honest player other than the targeted player.

3 Our Solution

In our model, all users can communicate with the LSP and the users can communicate with each other without the help of the LSP. While forming a group, group members exchange their identities (e.g., IP address, phone number) with other members. A member sends all group members' identities and the POI type to the LSP and inquires about the GNN. The LSP runs either LOOP or H-LOOP. During the course of execution, the LSP communicates with the users to direct the GNN search. The users communicate with each other to compute and compare ADs. Computation and comparison of ADs are done with two privacy preserving protocols, which we study in Sec. 3.1. Finally, the LSP broadcasts the answer to all users.

3.1 Privacy Preserving Protocols

In this section, we present two privacy preserving protocols. These protocols safeguard users' location privacy. We present our algorithms to process the GNN query using these protocols in Sec. 3.2.

Private Aggregate Distance (PAD) Protocol. We use PAD to calculate the AD of all group members to a point in a privacy preserving manner. As a result, users can calculate their ADs from any point without revealing their individual distances from the point. We assume that the attackers cannot observe communications between the users. This protocol is based on Sheikh et al. [8]'s work, described in Sec. 2.2. Let $\{U_1, U_2, \ldots, U_n\}$ be the group of users. Distance from a user U_i to a point or a rectangle is d_i, which is kept private. The AD has to be computed as

$$D = \sum_{i=1}^{n} d_i.$$

The protocol to solve this problem is as follows:

Each user U_i does:

1. Choose an integer random number α_i, where $0 \leq \alpha_i < n$.
2. Choose α_i users from the group, excluding U_i. Let $\{U_{i1}, U_{i2}, \ldots, U_{i\alpha_i}\}$ be the chosen subset of users.
3. Choose α_i random numbers d_{ij}.
4. Calculate
$$\Delta_i = d_i - \sum_{j=1}^{\alpha_i} d_{ij}$$
5. Send d_{ij} to U_{ij}, where $j = \{1, 2, \ldots, \alpha_i\}$.
6. Receive numbers from other users if sent. Let $\Delta_i^{'}$ be the sum of the received numbers.
7. Calculate $d_i^{'} = \Delta_i + \Delta_i^{'}$ and send $d_i^{'}$ to the coordinator.

Finally, the coordinator calculates

$$D = \sum_{i=1}^{n} d_i^{'}$$

A user discloses only d_{ij} to U_{ij} and $d_i^{'}$ to the coordinator in step 5 and 7 respectively. The user keeps all other information private.

An LSP or any user or any third party can act as a coordinator. The number of messages sent by a user U_i is α_i. Since an attacker is not aware about communications between users, even if a user chooses $\alpha_i = 0$ and does not send any number to any user, the attacker still cannot learn the value of α_i. Hence, privacy is assured even at a lower value of α. If α_i has a mean value of α, total number of messages is $n\alpha$. When calculating ADs to a set of points/rectangles, users exchange an array of values instead of just one value in each communication. PAD cannot protect a user if all other users collude. Therefore, the group size must be at least three.

Comparing with Minimum Disclosure (CWMD) Protocol. Let there be n users and each user U_i has a set of POIs S_i. Suppose a user knows ADs of all POIs of her set. The users want to find out the POI from $S_1 \cup S_2 \cup \ldots \cup S_n$, which has the minimum AD. An attacker can use ADs to multiple points in order to find out the group's locations. Therefore, our goal is to find out the POI with smallest AD with minimum disclosure. CWMD solves this problem. If CWMD is used, a user cannot learn the AD of any additional POI and an LSP cannot learn enough to perform an attack.

Let $Dmin_i$ be the smallest AD of a POI in S_i. The protocol is as follows:

1. Each user U_i sends the smallest AD $Dmin_i$ to the LSP.
2. Let $Dmin_j$ be the smallest in $\{Dmin_1, Dmin_2, \ldots Dmin_n\}$. The LSP sends a disclosure request to U_j.
3. Let the POI P_{min} has the minimum AD in S_j. U_j sends P_{min} to the LSP.
4. The LSP publishes P_{min}.

There are $2n$ unknown coordinate values of n user locations. Therefore, an LSP needs at least $2n$ POIs and their ADs to perform an MPAD attack. In CWMD, an LSP can learn n ADs and the ID of only one POI. It cannot learn the IDs of the remaining related POIs. Even if the LSP cheats it can learn ADs of at most n POIs. It cannot invoke the protocol multiple times because the protocol needs active participation of the users. As a result, the LSP cannot perform an MPAD attack.

3.2 Privacy Preserving Algorithms

We present two privacy preserving GNN query processing algorithms. These algorithms are named as Location Oblivious Private (LOOP) algorithm and Heuristic Location Oblivious Private (H-LOOP) algorithm. They are based on PAD and CWMD.

Location Oblivious Private (LOOP) Algorithm. Let there be n users in the group. The LSP divides all POIs into n subsets such that POI P_j is an element of set S_l, where $l = (j \bmod n) + 1$. Then the LSP assigns each user U_i as the coordinator of POIs in S_i, where $i = 1, 2, ..., n$. Next the LSP sends a list of all POIs along with their locations and assigned coordinators' IDs to the users. The coordinator users communicate with other users and calculate the ADs of their assigned set of POIs using PAD. In this way, ADs of all POIs are calculated. Then the coordinators use CWMD to find out the POI with the smallest AD, which is the GNN.

Figure 2 shows an example of the flow of LOOP. At first, the LSP distributes 8 POIs among 4 users. Then the users communicate among themselves according to PAD and calculate ADs of all POIs. Finally, the users perform CWMD to find out the POI with the minimum AD.

Let there be x users in the group who are colluding. They want to calculate locations of the remaining $n - x$ users. So they need at least $2(n - x)$ ADs to find out $2(n - x)$ unknown coordinate values. Hence, ADs of at least $2(n - x) - 1$ POIs can be calculated safely. If the total number of POI is N, the colluders can learn ADs of at most $x\lceil \frac{N}{n} \rceil$ points. Thus, in order to prevent MPAD attack $2(n - x)$ must be greater than $x\lceil \frac{N}{n} \rceil$.

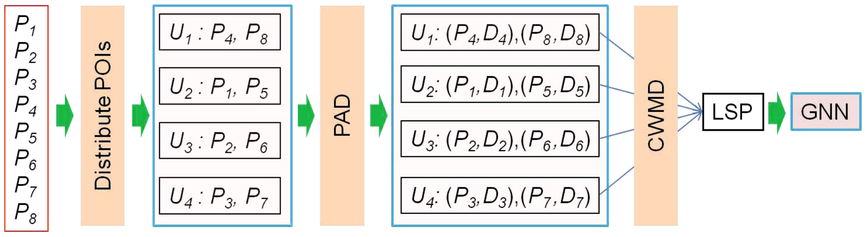


Fig. 2. Flow of LOOP

Hence, the maximum number of POIs can be handled by LOOP while resisting an MPAD attack from a group of x colluding users is

$$N_{max} = \begin{cases} n \times \lfloor \frac{2(n-x)-1}{x} \rfloor, & \text{if } 2(n-x)-1 \geq x. \\ 2(n-x)-1, & \text{otherwise.} \end{cases} \tag{1}$$

Table 1 shows N_{max} for a group of 10 users at different resistance levels. If the resistance level is y, MPAD attack from a group of y colluding users can be resisted.

Table 1. N_{max} at different resistance levels according to Equation 1

Number of colluders or resistance level	1	2	3	4	5,6	7	8
N_{max}	170	70	40	20	10	5	3

If there are a large number of POIs, i.e. larger than N_{max}, malicious users can learn sufficient number of ADs to perform MPAD attack. Since LOOP is a brute force method, users have to learn locations of all POIs and calculate their ADs. However, an LSP may not be willing to reveal locations of all POIs. In order to overcome above mentioned limitations, we need to reduce the number of POIs whose ADs are calculated. Next we propose a solution, H-LOOP, which prunes the search space and predicts the location of the GNN and searches in the surrounding area only.

Heuristic LOOP (H-LOOP) Algorithm. The base idea for H-LOOP is to locate the GNN in the part of the search space which has the smallest minimum AD and consider only the POIs located in that part as candidates. The LSP selects the smallest square which bounds all the POIs as the current search space and counts the number of POIs. If the number of POIs is larger than a threshold value β, then it divides the search space into γ equal sized non-overlapping squares. In H-LOOP, the users reach a consensus about the resistance level and then calculate N_{max} with (1). The users send N_{max} with the the query to the LSP. β is the minimum of N_{max} and the number of POI's location an LSP is willing to reveal. Lower value of β leads to higher resistance to MPAD attack. Then the LSP and users calculate aggregate minimum distances of these squares with PAD. Note that the minimum distance of a point from a region is defined as the distance of that point to its closest point in that region. If the concerned point is located inside the region, the minimum distance is zero. Next it selects δ squares with minimum ADs and narrows down the search space to those squares. Then it again counts the number of POIs in the new search space. The LSP divides each selected square into γ squares and narrows down the search space recursively until the number of POIs in the search space is less than β. Finally, LOOP is performed considering only those limited number of POIs in the search space to find out the GNN.

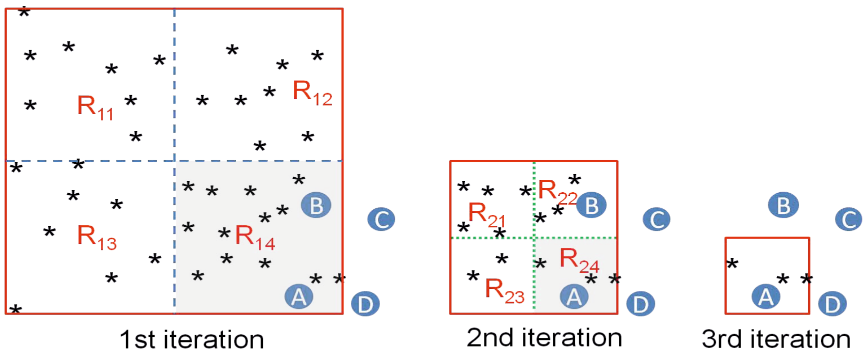


Fig. 3. Reduction of the search space in H-LOOP

We illustrate our approach with an example in Fig. 3. In this figure, asterisks are POIs and A, B, C and D are users. The square with solid boundary is the initial search space. Let $\gamma = 4, \delta = 1$ and $\beta = 4$. Since there are more than β POIs in the search space, the LSP divides it into four squares with dashed lines. These squares are R_{11}, R_{12}, R_{13} and R_{14}. Then the LSP calculates their centers and lengths and passes them to the users. Next the users calculate their minimum distances from these squares. Then they calculate their aggregate minimum distances from each of these squares using PAD and the LSP coordinates the calculation. The LSP selects R_{14} as the new search space since it has the minimum AD. The LSP divides the new search space again by dotted lines into four squares because it includes more than β POIs. It sends new squares R_{21}, R_{22}, R_{23} and R_{24} to the users. Again the users communicate with the LSP and calculate their aggregate minimum distances from each of these squares using PAD. The LSP selects R_{24} as the new search space since it has the minimum AD. Now the new search space has less than four POIs. Finally, LOOP is performed considering the POIs located inside R_{24} to find out the GNN.

An LSP runs the following algorithm for searching the GNN.

Input: A list of POIs with their locations, Division factor γ, Selection factor δ, Division threshold β
Output: The GNN

```
1  $SearchSpace \leftarrow$ bounding square of all POIs;
2  **While** number of POIs in $SearchSpace > \beta$
3      $sq \leftarrow$ divide($SearchSpace$,$\gamma$);
4      doPAD($sq$);
5      $SearchSpace \leftarrow$ top $\delta$ squares with minimum ADs;
6  **end**
7  Perform LOOP considering POIs in $SearchSpace$;
8  **Return** the POI returned by LOOP;
```

The function 'divide($SearchSpace$, γ)' divides each square of $SearchSpace$ into γ non-overlapping equal sized squares. A square number should be used as γ. In the function 'doPAD()' the LSP communicates with the users and coordinate AD calculation according to PAD. The number of iteration in the "While loop" depends on γ and β. The search space shrinks faster if a larger γ is used. β regulates the number of candidate POIs, whose AD will be calculated. Thus the robustness of H-LOOP to MPAD attack depends on β.

Privacy Analysis. In order to recover the location of a single user, her individual distances to multiple points must be known. PAD enables a user to keep her distances private if there is at least another honest user. Therefore, in this case, a user's location cannot be recovered by traditional 2D trilateration.

As discussed in Sec. 3.2, if x users collude, they must learn at least $2(n-x)$ ADs to points to perform an MPAD attack. Since an LSP can learn only one AD to a POI, it cannot perform MPAD attack. Malicious users may perform MPAD attack in LOOP if the number of POIs is large. However, in H-LOOP, if the number of colluders is within the resistance level set by the users, the colluders cannot learn sufficient number of ADs to points and consequently cannot do MPAD attack. The users can choose any value that is less than $n-1$ as resistance level. Although the LSP can learn aggregate minimum distances of multiple squares, this information cannot be used to solve (1) which requires ADs to points. An LSP can contribute less than a malicious user in MPAD attack because it knows AD of only one POI. Therefore, even if the LSP colludes with malicious users, their MPAD attack can be resisted.

Communication Cost. The communication cost in LOOP is low because both PAD and CWMD are performed only once. Though H-LOOP requires several rounds of communication like other privacy preserving methods, but in return H-LOOP can improve privacy level of users significantly compared to other methods. Hashem et al.'s [3] method needs anonymous communications between a user and an LSP. Talouki et al.'s [6, 10] methods setup an anonymous veto network at first. Users transfer data using Oblivious Transfer in Huang et al.'s [7] method. Anonymous communication and Oblivious Transfer are multi-step communication processes.

4 Experimental Setup

In this section we evaluate the performance of our proposed algorithms and compare them with related works through extensive experiments. We used both synthetic and real data in our experiments. The data space was normalized to an area of 1024 $\times$ 1024 square units. The data space can be considered as a map of a real city. We generated synthetic data sets using uniform and Zipfian distributions. We varied the size of datasets as 2K, 5K, 10K, 20K and 50K point locations. As real dataset we used all parks of California as POIs and locations of flats as user locations[1]. Since there are two major metropolitan areas in California, i.e., Los Angeles and San Francisco, we divided the dataset into two parts taking longitude -120 as a dividing line. California contains 6728 parks and 2700 flats. L.A. has 3407 parks and 1266 flats, where San Francisco has 3321 parks and 1434 flats. Flats were randomly selected as users' locations. As mentioned in Sec. 3.2, H-LOOP can be adjusted to prevent MPAD attack from a single user or a group of colluding multiple users. We varied the value of the division threshold β to deal with MPAD attacks by one to eight attackers. The value of both division factor γ and selection factor δ were set to four. Table 2 summarizes the values used for each parameter in our experiments and their default values. In our default scenario, we imagine 10 friends who want to select a park from nearby 2000 parks to have a picnic.

Table 2. Experimental setup

Parameter	Range	Default
Group size	3,5,10,15,20	10
POI distribution	Uniform, Zipfian	Uniform
User distribution	Uniform, Zipfian	Uniform
Number of POIs	2K, 5K, 10K, 20K, 50K	2K
Number of colluders	1,2,3,4,5,6,7,8	1

We considered 10000 private GNN queries for each set of experiments, evaluated the proposed algorithms for each of these GNN queries and determined the average experimental results. For each query we generated new set of POIs' locations and users' locations. We ran our experiments on a desktop computer with Intel(R) Core(TM) i7-2600 3.40GHz processor and 8GB RAM.

5 Results and Discussion

In this section, we present our results. We did a comparative analysis with [6], the most recent method. Very few techniques are currently available that consider privacy while

[1] `http://www.cs.utah.edu/$\sim$lifeifei/SpatialDataset.htm`

executing GNN queries. We do not consider [3, 10, 11] for comparison because of their limited privacy protection and extremely high cost as discussed in Sec. 2.1.

We measure *error rate* and *error size* to evaluate accuracy of the methods. Error rate means the percentage of wrong answers an algorithm produces. If POI x is the true GNN and we get POI y as answer using an algorithm then

$$\text{Error size} = \frac{\text{AD of } y}{\text{AD of } x} - 1$$

We report the error size in percent. If x and y are the same POI, then error size is 0%.

We measure computation time and number of rounds to evaluate cost. The number of rounds is the number of times the search space is reduced in H-LOOP.

5.1 Effect of User and POI Location Distribution

We studied the effect of user and POI location distributions by varying their distributions. Other parameters were set to the default values listed in Table 2.

Accuracy: Since H-LOOP and Talouki et al.'s work [6] are the only two approximate privacy preserving methods, we investigated their accuracy. Our experiments used four types of synthetic and three real datasets with default setup of Table 2. Figure 4 shows that H-LOOP outperformed Talouki et al.'s [6] work by three orders of magnitude in all cases. From Fig. 4, Talouki et al.'s [6] method has a probability of 0.45 to incur an error of size 1% or more, whereas in case of H-LOOP this probability is 0.0001. In 8% of the cases, Talaouki et al.'s [6] method incurred errors of size 10% or more, whereas the worst error size of H-LOOP was 7.31%. Mean and standard deviation of H-LOOP's error size were 5.013E-4% and 0.0425%, in contrast to 2.5547% and 4.3728% of Talouki et al.'s method. In practical applications, if the users are spread out over an area of 50 Km diameter, Talouki et al.'s [6] method will incur errors in the order of

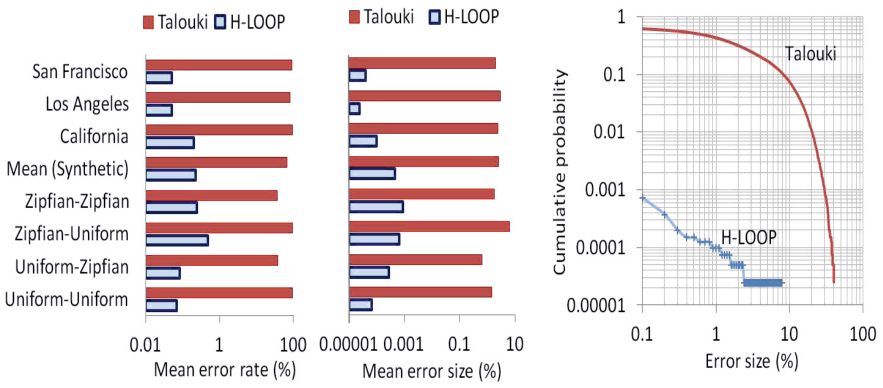


Fig. 4. Error in different datasets

Kms, whereas H-LOOP will incur errors in the order of meters. Even when the users were located in multiple cities of California H-LOOP performed well.

Cost: We randomly selected 2000 POI locations from each real dataset because the computation time depends on the number of POIs. Since LOOP takes into account all POIs, the distribution of user locations and POI locations have no effect on computation time. Overall average computation time per user in LOOP was 0.76ms. Table 3 shows the effect of location distribution on cost of H-LOOP. In H-LOOP, the average round count and computation time were higher when the POIs had Zipfian distribution. The average computation time per user was below one ms in all setups. Even if a smartphone is three orders of magnitude slower than a desktop computer [11], the computation of H-LOOP and LOOP will still take less than one second. This indicates that LOOP and H-LOOP are practical solutions. [In a compatible setup, Huang et al. [7] reported that their method took 4 seconds per user in a decentralized approach and 20 seconds in a centralized approach. These values translate to unrealistic times on today's smartphones. We did not investigate the computation cost of Talouki et al.'s [6] method because we felt that their method is not practically useful due to large errors.]

Table 3. Effect of Location Distribution on Cost of H-LOOP

User distribution	POI distribution	Average round count	Average computation time per user(ms)
Uniform	Uniform	3.00	0.53
Zipfian	Uniform	3.01	0.54
Uniform	Zipfian	3.99	0.68
Zipfian	Zipfian	4.02	0.70
San Francisco	San Francisco	3.35	0.60
Los Angeles	Los Angeles	3.04	0.55
California	California	3.33	0.59

5.2 Effect of Group Size

We studied the effect of group size by varying the group size. Other parameters were set to the default values listed in Table 2. Figure 5 shows that the error decreases with an increase of the group size in H-LOOP. There are two reasons for that. First, from (1) we can say that the number of POIs LOOP can handle while defending an MPAD attack increases rapidly with a group size increase. Thus, H-LOOP prunes the search space less when the group is larger. The number of rounds, as shown in the lower left part of Fig. 5, verifies this fact. A larger search space has a higher probability to include the GNN. Second, our heuristic of estimating a POI's AD with its bounding square's minimum AD works better for larger group sizes. The best error rate and error size of Talouki et al.'s [6] method were 94.1% and 0.68%, respectively.

Figure 5 also shows the effect of the group size on computation costs in H-LOOP. A user's computation cost can be mainly divided into two parts: calculating distances and running PAD. A user has to calculate distances to cells and POIs. The distances to cells are calculated in multiple rounds when H-LOOP is reducing the search space. As the number of rounds decreases with a larger group size, the time spent in calculating distances to cells also decreases. The distances to POIs are calculated in the final round. Since the number of candidate POIs increases, the time spent in calculating distances to POIs also increases. In our experimental setup, a user sends messages to all other users in PAD, which means that the number of messages is proportional to the group size, and the time spent in PAD is higher for a larger group. In case of LOOP, the computation time increased linearly up to 1.3 ms with the increase of group size.

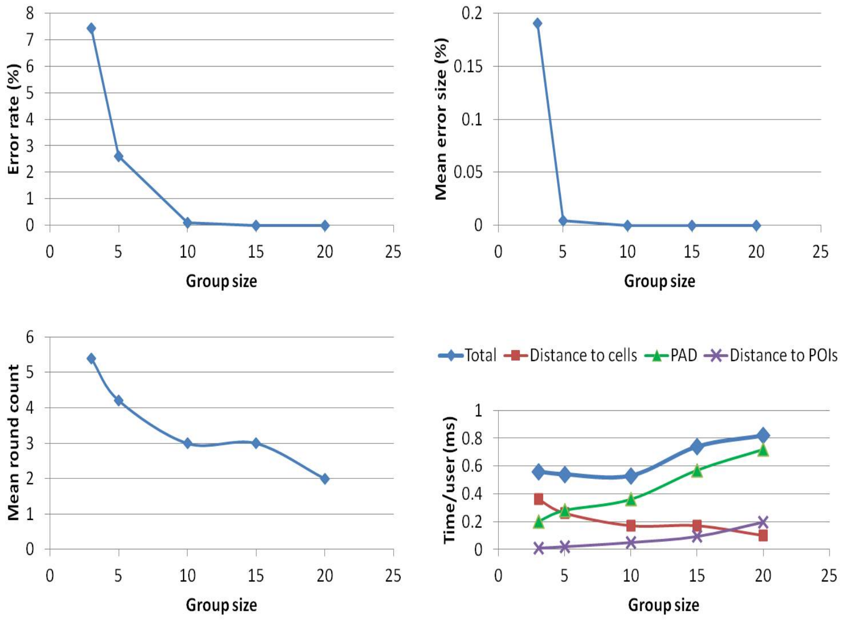


Fig. 5. Effect of group size on H-LOOP

We also studied the effect of the group size while keeping the percentage of colluders constant. In LOOP the POIs are divided into subsets and each subset is assigned to a group member for AD computation. When the the colluder percentage remains constant the subset size also remains constant. However, since there are more users in a larger group, the number of subsets are larger. Therefore, the number of POIs LOOP can handle while resisting an MPAD attack increases with a group size increase. As a result, both error rate and error size are smaller in H-LOOP for larger group sizes. Figure 6 shows the error rate and error size for different group sizes when 40% of the group members are colluders. The figure shows that error rate and error size decreases with the increase of group size.

5.3 Effect of the Number of POIs

We studied scalability by varying the number of POIs. Other parameters were set to default values listed in Table 2. The left part of Fig. 7 shows that the mean computation time per user in H-LOOP increased logarithmically with the increase of POI. H-LOOP reduces the search space exponentially in every round. So the number of rounds increases logarithmically. The communication and computation cost of users are both proportional to the number of rounds. Therefore, the communication and computation cost are logarithmic to the number of POIs. In case of LOOP, the computation time increased linearly up to 18 ms with the increase of POI. The right part of the figure shows that although the error rate increased linearly with the increase of POI numbers, it remained small. Even when the number of POIs is 50000, the error rate was only 1.4%. The mean error size did not show any trend and 1.37E-4% was the largest mean error size. Therefore, H-LOOP is scalable with respect to the number of POIs. The best error rate and error size of Talouki et al.'s [6] method were 98.5% and 1.22%, respectively.

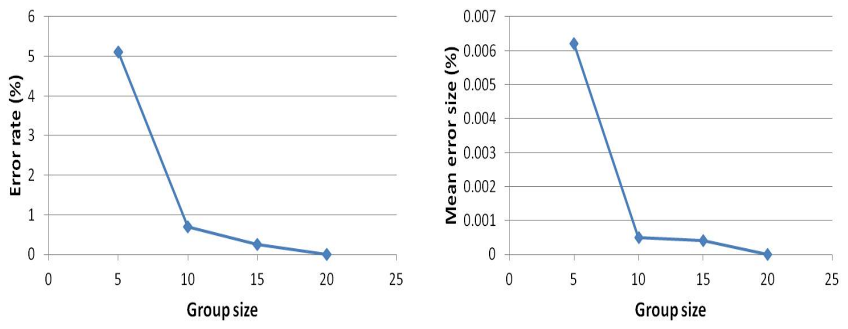


Fig. 6. Effect of group size on error when 40% of the group members are colluding

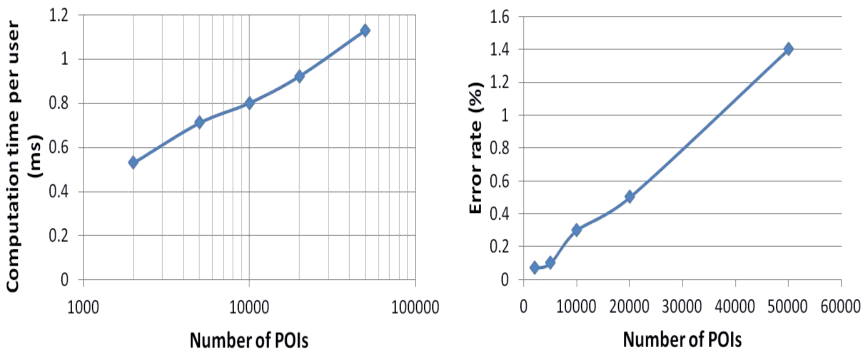


Fig. 7. Scalability of H-LOOP: number of POIs

5.4 Effect on H-LOOP for Different Collusion Group Sizes

We set the value of the division threshold β equal to the N_{max} listed in Table 1 and examined errors at different resistance levels. Other parameters were set to the default values listed in Table 2. Figure 8 shows our experimental results. Although the error increased with an increase of the robustness level, both the error rate and the mean error size remained very low. Even when 8 out of 10 users collude, their attack can be defended easily. In this case, the users travel on average 0.0026% more due to error.

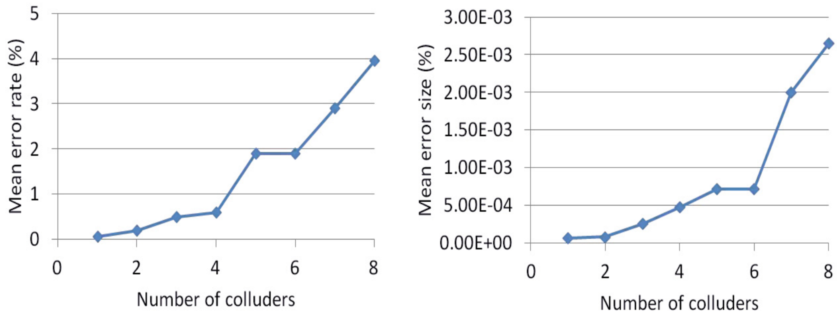


Fig. 8. Error in different resistance levels

Table 4. Practicality of privacy preserving GNN query methods

Algorithms	Resistance to attack	Accuracy	Cost
Hashem [3]	Low	Perfect	Low
Huang [7]	Medium	Perfect	High
Talouki [6]	High	Low	Low
Talouki [10]	Low	Perfect	Low
H-LOOP	High	High	Low

6 Conclusion

We proposed a framework for privacy preserving GNN queries. First, we developed two privacy preserving protocols, i.e., PAD and CWMD in order to privately compute and compare ADs respectively. Then we proposed a basic algorithm, LOOP, and a heuristic approach, H-LOOP, to search for the GNN.

A privacy preserving GNN query algorithm must be strong against attacks, highly accurate and fast to be applicable in real world. In Table 4, we compare H-LOOP, our best method, with existing works with respect to these three criteria based on our findings. We can observe that H-LOOP is a practically applicable solution.

We assumed that users and POIs are located in a 2D Euclidean space without any constraints on movements. In future, we intend to extend our work to road networks.

Acknowledgments. This research was partially supported under Australian Research Council's Discovery Projects funding scheme (project number DP110100757).

References

1. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: MobiSys, pp. 31–42 (2003)
2. Chow, C.Y., Mokbel, M.F., Liu, X.: A peer-to- peer spatial cloaking algorithm for anonymous location based service. In: ACMGIS, pp. 171–178 (2006)
3. Hashem, T., Kulik, L., Zhang, R.: Privacy preserving group nearest neighbor queries. In: EDBT, pp. 489–500 (2010)
4. Duckham, M., Kulik, L., Birtley, A.: A spatiotemporal model of strategies and counter strategies for location privacy protection. In: Raubal, M., Miller, H.J., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2006. LNCS, vol. 4197, pp. 47–64. Springer, Heidelberg (2006)
5. Shokri, R., Freudiger, J., Hubaux, J.P.: A unified framework for location privacy. In: HotPETs (2010)
6. Ashouri-Talouki, M., Baraani-Dastjerdi, A., Aydın Selçuk, A.: GLP: A cryptographic approach for group location privacy. Computer Communications 35(12), 1527–1533 (2012)
7. Huang, Y., Vishwanathan, R.: Privacy preserving group nearest neighbour queries in location-based services using cryptographic techniques. In: IEEE GLOBECOM, pp. 1–5 (2010)
8. Sheikh, R., Kumar, B., Mishra, D.: A distributed k-secure sum protocol for secure multi-party computations. Journal of Computing 2(3), 68–72 (2010)
9. Papadias, D., Shen, Q., Tao, Y., Mouratidis, K.: Group nearest neighbor queries. In: ICDE, pp. 301–312 (2004)
10. Ashouri-Talouki, M., Baraani-Dastjerdi, A.: Homomorphic encryption to preserve location privacy. International Journal of Security and Its Applications 6(4), 183–189 (2012)
11. Huang, Y., Chapman, P., Evans, D.: Privacy preserving applications on smartphones. In: USENIX Workshop on Hot Topics in Security, p. 4 (2011)
12. Mood, B., Letaw, L., Butler, K.: Memory efficient garbled circuit generation for mobile devices. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 254–268. Springer, Heidelberg (2012)