Agent-Based System for Efficient *k*NN Query Processing with Comprehensive Privacy Protection

Mohamad Shady Alrahhal[1], Maher Khemakhem[2], Kamal Jambi[3]
King Abdulaziz University (KAU)
Jeddah, Saudi Arabia

Abstract*—Recently, location based services (LBSs) have become increasingly popular due to advances in mobile devices and their positioning capabilities. In an LBS, the user sends a range of queries regarding his k-nearest neighbors (kNNs) that have common points of interests (POIs) based on his real geographic location. During the query sending, processing, and responding phases, private information may be collected by an attacker, either by tracking the real locations or by analyzing the sent queries. This compromises the privacy of the user and risks his/her safety in certain cases. Thus, the objective of this paper is to ensure comprehensive privacy protection, while also guaranteeing the efficiency of kNN query processing. Therefore, we propose an agent-based system for dealing with these issues. The system is managed by three software agents ($\mathtt{selector}_{DL}$, $\mathtt{fragmentor}_{Q}$, and $\mathtt{predictor}$). The $\mathtt{selector}_{DL}$ agent executes a Wise Dummy Selection Location (WDSL) algorithm to ensure the location privacy. The mission of the $\mathtt{selector}_{DL}$ agent is integrated with the mission of the $\mathtt{fragmentor}_{Q}$ agent, which is to ensure the query privacy based on Left-Right Fragmentation (LRF) algorithm. To guarantee the efficiency of kNN processing, the $\mathtt{predictor}$ agent executes a prediction phase depending on a Cell Based Indexing (CBI) technique. Compared to similar privacy protection approaches, the proposed WDSL and LRF approaches showed higher resistance against location homogeneity attacks and query sampling attacks. In addition, the proposed CBI indexing technique obtains more accurate answers to kNN queries than the previous indexing techniques.

***Keywords*—Agents; attacks; dummies; fragmentation; indexing; privacy protection; resistance**

I. Introduction

Location Based Services (LBSs) are services that are customized according to the location of the user. In recent years, LBSs have received substantial attention, especially since GPS-enabled devices (such as smart phones) became popular. One of the most important advantages of LBS-enabled applications is their ability to search for the nearest Point of Interests (POIs). Searching for the nearest POIs requires construction of a query on the LBS user side. Table I summarizes the units of the constructed query.

TABLE I. General Form of the LBS Query

Symbol	**<X, Y>**	**POI**	**R**	**ID**
Description	Coordinates of the real location	Queried interests	Queried range	The identity of the LBS user

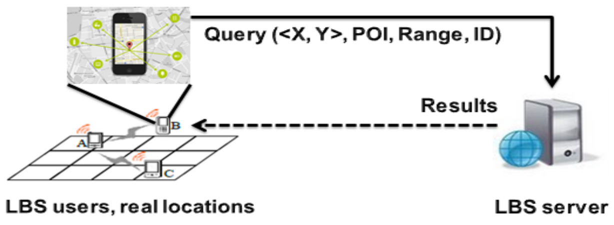


Fig. 1. Classical scenario of using LBS applications.

As a general example of LBS usage, Fig. 1 illustrates the classical scenario of using LBS-enabled applications based on the query units that are listed in Table I.

In Fig. 1, the LBS user constructs a query regarding a desired POI and sends it to the LBS server. Then, the LBS server processes the query and sends back the results. However, this classical scenario involves risk since the LBS user is forced to construct the query based on his/her real geographic location. This risk is directly related to the privacy issue of the LBS user. The reason behind this risk is that an attacker can track the real location of the LBS user [1] or intercept the sent query for analysis purposes [2]. In both cases, the attacker can collect sensitive or personal information about the LBS user, such as customs, habits, religion, or politic leanings. Then, this personal information can be misused to conduct attacks in real life, such as mugging, extortion or stealing. According to [3], these two methods of personal data collection can lead to branches of two kinds of privacy: location privacy and query privacy. Therefore, if we want to achieve full privacy protection, we need to protect these two kinds of privacy. However, achieving comprehensive privacy protection requires protecting the query privacy (in addition to the location privacy) at the sending, processing, and responding levels. Comprehensive LBS privacy protection has not been addressed previously to the best of our knowledge.

The queried POI, are either static POIs (such as the nearest hotels, hospitals, or sports clubs in a defined range) or moving POIs (such as the nearest taxis that will enter a defined range). When an LBS user searches for a moving POI, it is referred to as a range query or k-nearest neighbor (*kNN*) query [4]-[7]. In manipulating *kNN* queries, two major issues arise: The first is related to ensuring the privacy protection of the *kNN* queries, which in turn ensures the privacy of the LBS user. The second is related to guaranteeing the accuracy of the retrieved results (i.e., the retrieved locations of the queried moving POI) [8], [9]. Fig. 2 illustrates the uncertainty problem, which is considered a real-time problem.

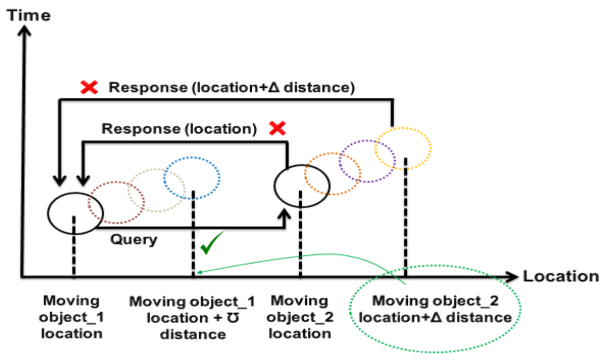

Time
Response (location+Δ distance)
Response (location)
Query
Location
Moving object_1 location
Moving object_1 location + Ʊ distance
Moving object_2 location
Moving object_2 location+Δ distance

Fig. 2. Real-time uncertainty problem for k-NN queries.

According to Fig. 2, the first Moving Object (MO), as a query issuer, searches for a second MO. Because of the continuous updating of the locations of both the 1st MO and the 2nd MO in the real-time temporal and spatial domains, the query issuer will receive an unmatched value that is related to the exact location of the queried MO. The correct response to the query is (location + Δ), which must be delivered to the query issuer location (location + Ʊ).

Motivation. Many efforts have been proposed to protect the privacy of continuous *kNN* queries and overcome the real-time uncertainty problem. One of the most important proposed approaches is the use of dummies. In the context of LBS privacy protection, a dummy is a query that is constructed based on a fabricated location or fabricated properties. If the LBS user surrounds his/her real location by some fabricated (or false) locations, location privacy protection will be achieved [10]-[12]. If the LBS user tampers with the properties of the query itself (changing the queried interest or POI, for example), query privacy protection will be asserted [13]. In both cases, the current query (real query) is mixed with a number of false queries (dummies) so that the attacker cannot recognize the real query among the dummies. This process (i.e., mixing process) aims at achieving k-anonymity in which the attacker cannot identify the real query among k-1 dummies. However, achieving full privacy protection (i.e., location privacy and query privacy) by using dummies has not been addressed. Moreover, generating weak dummies allows an attacker to filter these dummies, thereby determining the accurate location of the LBS user. Beyond generating weak dummies, some inference attacks, such as location homogeneity attack [14] (which targets location privacy) and query analysis attack, such as query sampling attack [15] (which targets query privacy), can be applied by an attacker to circumvent the privacy protection methods. In both inference attacks and query analysis attacks, the attacker does not need to know the accurate location of the LBS user to infer the personal data. This, in turn, means that achieving robust privacy protection is a pressing need. Rregarding the manipulation of *kNN* queries, many techniques have been proposed, such as R*-tree [16], D-tree [17], and Grid-partition [18]. However, these techniques rely on Euclidean space to manipulate the *kNN* queries, whereas, in many real-life applications, the objects' movements are constrained in a road network. Moreover, these techniques cannot be applied in road networks because the network distance (i.e., the shortest path distance) cannot be computed using the boundary of the minimum bounding rectangle (MBR) or grid cell. This, in turn, leads to a poor manipulation of the real-time uncertainty problem for *kNN* queries. Therefore, an efficient technique for manipulating *kNN* queries is a top requirement.

In this paper, based on agent software technology, we propose an agent-based system architecture for privacy protection of LBS users. Three main missions are assigned to three software agents, which are integrated with one another to ensure comprehensive privacy protection of *kNN* queries and overcome the real-time uncertainty problem. The main contributions of this work are as follows:

- To protect the location privacy of LBS users, we introduce a novel Wise Dummy Selection Location (WSDL) algorithm. The objective of our WSDL algorithm is to select strong dummy locations that cannot be distinguished from the real location of the LBS user. The power of the proposed WSDL algorithm comes from taking into consideration two main factors: 1) selecting the dummy locations based on the historical query probability of each cell; and 2) selecting dummy locations that are far away from one another based on the products of the distances among the selected dummies. This, in turn, gives the WSDL algorithm strong resistance against location homogeneity attack.

- To protect the query privacy, we introduce a novel Left-Right Fragmentation (LRF)-based algorithm. Our LRF-based algorithm extracts the sensitive units of the constructed query, encrypts them, and randomizes them to ensure resistance to query sampling attacks.

- To enhance the real-time uncertainty problem, we introduce a novel indexing technique called Cell-Based Indexing (CBI). Our indexing technique performs efficient motion modeling with a prediction phase to ensure that the exact locations of the queried MOs are retrieved.

The rest of this paper is structured as follows: Section II discusses related work. The threat model is provided in Section III. Our proposed agent-based architecture is provided in Section IV. Section V discusses the security analysis. In Section VI, we present the metrics that are used. Section VII presents our experimental results and the conducted evaluations. Finally, we conclude the paper in Section VIII.

II. Related Work

This section reviews some of the related work on privacy protection approaches in the LBS research field. In addition, we discuss some of the related work on techniques that are used to manipulate *kNN* queries.

A. *LBS Privacy Protection Approaches*

Many efforts have been made to classify the privacy protection approaches in the domain of LBS, such as [3], [19], [20]. There are two major categories of LBS privacy protection approaches: server-based approaches and user-based approaches. In this subsection, we review some existing approaches from the user-based category that aim at protecting location privacy or query privacy.

The authors of work [10] proposed a dummy data array (DDA) algorithm for generating dummy locations to protect the location privacy of LBS users. For a given region, which is divided into a grid of cells, the key idea of the DDA algorithm is to calculate both the vertices and the edges of each cell in the grid. Then, the DDA algorithm randomly selects some of the cells as dummy locations. To select strong dummy locations and achieve k-anonymity, the DDA algorithm selects k cells of equal area. Similarly, [11] uses dummies to protect the location privacy of LBS users, but with a different dummy generation method. The authors proposed two algorithms. The first is called CirDummy, which generates dummies based on a virtual circle that contains the real location of the LBS user. The second is called GridDummy, which generates dummies based on a virtual grid that covers the real location of the LBS user. In [12], a dummy generation method called the Destination Exchange (Dest-Ex) method was proposed. In this method, historical motion trajectories are used to generate the dummies. To ensure that the generated dummies are strong, the Dest-Ex method chooses the historical trajectories that intersect with the current trajectory of the LBS user. Therefore, the attacker is confused when trying to determine the correct LBS user, who has several motion trajectories with different destinations. However, the main objective of all of these previous approaches was location privacy protection. To achieve query privacy protection, the authors of [13] proposed an approach called DUMMY-Q. The DUMMY-Q approach depends on the strategy of generating dummies, but the strategy is applied to the query, rather than the location. Therefore, dummy queries of different attributes from the same location are generated to hide the real query. To make the generated dummies stronger, two aspects are taken into consideration: 1) the query context; and 2) the motion model.

Encryption techniques have been employed to protect the privacy of LBS users. The authors of [21] proposed the idea of using buddies to protect both location privacy and query privacy against the LBS server (a malicious party). This approach depends on notifying the friends (buddies) of an LBS user who are located in the vicinity, thereby avoiding the revelation of any personal data to the LBS server. This approach assumes that each user shares a secret with each of his buddies and uses symmetric encryption techniques. Another approach was proposed based on using Private Information Retrieval (PIR) [22] to achieve full privacy protection. The key idea of the PIR technique depends on the quadratic residuosity assumption, which states that it is computationally hard to find the quadratic residues in modulo arithmetic of a large composite number for the product of two large primes. Therefore, the LBS server can process and answer the query without knowing any sensitive information about the query.

B. *Techniques of kNN Query Manipulation*

The Global Positioning System (GPS), which is integrated with the mobile devices of the LBS users, allows the users to obtain their locations from the satellite and send them to the LBS server. During movement, the locations of the LBS users are continuously updated on the LBS server side. This results in inaccurate retrieved locations when the LBS user asks for the *kNN* MOs as POIs. Therefore, the final goal of any techniques that is used for manipulating the *kNN* queries is to retrieve approximate locations of the MOs as responses to the *kNN* queries.

Many techniques have been proposed for manipulating the *kNN* queries. In [16], a traditional method called P*-tree was proposed for supporting range queries. The P*-tree technique efficiently manipulates range queries with static POIs, but not moving POIs. Another technique was provided in [17], which is called D-tree. The key idea of D-tree is to index the data regions based on the divisions among them so that a binary D-tree index is constructed. For a given *kNN* query, two main phases are used to find and retrieve the queried POIs: region partitioning and location-dependency query processing based on paging the D-tree index. The authors of [18] developed the D-tree technique, proposing a Grid-partitioning technique. The authors used the Voronoi Diagram to partition the service area into disjoint Voronoi cells (VCs), with each corresponding to one object. An object a, is guaranteed to be the nearest neighbor to any client that is located inside the same VC. In [23], a new *kNN* query processing technique was proposed by Jang et al. based on the density of the POIs. A PIR protocol was used to search for the POIs within a clocking region, so that the clocking region was expanded to overlap other regions based on the k-d overlap index. However, in all the previous techniques, the index is constructed for large regions, thereby ignoring the cells that are included in the divided regions.

III. Threat Model

In this section, we define the threat model, which specifics the attacker and his/her objective. In addition, we determine the ways that are used by the attacker to collect personal information about the victim, in addition to inference and analysis attacks.

A. *Attacker and His/Her Objective*

The objective of the attacker is to obtain privacy information about a particular LBS user, including location, POI and queried range. To achieve his/her objective, the attacker can track the location of the LBS user or analyze the sent query, as shown in Fig. 3 below.

In the context of the threat model, we define two terms: passive attack and active attack. In a passive attack, any LBS user can act as an attacker. In an active attack, the LBS server (or its maintainer) is an attacker and all the information (related to the trajectories of the LBS user's motion) that is stored in the LBS server is accessible. Since an active attack is stronger than a passive attack, we only address active attack.

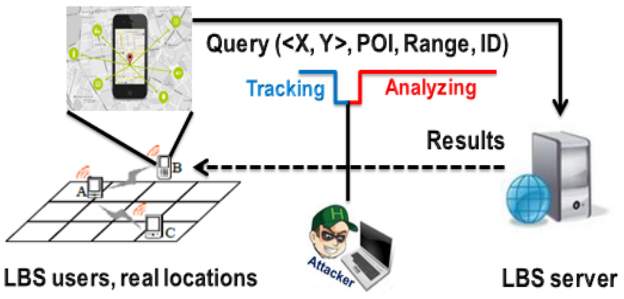


Fig. 3. Attacking the privacy of LBS users.

B. Inference Attacks and Query Analysis Attacks

The LBS server (an attacker) can apply inference attacks, such as location homogeneity attack, and query analysis attacks, such as query sampling attack.

In a location homogeneity attack, the attacker analyzes the locations of all LBS users. If their positions are almost identical, then the position information of each member is revealed. For instance, if the users are located in a place that represents a landmark such as a hospital, the attacker can infer that those users (including the victim) have problems related to their health, without needing to accurately identify their locations. Fig. 4 illustrates a location homogeneity attack.

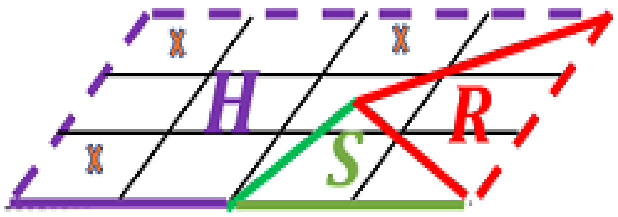


Fig. 4. Location homogeneity attack: (H) hospital or medical area, (S) sprot cub or athletic area, (R) restaurant or rest area.

In a query sampling attack, the attacker employs the uneven location distribution of the LBS users for his own malicious purposes. This attack targets isolated users in sparse regions, as illustrated in Fig. 5. Therefore, it relies on the traffic statistics of the environment where the users are located. In detail, the attacker tries to calculate a probability distribution function of the user location over a given area. If the distribution is not uniform, then the attacker can determine the areas where the user is located with a high probability. Once the location of the victim is determined, the attacker focuses on analyzing the sent queries.

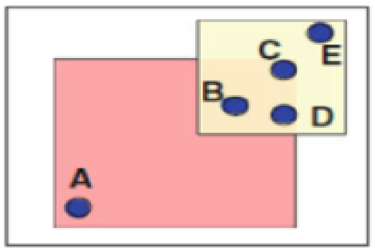


Fig. 5. Query sampling attack.

IV. Our Proposed Privacy Protection Architecture

In this section, we provide our agent-based privacy protection architecture, followed by the roles of the agents. The details of the architecture are represented by a sequence diagram.

The framework of the proposed architecture consists of an untrusted LBS server (a malicious party) and a group of mobile devices, which are connected via a network. The system is managed by three agents ($selector_{DL}$, $fragmentor_{Q}$, and $predictor$), as shown in Fig. 6.

Table II lists the agents and identifies the main mission of each one, its type, and where it is installed.

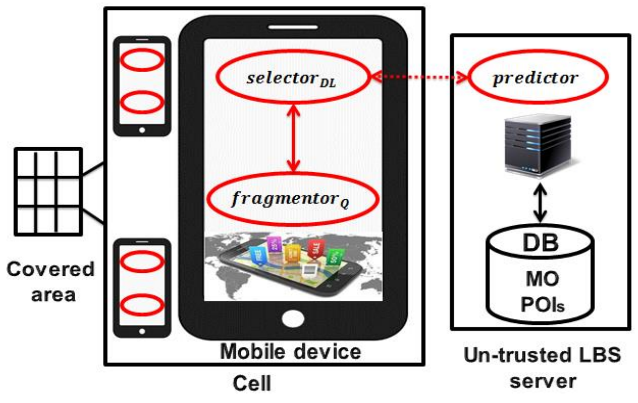


Fig. 6. Our agent-based architecture.

TABLE II. Agents

Agent Name	Type	Main Mission	Location
Selector_{DL}	Stationary	Location privacy protection	Each mobile device
Fragmentor_{Q}	Mobile	Query privacy protection	Each mobile device
Predictor	Stationary	Uncertainty real-time problem solution	LBS server

A. Roles of the Agents

$\boldsymbol{Selector_{DL}}$**:** This stationary agent executes the Wise Dummy Selection Location (WSDL) approach. It targets the location privacy protection against the untrusted LBS server, which can apply location homogeneity inference attack, as described below.

1) Wise Dummy Selection Location (WDSL) approach

The final objective of the WSDL approach is to generate strong dummy locations to protect the location privacy of the LBS user. In the dummy generation process, suitable locations are selected that cannot be distinguished from the real location of the LBS user. Consider a region (G) divided into a grid of cells. Each cell has a probability of being queried, which is based on past queries. This is referred to as the query probability. For a given LBS user in a cell within G, randomly selecting cells to be the dummy locations, as proposed in the DDA approach [10], for an example, it is a poor strategy. In contrast, selecting the cells (to be a dummy locations) that have the same query probabilities as the cell where the LBS user is located is an efficient solution. Fig. 7 illustrates this solution, where G is divided according to the coordinates (X, Y).

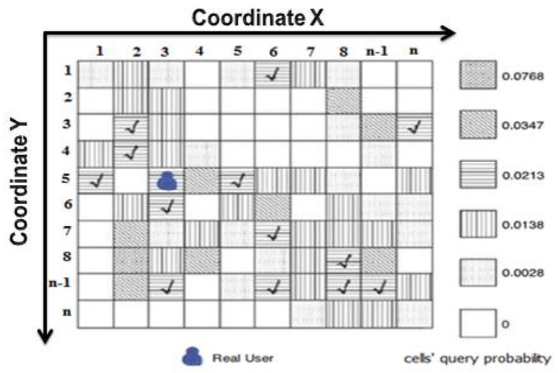


Fig. 7. Dummy locations selection in the WSDL approach.

In Fig. 7, if the LBS user who is located in the cell that is identified by row number five and column number three (i.e., the coordinates C[5, 3]) wants to protect his/her location privacy by achieving 4-anonymity level (i.e., k=4), he/she can select three of the cells that are marked by the √ symbol. Since the query probability of any of the three selected cells equals the query probability of the original cell, the attacker cannot determine the real location of the LBS user among the k-1 dummy locations.

In a formal way, for a given region (G) that is divided into $(n \times n)$ cells, let (qp) refers to the query probability of a cell. Then, $\sum_{i=1}^{n^2} \mathrm{qp_i} = 1$. Each of the k locations (i.e., cells) that are contained in a query, which include one real location and $(k-1)$ dummies, has a conditional probability of being the real location. Let $\acute{p}_i$ $(i = 1, 2, \dots, k)$ denote the probability that the i$^{\text{th}}$ location is the real location. Then, $\acute{p}_i = \frac{qp_i}{\sum_{j=1}^{k} qp_i}$.

The entropy (E) of identifying the real location out of the dummy set is defined as:

$$E = -\sum_{i=1}^{k} \acute{p}_i \times \log_2 \times \acute{p}_i \qquad (1)$$

The first factor that is taken into consideration is the maximization of the entropy value in the dummy selection process.

$$\mathrm{Max}\,(-\sum_{i=1}^{k} \acute{p}_i \times \log_2 \times \acute{p}_i) \qquad (2)$$

2) *Danger of location homogeneity inference attack*

If the LBS user selects cells C[5, 1], C[4, 2], and C[6, 3], as shown in Fig. 8, some personal information can be inferred by the attacker without the need to determine the real location of the LBS user. This occurs because the three selected dummy locations are close to one another. If these selected dummy locations belong to a medical area (which includes hospitals as a POIs, for example), then the attacker can infer that the LBS user has a health problem. Therefore, it is better to select the following three cells, for example, C[3, n], C[n-1, n-1], and C[1, 6].

To defend against location homogeneity attacks, a second factor is taken into consideration in the process of dummy location selection: "the selected dummy locations must be far away from one another". In this context, the question arises as to how to determine the furthest dummy location from the real location of the LBS user and spreads away from the other dummy locations. This can be accomplished by calculating the distance between the real location of the LBS user and each dummy location based on the product distance rather than the normal sum distance. Fig. 8 illustrates the strategy of wise dummy location selection.

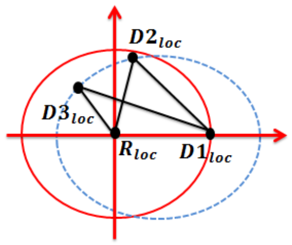


Fig. 8. Wise dummy location selection.

In Fig. 8, R_{loc} represents the real location of the LBS user, and $D1_{loc}, D2_{loc}$, and $D3_{loc}$ represent the dummy locations, where the query probability of each dummy location equals the query probability of the real location. Let the distance between two points PT_i and PT_j be given by $\sum_{i \neq j} \mathrm{dis}(PT_i, PT_j)$. $D1_{loc}$ is the first dummy location that can be directly selected since it is the furthest location from R_{loc}. If we want to achieve $3-$anonymity level, we can choose $D2_{loc}$ or $D3_{loc}$. If we consider the sums of distances between pairs of dummy locations, we can choose either of them ($D2_{loc}$ or $D3_{loc}$) because ($|D2_{loc}R_{loc}| + |D2_{loc}D1_{loc}|$ = ($|D3_{loc}R_{loc}| + |D3_{loc}D1_{loc}|$). However, to achieve higher resistance $D2_{loc}$ is preferred over $D3_{loc}$ since it spreads dummy locations farther. Therefore, instead of using the sum of distances between pairs of dummy locations, we can use their product. Note that ($|D2_{loc}R_{loc}| \times |D2_{loc}D1_{loc}|$ > ($|D3_{loc}\,R_{loc}| \times |D3_{loc}\,D1_{loc}|$). This leads to the choice of $D2_{loc}$ as the second dummy location.

Mathematically, the two previous factors form two objectives in a Multi-Objective Optimization Problem (MOP). Let $DL = [D1_{loc}, D2_{loc}, D3_{loc}, \dots, Dk_{loc}]$ denote the set of real and dummy locations. The MOP is defined as:

$$FD_{loc} = \arg\max \left\{-\sum_{i=1}^{k} \acute{p}_i \times \log_2 \times \acute{p}_i\;, \prod_{i \neq j} \mathrm{dis}(Di_{loc}, Dj_{loc})\right\} \qquad (3)$$

Where, FD_{loc} represents the final selected dummy locations.

The first objective of the MOP was previously optimized in formula 2 because, from all the given dummy locations (i.e., all cells that form the region G), we select a set of dummy locations based on similarity of query probability. This set is called the set of candidate dummy locations (CD_{loc}), which yields the maximum entropy value. Out of the candidate dummy locations, we optimize the second objective of the MOP as follows, which determines the final selected dummy locations:

$$FD_{loc} = \arg\max \left\{\prod_{i \neq j} \mathrm{dis}(Di_{loc}, Dj_{loc})\right\} \qquad (4)$$

In steps, we first sort the cells according to their query probabilities. Second, we select 4k cells from outside the queried range (R) of the real query (k cells from each direction around the real location of the LBS user R_{loc}). All 4k selected cells have the same query probability as the cell of the real location of the LBS user. The 4k selected cells form the candidate set of dummy locations. Third, out of the candidate set, we randomly select the furthest $(k-1)$ cells as the actual and final dummy locations. Algorithm 1 provides details of the WSDL approach.

Algorithm 1: Wise Dummy Selection Location (WDSL)

Input: qp (query probability of each cell), R_{loc} (the real location of the LBS user), k (anonymity level).

Output: FD_{loc}.

```
1: sort cells based on their query probabilities;
2: for (direction=1; direction <4; direction ++)
3:      $CD_{loc} = FD_{loc} = \emptyset$;
4:      select k cells from each direction around $R_{loc}$;
5:      Count ← 0;
```

```
6:          while (count candidate $\neq$ k)
7:              if ($qp(C_i) = qp(R_{loc})$) then
8:                  $CD_{loc} \leftarrow CD_{loc} \cup C_i$;
9:                  Count←count + 1;
10:             end if
11:         end while
12:         for ($i = 1; i \leq length\ (CD_{loc}); i++$)
13:             Dis-Array-core[i] ← calculate $distance\ (C_i, R_{loc})$;
14:         core candidate ← max (Dis-Array-core);
15:         for ($j = 1; j \leq length\ (CD_{loc}); j++$)
16:             $dis_1 = dis(core\ candidate, candidate_j)$;
17:             $dis_2 = dis(R_{loc}, candidate_j)$;
18:             Dis-Array[j] ← $dis_1 \times dis_2$;
19:         end for
20:         Selected-Dummies [direction] ←
21:                 {Top (Sort (Dis-Array) , $\frac{k-1}{4}$) ∪ core candidate};
22: end for
23: $FD_{loc} \leftarrow \bigcup_1^{direction=4}$ Selected − Dummies [direction];
24: output $FD_{loc}$
```

After generating the final $(k-1)$ dummy locations, the $selector_{DL}$ agent delivers them (as a set of coordinates) to the $fragmentor_Q$ agent to start its mission, as described below.

$\boldsymbol{Fragmentor_Q}$**:** The final goal of this mobile agent is to protect the privacy of the issued query during the sending and processing phases. To complete this mission, the $fragmentor_Q$ agent constructs k queries ($k-1$ queries based on the $k-1$ dummy locations that are received from the $selector_{DL}$ agents, plus the query based on the real location of the LBS user). Then, it executes a fragmentation approach called Left-Right-Fragmentation (LRF) to protect the privacy of each constructed query. After that, it migrates to the LBS server, carrying the protected queries, which are manipulated and answered there with the help the $predictor$ agent. After the queries are answered on the LBS server side, the $fragmentor_Q$ migrates back to the home machine (i.e., the mobile device of the LBS user) to deliver the results.

3) *Left-Right-Fragmentation (LRF) approach*

The $fragmentor_Q$ agent receives the set of actual dummy locations that were generated by the $selector_{DL}$ agent. Each dummy location has its own coordinates (X, Y). Let $FD_{loc-coor}$ denote the set of the coordinates of the generated dummy locations, where:

$$FD_{loc-coor} = \{\langle X_1, Y_1\rangle, \langle X_2, Y_2\rangle, \langle X_3, Y_3\rangle, \ldots, \langle X_{k-1}, Y_{k-1}\rangle\} \quad (5)$$

For each coordinate $\langle X_i, Y_j\rangle \in FD_{loc-coor}(i, j = 1, 2, \ldots, k-1)$, a query is built according to the format that is specified in Table I, which consists of the following units: coordinates of the LBS user $\langle X, Y\rangle$, queried interest POI, queried range R, and identity of the LBS user ID. Each constructed query is referred to as an original query.

The key idea of the fragmentation technique is to extract the sensitive data from the query, encrypt them, and then randomize them, as shown in Fig. 9.

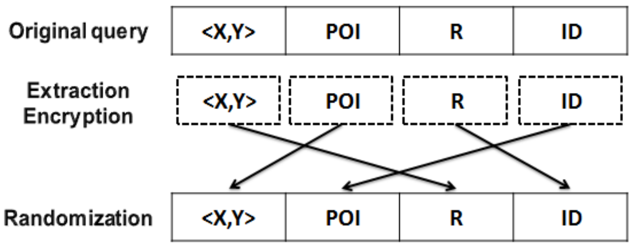


Fig. 9. Fragmentation technique.

In the context of fragmentation, we address the sensitive units of the query and the sensitive associations among the units because the attacker focuses on either one unit or the associations among two or more units to infer personal information. For instance, if the LBS user always queries the nearest hospitals as POIs, then the attacker can infer that the LBS user has a health problem. Meanwhile, if the attacker associates the ID of the LBS user with the queried POIs, then he/she can accurately identify the LBS user who has a health problem. Therefore, protecting the sensitive association is more important than protecting the sensitive units.

In this paper, the sensitive units of a given query are ($\langle X, Y\rangle$, POI, and R). For the LBS user ID, it is not considered a sensitive unit because the attacker cannot gain any private information from the ID unit alone. Moreover, even if the attacker associates the ID unit with any of the other units, he/she will fail to gather private information due to the encryption and randomization processes. Thus, if the attacker applies a query analysis attack, he/she will obtain, for instance, the following information: "the LBS user whose ID is (Bob-1) issues a query from an unknown location that asks for nameless POIs that are located in non-existent range R". This statement does not reveal any private information.

In detail, the sensitive units are protected by public key infrastructure (PKI) and the sensitive associations are protected by a randomization phase. This forms our proposed Left-Right-Fragmentation (LRF) algorithm.

Formally, for a given set of queries $Q = (q_1, q_2, \ldots, q_i)$, where $(i = 1, 2, \ldots, k-1)$, encoding a query (q_i) consists of splitting it into two main parts: the left part $(P^l_{q_i})$ and the right part $(P^r_{q_i})$. Both parts are necessary for reconstructing the original query q_i.

$$q_i = P^l_{q_i} \circ P^r_{q_i} \quad (6)$$

In the first step of the randomization phase, we place the ID unit in the middle since it is not considered a sensitive data, as shown in Fig. 10.

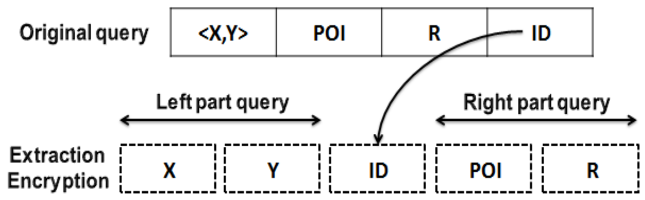


Fig. 10. First step in the randomization phase of the LRF-based algorithm.

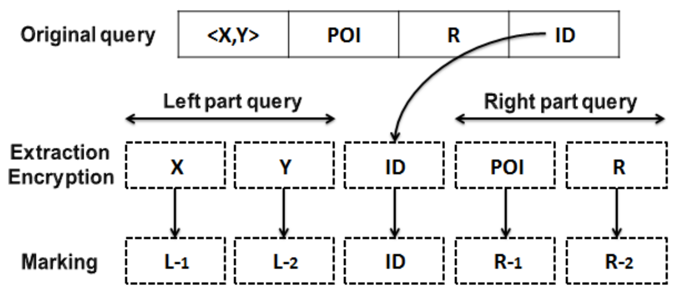


Fig. 11. Marking step in the randomization phase of the LRF-based algorithm.

The left part of the query includes one unit, which consists of two sub-units (X, Y). As for the right part query, it includes two units: (POI and R). In the second step of the randomization phase, we mark each unit or sub-unit by a (letter-number) pair that indicates the correct order in the original query, as shown in Fig. 11.

In Fig. 11, for instance, R-1 indicates that the encrypted unit (POI) must be placed directly to the right of the ID unit.

Since we have five different sites for ordering the units of the original query, there are ($1 \times 2 \times 3 \times 4 \times 5 = 12$ probable sites) for randomizing the original units. Thus, the $fragmentor_Q$ agent can periodically change the randomization strategy, which prevents the attacker from discovering the correct order of the original query's units. Fig. 12 illustrates one possible choice and the reconstruction process.

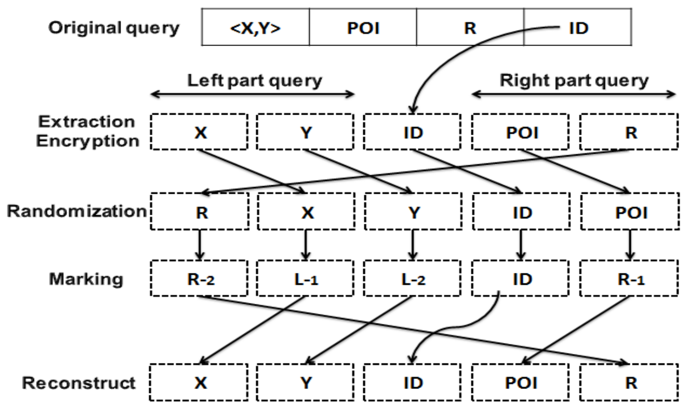


Fig. 12. Left-Right-Fragmentation (LRF) algorithm and reconstruction.

Reconstructing the original query is necessary for the stationary $predictor$ agent to perform its mission (i.e., manipulating and answering the queries). The reconstruction process is carried out in four steps:

1) Putting the ID unit in the middle.
2) Decrypting the units of the query based on the shared encryption key between the $fragmentor_Q$ agent and the $predictor$ agent.
3) Performing the marking step.
4) Moving the ID unit to the end.

Algorithm 2 illustrates the details of the left-right fragmentation approach.

Algorithm 2: Left-Right Fragmentation (LRF)

```
Input: kNN (⟨X,Y⟩, POI, R, ID) query.
Output: protected kNN (⟨X,Y⟩, POI, R, ID) query.
1: units{} = extract (X, Y, POI, R);obtaining the sensitive data.
2: units{} = encrypt (units )using 3DES algorithm;
3: new-units{}=null;
// randomization
4: count =0; rand-array [5] ={-1};
5: while (count <5)
6:         random-value = rand(4);
7:         if (! contains (rand-array, random-value))
8:                 rand-array[count] = random-value;
9:                 Count ++;
10:        end if
11: end while
12: for (i=0; i<5;i++)
13:    new-units {i}= units{ rand-array[i] };
14: Return new-units;
```

By the LRF-based algorithm, all queries that are constructed on the LBS user side are protected before being sent to the LBS server. Then, all the queries are packaged and carried together by the $fragmentor_Q$ agent to the LBS server, which in turn means that the queries are protected during the sending phase. Because the LRF-based algorithm mixes the real query with $k-1$ dummy queries (which are constructed based on the $k-1$ dummy locations and selected by the $selector_{DL}$ agent) and the mission of manipulating the queries is assigned to the $predictor$ agent, the queries are protected during the processing phase. The task of protecting the queries during the responding phase is included in the role of the $predictor$ agent.

$\boldsymbol{Predictor}$: This stationary agent receives the k queries that were constructed and carried by the $fragmentor_Q$ mobile agent. Then, it manipulates each query individually. After answering the received queries, the results are delivered to the $fragmentor_Q$ mobile agent, which, in turn, migrates back to the mobile device of the LBS user (i.e., the home machine). The process of manipulation requires the reconstruction of the k protected queries. This is performed according to the four steps that are listed above, where the same shared encryption key as was used to encrypt the units of the queries is used for decryption. After reconstructing the queries, the $predictor$ agent manipulates each query according to an indexing technique, as described below.

4) *Cell-Based Indexing (CBI) technique*

In the *kNN* queries, the LBS user asks for the nearest k moving POIs that are located within a specified range R of the LBS user. Because of the continuous updatings of the locations of the moving POIs, the locations of the queried moving POIs are updated during the sending and processing of the queries. In addition, the location of the query issuer is also updated since it is considered an MO. Therefore, we need to retrieve the new exact locations of the queried moving POIs, and these new locations must be delivered to the new exact location of the query issuer. To achieve this, we model the motion of the moving POIs first. Then, the $predictor$ agent indexes the moving POIs and, based on their indices, predicts their new locations.

The given region (G), which is divided into $(n \times n)$ cells of equal size, is modeled as an undirected graph $\mathrm{GR}(\mathrm{H}, \mathrm{A})$, where H represents the headers, and A represents the arms. The numbers that are associated with the arms denote weights (W), which represent the physical distances between two headers, as shown in Fig. 13.

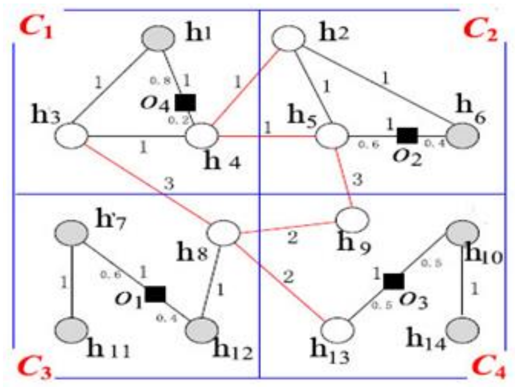


Fig. 13. Modeling the motion of the POIs.

In the context of the model, the terms path, boundary path, and MO are defined as follows:

Definition 1. For a given undirected graph $\mathrm{GR}(\mathrm{H}, \mathrm{A})$, a path P between a start header $\mathrm{h_s}$ and end header $\mathrm{h_e}$, which is illustrated as a black line in Fig. 13, is expressed by the following formula:

$$\mathrm{P}(\mathrm{h_s}, \mathrm{h_e}) = \{ \mathrm{h_s}, \mathrm{h_{p1}}, \mathrm{h_{p2}}, \dots, \mathrm{h_{pm}}, \mathrm{h_e} \} \tag{7}$$

where, $\mathrm{h_{pm}}$ represents a sub path in the case in which there exist many headers from the start to the end.

Definition 2. For a given path P, a path is called a boundary path, if its start header $\mathrm{h_{ps}} \in \mathrm{C_i}$ and its end header $\mathrm{h_{pe}} \in \mathrm{C_j}$, where C represents a cell and $\mathrm{i} \neq \mathrm{j}$ (i.e., passing from one cell to another). Boundary paths are shown as red lines in Fig. 13.

Definition 3. For a given path P, an MO that is located on a path at time t is expressed by the following triple:

$$\mathrm{mo^t} = < \mathrm{cl}, \mathrm{p}, \mathrm{d} > \tag{8}$$

where cl denotes the current location of the MO, p denotes the path that is linked to the MO, and d denotes the direction of the MO from the start header to the end header.

Based on the previous three definitions, four neighboring cells are shown in Fig. 13. The MOs are illustrated as black boxes. $\mathrm{C_1}$ contains three boundary paths ($\{\mathrm{h_4}, \mathrm{h_2}\}$, $\{\mathrm{h_4}, \mathrm{h_5}\}$, $\{\mathrm{h_3}, \mathrm{h_8}\}$), which are weighted as 1, 1, and 3, respectively. The moving object $\mathrm{mo_1}$ resides in $\mathrm{C_3}$ on the path between $\mathrm{h_7}$ and $\mathrm{h_{12}}$, and moves in the direction of $\mathrm{h_{12}}$, with a distance of 0.4.

Based on the model that was presented above, the *predictor* agent creates and manages an index at the cell level. This index includes two parts: an index part and a data part, as shown in Fig. 14.

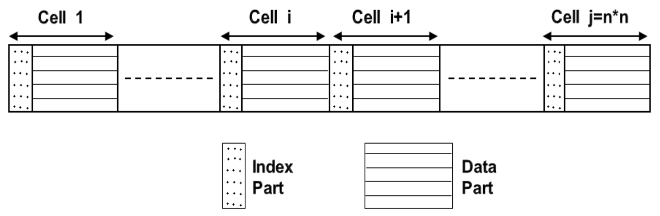
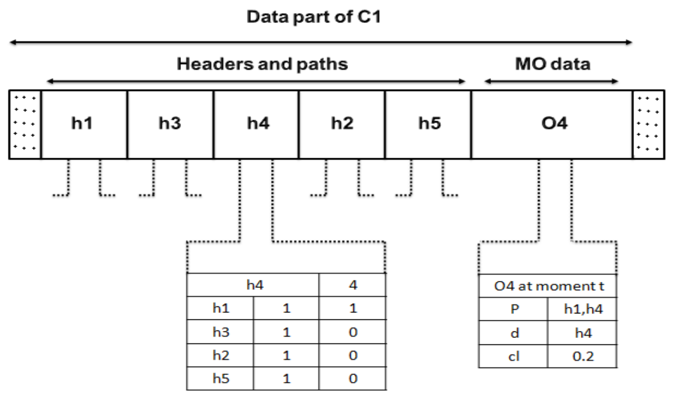


Fig. 14. General structure of CBI.



h4		4
h1	1	1
h3	1	0
h2	1	0
h5	1	0

O4 at moment t	
P	h1,h4
d	h4
cl	0.2

Fig. 15. Data part structure of $\mathrm{C_1}$.

The data part holds detailed information about both the cell and the MOs that are located within the cell. This information mainly includes headers, paths, numbers of MOs on the paths, and data about the MOs, as illustrated in Fig. 15.

Two tables are shown in Fig. 15. The first record in the left table indicates that there are 4 headers that are linked to the header $\mathrm{h_4}$ and contained in $\mathrm{C_1}$, which form two paths ($\{\mathrm{h_4}, \mathrm{h_1}\}, \{\mathrm{h_4}, \mathrm{h_3}\}$) and two boundary paths ($\{\mathrm{h_4}, \mathrm{h_2}\}, \{\mathrm{h_4}, \mathrm{h_5}\}$). The rest of the records carry information about the physical distances (or weights) of the formed paths/boundary paths and the number of MOs on each. For example, the second record states that $\mathrm{w}(\mathrm{h_4}, \mathrm{h_1}) = 1$, and this path has one moving object. The right table states that moving object $\mathrm{O_4}$ moves on path $\mathrm{p} = \{\mathrm{h_4}, \mathrm{h_1}\}$ towards $\mathrm{h_4}$, and its current location is a distance of 0.2 from $\mathrm{h_4}$.

Based on the information that is related to the MO ($\mathrm{O_4}$ in the right table of Fig. 15), the *predictor* agent can calculate the speed of the MO based on its two previous consecutive locations and moments, as follows:

$$\mathrm{speed} = \frac{\mathrm{distance}}{\mathrm{time}} = \frac{|\,\mathrm{cl\ at\ the\ moment\ } t_2 - \mathrm{cl\ at\ moment\ } t_1\,|}{t_2 - t_1} \tag{9}$$

After calculating the speed, the *predictor* agent can estimate the future location of the MO by calculating the Δ distance (illustrated in Fig. 2 in the introduction section) and adding it to the current location of the MO, taking into consideration the direction of the MO.

The index part of a given cell contains the cell identifier ($\mathrm{C_{i,j}}$); the area of the cell, which is represented by the width of the cell ($\mathrm{wth^2}$); and the number of MOs that are located in the cell. In addition, it includes the same previous information about the eight cells that surround the given cell, as shown in Fig. 16.

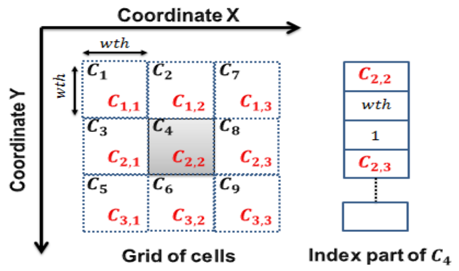

Coordinate X
Coordinate Y
wth
wth
C_1 $C_{1,1}$ C_2 $C_{1,2}$ C_7 $C_{1,3}$
C_3 $C_{2,1}$ C_4 $C_{2,2}$ C_8 $C_{2,3}$
C_5 $C_{3,1}$ C_6 $C_{3,2}$ C_9 $C_{3,3}$
Grid of cells
$C_{2,2}$
wth
1
$C_{2,3}$
Index part of C_4

Fig. 16. Index part structure of C_4.

The index part will be the input of a bloom filter [24]. The benefit of the bloom filter is that it can give a direct answer regarding the existence or non-existence an element within a set. We exploit this to determine whether there is an MOs within the cells that are covered by the range R, which is specified in the *kNN* query. If no MOs are found in a cell, it is not necessary to search inside the cell. Thus, we can move to the next cell. Thisgreatly speeds up both the response time and the processing time of *kNN* queries since time is not wasted on examining empty cells.

In detail, for a given *kNN* query with a range R, we first determine which cells are covered by R. Then, the index part of each cell is used to determine which contain the queried MOs using the bloom filter. For only the cells that have MOs, the actual search is performed on the data part of each limited cell with a prediction phase; to retrieve the future locations of the queried MOs. Algorithm 3 illustrates the steps of processing a *kNN* query based on the proposed CBI technique.

After retrieving the results (i.e., the predicted locations of the queried MOs), the $predictor$ agent encrypts the results and delivers them to the $fragmentor_Q$ agent. The $fragmentor_Q$ mobile agent migrates back to the home machine to deliver the results to the LBS user. The process of encrypting the results ensures the privacy protection of the queries during the responding phase. Algorithm 4 describes the itinerary of the $fragmentor_Q$ mobile agent.

Algorithm 3: CBI based *kNN* query processing

```
Input: cells, $R_{loc}$ real location , range $R$.
Output: POIs [] //moving objects.
1: covered-cells[]=null;
2: for (i=1; i<=count(cells); i++)
3:     distance $_{R_{loc},Cell_{loc}} = \sqrt[2]{(Rx_{loc} - Cellx_{loc})^2 + (Ry_{loc} - Celly_{loc})^2}$;
4:     if (distance $_{R_{loc},Cell_{loc}} \leq R$)
5:         add (cell[i], covered-cells);
6: end for
7: foreach cell in covered cells
8:     if (bloom (index-part of cell))
9:         fetch (data-part of cell)
10:        foreach path in data-part
11:            if (path contains MO)
12:                future-cell=prediction (MO);
13:                add(future-cell, POIs);
14:            end if
15:        end foreach
16:    end if
17: end foreach
18: return POIs;
```

Algorithm 4: Trip of $fragmentor_Q$ mobile agent

```
Input: $kNN$ query.
Output: report results.
1: agent = new $fragmentor_Q$(); (create an agent)
2: itinerary = new itinerary ();
3: itinerary.Adddistenation ( "LBS server", "execute
   encryption method");
4: itinerary.Adddistenation ( "LBS mobile device", "execute
   report results method");
5: output: report results;
```

Since the query issuer (i.e., the LBS user) is an MO, his/her location changes during the sending and processing the query. Therefore, the results must be delivered to the query issuer according to his/her new location. Because the $fragmentor_Q$ is a mobile agent that is created in the mobile device of the LBS user, which represents the home machine, it must return back to the same home machine without any additional predictions on the location of the query issuer, as shown in Algorithm 4. Therefore, the future locations of the queried MOs are calculated in the prediction phase, while the future location of the query issuer is naturally obtained due to the returning step in the itinerary. In other words, it is not necessary to compute the Ʊ distance. As a result, the two parts of the real-time uncertainty problem are solved, as shown in Fig. 17.

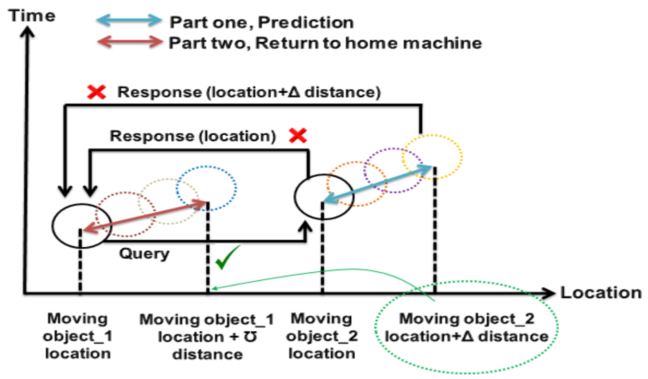

Time
Part one, Prediction
Part two, Return to home machine
Response (location+Δ distance)
Response (location)
Query
Location
Moving object_1 location
Moving object_1 location + Ʊ distance
Moving object_2 location
Moving object_2 location+Δ distance

Fig. 17. Solved uncertainty real-time problem.

B. Details of Our Proposed Architecture

We use sequence diagrams to illustrate the general scenario of our proposed agent-based architecture. Fig. 18 shows the steps for processing a *kNN* query with comprehensive privacy protection.

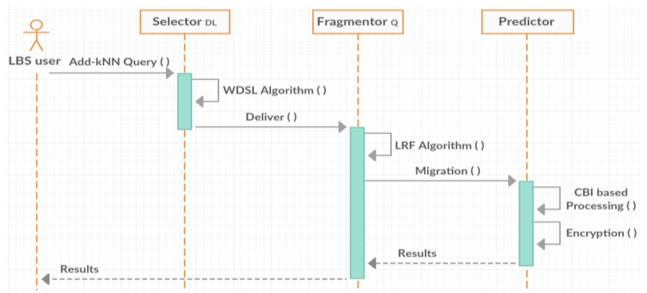


Fig. 18. Sequence diagram of processing a kNN query.

V. Security Analysis

In this section, we discuss two main security issues. The first is related to the agents themselves and the second is related to the WSDL and LRF-based algorithms, which were proposed for privacy protection.

A. Security of Agents

The main obstacle to the widespread deployment of the mobile agent technology is the security issue, in particular, the problem of protecting a mobile agent from malicious hosts that may completely block the agent or modify its carried data. Since the scope in this paper is privacy protection, security is out of scope. Therefore, we assume that all the agents are secure. Specifically, the approach that was proposed in [25] is followed. The integrating of privacy protection with security agents will be considered in future work.

B. Security against Inference Attacks and Query Analysis Attacks

In this subsection, we prove that our proposed architecture is robust by discussing the resistance of the WSDL algorithm and the LRF-based algorithm against location homogeneity attack and query sampling attack, respectively. Since we consider active attack (as discussed in Section III), Table III lists the capabilities of the LBS server (the attacker).

TABLE III. Capabilities of the Attacker (LBS Server)

Cap-No	Description
1	Can eavesdrop on the wireless channel.
2	Can monitor the current queries of LBS users.
3	Can obtain all the stored information that is related to the LBS users.
4	Can obtain the historical location data of the LBS users.
5	Knows the query privacy protection method (LRF-based algorithm).
6	Knows the location privacy protection method (WDSL algorithm).

We follow the definition-theorem-proof style in discussing resistance against inference attacks.

Definition 1. An algorithm is query sampling attack resistant if the units of the sent query cannot be obtained and correctly reordered.

Theorem 1. The proposed LRF-based algorithm is query sampling attack resistant.

Proof 1. Obtaining the units of a query requires eavesdropping on the wireless channel. Since a cryptographic technique (PKI) is used to protect the sensitive data (the units of the query), the attacker cannot obtain the units. Moreover, even if the attacker were to successfully break the encryption phase, he/she would need to form the query in a correct order due to the randomization phase. Furthermore, if the attacker tries to reverse the LRF-based algorithm, he/she will fail because of the periodic changing of the query's units in the LRF-based algorithm, which confuses the attacker and forces him/her to randomly guess the correct order of the units to form the original query. This means that the query sampling attack fails.

Definition 2. An algorithm is location homogeneity attack resistant if the probability of successfully guessing the real location of an LBS user is very low.

Theorem 2. The proposed WSDL algorithm is location homogeneity attack resistant.

Proof 2: We assume that the attacker completely breaks the LRF-based algorithm, thereby obtaining the location of the LBS user. In addition, the information that the attacker holds is the query probability of each individual cell (qp) and all the submitted (k) locations $l_1, l_2, \dots, l_k$ (i.e., the mixture of real and dummy locations). Let $PS_{(event)}$ refer to the probability of the attacker successfully guessing whether (event) is true. The WDSL algorithm is resistant to location homogeneity attack if the following two conditions are satisfied:

1) $PS_{(l_i)} = PS_{(l_j)} \ \forall \ (0 < i \neq j \leq k)$ (10)
2) $dis\left(l_i, l_j\right) \ is \ long.$

First, since the dummy locations are selected based on the query probabilities (qp) of the cells being similar to the query probability of the LBS user's cell (i.e., his/her real location), the attacker can obtain no benefit from employing the query probabilities to determine the real location of the LBS user. Second, since we have ksubmitted locations, the probability of successful guessing the real location is $\left(\frac{1}{k}\right)$. The previous probability value is the same for all k submitted locations because no benefit is obtained from knowing the query probabilities of the locations. This means that the first condition is satisfied. Third, since the dummy locations are selected based on the product of the distances rather than the sum of the distances, the second condition is satisfied. Moreover, even if the attacker tries to reverse the WDSL algorithm, he/she will fail to determine the real locations of the dummies. That is because of the random selection of the final and actual dummy locations, which leads to uncertainty in the dummy selection results. Therefore, the attacker can only randomly guess the real location of the LBS user. As a result, the location homogeneity attack fails.

VI. Metrics

In this section, we provide the metrics that are used for evaluation purposes. In this paper, two kinds of metrics are employed: privacy metrics and performance metrics.

A. Privacy Metrics

We use two privacy metrics: the entropy E and a metric that is derived from the entropy. To evaluate the location privacy, we employ E to quantify the privacy. It is better to achieve a higher E value. E is defined by formula 1.

Suppose an LBS user sends a query to dummy locations to protect his/her privacy. The highest entropy value that can be achieved is $\log_2(k)$, which is achieved when all the submitted locations have the same probability of being treated as the real location of the query issuer (LBS user). Therefore, if the LBS user achieves an entropy value that is less than $\log_2(k)$, the extent to which the privacy was breached by the attacker (LBS server) will be $(\log_2(k) - E)$. As time progresses, the attacker achieves a small success with each sent query. The sum of these small successes represents the degree of danger that the privacy will be compromised, which represents the second privacy metric.

More formally, let $\text{Ʈ} = (\text{ƫ}_1, \text{ƫ}_2, \text{ƫ}_3, \ldots, \text{ƫ}_n)$ refer to the moments at which the LBS user issues queries, where each query is protected by $k - 1$ dummy locations. The degree of danger D_{danger} is defined as:

$$D_{danger} = \sum_{i=1}^{n}\left(\log_2(k) - E(\text{ƫ}_n)\right), \text{where } \text{ƫ}_n \in \text{Ʈ} \quad (11)$$

When an encryption technique is used to protect the privacy, no privacy metric is used to quantify the privacy. This is clearly stated in the survey in [3]. Therefore, we rely on a performance metric for evaluating the query privacy protection.

B. Performance Metrics

Since we used encryption in the proposed LRF-based algorithm to protect the query privacy, we introduce the computation time T_{comp} for evaluation. Here, the computation time refers the time that is spent on both sides (the LBS mobile device's side and the LBS server side). On the LBS mobile device's side, the computation includes the time spent constructing a query based on a dummy location and passing sensitive units, which is equal to the sum of the durations of the extraction, encryption, randomization, and marking phases.

$$T_{Q-cons} = T_{ext} + T_{enc} + T_{rand} + T_{mark} \quad (12)$$

On the LBS server side, the computation time is the time that is spent preparing the query (i.e., reconstructing the query), which is equal to the sum of the durations of the decryption, marking, unit ordering, and processing phases.

$$T_{Q-recons} = T_{dec} + T_{mark} + T_{ordering} + T_{processing} \quad (13)$$

Thus, the computation time is defined as:

$$T_{comp} = T_{Q-cons} + T_{Q-recons} \quad (14)$$

To evaluate the proposed indexing technique, we use two times as performance metrics: access latency and tuning time. Access latency (T_{AL}) refers the elapsed time between the moment when a query is issued and the moment when it is satisfied. Therefore, it depends on T_{comp} as follows:

$$T_{AL} = T_{comp} + T_{sending} + T_{receiving} \quad (15)$$

The tuning time T_{TU} is the time that the mobile LBS user stays active to receive the requested data.

VII. Experimental Results and Evaluations

A. Simulation Setup

In this paper, Matlab software is used to implement the proposed algorithms, with the help of Java Agent DEvelopment Framework (JADE). The performance evaluation is simulated on a Genuine Intel(R) 2.4 GHz PC with 4.00 G RAM, running Microsoft Windows 7 Ultimate. Table IV lists the parameter settings. A data base is constructed for the moving POIs, where timestamps are attached to each POI and each query. The query probability is generated randomly with the help of the Google Maps API.

TABLE IV. Parameter Settings

Parameter	Setting
Number of cells (n × n)	160×160
Number of headers (H)	21,103
Number of arms (A)	21,246
Number of users	10,000
Number of moving POIs	500

For comparison, we selected three dummy-based approaches for location privacy protection: DDA [10], CirDummy [11], and Dest-Ex [12]. The Buddies [21] and PIR [22] approaches are selected for query privacy protection. As *kNN* query processing techniques, we selected D-tree [17] and density [23].

B. Evaluations of Resistance Against Aattacks

There is a direct correlation between the $k -$ anonymity level and the resistance against attacks because a higher $k -$ anonymity level provides higher resistance. Increasing the $k -$ anonymity level requires increasing the number of generated dummies. Therefore, based on the entropy value, we first measure the privacy protection level against the $k -$ anonymity level, assuming that the defenses of the fragmentation technique have been broken. Then, we calculate the number of LBS users that reach dangerous states based on the D_{danger} privacy metric.

Fig. 19 below shows a snapshot at a time progress of 120 minutes. Among the approaches, the DDA approach performs the worst because DDA fills the array of dummies by selecting locations in a random way based on the principle that “the dummy locations must be equal in area”. Thus, the entropy of the dummy locations mainly relies on the current query probabilities of the grid of cells. The CirDummy approach slightly outperforms the DDA approach. That is because the selected dummy locations are limited by a virtual circle. Since the variation of the query probabilities is not large within the circle, which covers only a few cells (a small region), the corresponding entropy value is only slightly higher. The Dest-Ex approach outperforms both DDA and CirDummy. The main factor that contributes to the enhancement of the entropy values is the direction, which may be changed to include more cells with the same query probability. Compared to the previous approaches, the WDSL approach performs the best. The underlying reason is that the dummy locations are selected based on having similar query probabilities to the real location. This guarantees much higher entropy values and higher corresponding privacy levels.

Regarding to the evaluation based on the D_{danger} privacy metric, we evaluated the situations of LBS users under location homogeneity attack. In this context, a threshold is defined $(thr = 0.75)$ at which the LBS user is considered vulnerable to attack by the LBS server. The level of anonymity is fixed to $(k = 6)$ (i.e., at any moment, the sent query is protected by five queries, which are built based on dummy query locations). Twenty LBS users are randomly selected from each of the compared approaches and a snapshot at $(t = 120)$ is taken, as shown in Fig. 20.

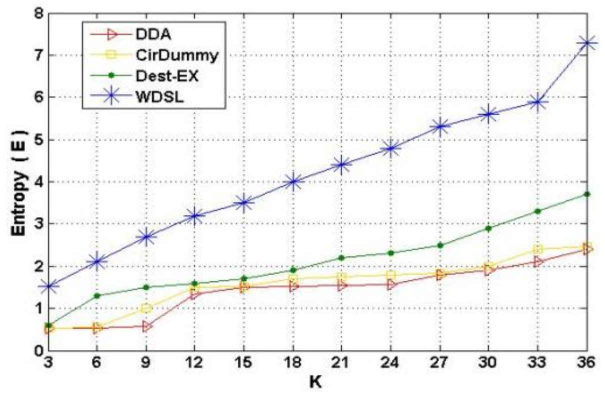


Fig. 19. Entropy vs. k, $t = 120$.

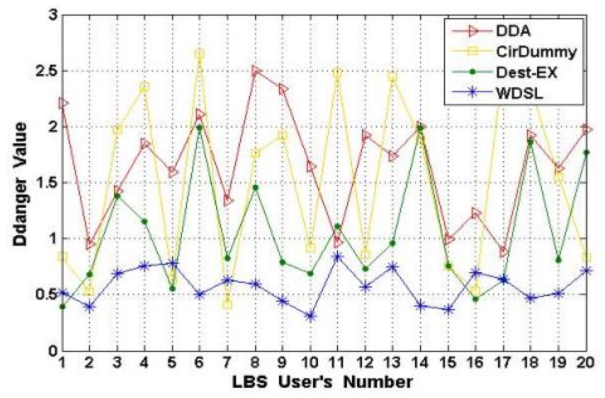


Fig. 20. D_{danger} values for 20 LBS users, $k = 6$, $t = 120$.

TABLE V. COMPARISON OF VULNERABILITY STATUSES OF LBS USERS

Settings: $t = 120, k = 6, thr = 0.75$.		
Term / Approach	*Number of users that exceed the threshold*	*Percentage of encroachment*
WDSL	3	0.15
Dest-Ex	10	0.5
CirDummy	16	0.8
DDA	20	100

Table V shows that all LBS users in the DDA approach exceeded the threshold. That is because all the selected dummy locations are close to one another since they are formed by the vertices and the edges of the grid. More than three-quarters and half of the LBS users exceeded the threshold in CirDummy and Dest-Ex, respectively. Compared to DDA, the CirDummy has higher resistance against location homogeneity attack since the radius of the circle may be enlarged to include some dummy locations that are far away from the real location of the LBS user. Dest-Ex achieved a higher resistance than CirDummy because the directions can be changed to include dummy locations that are further away from the real location. The proposed WDSL approach performs the best since it has the minimum number of LBS users that exceeded the threshold, and, consequently, the highest resistance against the location homogeneity attack. That is because the dummy locations are selected based on the product of their distances.

Under different threshold values, snapshots, and numbers of LBS users, Table VI supports the results in Table V.

TABLE VI. PERCENTAGE OF ENCROACHMENT OF THE PREDEFINED THRESHOLDS

Try NO	**NO of LBS users**	***t***	***thr***	**Percentage of encroachment**			
				WDSL	*Dest*	*Cir*	*DDA*
1	40	130	0.7	0.11	0.5	0.62	100
2	60	140	0.65	0.12	0.53	0.78	100
3	80	150	0.6	0.2	0.4	0.61	100
4	100	160	0.55	0.18	0.41	0.55	100
5	120	170	0.5	0.13	0.34	0.53	100

C. Evaluations of Computation Costs

We use the T_{comp} performance metric to evaluate the efficiency of the proposed LRF-based approach against the buddy and PIR approaches. Two aspects are considered in the evaluation: the impact of increasing the k value that is associated with a query and the impact of increasing the number of sent queries.

In general, the computation time increases as k increases. Fig. 21 shows a snapshot at $t = 120$, where we randomly selected an LBS user who sends a privacy-protected query at different levels of k. The PIR-based approach performs the worst since it performs many computations to protect the privacy of the query. Despite the times spent in the various phases (i.e., the extraction, encryption, randomization, and marking phases), our proposed LRF fragmentation technique performs the best. The reason behind this is the efficient employment of the bloom filter to enhance the processing time of the query. Specifically, the help that is provided by the $predictor$ agent through the proposed CBI technique efficiently contributes to the shortening of the query processing time. In depth, the process of encapsulating the index part by the bloom filter has a positive impact on the search time, as it avoids searching in the empty cells.

The results that are shown in Fig. 21 are supported by those in Fig. 22, in which the number of protected queries increased. Again, the bloom filter is the underlying feature that accelerates the answering of the queries, which is not utilized by the other approaches.

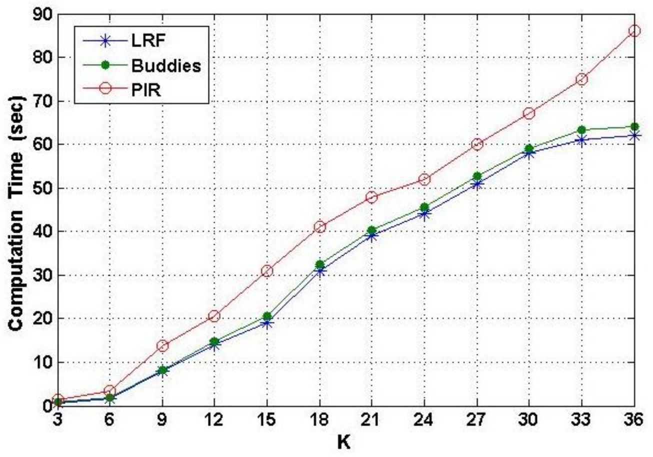


Fig. 21. T_{comp} vs. k, $t = 120$.

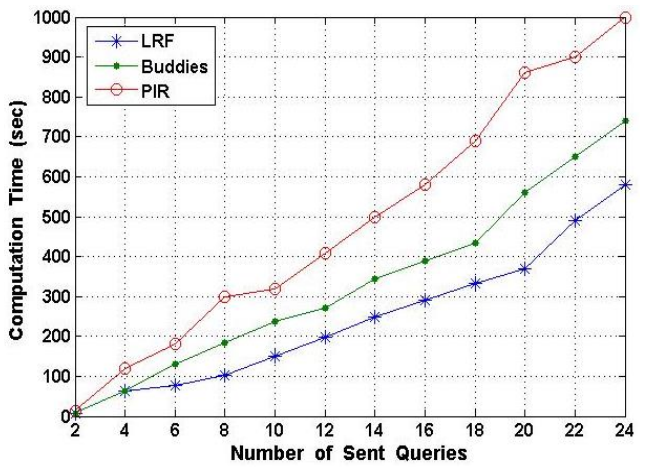


Fig. 22. T_{comp} vs. Number of sent queries, $k = 6, t = 120$.

D. Access Latency (T_{AL}) and Tuning Time (T_{TU}) Evaluations

In the previous subsection, the sending time of the query (from the mobile device of the LBS user to the LBS server) and the receiving time of the query's answer (from the LBS server to the mobile device of the LBS user) are completely ignored. When evaluating T_{AL} and T_{TU}, the two previous times must be taken into account. We assume that the sending time of the query is the same for all of the compared techniques (i.e., D-tree, Density, and the proposed CBI). Fig. 23 shows the access times for different numbers of sent queries.

As shown in Fig. 23, the proposed CBI technique outperforms the Density and D-tree techniques. The main factor in this is the migration of the $fragmentor_Q$ mobile agent back to the home machine, to deliver the answers to the sent queries. Meanwhile, in both the Density and D-tree techniques, a significant amount of time is needed to search for the queries' issuer (since it is considered an MO) to deliver the answers. In other words, the receiving time of the queries' answers is longer for these two methods than for the proposed CBI technique. The access latency time reflects the efficiency of the proposed CBI technique in solving the second part of the real-time uncertainty problem (see Fig. 17).

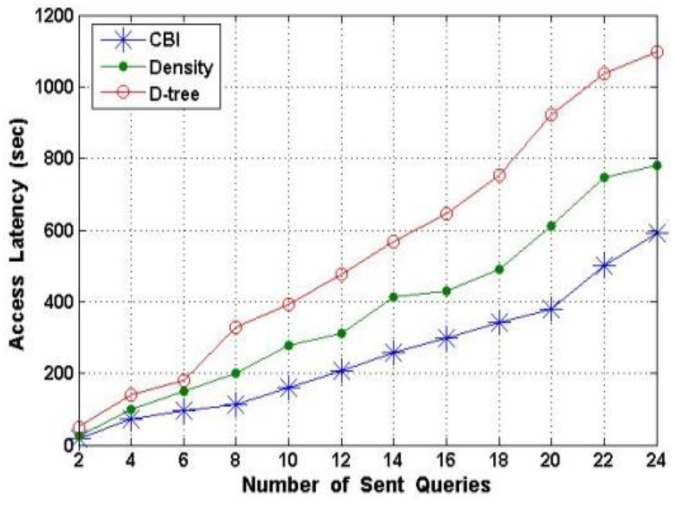


Fig. 23. T_{AL} vs. Number of sent queries, $k = 6, t = 120$.

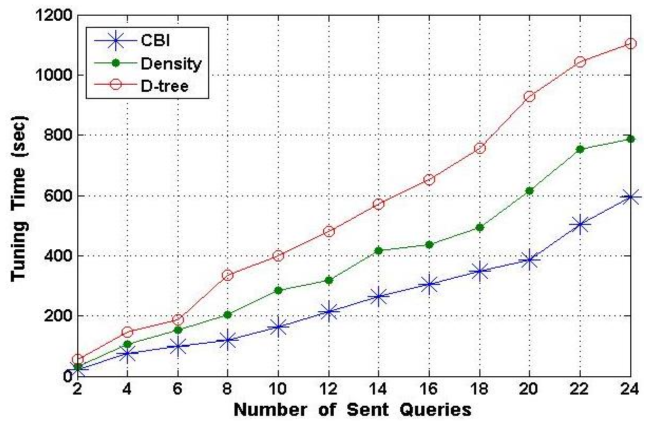


Fig. 24. T_{TU} vs. Number of sent queries, $k = 6, t = 120$.

The results that are shown in Fig. 24 support those that are illustrated in Fig. 23, but with higher tuning time values since the LBS user spends additional time preparing the queries and exploring the received answers. However, the proposed CBI technique provides the minimum tuning time values. Since the tuning time refers the time that mobile device of the LBS user stays active, the proposed CBI technique reduces the battery consumption of the mobile device. Short battery life is a main drawback of user-based privacy protection approaches.

E. Evaluations of the Prediction Phase of the CB Technique

The migration of the $fragmentor_Q$ mobile agent back to the home machine contributes to solving the second part of the real-time uncertainty problem, and the prediction phase in the proposed CBI technique contributes to solving the first part of the real-time uncertainty problem (see Fig. 17 above). In this context, we evaluate the number of retrieved moving POIs and the precisions of the locations of the retrieved moving POIs.

Fig. 25 shows the number of retrieved moving POIs when the LBS user searches for the nearest 6 taxis that are located within a 0.5 km range around different real locations of the LBS user. For instance, in response to the query $(< 70,70 >, \text{taxis}, 0.5, \text{Bob})$, 3, 6, and 11 moving taxis were retrieved by the D-tree, Density, and CBI techniques, respectively. The Density technique outperforms the D-tree technique since it uses the overlap among the cells to build the index. The proposed CBI technique outperforms the Density technique due to two factors: First, the index is built on the level of cells, which accurately covers all the cells that are included in the 0.5 km range. Second, in the prediction phase, because of the motion of the queried POIs, many additional POIs may enter the cells (covered by the given range) from the surrounding cells. Therefore, the prediction phase can include the POIs that entered the range in the answer to the query.

Since the number of the retrieved POIs does not accurately reflect the efficiency of the proposed indexing technique, we evaluate the precision of the retrieved locations of the moving POIs. Here, the precision term is the degree of matching between the current location (i.e., the exact future location of the moving POI) and the predicted one. From Fig. 25, we select the nine retrieved taxis that are related to the query $(< 90,90 >, \text{taxis}, 0.5, \text{Bob})$ for evaluation.

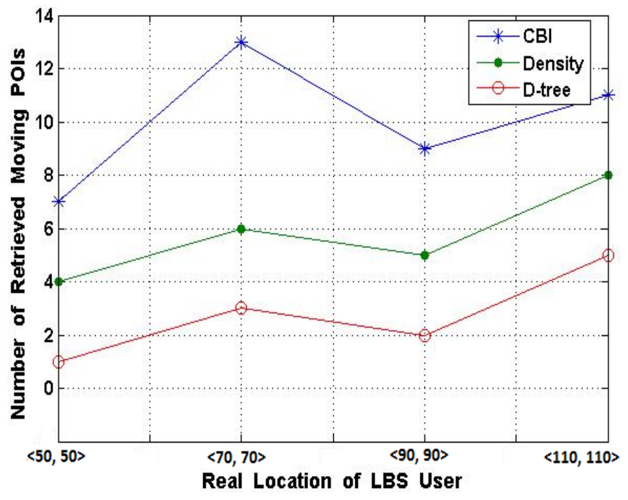


Fig. 25. Number of retrieved POIs, $k = 6, R = 0.5\ km, t = 120$.

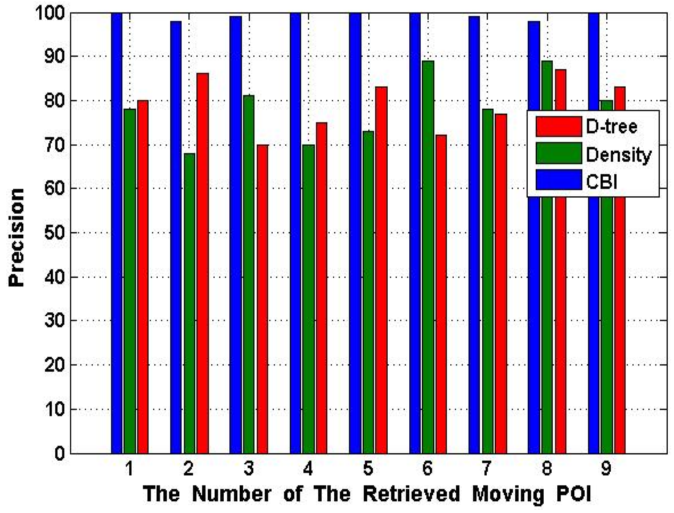


Fig. 26. Locations precision of the retrieved moving POIs.

As shown in Fig. 26, the precision of the retrieved locations is variable for both the D-tree and Density techniques. In contrast, the stability of the precision of the locations that were retrieved by the proposed CBI technique high: it varies between 100% and 98%. This is due to the prediction phase.

VIII. Conclusion

With the impressive development of both wireless networks and mobile devices, Location Based Services (LBSs) have become popular. LBSs enable network users to perform range queries or k-Nearest Neighbor (*kNN*) queries. However, it is extremely important to ensure comprehensive privacy protection, in addition to guaranteeing the efficiency of *kNN* query processing. We propose a Wise Dummy Selection Location (WDSL) approach for ensuring the location privacy of *kNN* queries. To ensure a high protection level of location privacy, the WDSL approach selects dummy locations that satisfy two conditions: (1) the query probabilities of the selected dummy locations are the same as that of the real location of the LBS user and (2) the selected dummy locations are distributed over a wide region to ensure resistance against location homogeneity inference attack. Resistance against query sampling attack, which targets the query privacy, is considered. Extracting, encrypting, and randomizing the sensitive units of the sent query based on a Left-Right Fragmentation (LRF) technique results in robust defense against the query sampling attack and ensures the query privacy. The integration of the WDSL approach and the LRF technique ensures the *kNN* query privacy during the sending, processing, and responding phases. To manipulate the *kNN* query efficiently, an index is built based on an efficient motion model at the level of cells, in which the moving POIs are moving. The index consists of two parts: a data part and an index part. The data part is supported by a prediction phase, which estimates the future locations of the queried moving POIs. The index part is encapsulated by a bloom filter to speed up the response to the *KNN* query. In terms of resistance against inference attacks and query analysis attacks, computational cost, and number and accuracy of retrieved moving POI locations, the proposed system outperforms similar approaches and techniques.

In future work, we intend to ensure the integrating of agents security and privacy. In addition, we intend to develop defenses against other inference attacks, such as map matching attacks and semantic location attacks.

References

[1] Dardari, Davide, Pau Closas, and Petar M. Djurić. "Indoor tracking: Theory, methods, and technologies." IEEE Transactions on Vehicular Technology 64.4 (2015): 1263-1278.

[2] Wernke, Marius, et al. "A classification of location privacy attacks and approaches." Personal and Ubiquitous Computing 18.1 (2014): 163-175.

[3] Shin, Kang G., et al. "Privacy protection for users of location-based services." IEEE Wireless Communications 19.1 (2012).

[4] Yi, X., Paulet, R., Bertino, E., & Varadharajan, V. (2014, March). Practical k nearest neighbor queries with location privacy. In Data Engineering (ICDE), 2014 IEEE 30th International Conference on (pp. 640-651). IEEE.

[5] Ni, Weiwei, Mingzhu Gu, and Xiao Chen. "Location privacy-preserving k nearest neighbor query under user's preference." Knowledge-Based Systems 103 (2016): 19-27.

[6] Yi, Xun, et al. "Practical approximate k nearest neighbor queries with location and query privacy." IEEE Transactions on Knowledge and Data Engineering 28.6 (2016): 1546-1559.

[7] Ma, Tinghuai, et al. "Protection of location privacy for moving kNN queries in social networks." Applied Soft Computing 64.2 (2017): 485-158.

[8] Dai, Jian, Zhi-Ming Ding, and Jia-Jie Xu. "Context-Based Moving Object Trajectory Uncertainty Reduction and Ranking in Road Network." Journal of Computer Science and Technology 31.1 (2016): 167-184.

[9] Zhang, Xu, et al. "A novel location privacy preservation method for moving object." International Journal of Security and Its Applications 9.2 (2015): 1-12.

[10] Alrahhal, Mohamad Shady, et al. "AES-Route Server Model for Location based Services in Road Networks." INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS 8.8 (2017): 361-368.

[11] Lu, Hua, Christian S. Jensen, and Man Lung Yiu. "Pad: privacy-area aware, dummy-based location privacy in mobile services." In Proceedings of the Seventh ACM International Workshop on Data Engineering for Wireless and Mobile Access, pp. 16-23. ACM, 2008.

[12] Hara, Takahiro, et al. "Dummy-Based User Location Anonymization Under Real-World Constraints." IEEE Access 4 (2016): 673-687.

[13] Pingley, Aniket, Nan Zhang, Xinwen Fu, Hyeong-Ah Choi, Suresh Subramaniam, and Wei Zhao. "Protection of query privacy for

continuous location based services." In Infocom, 2011 Proceedings IEEE, pp. 1710-1718. IEEE, 2011.

[14] Pan, Xiao, et al. "Protecting personalized privacy against sensitivity homogeneity attacks over road networks in mobile services." Frontiers of Computer Science 10.2 (2016): 370-386.

[15] Wang, Yong, et al. "A fast privacy-preserving framework for continuous location-based queries in road networks." Journal of Network and Computer Applications 53 (2015): 57-73.

[16] Hambrusch, Susanne, Chuan-Ming Liu, Walid G. Aref, and Sunil Prabhakar. "Query processing in broadcasted spatial index trees." In International Symposium on Spatial and Temporal Databases, pp. 502-521. Springer, Berlin, Heidelberg, 2001.

[17] Xu, Jianliang, Baibua Zheng, W-C. Lee, and Dik Lun Lee. "Energy efficient index for querying location-dependent data in mobile broadcast environments." In Data Engineering, 2003. Proceedings. 19th International Conference on, pp. 239-250. IEEE, 2003.

[18] Zheng, Baihua, et al. "Grid-partition index: a hybrid method for nearest-neighbor queries in wireless location-based services." The VLDB Journal—The International Journal on Very Large Data Bases 15.1 (2006): 21-39.

[19] A. Beresford and F. Stajano, ―Location Privacy in Pervasive Computing,‖ IEEE Pervasive Computing, vol. 2, no. 1, 2003, pp. 46–55.

[20] Zhang, Xu, and Hae Young Bae. "Location Positioning and Privacy Preservation Methods in Location-based Service." International Journal of Security and Its Applications 9.4 (2015): 41-52.

[21] Mascetti, Sergio, et al. "Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies." The VLDB Journal—The International Journal on Very Large Data Bases 20.4 (2011): 541-566.

[22] Ghinita, Gabriel, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. "Private queries in location based services: anonymizers are not necessary." In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp. 121-132. ACM, 2008.

[23] Jang, Mi Young, and Jae Woo Chang. "A new k-nn query processing algorithm enhancing privacy protection in location-based services." In Computer and Information Technology (CIT), 2011 IEEE 11th International Conference on, pp. 421-428. IEEE, 2011.

[24] Solomon, Brad, and Carl Kingsford. "Improved Search of Large Transcriptomic Sequencing Databases Using Split Sequence Bloom Trees." In International Conference on Research in Computational Molecular Biology, pp. 257-271. Springer, Cham, 2017.

[25] Madkour, Mohamed A., et al. "Securing mobile-agent-based systems against malicious hosts." World Applied Sciences Journal29.2 (2014): 287-2