Privacy-area Aware All-dummy-based Location Privacy Algorithms for Location-based Services

Huan ZHAO\textsuperscript{1,a,*}, Xiao-Ling YI\textsuperscript{1,b}, Jiao-Long WAN\textsuperscript{1,c}

\textsuperscript{1}College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan, China

\textsuperscript{a}hzhao@hnu.edu.cn, \textsuperscript{b}yixiaoling10@hnu.edu.cn, \textsuperscript{c}hnu_long@hnu.edu.cn

*Corresponding author

Keywords: Location Privacy, Location-Based Services (LBSs), K-anonymity, Dummy.

Abstract. Location-Based Services (LBSs) have become one of the most popular activities in our daily life. With the rapid advance of LBSs, there are more threats to users’ privacy. For this reason, while enjoying the convenience provided by LBSs, we have to protect our location privacy. In this paper, we propose two all-dummy-based location privacy algorithms to achieve k-anonymity for privacy-area aware users in LBSs. Different from previous work, on the client side, our method can prevent the center attack and border attack through transforming the actual location to an anchor and the request we send to the LBSs doesn’t contain the actual location of the user. On the server side, we use a rough query before precise query to reduce the processing time and transmission bandwidth. It only returns the results that the client needs. Evaluation results show that our methods can provide more effective privacy protection and lower computation and communication cost.

Introduction

With the popularity of mobile devices and the rapid development of social network, location-based services (LBSs) have been one of the most popular activities in our daily life. Nowadays people can enjoy the benefits from various tailored and personalized LBS anytime and anywhere. For example, you can get the nearest restaurant or the gas station via LBS.

While enjoying various LBSs, users also face the threats of location privacy disclosure. The users’ location information is usually closely linked with the users’ identity information, such as, the name of their company, home address, health status, religious beliefs, etc. Therefore, more and more users focus on the location privacy protection.

Spatial cloaking is a widely used method in location privacy protection. It ensures a user’s location is mixed with at least k candidates. It collects k users’ locations and sends the minimum region including those k users to the server as q query instead of the user’s exact location. Gruteser et al. [10] first introduced k-anonymity into location privacy. It tries to blur a user’s exact location by generating a cloaking region which includes a query issuer and k-1 other users. Current research aligns on developing techniques to elaborate on k-anonymity [3-8, 15] that process a user’s query with protecting privacy during the access of LBSs. However, these kinds of approaches always rely on a location anonymizer to enlarge the cloaking region.

Dummy-based anonymization method solves these selection problems. User location anonymization methods using dummies are the representative approach [2, 9, 12, 14]. Through generating dummies and sending their locations with the user’s actual location to the server [2, 11-13], the user can hide the true location. According to the query, the server is performed in accordance with all position candidates and the server returns the query results to the user. To address these aforementioned problems, Lu et al. [2] propose two dummy location generating methods called CirDummy and GridDummy, which
achieve $k$-anonymity for mobile users without employing the location anonymizer and considering the privacy-area. However, the real location of the user is on the border, and the border may still be attacked.

To solve the problem, in this paper, we propose two privacy-area aware dummy-based location privacy algorithms, which are developed from the methods mentioned in [2] to achieve $k$-anonymity without the true location and against the center attack and border attack. We use a simple example shown in Fig. 1 to illustrate our concerns. In Fig. 1(a), the adversary can easily to guess the actual location of the user. Fig. 1(b) is the method in [2] to achieve a bigger privacy area, but it still can be attacked on the border. In our proposed method, we send the whole dummy locations to the server like Fig. 1(c). Due to the lack of true location, we cannot directly calculate two location’s relation and we use an estimation method. We follow client/server modal and don’t need the trusted third party. Our proposed method just requires the server returns the results what the client needs, which can save a lot of bandwidth.

The rest of this paper is organized as follows. In section II, we review related word of our methods. Then, in section III, we introduce some preliminaries of our work and we describe our proposed method and implementation in detail. Following in section IV, we explain our experiment and empirically evaluate our solutions. Finally, we conclude in section V.

![Figure 1. Motivation.](image)

**Proposed Method**

Our proposal assumes a client/server architecture. And our methods anonymize a user’s location with all the dummies based on Shokri et al.’s framework [1] in a real environment.

**Definitions and Assumptions**

Definition 1: $POS = (pos_1, pos_2, ..., pos_k)$ is the position that user submitted to the server, where the $pos_i$ $(1 \leq i \leq k)$ is a dummy location around the user’s actual location.

Definition 2: Query in this paper is a location-dependent query, which is abstracted as $Q = (pos, d, <S, k>, K, req)$, where parameter $pos$ is the mobile user location, parameter $d$ is the query radius, parameter $<S, k>$ means it should provide $k$ locations and its privacy area is no smaller than $S$, parameter $K$ is the number of required return and parameter $req$ denotes user-specified predicates. With the privacy-area aware and location dummy approach, it is converted into $Q' = (POS, d, <S, k>, K, req)$, where $POS$ contains $k$ dummy locations. The query $Q'$ hides the user location.

Assumption 1: For the two groups of users’ position data, we compare their distance with $k^2$ times. If their distance is less than $d$ ($d$ represents query distance), we call overlap, and assume the more the number of overlap, the closer to users each other. This assumption is easy to prove, because if two
users are closer, the greater number generated in a hidden area of the dummy location falls within the scope of the query.

Privacy-Area Aware Dummy Generation Algorithms

We are interested in approaches that aid in satisfying $<S, k>$ and can against the center attack and border attack. We thus propose two improved privacy-area aware dummy generation algorithms based on [2]. The one algorithm employs a circle to constrain all dummy positions, while the other uses a uniform grid.

Circle-Based All Dummy Algorithm. We use an example shown in Fig. 2(a) to illustrate our circle-based all dummy algorithm. We consider the example where $k=8$ and pos is the user location. The user-defined privacy region $S$ provides us a way to determine the radius of the virtual circle. From paper [2], taking into account the requirement $S$, we could get an upper bound radius $r = \sqrt{2 \cdot S/(k \cdot \sin(2\pi/k))}$. Let $r_{min} = \rho \cdot r$, we then get:

$$S \geq \bar{s} \geq \frac{1}{2} \cdot k \cdot r_{min}^2 \cdot \sin \frac{2\pi}{k} = \frac{1}{2} \cdot k \cdot (\rho \cdot r)^2 \cdot \sin \frac{2\pi}{k} = \rho^2 \cdot S \ (0 < \rho \leq 1) \tag{1}$$

This indicates that by carefully choosing $\rho$, we can get a guarantee on the privacy area of the location privacy query generated based on a virtual circle.

The algorithm, called CircleAllDummy, is shown in Algorithm 1. It first initializes set $P$ and calculates the angel $\theta$ between positions (line 1). Next, it calculates the upper-bound radius $r$ of the virtual circle (line 2) and it determines the circle center $pos_0$ at random (lines 3-5). The $k$ dummy locations distance to the virtual circle center $pos_0$ are constrained by $(\rho \cdot r, r]$, while they are scattered evenly in terms of their angles with respect to $pos_0$ (lines 6-11). Finally, dummy location set $P$ is returned (line 12).
As it’s considered in paper [2], the more coefficient $\rho$ is close to 1, all positions are closer to the virtual circle edge and the real user location is less randomly. But our method sends all dummies to the server without the real user’s location, it is safer than those send $k\!-\!1$ dummies with one real location.

Algorithm 1 CircleAllDummy

Input: user position $pos$, privacy area $S$, anonymity $k$, coefficient $\rho$

Output: dummy location set $P$

1: $P \leftarrow \emptyset$, $\theta \leftarrow 2\pi/k$
2: $r \leftarrow \sqrt{2 \cdot S/(k \cdot \sin \theta)}$
3: $r' \leftarrow \rho \cdot r$
4: $pos_{0x} \leftarrow ((\text{rand}(1) \geq 0.5) \cdot 2 - 1) \cdot \text{rand}(r', r) + pos_x$
5: $pos_{0y} \leftarrow ((\text{rand}(1) \geq 0.5) \cdot 2 - 1) \cdot \text{rand}(r', r) + pos_y$
6: for $i$ from 1 to $k$ do
7: $r_i \leftarrow \text{rand}(r', r)$
8: $pos_{ix} \leftarrow r_i \cdot \cos(i \cdot \theta) + pos_{0x}$
9: $pos_{iy} \leftarrow r_i \cdot \sin(i \cdot \theta) + pos_{0y}$
10: append $pos_i$ to $P$
11: end for
12: return $P$

Algorithm 2 GridAllDummy

Input: user position $pos$, privacy area $S$, anonymity $k$, coefficient $\tau$

Output: dummy location set $P$

1: $P \leftarrow \emptyset$, $a \leftarrow \sqrt{S}$, $c \leftarrow \lceil \sqrt{k} \rceil$
2: $g \leftarrow a/(c - 1)$
3: $r \leftarrow \tau \cdot a$
4: $pos_{0x} \leftarrow \text{rand}(0, r) + pos_x$
5: $pos_{0y} \leftarrow \text{rand}(0, r) + pos_y$
6: $idx \leftarrow \text{rand}(0, c - 1)$
7: $idy \leftarrow \text{rand}(0, c - 1)$
8: for $i$ from 1 to $k$ do
9: $pos_{ix} \leftarrow (\text{rand}(0, c - 1) - idx) \cdot g + pos_{0x}$
10: $pos_{iy} \leftarrow (\text{rand}(0, c - 1) - idy) \cdot g + pos_{0y}$
11: append $pos_i$ to $P$
12: end for
13: return $P$

Grid-Based All Dummy Algorithm. We use an example shown in Fig. 2(b) to illustrate our grid-based all dummy algorithm. We consider the example where $k=6$ and $pos$ is the user’s location. Taking into account the privacy requirement $S$, we get the side length $a = \sqrt{S}$. Then we divide the area with side length $a$ into $c$ cells, where $c = \lceil \sqrt{k} \rceil$, and we let each cell’s length $g = a/(c - 1)$.

The algorithm, called GridAllDummy, is shown in Algorithm 2. It first initializes an empty set $P$, calculates the number of cells $c$, the side length $a$ and each cell’s length $g$ (lines 1-2). Then, it determines the virtual user $pos_{0}$ at random (lines 3-5), so that in either direction its distance to $pos$ falls in $(0, \tau \cdot \sqrt{S})$, where $\tau$ is the offset coefficient of the user position, and $0 < \tau \leq 1$. It attaches the virtual user position $pos_{0}$ to each direction (lines 6-7). It then generates $k$ dummy locations randomly based on $pos_{0}$ from that virtual grid. Finally, dummy location set $P$ is returned.

Server-Side Processing

Having received a location privacy query $Q' = (POS, d, <S, k>, K, req)$ from a mobile client, the server processes this according to predicate $req$, and returns $K$ required query results.

In our method, without the user’s true location, we cannot directly calculate distance of two points. In this paper, we assume that the two users’ location distribution that a server received shown in Fig. 3, when A wants to know whether B locates in its range, A should compare all its $k$ dummy locations with B’s $k$ dummy locations. It is easy to know that the more B’s dummy locations are in the range of A’s dummy locations, the closer B to A. In order to reduce the computational complexity, we first use the
rough radius $D$ which satisfied the inequality $D \geq d + \sqrt{S}$. Then, we use the record of the times less than $d$ as the first metric of two locations’ relationship and we use the summation of distance as the second metric.

The algorithm underlying the server-side module is presented in Algorithm 3. We first initializes empty set $R$, $T$, $W$ and server gets the target user’s position data (lines 1-2). Next, server randomly selects a dummy location from the target user’s query and also randomly selects a dummy location from other user, and calculates the distance. If their distance is less than $D$, server will add user’s ID and position data to $W$ using hash table (lines 3-15). Then, for each user in $W$, server compares their distance with target user, records the overlap time and calculates the distance (lines 18-30). Then, if overlap time is greater than 0, server adds a triple to $T$, otherwise continue (lines 31-36). Then, server sorts $T$ by primary keyword repeat descend and secondary keyword distance descend (line 38). Then, server packages the first $K$ in $T$ as a result set (line 39). Finally, $R$ is returned (line 40).

---

Algorithm 3 ServerCompare

**Input:** position privacy query $Q'$, privacy area $S$, anonymity $k$, query radius $d$, nearest neighbor number $K$, others’ position set userDataSet

**Output:** K-nearest neighbor set $R$

1: $R \leftarrow \emptyset, T \leftarrow \emptyset, W \leftarrow \emptyset$
2: $userQueryPos \leftarrow get\ position\ from\ Q'$
3: $i \leftarrow \text{rand}(0, k - 1)$
4: $pos_i \leftarrow userQueryPos[i]$
5: for all userDataSet $\in userDataSet$ do
6:  $j \leftarrow \text{rand}(0, k - 1)$
7:  $userPos \leftarrow get\ position\ from\ userData$
8:  $pos_j \leftarrow userQueryPos[j]$
9:  if $\text{dist}(pos_i, pos_j) < D$ then
10:     $userID \leftarrow get\ user\ ID\ from\ userData$
11:     $W[userID] \leftarrow userPos$
12: else
13:     continue
14: end if
15: end for
16: for all $userID \in W$ do
17:     count $\leftarrow 0$, distance $\leftarrow 0$
18:     $userPos \leftarrow W[userID]$
19:     for $i$ from 1 to $k$ do
20:         $pos_i \leftarrow userQueryPos[i]$
21:         for $j$ from 1 to $k$ do
22:             $pos_j \leftarrow userPos[j]$
23:             if $\text{dist}(pos_i, pos_j) < d$ then
24:                 count $\leftarrow count + 1$
25:             end if
26:         end for
27:         distance $\leftarrow distance + \text{dist}(pos_i, pos_j)$
28:     end for
29: end for
30: if count $> 0$ then
31:     $t \leftarrow (userID, count, distance)$
32:     append $t$ to $T$
33: else
34:     continue
35: end if
36: end for
37: $T \leftarrow \text{sort}(T)\ by\ primary\ keyword\ repeat\ descend\ and$
38: secondary\ keyword\ distance\ descend
39: $R \leftarrow T(1, K)$
40: return $R$
Communication Cost Analysis

**Upstream Communication Cost.** It is easy to know that a 2D location takes up 8 bytes. Every user sends $k$ locations to the server, so the size of a raw request is expressed as:

$$req_{raw} = 8k \quad (2)$$

Restructuring a raw query request by putting all $x$ coordinate values in front of $y$ values. The cost can be reduced to roughly $16\sqrt{k}$ [2].

**Downstream Communication Cost.** In our proposed method, we return the request K groups of locations to the user. The size in bytes of a raw result message without packing is:

$$res_{raw} = 8k \cdot K \quad (3)$$

With the packing described in reference [2], the result message size shrinks to $8K$.

**Performance Evaluations**

**Settings**

In our experiments, we use the central part of the Changsha, Hunan Province, China to simulate data set. All datasets are normalized to a 10000*10000 square 2D space on the server side. All algorithms run on a Windows 7 PC with 3.2GHz Intel Dual-Core E6700 CPU and 2GB RAM. The parameters are shown in Table 1 with default values given in bold. And we use $\rho^2 = 0.75$ as the default in the CircleAllDummy algorithm and $\tau = 0.5$ as the default in the GridAllDummy algorithm.

![Figure 4. Downstream communication cost.](image)

**Table 1. Simulation Parameters Settings.**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial extents ($m^2$)</td>
<td>1000-1000</td>
</tr>
<tr>
<td>User dataset cardinality N</td>
<td>50K, 100K, 200K, <strong>500K</strong>, 800K, 1000K</td>
</tr>
<tr>
<td>Anonymity k</td>
<td>2, 4, 8, 16, 32, 64</td>
</tr>
<tr>
<td>Privacy region areas S ($m^2$)</td>
<td>80$^2\pi$, <strong>100^2\pi</strong>, 200$^2\pi$, 300$^2\pi$, 400$^2\pi$, 500$^2\pi$</td>
</tr>
<tr>
<td>Query radius d (m)</td>
<td>100, 200, 300, 500, 800, <strong>1000</strong></td>
</tr>
<tr>
<td>Nearest neighbor number K</td>
<td>10,20,<strong>30,40,50</strong></td>
</tr>
</tbody>
</table>
Communication Cost

Our method in the paper just reduces the downstream communication cost, so only the downstream communication cost is analyzed here. As is shown in Fig. 4, under different anonymity $k$, the server returns the message size in bytes in our ServerCompare method is smaller than the PAD’s. The PAD not only returns the real user requirement, but also returns extra $k-1$ group useless results, thus resulting in a tremendous waste of bandwidth. However, our proposed method only returns the results the client needs, significantly reducing bandwidth utilization.

Server-Side Cost

Fig. 5 reports on how the server-side module processing time is affected by five parameters against the PAD and our method. As we known, the processing time is influenced by user density, query radius $d$, privacy area requirement $S$, nearest neighbor number $K$ and anonymity $k$. It is seen that our methods can effectively reduce the processing time on the server side under different condition.

Privacy Area

For each algorithm we use the convex hull of all positions in each location privacy query as the privacy area. The results are reported in Fig. 6. Fig. 6(a) (b) show that our algorithms can cover acceptable range of privacy area when varying the anonymity $k$ and privacy area requirement $S$. And CircleAllDummy is a little better than our GridAllDummy. And CircleAllDummy achieves the guarantee indicated by $\rho$. Fig. 6(c) reports on the effect on the privacy area of CircleAllDummy for varying $\rho$. We can know that the privacy area is at least 75% of the privacy area requirement $S$.
**Result Accuracy Rate**

In the server side, by $k^2$ group of data comparison, we are still unable to accurately determine whether the true location of B is in the range of A for the reason that we use all dummy locations as shown in Fig. 3. It can bring some errors. It’s necessary to analyze the accuracy of the query results.

Fig. 7(a) (b) (c) (d) (e) report on the effect on the result accuracy rate of CircleAllDummy and GridAllDummy for varying user number $N$, privacy area requirement $S$, query radius $d$, nearest neighbor $K$ and anonymity $k$. It reports that the accuracy of GridAllDummy algorithm is a little better than CircleAllDummy algorithm. Fig. 7(a) shows that as user number increases, the accuracy of results decrease. The reason is that when user number increases, the user density is rising. Fig. 7(b) (c) report that as privacy area requirement $S$ growing, the accuracy of results drop dramatically, and as query radius $d$ grows, the result accuracy totally improved. Fig. 7(b) (c) indicate that as the ratio of $S/d$ larger, the accuracy becomes lower. Fig. 7 (d) shows that as $K$ increases, the accuracy of results is also rising. The reason is that when $K$ is large enough, the impact from random error become very weak. Fig. 7(e) describes that as the anonymity $k$ grows, the result accuracy doesn’t improve significantly. Fig. 7 (f) reports on the effect on the accuracy rate of GridAllDummy for varying $\tau$. When $\tau$ is smaller than 0.3, the result accuracy rate grows with $\tau$. However, when $\tau$ is larger than 0.3, the accuracy rate will drop with growing of it.

Overall, it is seen that the accuracy rate is at least 70%.
Conclusion

In this paper we have proposed privacy-area aware all-dummy-based location privacy algorithms for location-based services and we have used a new query anonymization method in server-side. The results show that our method can effectively reduce communication cost. We design two location transformation and all dummy generation algorithms that take into account privacy area requirements. The followed performance evaluation shows the effectiveness of our proposed algorithms. Our algorithms can produce acceptable range of privacy area. And there are some errors because user sends all dummy locations to the server. We have analyzed how the different factors affect the error. One of our future research is to minimize the errors through choosing the best parameters.

Acknowledgement

This work was sponsored by the Scientific Research Plan of Changsha Science and Technology Department (k1403027-11).

References


