PAD: Privacy-Area Aware, Dummy-Based Location Privacy in Mobile Services

Hua Lu   Christian S. Jensen   Man Lung Yiu
Department of Computer Science, Aalborg University, Denmark
{luhua, csj, mly}@cs.aau.dk

ABSTRACT

Location privacy in mobile services has the potential to become a serious concern for service providers and users. Existing privacy protection techniques that use k-anonymity convert an original query into an anonymous query that contains the locations of multiple users. Such techniques, however, generally fail in offering guaranteed large privacy regions at reasonable query processing costs. In this paper, we propose the PAD approach that is capable of offering privacy-region guarantees. To achieve this, PAD uses so-called dummy locations that are deliberately generated according to either a virtual grid or circle. These cover a user's actual location, and their spatial extents are controlled by the generation algorithms. The PAD approach only requires a lightweight server-side front-end in order for it to be integrated into an existing client/server mobile service system. In addition, query results are organized according to a compact format on the server, which not only reduces communication cost, but also facilitates the result refinement on the client side. An empirical study shows that our proposal is effective in terms of offering location privacy, and efficient in terms of computation and communication costs.

1. INTRODUCTION

1.1 Overview

The Internet is on the brink of going mobile. An infrastructure is rapidly emerging that encompasses large numbers of users equipped with mobile terminals that posses geo-positioning capabilities, e.g., built-in GPS receivers, and data communication capabilities.

Location-based services are increasingly becoming available that return results relative to the locations of their users. For example, a service may return the nearest gas station or all four-star Chinese restaurants downtown. Users who disclose their locations to a server in order to receive such a service are not afforded any location privacy, and knowledge of their locations can be misused [20].

We see location privacy as an enabling technology for the diffusion of the mobile Internet and the use of location-based services. By offering users the ability to choose different types and degrees of location privacy, users are more likely to want to use mobile services.

The paper assumes a typical service usage scenario. To be able to use a service provider's services, a user needs to create an account and to log onto that account—this is akin to, e.g., iGoogle$^{\text{TM}}$. Consequently, the service provider knows the identity of each service user. In this setting, we assume that the user employs a service that relies on queries that take the user's current location as an argument; and we assume that the user wishes to not disclose his or her exact location to the service provider.

Our objective is then to provide a technique that enables the user to employ such services, but without disclosing his or her exact location. The service provider is expected to be able to infer the exact location of the (known) user with low probability.

1.2 Technical Motivation

The earliest proposal for location privacy protection is *spatial cloaking* [11]. Instead of sending a single user's exact location to the server, spatial cloaking techniques [3,4,8–11,17] collect k user locations and send a corresponding (minimum) bounding region to the server as the query parameter. The collection of different mobile user locations is done either by a trusted third-party component [8, 11, 17] in-between the clients and the server, or via a peer-to-peer collaboration [4, 9, 10] among mobile users. Because of the k-anonymity [19] achieved, an adversary can only identify a location's user with probability no higher than $1/k$.

Next, *obfuscation* techniques [1, 6, 7, 15] include fake or fixed locations, rather than those of other mobile users, as parameters of queries sent to the server. Fake dummy locations are generated at random [15], and fixed locations are chosen from special ones such as road intersections [6, 7]. Either way, the exact user locations are hidden from the server.

Other existing techniques include *cryptographic* protocol-based approaches [12,14]. Based on a specific transformation fully known only to the clients, the server processes user queries without the ability to decipher exact user locations. The drawback of such techniques is that the query results returned by the server do not offer correctness guarantees.

Existing location privacy techniques exhibit two significant limitations. First, some require a trusted third-party anonymizer that maintains all user locations. Such a component may not always be available, and it may itself present security/privacy problems. Second, the underlying k-anonymity that some techniques borrow directly from techniques for identity privacy protection is generally inadequate for location privacy, where the notion of distance between locations is important (unlike distances between identities).

For example, Figure 1 contains a total of 100 mobile users within a 1 km by 1 km region of interest. Assume that a query by the user with location q is given 10-anonymity. Because the user is in a

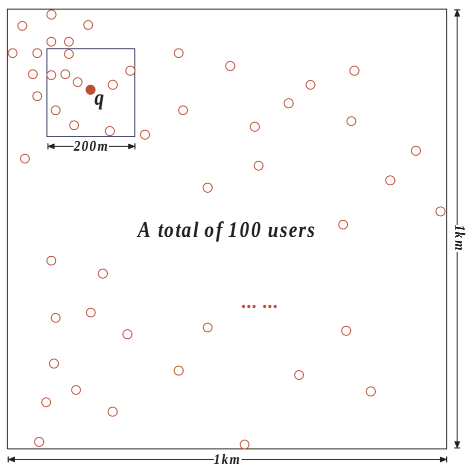


Figure 1: k-anonymity in A Spatial Context

dense part of space, the small 200 m by 200 m dashed anonymity region in the figure is generated. This region does not afford the user the desired location privacy. For example, an adversary can easily track down the user.

To obtain a larger privacy region, we intuitively need to use a larger value for k. In the example, if we need a 1 km by 1 km cloaking region, k must be set at 100. Unfortunately, simply increasing k may not help. If the user is in a high-density sub-region, e.g., at a bus stop with many people waiting for buses, we may need a very large k in order to obtain a suitably large privacy region. In contrast, if the density is low, even a small k can bring about a very large cloaking region, which may then result in a very large intermediate query result. Consequently, communication cost between the server and third-party anonymizer will become higher, as will the result refinement cost on the anonymizer. In other words, k-anonymity faces problems in offering region-based location privacy.

The example highlights two important observations. First, k-anonymity by itself is not enough to ensure location privacy. Rather, we should take privacy area constraints into account as well. Second, the effect of k-anonymity-based location privacy depends heavily on the distribution and density of the mobile users, which, however, are beyond the control of the location privacy technique. The Casper [17] requires each mobile user to specify a privacy profile that includes two parameters k and A_{min}, where k reflects the anonymity requirement and A_{min} indicates at least how large the cloaking region should be. Although both aspects are taken into account, the Casper is unable to balance between them, but is dependent on the mobile user distribution, due to the cloaking approach used.

For techniques based on cryptographic transformation, the key problem is the impaired query accuracy. Also, such techniques require a trusted third entity to accomplish the cryptographic transformation. This requirement may be difficult or impossible to meet.

1.3 Proposal Overview

Motivated by the observations above, we propose PAD, a privacy-area aware, dummy-based approach to user location privacy in mobile services. This approach takes into account the number of locations in one query request being sent to the server, as well as the area of the region covered by those locations. This duality makes it more appropriate in spatial contexts than purely k-anonymity-based techniques.

We present two dummy generation algorithms. One generates dummies based on a virtual grid covering the user location. The other generates dummies based on a virtual circle that contains the user location. These algorithms are flexible in that the dummy generation is configurable and controllable, thus offering means of controlling the location privacy of a user. This contrasts past work where location dummies are either generated totally at random [15] or are selected among fixed and constrained options [6, 7].

The PAD approach can be easily integrated into existing systems that employ client/server architectures. It does not require a trusted third-party component to server as an anonymizer, and nor does it assume that the server is trustable. In the server side, a lightweight front-end module suffices to render the approach functional. In addition, the query results to be sent to clients are organized according to a compact format with respect to all dummy locations in a query. This format not only reduces the communication cost, but also facilitates the result refinement on the client side.

Empirical evaluation show that our proposals are effective in terms of offering location privacy, are efficient in terms of computation and communication costs, and are flexible in terms of balancing between privacy requirements.

1.4 Paper Outline

The rest of the paper is organized as follows. Section 2 presents preliminaries relevant to the PAD approach. Section 3 details the PAD approach, covering two specific dummy generation algorithms and the integration of PAD into existing systems. Section 4 empirically evaluates our solution. Section 5 briefly revisits related work, and Section 6 concludes the paper.

2. PRELIMINARIES

Our proposal assumes a client/server architecture without a third-party anonymizer. This is motivated by several considerations. First, the client/server architecture is used widely, which affords our proposal wide applicability. Second, a mobile terminal does not need to report its location periodically to an anonymizer, as is needed in spatial cloaking solutions where the anonymizer needs up-to-date location information from all mobile terminals in order to do the cloaking. In our proposal, a mobile terminal only issues queries to the server on demand without any periodical location reports. Third, our proposal setting is based on the seemingly realistic assumption that the adversary knows what the server knows, i.e., the parameters and result of a query, and the identity of that query issuer. We focus on protecting user location privacy when they issue snapshot queries which are expected to occur frequently in practice.

2.1 Definitions

A location-dependent query is abstracted as $Q = (pos, \mathcal{P})$, where parameter pos is the mobile user location and parameter $\mathcal{P}$ denotes user-specified predicates. We call such a query Q the *original query*. With the location dummy approach, the original query is typically converted into a query $Q' = (pos_1, pos_2, \ldots, pos_k, \mathcal{P})$, where the pos_i include the user's real location and $k - 1$ dummy locations, and $\mathcal{P}$ is the original query predicate that applies to all k locations. We call query Q' an *location privacy query*, since it hides the user location.

Note that we use the same predicates $\mathcal{P}$ for all positions in Q', namely the one from the original query. There is no need for varying the predicates for different dummy locations—the variance in locations already yields the desired location privacy protection. In fact, the use of different $\mathcal{P}$s might be exploited by an adversary.

As pointed out in Section 1, the use of k-anonymity for offering location privacy has a crucial drawback: no guarantees are given of the form that user's location cannot be identified within a certain region. As we consider it important to be able to offer such guarantees, we proceed to introduce the relevant concepts.

DEFINITION 1. **(Privacy Region)** *The* privacy region *of a location privacy query is the convex hull of all positions contained in the query.*

Given that the convex hull is the minimum convex region that covers all positions in a location privacy query, it is a natural and conservative guess on the part of an adversary who knows all pos_i of where the user may be located. Put differently, the convex hull is a lower bound on the region within which an adversary may expect to find the real user location.

The area of a privacy region indicates the effort that an adversary needs to expend in an attempt at identifying the user's actual location. We therefore define the area of the privacy region of a location privacy query as the query's *privacy area*.

With this notion of privacy area, we are able to define our own location privacy notion, which is similar to, but semantically different from that employed by Casper [17].

DEFINITION 2. **($\langle k, s\rangle$-privacy)** *A location privacy query obeys* $\langle k, s\rangle$-privacy *if it contains k locations and its privacy area is no smaller than s.*

This notion of $\langle k, s\rangle$-privacy takes into account both the number of privacy locations and the privacy area. It should not simply be regarded as an extension of pure k-anonymity, as the k has a different meaning from that used in k-anonymity, which requires that at least k different users are unidentifiable from each other so that an adversary is unable to associate some sensitive information to any specific individual user. In contrast, we use $\langle k, s\rangle$-privacy to protect user locations, not user identities, from being known by the service provider. The additional area constraint s controls the size of the region covered by a location privacy query and thus ensures location privacy in a "real" spatial sense.

2.2 Enforcing $\langle k, s\rangle$-Privacy

A conventional spatial cloaking approach does not ensure that its anonymous queries obey $\langle k, s\rangle$-privacy. Most existing spatial cloaking techniques focus on satisfying the k-anonymity, which is intended to protect user identities, rather than the user locations, from being inferred by an adversary. As a result, the implicit privacy area requirement is downplayed. Although Casper [17] improves the cloaking by introducing an area constraint into its privacy profiles, it still faces difficulty because it depends heavily on the distribution of actual mobile users as indexed by a hierarchy of grid structures. If a grid cell is dense with respect to mobile users, Casper merges adjacent cells to enlarge the privacy region. However, this may group too many users and result in privacy regions that are overly large, which may increase the cost of query processing, including that in result refinement. If there are too few mobile users, any spatial cloaking technique fails at ensuring k-anonymity, let alone $\langle k, s\rangle$-privacy.

Next, existing obfuscation techniques largely ignore the indication of privacy areas. They choose as dummies either prefixed locations or randomly generated, fake positions, which may fail in ensuring privacy areas that comply with the expected privacy area requirement s.

Our proposal is based on dummies: it generates multiple location dummies and converts the original query Q into a location privacy query $Q' = (pos_1, pos_2, \ldots, pos_k, \mathcal{P})$. The dummy generation takes into account the privacy area requirement s, so that the area of the privacy region covered by all k positions (those of the dummies and the user's actual location) is equal to or close to s.

The actual user location is hidden among the k locations in Q', so the server cannot tell which location belongs to the user. In contrast, the client maintains a *privacy index*, namely the index of the exact user position in the position sequence sent to the server. This privacy index, denoted idx, is used by the client to determine the real query result from all points received from the server, as will be shown towards the end of Section 3.2.

2.3 Pitfalls in $\langle k, s\rangle$-Privacy

Extreme cases exist in which $\langle k, s\rangle$-privacy may fail to offer the location privacy expected by a user. Such extreme cases are related to the distribution of the positions in a location privacy query. For example, if k' positions, where k' is close to k, in query $Q' = (pos_1, pos_2, \ldots, pos_k, \mathcal{P})$ are very close, this can have two negative consequences for $\langle k, s\rangle$-privacy. An adversary will probably ignore the few outliers and infer with high probability that the user's location is among the k' ones. This effectively causes a smaller k in the $\langle k, s\rangle$-privacy. Further, the privacy region becomes the convex hull of the k' positions, yielding a privacy area that may be much smaller than the expected s.

To counter these problems, we must have some control over the distribution of the dummy positions in location privacy queries. In particular, we must avoid extreme cases in which many dummy positions are clustered very closely. Our dummy generation algorithms, to be presented in Section 3.1, avoid extreme distribution cases by generating all dummy locations evenly with respect to the exact user location.

3. PAD: PRIVACY-AREA AWARE DUMMY

We proceed to detail our proposal: Privacy-Area Aware Dummies-based location privacy (PAD). We propose two dummy generation algorithms, describe the corresponding server-side query processing, and present techniques for reducing the client/server communication costs.

3.1 Privacy-Area Aware Dummy Generation Algorithms

Generating the dummy locations totally at random provides little control [15]. We are interested in approaches for generating the dummies that aid in satisfying $\langle k, s\rangle$-privacy. We thus propose two privacy-area aware dummy generation algorithms. The one algorithm employs a circle to constrain all dummy positions, while the other uses a uniform grid.

3.1.1 Circle-Based Dummy Generation

To understand the idea behind the circle-based dummy generation, consider the example in Figure 2(a), where $k = 9$ and pos is the user location. All locations, including the user location and the dummy locations, are constrained by a circle centered at position pos' with radius r. The center pos' is determined at random such that $r_{min} \leq dist(pos, pos') \leq r$.[1] We will show how to determine the values r_{min} and r given the privacy area requirement s. For each pair of clock-wise consecutive positions and pos', they determine an angle θ. All positions are distributed in such a way that all θs are equivalent.

We first consider the hull of all positions, whose area is the sum

[1] $dist(p, q)$ denotes the Euclidean distance between points p and q.

of areas of k triangles:

$$\widetilde{s} = \sum_{i=1}^{k} \frac{1}{2} \cdot r_i \cdot r_{i+1} \cdot sin\theta = \frac{1}{2} \cdot \sum_{i=1}^{k} r_i \cdot r_{i+1} \cdot sin\frac{2\pi}{k},$$

where r_i is $dist(pos_i, pos')$. To ease the representation, we let r_{k+1} be equal to r_1. It is easy to understand that this hull is not necessarily convex and that $\widetilde{s} \leq \widehat{s}$, where $\widehat{s}$ is the area of the corresponding convex hull.

If we assume that all positions have identical distance to pos', the hull determined by them is necessarily convex. Thus, taking into account the privacy area requirement s, we have:

$$\widetilde{s} = \widehat{s} = \frac{1}{2} \cdot k \cdot r_i^2 \cdot sin\frac{2\pi}{k} = s$$

Solving this, we get an upper bound $r = \sqrt{(2 \cdot s)/(k \cdot sin\frac{2\pi}{k})}$. Let $r_{min} = \rho \cdot r$, where $0 < \rho \leq 1$. We then get:

$$\widehat{s} \geq \widetilde{s} \geq \frac{1}{2} \cdot k \cdot r_{min}^2 \cdot sin\frac{2\pi}{k} = \frac{1}{2} \cdot k \cdot (\rho \cdot r)^2 \cdot sin\frac{2\pi}{k} = \rho^2 \cdot s$$

This indicates a lower bound of the privacy area of all the k positions, i.e., $\widehat{s} \geq \rho^2 \cdot s$. As a result, by carefully choosing ρ, we can get a guarantee on the privacy area of the location privacy query generated based on a virtual circle. For example, if we choose $\rho = \sqrt{3}/2$, we can ensure that the resulting privacy area is not smaller than three quarters of s.

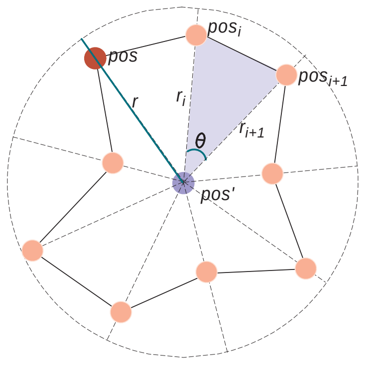
The algorithm, called **CirDummy**, is shown in Algorithm 1. The

Algorithm 1 CirDummy (User position pos, anonymity k, privacy area s, coefficient ρ)

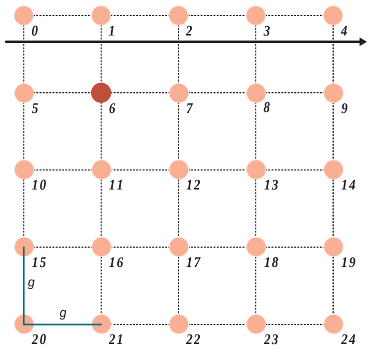
```
1: $\theta \leftarrow 2 \cdot \pi / k$;  $r \leftarrow \sqrt{2 \cdot s / (k \cdot \sin\theta)}$
2: determine a position $pos'$ at random s.t. $dist(pos, pos') \in [\rho \cdot r, r]$
3: initialize $K[0..k-1]$ to be an empty array; $K[1] \leftarrow pos$
4: for $i$ from 1 to $k-1$ do
5:     determine a position $p$ at random s.t. $dist(pos', p) \in [\rho \cdot r, r]$ and $\angle p\, pos'\, K[i-1] = \theta$
6:     $K[i] \leftarrow p$
7: $idx \leftarrow random(0, k-1)$
8: switch $K[0]$ with $K[idx]$
9: return $K$ and $idx$
```

algorithm first calculates the upper-bound radius r of the virtual circle and the angle θ between any consecutive position pair (line 1). Next, it determines the virtual circle center pos' at random (line 2) so that its distance to pos falls in $[\rho \cdot r, r]$. Then, it initializes an empty array K and makes the user position pos its first element (line 3). The $k-1$ dummy locations are generated as discussed already: their distances to the virtual center pos' are constrained by $[\rho \cdot r, r]$, while they are scattered evenly in terms of their angles with respect to pos' (lines 4–6). Having generated all dummies, a random index idx between 0 and $k-1$ (both inclusive) is chosen for the user position pos, which is switched with the idx-th element (lines 7–8). Finally, K and idx are returned.

When the coefficient ρ is very close to 1, it may happen that all positions are too close to the virtual circle edge. This leaves a large blank region around the center pos' of the virtual circle, which is not desirable. The CirDummy algorithm can be adapted to handle such cases. We can generate half the dummies using the CirDummy algorithm, and generate the other half at random making their distances to pos' fall in $(0, \rho \cdot r]$. To make it safer, the distance between pos and the user position pos is not constrained by ρ, but falls in $(0, r]$ (line 2 in Algorithm 1).



(a) Circle Based



(b) Grid Based

Figure 2: Privacy-Area Aware Dummy Generation Examples

3.1.2 *Grid-Based Dummy Generation*

The grid-based dummy generation works as follows. A (virtual) uniform, square grid is created, such that (1) it has k vertices, (2) its area is equal to s, and (3) the user position pos is one of the k vertices. The $k-1$ other vertices will be used as dummy locations, to be sent to the server together with the user position pos.

Algorithm 2 GridDummy (user position pos, anonymity k, privacy area s)

```
1: $c \leftarrow \sqrt{k}$
2: $id_x \leftarrow random(0, c-1)$; $id_y \leftarrow random(0, c-1)$
3: $g \leftarrow \sqrt{s}/(c-1)$
4: initialize $K[0..k-1]$ to be an empty array
5: for $i$ from 0 to $c-1$ do
6:     for $j$ from 0 to $c-1$ do
7:         $x \leftarrow (i - id_x) \cdot g + pos.x$
8:         $y \leftarrow (j - id_y) \cdot g + pos.y$
9:         $K[j * c + i] \leftarrow (x, y)$
10: return $K$ and $id_y * c + id_x$    // the index of $pos$ in $K$
```

The algorithm, called **GridDummy**, is shown in Algorithm 2. It first calculates the number of vertices in either direction (the x and y axes) as the square root of k (line 1). Next, it attaches the user position pos to one of the vertices, by generating the corresponding x and y indices at random (line 2). Based on the given area s and value c, the algorithm is able to determine the side length g of each grid cell (line 3). It then calculates the position for each grid vertex in row-major order and enters all positions into an array K in that order (lines 4–9). Finally, it returns both the position array K and the index of the user position pos in K.

We assume that k is a square number. If k is not a square number, we can simply use the smallest square number $k' > k$.[2] For a location privacy query with dummy positions generated by the GridDummy algorithm, its privacy region is simply the rectangle (square) indicated by the grid, as such a rectangle perfectly covers all the k locations. It does not make sense for an adversary who is aware of all positions in the query to assume any other shape in an attack.

An example of grid-based dummy generation is shown in Figure 2(b), where k is 25, with 5 vertices in either direction. All vertices are numbered in row-major order. The user position (dark color) is attached to vertex 6, which means $id_x = id_y = 1$ in the algorithm. The array K returned contains all positions from vertices 0 to 24.

3.2 Server-Side Processing

Having received a location privacy query $Q' = (pos_1, pos_2, \ldots, pos_k, \mathcal{P})$ from a mobile client, the server processes this according to predicate $\mathcal{P}$.

An extension is needed to ensure that the location privacy query is served correctly. Specifically, we place a module in front of the query engine that sends all positions in Q' to the query engine one by one as separate, individual queries and that assembles the query results returned by the query engine. This module differs from the secure third-party component adopted by previous work [13, 17]; notably, the module needs not be trusted, as the query request already hides the user location.

To reduce the server-to-client communication, the results for all k positions that need to be sent back to the client are organized in a compact manner as follows. Assume that there are l distinct points in the results. Then the result is represented as $R = (\langle r_1, bmp_1\rangle, \langle r_2, bmp_2\rangle, \ldots, \langle r_l, bmp_l\rangle)$, where each r_i is a point retrieved by the query engine, and each bmp_i is a bitmap telling which positions in Q' have r_i in their results.

The algorithm underlying the server-side module is presented in Algorithm 3. The first step is to initialize two vectors, R for all points retrieved and L for all bitmaps. For each position pos_i in Q', the algorithm sends this position together with the predicate to the query engine; it receives the result R_i, and it then updates the result R and bitmaps in L accordingly (lines 3–11). To facilitate the checking of whether a point pt in R_i has already been included in the result R (line 6), we use a memory resident R-tree to index all points in R. If pt has appeared, the corresponding bitmap in L will have its i-th bit set, to indicate it belongs to the result of the i-th position (lines 6–7). Otherwise, pt will be appended to R, and a new bitmap is appended to L (lines 9–11). Finally, both R and L are returned to the client.

Algorithm 3 ServerModule (Location privacy query Q')

```
1:  initialize the final-result vector $R$
2:  create a vector $L$ to contain all bitmaps
3:  for $i$ from 1 to $k$ do
4:      send $(Q'.pos_i, Q'.\mathcal{P})$ to query engine and get its result $R_i$
5:      for each point $pt \in R_i$ do
6:          if $pt$ appears in $R$ as the $j$-th one then
7:              set bit $i$ of $L[j]$
8:          else
9:              append $pt$ to $R$
10:             create a bitmap $bmp$ with bit $i$ set
11:             append $bmp$ to $L$
12: send $R$ and $L$ to the client
```

[2]As a variation, k can be a composite number, meaning that the virtual grid becomes a rectangle. The x and y directions then have different numbers of vertices. Due to space limitations, we present the fundamental idea only and omit minor variations possible.

Like the server-side packing, the client-side result refinement is quite simple: a single loop on all received points is enough. For each point r_i, we apply a bit-wise AND operation to its bitmap bmp_i and the value of operation $1 << (k - idx - 1)$. Point r_i is in the final result if the AND operation returns a value larger than zero.

3.3 Communication Cost Analysis

3.3.1 Upstream Communication Cost

With a 2D location taking up 8 bytes, the size of a raw request containing a total of k locations is expressed as:

$$|Req_{raw}| = 8k \tag{1}$$

Restructuring a raw query request [15] by putting all x coordinate values in front of y values, the cost can be reduced to roughly $16\sqrt{k}$.

When we employ the virtual grid approach to generate dummies, we can further reduce the cost by sending the grid configuration only in the request. The configuration of a uniform grid is given by 3 parts: the top-left corner location (8 bytes), the side length of each square grid cell (4 bytes), and the number of grid cells in the horizontal/vertical direction (1 byte). Therefore, a request with the grid configuration consumes 13 bytes only.

3.3.2 Downstream Communication Cost

Let R_i be the result that corresponds to the i-th position in query Q'. Then the size in bytes of a raw result message without packing is:

$$|Res_{raw}| = 8\sum_{i=1}^{k} |R_i| \tag{2}$$

With the packing described in Section 3.2, the result message size shrinks to:

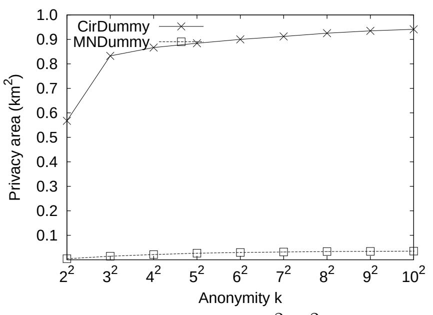
$$|Res_{packed}| = (8 + \lceil k/8 \rceil)|\bigcup_{i=1}^{k} R_i| \tag{3}$$

The data reduction rate achieved by the packing is therefore defined as:

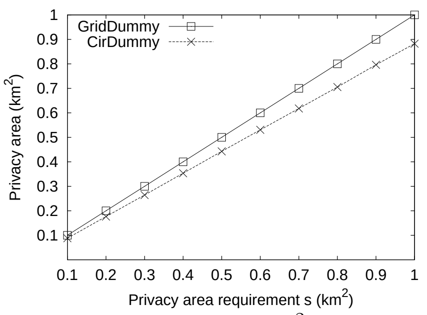
$$DRR = \frac{|Res_{raw}| - |Res_{packed}|}{|Res_{raw}|} \tag{4}$$

3.4 Summary

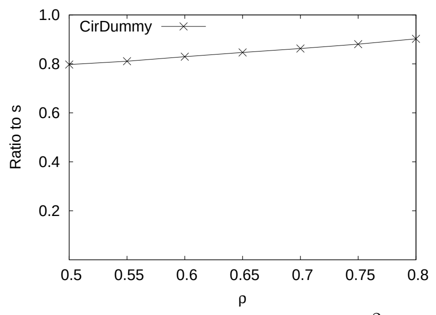
The PAD approach has several noteworthy properties. First, the dummy generation algorithms are aware of the privacy area requirement, enabling them to support region-based location privacy—privacy regions with desired sizes may be returned. Second, PAD is easy to implement and to integrate into a client/server architecture because it does not assume a trusted third-party component. Its dummy generation algorithms are quite simple yet effective, especially compared to cloaking techniques [8, 17] that depend heavily on the distribution and density of the mobile-user population. Third, PAD incorporates techniques that reduce both the upstream and downstream communication between client and server. We proceed to evaluate PAD empirically on both real and synthetic datasets.



(a) Effect of k ($s = 1\text{km}^2$, $\rho^2 = 0.75$)



(b) Effect of s ($k = 25$, $\rho^2 = 0.75$)



(c) Effect of ρ ($k = 25$, $s = 1\text{km}^2$)

Figure 3: Privacy Area

4. EMPIRICAL EVALUATION

In the empirical evaluation, we consider three performance aspects: privacy area, communication cost, and server module cost.

4.1 Settings

We use randomly generated uniform (UI) datasets and a real-world dataset in our experiments. The latter, called SC, contains 162,896 schools in the US mainland. It is obtained from a complete set of 172,188 schools across the entire country[3]. All datasets are normalized to a 10,000 × 10,000 square 2D space on the server side and are indexed by R-trees with a 1K byte page size. Within the data space, we generate 100 points at random and use them as query positions in each experiment. The performance figures reported are averages of all 100 queries.

All algorithms are written in Java and run on a Windows XP PC with a 2.8GHz Intel Pentium D CPU and 1GB RAM. All experimental parameters and their settings are listed in Table 1, with default values given in bold.

Parameter	**Setting**
Spatial extents	10000 × 10000
UI dataset cardinality, N	100K, 200K, **500K**, . . ., 1000K
Anonymity, k	2^2, 3^2, . . ., $\mathbf{5^2}$, . . ., 10^2
Privacy region area, s	$0.1{\cdot}1000^2$, $0.2{\cdot}1000^2$, . . ., $\mathbf{1000^2}$
K in *KNN* queries	10, 20, 30, 40, 50
Range in range queries	100, 200, 300

Table 1: Parameters Used in Experiments

4.2 Privacy Area

This batch of experiments compares PAD's two dummy generation algorithms with the MN dummy generation algorithm [15], which places the next dummy position within the neighborhood of the current one. For both the MN algorithm and the CirDummy algorithm, we use the convex hull of all positions in each location privacy query as the privacy area. We use $\rho^2 = 0.75$ as the default in the CirDummy algorithm.

According to the results reported in Figure 3(a), the privacy area of CirDummy is always considerably larger than that of the MN algorithm when varying the anonymity k. This is attributed to the fact that CirDummy is aware of the privacy area requirement, while MN is not. When k changes from 4 to 9, the privacy area of CirDummy exhibits an apparent increase, and then it grows slowly as k increases. This indicates that CirDummy does not achieve good privacy areas with only few dummies, but with enough dummies (here, at least 9) its result is better than the expectation indicated by the coefficient ρ.

Figure 3(b) plots the achieved privacy areas of CirDummy and GridDummy when varying the privacy area requirement s. As s increase, GridDummy always produce exactly sized privacy regions, and CirDummy achieves the guarantee indicated by ρ. Figure 3(c) reports on the effect on the privacy area of CirDummy for varying ρ. It is seen that the privacy area is at least 80% of the privacy area requirement s.

As the GridDummy algorithm always produces exactly sized privacy regions, we focus on it in the sequel.

4.3 Communication Cost

We first consider the upstream communication cost, i.e., the query request message size in bytes. Figure 4 covers three approaches: all positions are sent without any reduction ("Dummy"); dummy reduction [15] is used ("Dummy with reduction"); the virtual grid configuration is sent ("GridDummy") as described in Section 3.3.1. GridDummy is a clear winner, as it needs only send the grid configuration rather than all positions. When k grows the communication cost savings therefore increase.

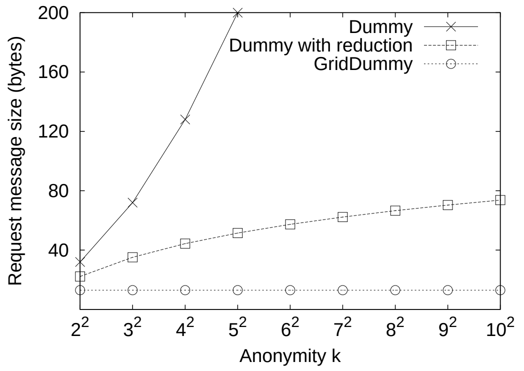
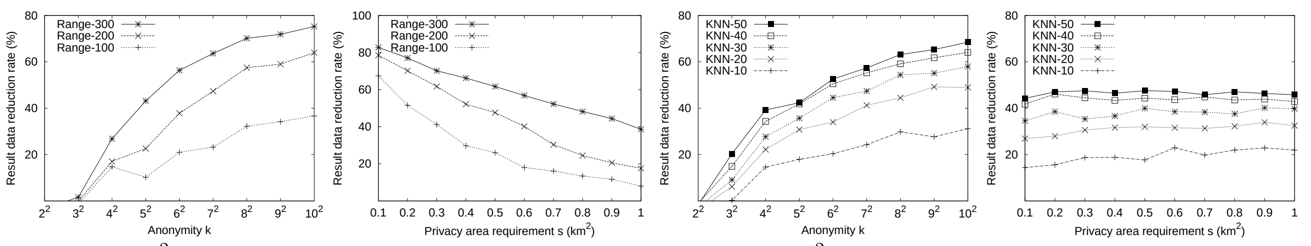


Figure 4: Upstream Communication Cost

We next investigate how the downstream communication cost, i.e., the query result message size, is affected by k and s. Figure 5 reports on the result data reduction rate (Formula 4) for the SC dataset.

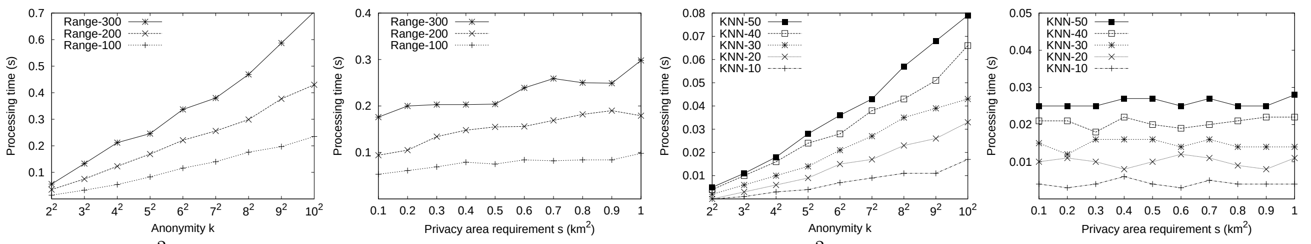
Figures 5(a) and 5(b) cover range queries. It is shown in Figures 5(a) that as the anonymity requirement k grows, the communication cost savings also increase. Given a fixed privacy area s, the more positions a location privacy query contains, the more overlap these positions' results have. Consequently, the packing of results gains and becomes more efficient.

[3]U.S. Board on Geographic Names, `http://geonames.usgs.gov/index.html`.



(a) $s = 1\mathbf{km}^2$, Range query (b) $k = 25$, Range query (c) $s = 1\text{km}^2$, KNN query (d) $k = 25$, KNN query

Figure 5: Result Data Reduction Rate on SC Dataset



(a) $s = 1\text{km}^2$, Range query (b) $k = 25$, Range query (c) $s = 1\text{km}^2$, KNN query (d) $k = 25$, KNN query

Figure 6: Server-Side CPU Cost on SC Dataset

Next, Figure 5(b) shows that an increasing privacy area requirement s decreases the communication cost saving. As the privacy region expands for a fixed number of positions, these positions are distributed more apart. This results in less overlap among their results, which reduces the result reduction. Both figures also indicate that when other settings are fixed, the result reduction savings increase as the query range predicate increases. A larger range predicate retrieves more data points, increases the overlap among results of different positions, and therefore improves the result reduction. Figures 5(a) and 5(b) indicate that small anonymity requirement k values are not good choices, but a medium-sized privacy area requirement s strikes a good balance between location privacy and the overall result transmission cost for range queries.

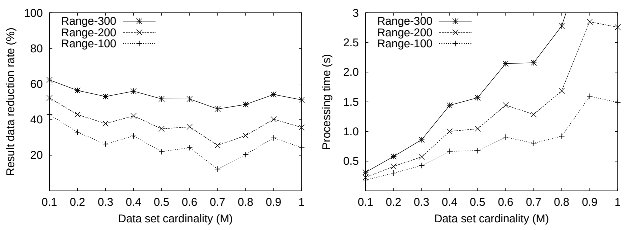
Figures 5(c) and 5(d) report results for KNN queries. Here, the privacy area requirement s has little effect on the communication cost savings, as shown in Figure 5(d). Nevertheless, both larger anonymity requirement k values and larger K values have a positive effect. The reason for the former is similar to that mentioned earlier for range queries. A larger value of K increases the overlap among the results of different positions when other settings are fixed, thus increasing the result data reduction. Because the result cardinality of a KNN query is usually smaller than that of a range query, the anonymity requirement k has a more significant effect than does the privacy area requirement s. Small k values are not good choices for KNN queries.

Next, we consider how the result data reduction is affected by the dataset cardinality. The results shown in Figure 7(a) indicate the effect of the cardinality is insignificant. Due to space limitations, we show results for range queries; those of KNN queries exhibit similar trends, but the values are slightly lower.

4.4 Server-Side Cost

For the experiments covered in Section 4.3, we also obtain the CPU processing time of the server-side module. The results obtained on the SC dataset are reported in Figure 6.

Referring to Figure 6(a) and Figure 6(c), larger anonymity requirement k values cause higher CPU processing time on the server side for both range queries and KNN queries. This is because a larger k value causes more positions in a location privacy query sent to the server-side module for processing. Figure 6(b) and Figure 6(d) show that the privacy area requirement s has a very limited impact on the CPU cost of the server-side module. The server-side module design does not depend on the privacy area in an incoming location privacy query. To some degree, however, the privacy area can determine which and how many data points are retrieved from a given dataset, which moderately affects the CPU processing time consumed by the server-side module. All results here suggest that the extra cost of the server-side module is very low, especially for KNN queries, which retrieve relatively few data points.



(a) Result Data Reduction Rate (b) Server-Side CPU Cost

Figure 7: Effect of N for UI ($k = 25$, $s = 1\text{km}^2$)

Figure 7(b) reports on how the server-side module processing time is affected by the dataset cardinality. It is seen that larger dataset cardinalities increase the CPU processing time. This is because more data points are retrieved by the module as query results.

5. RELATED WORK

In this section we review the existing location privacy protection techniques proposed in related work. Existing techniques can be roughly classified into three categories.

The first category contains *spatial cloaking* techniques [2–5, 8–11, 16, 17]. Based on the k-anonymity [19] privacy notion in information publishing, spatial cloaking techniques do not send a single user's exact location to the server. Instead, they collect at least k user locations and send a (minimum) region covering all these locations to the server as the query. This way, the k-anonymity [19] is achieved. Therefore, an adversary can only guess which location belongs to which user with probability no higher than $1/k$. To collect multiple mobile-user locations, the conventional client/server architecture is extended. Either a trustable third-party component [8, 11, 17] in-between the clients and the server is introduced, or peer-to-peer collaboration [4, 9, 10] is exploited.

The second category contains *obfuscation* techniques [1,6,7,15], which use fake or fixed locations rather than those of other mobile users when cloaking a query. One approach [15] generates at random a number of fake locations, called dummies, and sends them and the user's location to the server, thus hiding the user's location. Another approach [6,7] directly uses special locations such as road intersections as dummies. Motivated by inaccuracy in location positioning technologies, a recent obfuscation-based technique [1] employs a circular region to model each user location, and accordingly sends obfuscated regions instead of positions to the server.

The third category encompasses techniques based on *cryptographic* protocols [12, 14]. In relevant approaches, both the data stored on the server and the query location sent to the server are transformed into a specific format. Because that format is fully known by the clients only, the server has great difficulty in finding out the exact user locations when processing queries. However, such techniques only return approximate query results without correctness guarantees, and they require a trusted third-party component to carry out the secret data transformation.

Finally, the recent SpaceTwist proposal [18] does not belong to any category above. In SpaceTwist, a client asking for nearest neighbors sends a fake location, called an anchor, to the server. The server incrementally processes the KNN query with respect to the anchor, returning nearest neighbor points incrementally to the client. The client then continuously retrieves points from the server and updates its query result with respect to its exact location kept locally until it finds that the accurate nearest neighbors are contained among the points retrieved so far.

6. CONCLUSION AND RESEARCH DIRECTIONS

In this paper we propose PAD, a privacy-area aware, dummy-based location privacy protection technique for mobile services. We design two dummy generation algorithms that take into account privacy area requirements. The paper describes how to integrate PAD into a client/server architecture so that the location privacy of mobile users is preserved and the client/server communication costs are reduced, at the expense of insignificant server-side costs. We report on an empirical evaluation of PAD. It is noteworthy that PAD is effective in offering area-based location privacy, which is not the case for existing dummy-based techniques. In addition, PAD is efficient: it requires insignificant extra server-side costs and is capable of reducing the client-server communication costs.

Several direction for future research exist. First, it is possible to extend PAD to 2-dimensional space with obstacles, which are regions where service users cannot be located. The dummy generation algorithms must still satisfy the privacy requirements, but must take into account the obstacles. Second, it is relevant to consider privacy-area (or some other privacy metric) aware location privacy in a spatial network setting. Third, while the paper considers only snapshot queries, it is of interest to extend PAD to support also continuous queries that can be issued by mobile users.

7. REFERENCES

[1] C. A. Ardagna, M. Cremonini, E. Damiani, S. D. C. di Vimercati, and P. Samarati. Location Privacy Protection Through Obfuscation-Based Techniques. In *Proc. DBSec*, 2007.

[2] C. Bettini, S. Mascetti, X. S. Wang, and S. Jajodia. Anonymity in Location-Based Services: Towards a General Framework. In *Proc. MDM*, 2007.

[3] C.-Y. Chow and M. F. Mokbel. Enabling Private Continuous Queries For Revealed User Locations. In *Proc. SSTD*, 2007.

[4] C.-Y. Chow, M. F. Mokbel, and X. Liu. A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-based Services. In *Proc. ACM GIS*, 2006.

[5] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar. Preserving User Location Privacy in Mobile Data Management Infrastructures. In *Proc. PET*, 2006.

[6] M. Duckham and L. Kulik. A Formal Model of Obfuscation and Negotiation for Location Privacy. In *Proc. PERVASIVE*, 2005.

[7] M. Duckham and L. Kulik. Simulation of Obfuscation and Negotiation for Location Privacy. In *Proc. COSIT*, 2005.

[8] B. Gedik and L. Liu. Location Privacy in Mobile Systems: A Personalized Anonymization Model. In *Proc. ICDCS*, 2005.

[9] G. Ghinita, P. Kalnis, and S. Skiadopoulos. MobiHide: A Mobile Peer-to-Peer System for Anonymous Location-Based Queries. In *Proc. SSTD*, 2007.

[10] G. Ghinita, P. Kalnis, and S. Skiadopoulos. PRIVÉ: Anonymous Location-Based Queries in Distributed Mobile Systems. In *Proc. WWW*, 2007.

[11] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proc. USENIX MobiSys*, 2003.

[12] P. Indyk and D. Woodruff. Polylogarithmic Private Approximations and Efficient Matching. In *Proc. TCC*, 2006.

[13] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing Location-Based Identity Inference in Anonymous Spatial Queries. *IEEE TKDE*, 19(12):1719–1733, 2007.

[14] A. Khoshgozaran and C. Shahabi. Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy. In *Proc. SSTD*, 2007.

[15] H. Kido, Y. Yanagisawa, and T. Satoh. An Anonymous Communication Technique using Dummies for Location-based Services. In *Proc. ICPS*, 2005.

[16] W.-S. Ku, R. Zimmermann, W.-C. Peng, and S. Shroff. Privacy Protected Query Processing on Spatial Networks. In *Proc. ICDE Workshops*, 2007.

[17] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The New Casper: Query Processing for Location Services without Compromising Privacy. In *Proc. VLDB*, 2006.

[18] M. L. Yiu, C. S. Jensen, X. Huang, and H. Lu. SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services. In *Proc. ICDE*, 2008.

[19] L. Sweeney. k-Anonymity: A Model for Protecting Privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[20] J. Voelcker. Stalked by Satellite: An Alarming Rise in GPS-enabled Harassment. *IEEE Spectrum*, 47(7):15–16, 2006.