Non-Exposure Location Anonymity

Haibo Hu, Jianliang Xu

Department of Computer Science, Hong Kong Baptist University
Kowloon Tong, Kowloon, Hong Kong SAR, China
{haibo,xujl}@comp.hkbu.edu.hk

***Abstract*— Location cloaking has been proposed and well studied to protect user privacy. It blurs the accurate user location (i.e., a point with coordinates) and replaces it with a well-shaped cloaked region (usually a circle or a rectangle). However, to obtain such a cloaked region, all existing cloaking algorithms require to know the accurate locations of all users. Since such information is exactly what the user wants to hide, these algorithms can work only if all parties involved in the cloaking process are trusted. However, in practice this assumption rarely holds as any of these parties could be malicious. Therefore, location cloaking without exposing the accurate user location to any party is urgently needed. In this paper, we present such a non-exposure cloaking algorithm. It is designed for k-anonymity and cloaking is performed based on the proximity information among mobile users, instead of directly on their coordinates. We decompose the problem into two subproblems — proximity minimum k-clustering and secure bounding, and develop distributed algorithms for both of them. Experimental results consistently show that these algorithms are efficient and robust under various proximity topologies and system settings.**

I. Introduction

The recent consumer electronics market witnesses a booming sale of smart mobile devices. These devices, typically Smartphones and PDAs, are equipped with powerful CPU, large memory, positioning technology (e.g., GPS) and most importantly a complete set of wireless communication interfaces (e.g., Bluetooth, WiFi, and HSDPA). As such, these devices can establish not only Internet connections to external servers, but also point-to-point connections to nearby peer devices. The omnipotence of these devices opens up new applications for mobile subscribers. In particular, with the combination of GPS and wireless Internet, mobile users can enjoy *location-based services* (LBS), which provide dynamic content according to where the user is located. Typical LBS applications include road navigation, nearest point of interest (POI) query, and location-aware advertisement.

In order to enjoy such services, the mobile user must explicitly expose his/her accurate location to the server. For example, if the user asks for the nearest restaurant, he/she must provide the LBS server his/her accurate position in terms of GPS coordinates. In this sense, the user's location privacy is compromised in exchange for services. To address this issue, an intuitive solution is to cache the whole dataset of POI on the mobile device, which can then resolve location-based queries locally. However, due to limited resources of the mobile device, this solution cannot scale to large POI datasets, neither can it deal with data updates. Therefore, a more sophisticated strategy called *location anonymity* has been proposed and studied [1], [2], [3], [4], [5]. The objective is to allow the mobile user to request services without revealing the accurate location. Among various approaches proposed along this line, *location cloaking* is predominant [1], [2], [3], [6]. It blurs the accurate user location and replaces it with a well-shaped *cloaked region* (usually a circle or a rectangle), according to some anonymity metric such as *k-anonymity* (the cloaked region must contain at least k users) or *granularity* (the size of the cloaked region must exceed a threshold).

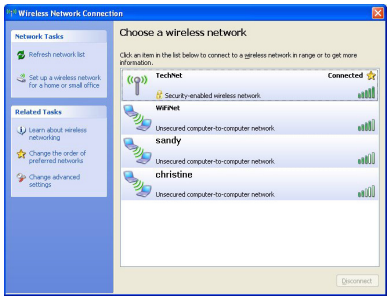


Fig. 1. RSS from WiFi Interface

In effect, location cloaking achieves privacy protection at the cost of a degrading service. The larger is the cloaked region, the more privacy is preserved, but the less specific is the request. Therefore, most existing location cloaking research focuses on minimizing the size of the cloaked region while still satisfying the anonymity metric. To this end, a number of location cloaking algorithms have been proposed for different anonymity metrics [1], [2], [3], [7], [8], [9].

However, to obtain the cloaked region and optimize its size, all existing algorithms require the accurate locations (i.e., the coordinates) of all users. As the accurate locations are exactly what the users want to hide, all existing work essentially imposes an assumption that all parties involved in the cloaking process must be trusted. Typical parties include the "anonymizers" that sit in between the user and the LBS server [2], [3], [7], and the user peers when the cloaking is performed in a peer-to-peer environment [8]. However, in practice any of these parties could be malicious and the exposure of the accurate location information to any party might reveal users' identity or other sensitive information. In this sense, existing algorithms have limited applications and location cloaking without exposing the accurate user location to any party is urgently needed.

This work was supported by the grants from the Research Grants Council, Hong Kong SAR, China (Project Nos. HKBU211206, HKBU210808, FRG/07-08/II-23, and HKBU 1/05C).

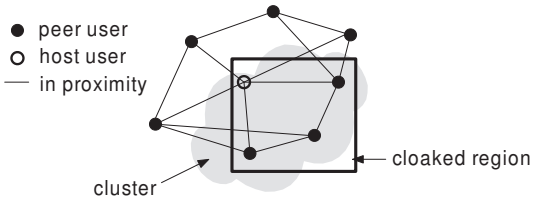


Fig. 2. Two-Phase Cloaking

In this paper, we present such a *non-exposure* location cloaking algorithm. It is designed for k-anonymity and cloaking is performed based on the *proximity* information among mobile users. Such proximity information is widely available in practice. For example, a mobile device is able to measure the closeness from its peers through its omnidirectional antenna, by either the received signal strength (RSS) from its peers (the stronger the closer), or the time difference of arrival (TDOA) of beacon signals from its peers (the shorter the closer). Fig. 1 shows the WiFi RSS of neighboring peers from a laptop computer and Fig. 2 shows a proximity graph that is based on such RSS information. A vertex in this graph stands for a user, and an edge means that the two users are WiFi neighbors.

The proposed non-exposure cloaking process is invoked by the *host user* who wants to request a location-based service. The process involves the surrounding users and is conducted in two phases. In the first phase, k users (including the host user) are identified through the proximity information. They and only they contribute to the resulted cloaked region. Moreover, if they become host users later, they will employ the same cloaked region. The shaded cloud in Fig. 2 encloses 4 users for a 4-anonymity cloaking request from a host user. In this paper, we show that this phase is equivalent to finding a cluster of size at least k in the graph. Furthermore, to minimize the size of the cloaked region, we should minimize the *diameter* of this cluster. This problem is thus called proximity *minimum* k-clustering. It is difficult particularly in the distributed environment because an earlier cluster result for a host user might significantly affect subsequent cluster results. We tackle this problem by defining an equivalent relation called t-*connected*. This leads to a nice property called *cluster-isolation*, where subsequent cluster results are immune to change. Based on this property, we present an efficient distributed algorithm for minimum k-clustering.

In the second phase, the cloaked region — a bounding box of all users in the cluster — is obtained without exposing their accurate locations. The solid box in Fig. 2 shows a possible bounding box for the cluster of 4 users. Finding this box is equivalent to obtaining lower and upper bounds of users' coordinates without revealing these coordinates. We call this problem *secure bounding*. It is related to secure multi-party computation (SMC) [10]. To reduce the size of the cloaked region, the objective of secure bounding is to obtain the bound as tight as possible with the lowest cost. We propose a progressive bounding algorithm in which a bound increases progressively until all users agree with this bound. By developing a sophisticated cost model for the communication cost, we derive the optimal increment value for this algorithm.

To summarize, our contributions in this paper are as follows:

1) To the best of our knowledge, this is the first study that explores the problem of location cloaking without exposing the accurate user locations.
2) We propose location cloaking on proximity and decompose this problem into two subproblems: proximity minimum k-clustering and secure bounding.
3) We define an equivalent relation that leads to *cluster-isolation*, based on which we design an efficient and distributed algorithm for minimum k-clustering.
4) We develop a cost model of the communication cost for secure bounding, based on which we design an optimal progressive bounding algorithm.
5) We conduct extensive experiments that show the robustness and effectiveness of the proposed algorithms for various network topologies and other system settings.

The rest of the paper proceeds as follows. Section II reviews existing work on location anonymity and privacy-aware location-based services. Section III introduces the system architecture and problem definition. Section IV presents the proximity k-clustering algorithm and Section V presents the secure bounding algorithm. The experimental results are shown in Section VI, followed by the concluding remarks in Section VII.

II. Related Work

Location anonymity has attracted intensive research as a solution to protecting user privacy in mobile computing, especially for location-based services. The objective is to allow a mobile user to request services without revealing his/her position. Among various anonymizing techniques, *location cloaking* is the predominant. It sends to the server a *cloaked region* (usually a circle or a rectangle) that contains the genuine user position and is large enough to satisfy some privacy metric. The two most widely adopted metrics are *k-anonymity* — this region must contain at least k users so that the genuine requesting user is indistinguishable from at least $k-1$ other users who have the same cloaked region, and *granularity* — the area of this region must exceed a threshold. Gruteser and Grunwald were the first to propose spatio-temporal cloaking [1] for k-anonymity, where a trusted middleware generalizes (i.e., cloaks) the spatial and temporal extents of user locations. More specifically, the middleware indexes all user locations using a Quadtree. Upon receiving a request, the middleware traverses the tree until it finds a quadrant containing the requesting user and other $k-1$ users. This quadrant is the cloaked region for this request. Gedik and Liu considered a personalized k-anonymity model and proposed Clique-Cloak, which constructs a clique graph to combine clients that can share the same cloaked region [2], [11]. A grid-based cloaking algorithm was suggested in the Casper framework [3] to address both the k-anonymity and granularity metrics. We addressed the issue when a client continuously requests location cloaking, and developed an optimal cloaking technique to resist trace analysis attacks [4].

There are some recent studies on location cloaking for distributed environments where a centralized and trusted anonymizer does not exist. Chow *et al*. extended k-anonymity cloaking to a peer-to-peer environment [8]. The main idea is to let the client form a group from his/her peers via multi-hop communication. The cloaked region of any subsequent request is then a region that covers all peers in this group. To reduce the size of the cloaked region while achieving the same k-anonymity, Ghinita *et al*. studied cloaking in a distributed environment and proposed $hilbASR$ to sort all users and store this ordering in a distributed annotated B^+-tree index [7]. In $hilbASR$, all users are sorted by Hilbert space-filling curve ordering according to their locations, and then every k users are grouped together in this order. They recently extended this framework to a Chord peer-to-peer environment [9] and used distributed hash tables, instead of the hierarchical B^+-tree to store user locations.

It is noteworthy that all these cloaking approaches require the user to expose the accurate positions to the trusted (centralized) anonymizer or peers. The approach proposed in this paper, to the best of our knowledge, is the first in the literature that eliminates this requirement.

On the server side, to support location cloaking, spatial query processing on cloaked regions has also been studied. Hu and Lee proposed *k range nearest neighbor* (kRNN) search that takes a rectangle instead of a point as the input for k nearest neighbor search. The Casper framework proposed by Mokbel *et al*. consists of both an anonymizer and a query processor. The processor evaluates spatial queries over the cloaked regions and returns a superset of results to the client for further filtering [3]. Cheng *et al*. proposed a similar framework based on location uncertainty [12], where the returned results are probabilistic results.

Besides location cloaking, other anonymizing techniques have also been proposed. *Pseudonym* decouples the mapping between the user identity and the location so that the server only receives the location without the user identity [13], [14]. However, such a technique is limited to those location-based services that do not require the user's identity. In particular, the lack of user identity makes the billing of these services impossible. *Dummy* generates fake user locations (called dummies) and mixes them together with the genuine user location into the request [15]. However, by monitoring long-term movement patterns of the user, the server may distinguish the genuine location from dummies. You *et al*. enhances this technique by generating consistent movement patterns for dummies in a long run [16]. More recently, Yiu *et al*. proposed *SpaceTwist* [5], where the user repeatedly issues kNN queries from dummies, which they called *anchors*, until the kNN result for the genuine location is guaranteed. Ghinita *et al.* proposed a similar framework that is based on Private Information Retrieval (PIR) [17]. The framework partitions the space into grid cells and then the user requests the content of the cell where he/she is located. Thanks to PIR, the user can encrypt which cell is requested while receiving the correct content. By setting proper content for each cell, this framework can support approximate and exact NN queries. Furthermore, the framework is shown to guard against correlation attacks, but with two limitations. First, this framework must ensure that the database is encrypted as described, and all original data is removed at the server; however, there is no trivial way of ensuring this on an untrusted server. Second, extending this framework to support kNN queries is difficult and costly: for each k, the server must create a unique version of content for each cell, which leads to high computation and storage overhead.

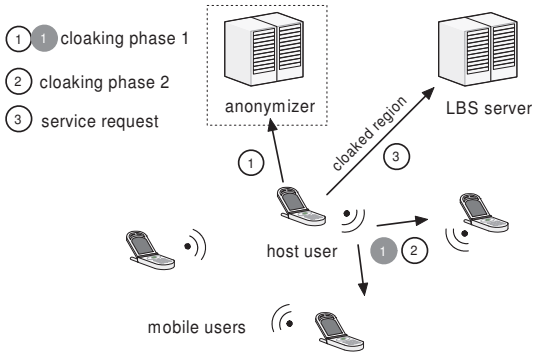


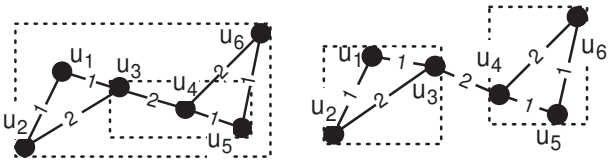
Fig. 3. System Model

III. System Architecture

In this section, we describe the system model and assumptions made in our study. As shown in Fig. 3, there are a large number of mobile users in the system. They carry wireless-enabled (Bluetooth, WiFi, or GPRS) devices such as PDAs and Smartphones. These devices can establish not only Internet connections to external servers, but also point-to-point connections to neighboring peer devices. To request location-based contents or services, these devices are also equipped with embedded positioning modules (such as GPS) to acquire their own positions and attach them in the service requests. However, to protect location privacy, before a user (called the *host user*) requests the service, he/she invokes location cloaking, which obtains for this user a cloaked region that satisfies k-anonymity — at least k users are in this region. Moreover, this region is also the cloaked region for any of them as a host user. Then the host user attaches this region, instead of the accurate location, in the service request, so that any adversary who intercepts this request cannot distinguish its owner from any of the other $k-1$ users.

Cloaking is performed on the *proximity information* among peer users, instead of directly on their spatial coordinates. A mobile device can measure such proximity information through the received signal strength (RSS), or the time difference of arrival (TDOA) of beacon signals. However, the proximity information alone can only identify k users for the k-anonymity, but does not suffice to obtain the cloaked region, which is a bounding box of these k users. In order not to expose their accurate locations, a *secure bounding* protocol must be carried out on these users after they are identified.

As such, the proposed non-exposure location cloaking for a host user is conducted in two phases. In the first phase, k users (including the host user) are identified through the



(a) Weighted Proximity Graph (b) Revised kNN Clustering

Fig. 4. k-Clustering with Proximity

proximity information. By identifying these k users, the first phase also restricts the number of participating users in the next phase. In the second phase, these users participate in the secure bounding to obtain the cloaked region. Note that, this region will serve as the cloaked region for all these k users.

The complete workflow of a service request from a host user is illustrated in Fig. 3. If this user has previously participated in the cloaking for some other user and hence has a cloaked region, the whole cloaking will be skipped — he/she can request the location-based service directly with this cloaked region (③). Otherwise, the first phase of location cloaking (i.e., k-clustering) is executed either at a centralized and dedicated server called anonymizer or distributedly at the host user. In the former case (①), the anonymizer has the complete proximity information submitted by all users. In the latter case (❶), the host user dynamically gathers such information through peer-to-peer communications. The second phase of cloaking (i.e., secure bounding) is executed at the host user and participated by all the k users through peer-to-peer communication (②). Finally, the host user requests the location-based service (③) with the cloaked region obtained in the second phase of cloaking. In the next two sections, we will discuss the algorithms in the two cloaking phases, respectively.

IV. Proximity k-Clustering

We formally model the proximity input for anonymization as follows. We are given a dataset $\mathcal{D}$ of all users, and each user in $\mathcal{D}$ has some peer users in proximity. The proximity input can be modeled as an undirected weighted graph where each vertex denotes a user and each edge (u, v) denotes that users u and v are in proximity; and the weight of (u, v) denotes the relative distance between u and v. This distance could be any measure that relates to distance (e.g., signal strength), as long as it is symmetric and agreed by both u and v. We call the resulted graph a weighted proximity graph (WPG). Fig. 4(b) shows a WPG where the relative distance is the signal strength. For example, the weights of edges (u_2, u_1) and (u_2, u_3) are 1 and 2, which means that the signal between u_2 and u_1 is stronger than that between u_2 and u_3.

By definition, location k-anonymity is to map a host user u to a set of peer users $S(u) \in \mathcal{D}$ such that the size of $S(u)$ exceeds k, i.e., $|S(u)| \geq k$. However, besides the size threshold, $S(u)$ must also satisfy the following: (1) $u \in S(u)$, otherwise the host user may not get the service he/she desires; and (2) $\forall v \in S(u)$, $S(v) = S(u)$, i.e., every user in set $S(u)$ must be mapped to the same $S(u)$, otherwise by knowing $S(v) \neq S(u)$, an adversary can simply deduce that v cannot be the host user and thus safely remove v from $S(u)$. The latter criterion is also called *reciprocity* property in the literature [7]. These two criteria together indicate that $S(u)$ forms a *k-clustering* in $\mathcal{D}$. More specifically, a k-clustering is a partition of $\mathcal{D}$ into a number of groups, each of which has a size of at least k. The following theorem confirms that location k-anonymity on WPG is equivalent to k-clustering.

Theorem 4.1: Location k-anonymity on WPG is equivalent to k-clustering.

Proof. We prove this by showing that $S(u)$ is an equivalence class, or more specifically, the binary relation "a is in the k-cluster of b" (denoted by $a \sim b$) is an equivalent relation. That is, it satisfies the following three properties:

Reflexibility: Since $u \in S(u)$, $u \sim u$.

Symmetry: If $u \sim v$, by definition it implies $u \in S(v)$. Then according to the reciprocity property, $S(u) = S(v)$. On the other hand, by reflexibility, $v \in S(v)$. So we have $v \in S(u)$, i.e., $v \sim u$.

Transitivity: If $u \sim v$ and $v \sim t$, by definition it implies $u \in S(v)$ and $v \in S(t)$. According to the reciprocity property, $S(u) = S(v) = S(t)$. On the other hand, by reflexibility, $u \in S(u)$. So we have $u \in S(t)$, i.e., $u \sim t$. □

The objective of location k-anonymity is to find a cloaked location with the smallest size for the host user in order to receive the best quality of service. In this paper, the cloaked location is a bound of all user locations in the cluster, and it strongly depends on the diameter (the maximum distance between any two vertices) of this cluster: the smaller the diameter is, the smaller the cloaked location will be. As such, we set our objective of k-clustering as to find for the host user a minimum-diameter cluster of size at least k, and hereafter call this problem *minimum k-clustering*.

The minimum k-clustering problem is different from existing graph clustering problems in two aspects. First, the criteria are two-folded — minimizing the diameter and bounding the size of this cluster, whereas most existing algorithms have a single criterion to optimize, such as minimum sum of weights [18]. Second, existing algorithms, especially those top-down fashioned (such as k-median), are centralized. A centralized algorithm requires the global knowledge of the whole WPG, and easily becomes a performance bottleneck. More importantly, such an algorithm finds the cluster for *every* vertex in the WPG; however, in our problem only the host users need k-clustering. As such, a distributed and local k-clustering algorithm that only finds the cluster for a host user is more desirable.

However, intuitive local k-clustering leads to a poor minimum k-clustering result for subsequent host users. Let us consider the local k-clustering algorithm *k nearest neighbor* (kNN). kNN clusters the host vertex and its $k-1$ nearest neighbors in the WPG. For example, in Fig. 4(a) we are asked for a 3-clustering out of 6 users, where the host vertex is u_4. kNN will cluster u_4 with u_3 and u_5, the two nearest users of u_4. However, since there are only three vertices left, they must form a cluster for subsequent 3-clustering. The resulted bound of this cluster, shown in the dotted box, is equal to

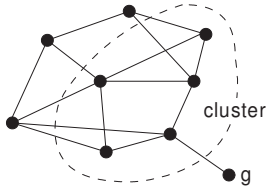


Fig. 5. Disconnected Problem

the bound of all 6 users. The reason for this poor clustering result is that although the first cluster has a small size, it significantly changes the topology of the rest in the WPG and may even make it disconnected (i.e., having an infinite diameter), so that subsequent k-clustering results are poor. In general, directly finding a cluster of size k from the host vertex is not sufficient, because removing this cluster from the WPG might cost some other vertices a larger cluster or even unable to find a cluster of size k. Fig. 5 shows an example. The dotted line encloses a cluster of 5 vertices; however, removing this cluster isolates vertex g from the rest of WPG. To address this issue, while obtaining a minimum-diameter cluster for the host vertex, the distributed minimum k-clustering algorithm must also guarantee the k-clustering result of the rest WPG are not affected. This property is called *cluster-isolation*, and a distributed k-clustering algorithm that satisfies this property is *cluster-isolated*.

Property 4.1: Let $\mathcal{C}(u)$ denote the cluster for vertex u under a distributed k-clustering algorithm. $\mathcal{C}(u)$ is *isolated* if for any vertex $v \neq u$, $\mathcal{C}(v)$ is the same for WPGs $\mathcal{G}$ and $\mathcal{G} - \mathcal{C}(u)$. This distributed algorithm is *cluster-isolated* if for any u, $\mathcal{C}(u)$ is isolated.

Fig. 4(b) shows a revised algorithm of kNN, together with its 3-clustering result of u_4 in the same WPG as in Fig. 4(a). The algorithm is the same as kNN, except that when two or more vertices are of the same distance, it breaks the tie by choosing the vertex with the smallest degree. As such, $\{u_4, u_5, u_6\}$ is clustered because u_5 and u_6 are the 2NN results of u_4. Note that this cluster is isolated because removing it does not affect the clustering result of any remaining users — u_1, u_2, u_3, because they will form a cluster with or without $\{u_4, u_5, u_6\}$. In this particular WPG, the 3-cluster of any vertex is isolated, so this algorithm is cluster-isolated on this WPG. Consequently, the resulted bounds of these clusters are much smaller than those in Fig. 4(a). It is noteworthy that this algorithm is not cluster-isolated in general. For example, if the weight of edge (u_4, u_6) is 3 instead of 2, then $\{u_3, u_4, u_5\}$ is clustered for u_4, and it does affect the clustering results of the remaining users — u_1, u_2, u_6. As such, we should design a general algorithm that proves to be cluster-isolated.

Before we propose a distributed and cluster-isolated minimum k-clustering algorithm, there is one remaining issue — the diameter of a cluster is complex and costly to derive in the clustering process. To remedy this, we use the *maximum edge weight* (MEW) instead of the diameter. In effect, we show in the following corollary that the diameter of a regular graph (all vertices have same degrees) of size k and degree d is bounded by the maximum edge weight w. Since the topologies of wireless networks tend to be clustered and small world graphs [19] which consist of regular graphs plus a few random edges, this corollary justifies the usage of MEW instead of the diameter in this paper.

Corollary 4.2: The diameter of a regular graph with k vertices and degree d is bounded by $w * (1 + \lceil \log_{d-1}((2+\epsilon)dk \log k) \rceil)$, where w is the maximum edge weight and $\epsilon > 0$ is a constant.

PROOF. From [20], we know that the diameter for an un-weighted regular graph is bounded by $1 + \lceil \log_{d-1}((2+\epsilon)dk \log k) \rceil$. As such, the diameter for the weighted graph is at most $w * (1 + \lceil \log_{d-1}((2+\epsilon)dk \log k) \rceil)$. □

A. Connectivity k-Clustering

In this subsection, we formally present the distributed k-clustering algorithm on WPG that minimizes the MEW in the cluster. It is derived from a centralized k-clustering algorithm and modified to be cluster-isolated. First, we introduce an equivalence relation called t-connected, based on which these two k-clustering algorithms are designed.

Definition 4.1: **t-connected relation**: Two vertices a and b are t-connected, if there is a path $a, v_1, v_2, ..., b$ in WPG such that no edge weight in this path exceeds t.

Theorem 4.3: t-connected is an equivalence relation.

PROOF. We prove that this relation satisfies the three properties of an equivalence relation:

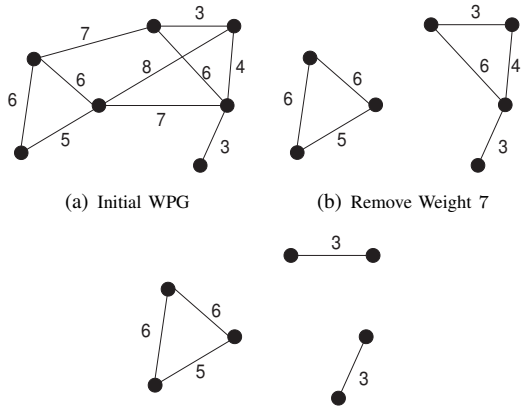
Reflexibility: For vertices a and a, since there is no edge in the path, t-connected is trivially satisfied.

Symmetry: Since WPG is an undirected graph, the weights on the path from a to b and those from b to a are equivalent.

Transitivity: If a and b are t-connected, b and c are t-connected, then we can connect the path from a to b and the path from b to c. As such, we obtain a path from a to c with no edge weight exceeding t. So a and c are t-connected. □

In general, an equivalence relation partitions all elements in a set into equivalence classes. We therefore obtain a clustering of the vertices in WPG through t-connectivity. In terms of graph notions, an equivalence class corresponds to a connected component in WPG whose edge weights do not exceed t. For different t, we obtain different clustering results. More specifically, if t is set to the MEW of the whole WPG, the clustering result has only a single connected component — the whole WPG. Then by decreasing t, this component is partitioned into smaller connected components, which form another clustering result.

To minimize the cluster size, the clustering algorithm should use the lowest t while keeping all connected components *valid*, i.e., their sizes are no smaller than k. Algorithm 1 shows the pseudocode. It partitions a connected component, i.e., a t-connectivity cluster, by removing edges in this cluster in the descending order of their weights, until this cluster is no longer connected and is thus partitioned into some smaller connected components, i.e., clusters. Each of these clusters is partitioned in the same way into even smaller clusters. The recursive partition continues until a further partition will lead to an invalid



(a) Initial WPG (b) Remove Weight 7

(c) Remove Weight 4

Fig. 6. Centralized t-Connectivity 2-Clustering

cluster, i.e., the size is smaller than k. As such, the resulted clusters are those that cannot be further partitioned, and we call them the *smallest valid t-connectivity clusters*. Fig. 6(a) though 6(c) illustrate a 2-clustering process. Starting from the MEW of the whole WPG (which is 8), the algorithm removes the edges of weights 8 and 7, and the WPG is partitioned into two clusters (Fig. 6(b)). For each cluster, starting from its own MEW, the algorithm removes edges of descending weights until it is no longer connected. In Fig. 6(b), the left-hand cluster is already a smallest valid t-connectivity cluster, while the right-hand cluster can be further partitioned by removing the edges of weights 6 and 4. The two resulted clusters, however, become smallest valid t-connectivity clusters, so the whole clustering process terminates. The final clusters are shown in Fig. 6(c). Note that throughout the recursive partition, the connectivity t's of the clusters are ever-decreasing, and the minimum MEW is guaranteed because any further decrease would invalidate some cluster(s).

Algorithm 1 Centralized t-Connectivity k-Clustering

Input: $\mathcal{G}$: the WPG
k: the anonymity requirement
Output: $\cup\mathcal{C}$: the set of resulted clusters
Procedure:
1: $\cup\mathcal{C} = \{\mathcal{G}\}$;
2: **while** there is $\mathcal{C}$ can be partitioned **do**
3: Q = descending sort $\mathcal{C}.E$ by their weights;
4: **while** $\mathcal{C}$ is still connected **do**
5: edge $e = Q.pop()$;
6: remove e from $\mathcal{C}$;
7: add the connected components to $\cup\mathcal{C}$;

Algorithm 1 is centralized, as it requires the knowledge of the whole WPG. In what follows, we extend it to a distributed version of t-connectivity k-clustering. It finds the cluster $\mathcal{C}$ for a particular vertex u. Intuitively, to minimize the cluster size (i.e., the MEW), the algorithm should find the smallest valid t-connectivity cluster of u by increasing t until the cluster size just exceeds k. However, a key observation is that this cluster is not necessarily isolated, that is, removing this cluster might affect the clustering result of some remaining vertex in the WPG. To remedy this, we give a sufficient condition of the smallest valid t-connectivity cluster being isolated.

Theorem 4.4: A sufficient condition of the smallest valid t-connectivity cluster being isolated is that, all **external border vertices** of this cluster (vertices adjacent but not belonging to it) can form a valid t-connectivity cluster in the remaining WPG.

PROOF. Let $\mathcal{C}$ denote this cluster, w denote an arbitrary vertex in the remaining WPG, and $\mathcal{C}(w)$ and $\mathcal{C}'(w)$ denote the clustering result of w in the original and remaining WPGs, respectively; that is, they are the smallest valid t-connectivity clusters that satisfy this sufficient condition. We prove $\mathcal{C}'(w) = \mathcal{C}(w)$ in the following two cases.

1) If w is an external border vertex, by this condition, with t-connectivity w already has a valid cluster in the original WPG, and this cluster does not contain any vertex in $\mathcal{C}$ because otherwise a connectivity higher than t is needed. Moreover, since $\mathcal{C}(w)$ is fully contained in this cluster, removing $\mathcal{C}$ does not remove any vertex or edge in $\mathcal{C}(w)$, which means $\mathcal{C}'(w) = \mathcal{C}(w)$.
2) If w is not an external border vertex, we can still prove that $\mathcal{C}(w)$ does not contain any vertex in $\mathcal{C}$. Otherwise, this cluster must contain at least one vertex v in $\mathcal{C}$ and one external border vertex s. Since the weight of edge (v, s) exceeds t, the connectivity t' of this cluster must exceed t. However, s already has a valid t-connectivity cluster even in the remaining WPG, which contradicts the fact that $\mathcal{C}(w)$ is the smallest valid cluster. Therefore, $\mathcal{C}'(w) = \mathcal{C}(w)$.

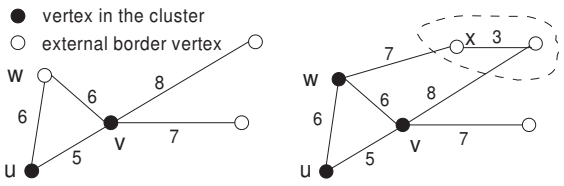
Then by definition, $\mathcal{C}$ is isolated. □

Corollary 4.5: A distributed algorithm that finds for a host vertex u the smallest valid t-connectivity cluster that satisfies the above condition is cluster-isolated.

Algorithm 2 details the distributed algorithm in three steps. In the first step (lines 1–6), it obtains the smallest valid cluster of u by spanning from u through edges with increasing weights until the size reaches k. In Algorithm 2, it always chooses the minimum-weight edge from the priority queue of to-be-spanned vertices, the result cluster $\mathcal{C}$ is guaranteed as the smallest valid cluster. In the second step (lines 7–15), the algorithm checks each external boundary vertex v in $\mathcal{C}$. If v cannot form a cluster of size k with t connectivity in the remaining WPG, v is added to $\mathcal{C}$ and t is thus updated. New vertices will then be spanned from $\mathcal{C}$ using the new t. In this process, some vertices become new external boundary vertices. The algorithm terminates when all external boundary vertices are checked. It is noteworthy that if an external boundary vertex passes the check once, it will either pass any subsequent check (because in the updated $\mathcal{C}$, t can only increase) or even be added to $\mathcal{C}$. In both cases, it does not need to be checked again. Finally in the third step (lines 16–17), since the size of $\mathcal{C}$ might be well above k and since all the edge weights in $\mathcal{C}$ are known, the algorithm calls the centralized k-clustering algorithm (Algorithm 1) to obtain the smallest valid cluster for u. Fig. 7(a) and 7(b) show the distributed t-connectivity 2-clustering process on the same WPG as Fig. 6 where u is the host vertex. In the first step, the distributed

Algorithm 2 Distributed t-Connectivity k-Clustering

Input: $\mathcal{G}$: the WPG
u: the host vertex
k: the anonymity requirement
Output: $\mathcal{C}$: the cluster of u
Procedure:
1: $\mathcal{C} = \{u\}$;
2: push all u's neighbors in $\mathcal{G}$ to H;
3: **while** $|\mathcal{C}| < k$ **do**
4: $v = H.pop()$;
5: $\mathcal{C} = \mathcal{C} \cup \{v\}$;
6: push all v's neighbors not in $\mathcal{C}$ to H;
7: E is the set of external boundary vertices;
8: t is the connectivity of $\mathcal{C}$;
9: **while** E is not empty **do**
10: $v = E.next()$;
11: **if** v does not have a t-connectivity cluster of size k **then**
12: $\mathcal{C} = \mathcal{C} \cup \{v\}$;
13: t=minimum weight of edge between v and $\mathcal{C}$;
14: span $\mathcal{C}$ with new t;
15: insert new external boundary vertices to E;
16: call $centralize\_k\_clustering(\mathcal{C}, k)$;
17: $\mathcal{C}$ is the cluster that contains u;



(a) Step 1 (b) Step 2

Fig. 7. Distributed t-Connectivity 2-Clustering

algorithm uses 5-connectivity to form a 2-cluster $\{u, v\}$ in solid dots (Fig. 7(a)). In the second step, the three external border vertices (the hollow dots) are checked. Among them, only w cannot form a 2-cluster with 5-connectivity. As such, w is added to the cluster and x is added as a new external border vertex (Fig. 7(b)). Since x can have a 2-cluster with 5-connectivity (shown in dashed line), the algorithm proceeds to the third step and returns $\{u, v, w\}$ as the 2-cluster result. Note that in this example, the distributed algorithm achieves the same cluster as the centralized algorithm. However, in general the result by the distributed algorithm depends on the host users, especially when there are many of them.

V. Secure Bounding

Now that the k-cluster is formed, the next and final step is to hide the true identity of the host user among the users in this cluster. This is achieved by generalizing the identifier or quasi-identifier that is embedded in the service request. Typical examples of such identifier include the geographic coordinates (longitude and latitude) and IP address. Without loss of generality, we assume that the identifier is a scalar and private attribute ξ of the user, and therefore the objective in this section is to obtain tight lower and upper bounds for the ξ values of all users in a cluster without revealing them.

This problem is closely related to secure multi-party computation (SMC) [10]. In general, a secure multi-party computation problem is to compute a function from multiple participants in a distributed network, each holding one input. The computation must be secure, that is, no more information is revealed to any participant except for those implied by this participant's input and the function output. The most fundamental problem in SMC is Yao's Millionaire Problem [21], where two millionaires wish to know who is richer, without revealing the exact figures of their fortune. The essence of this problem is to compare two private numbers. Theoretically, the Millionaire problem and secure multi-party computation problem in general can be solved using circuit evaluation protocol [22]. In this protocol, the function is represented by a Boolean circuit, and each party jointly evaluates the circuit without disclosing to the other parties their own inputs that are fed into this circuit. However, the communication cost of such protocol depends on the size of the circuit, which in turn depends on the size of the input domain and on the complexity of function expression [23].

Our problem can be reduced to an SMC problem if we want to obtain the tightest bound — the maximum and minimum ξ values. However, this is not favorable due to the following three reasons. First, the circuit evaluation protocol for the function of maximum or minimum is impractical. In fact, even for the most primitive SMC problem — the Millionaire problem — the communication complexity of the most efficient protocol (Cachin's algorithm) is polynomial to the number of bits of each input (i.e., the precision of ξ) and the number of participants (i.e., k in our problem). Therefore, solving an SMC problem is impractical, in particular in mobile environments where communication is extremely expensive. Second, returning the maximum and minimum ξ values exposes the actual ξ values of some users, which is not fair for them. Third, SMC assumes that participants have no apriori knowledge on the inputs and hence strict security can be guaranteed. However, in our problem, the fact that all participating users are in the same cluster already implies their ξ values are close and even within a certain range. As such, SMC is an overwhelming and inappropriate solution to our problem.

As such, in this section we present our own secure bounding protocol. For simplicity, we present the protocol for upper bounding, and we assume that users follow a *semi-honest* model [24]. In this model, the users follow the protocol properly except that they can record all intermediate results to deduce the ξ values of others.[1] Our protocol is progressive and follows the "hypothesis-verification" paradigm: in each iteration, a hypothetic bound is proposed and verified by all users in the cluster; if not all agree with this bound, a new iteration begins and a larger bound is proposed for those disagreeing users to verify. The protocol terminates when all users agree. Note that the bound can be computed either at a centralized server or at the host user in a distributed environment. Algorithm 3 shows the pseudocode of this protocol. The key factor in this protocol is the increment of the new bound from a disagreed bound. A smaller increment leads to a tighter

[1]This model is opposed to the *malicious* model where users may not follow the protocol at all. It has been shown that any protocol secure in the semi-honest model can be adapted to be secure in the malicious model by imposing the users to follow the protocol.

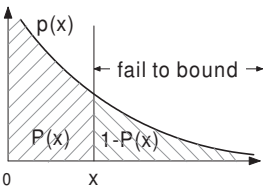


Fig. 8. Unary Bounding

bound, however, it is at the cost of more iterations and thus higher communication overhead for verification. On the other hand, a larger increment leads to a looser bound, and thus costs higher communication for the subsequent service request. In what follows, we derive the optimal bound to minimize the total communication cost.

Algorithm 3 Progressive Bounding

Input: $\mathcal{C}$: the cluster
u: the host user
Output: X: the upper bound
Procedure:
1: **while** X is not the upper bound for all users in $\mathcal{C}$ **do**
2: $\mathcal{C}$ = disagreeing users;
3: compute new X;

A. Unary Bounding

We are about to derive the increment of a new bound X from a disagreed bound X_0 in an iteration where there are still N users who disagree. Obviously, their ξ values must exceed X_0, so variables $\xi - X_0$ of the N users are all positive. Furthermore, they are also *independent and identically-distributed* (iid) random variables. In Fig. 8, the x-axis denotes such a variable, $p(x)$ ($x > 0$) denotes the probability density function, and $P(x)$ denote the cumulative density function.

As for the communication cost, the bound verification incurs only a round-trip fixed-size communication, so the cost per user is denoted by a constant C_b. On the other hand, the communication cost of the subsequent service request depends on the new bound X; without loss of generality, it is denoted by a function $R(X)$. The objective is to find an optimal bound X that minimizes the total communication cost.

First of all, we consider the simple case when $N = 1$, and the resulted bound is called *unary bound*. Since there is only one user, when the bound X is set to x, the total communication cost $C(x)$ is

$$C(x) = C_b + R(x) + (1 - P(x))C^*. \quad (1)$$

Here $1 - P(x)$ is the probability when x fails to upper bound the only user (see Fig. 8), and C^* denotes the subsequent communication cost in such case. Since there is only one user, this cost equals the minimum value of $C(x)$, which we are about to derive.

To obtain the minimum value of $C(x)$, we take the first derivative of x in Equation 1 and make it zero,

$$C'(x) = R'(x) - C^* p(x) = 0.$$

The x solution to this equation leads to the minimum $C(x)$ value, i.e., $C(x) = C^*$. So from Equation 1, we have

$$C^* = C_b + R(x) + (1 - P(x))C^*.$$

Combining these two equations, we obtain a differential equation of x as,

$$P(x)R'(x) = (C_b + R(x))p(x) \quad (2)$$

The optimal bound is the x solution to Equation 2. We show how this equation is solved in the following two examples.

Example 5.1: The x variable of a user (i.e., the $\xi - X_0$ value) follows a uniform distribution in the range of $(0, U)$, i.e., $p(x) = 1/U$, $P(x) = x/U$. The communication cost for the service request is proportional to the area of the bound, i.e., $R(x) = C_r * x^2$, where C_r is a constant. Then Equation 1 is reduced to $C_r * x^2 = C_b$, so $x = \sqrt{\frac{C_b}{C_r}}$. It is noteworthy that in this setting, the bound only depends on the ratio of C_b to C_r, not on U.

Example 5.2: The x variable of a user (i.e., the $\xi - X_0$ value) follows a negative exponential distribution, i.e., $p(x) = e^{-\lambda x}/\lambda$, $P(x) = 1 - e^{-\lambda x}/\lambda$. The communication cost for the service request is proportional to the length of the bound, i.e., $R(x) = C_r * x$. Then the Equation 1 is reduced to $\lambda e^{\lambda x} = (1 + \frac{C_b}{C_r}) + x$. An approximate solution to this transcendental equation can be obtained by Newton's method with an initial guess $x = [ln(1 + \frac{C_b}{C_r})/\lambda]/\lambda$. The bound again depends on λ and the ratio of C_b to C_r.

B. N-Bounding

We now extend unary bounding to N-bounding ($N > 1$). When there are N users, the total communication cost C is a function of both x and N,

$$C(x, N) = N * C_b + R(x) + \sum_{i=1}^{N} \binom{N}{i} (1 - P(x))^i P(x)^{N-i} C^*(i) \quad (3)$$

Here $\binom{N}{i}(1 - P(x))^i P(x)^{N-i}$ is the probability that i users disagree with bound x, and $C^*(i)$ is the subsequent communication cost in this case, which is in turn the optimal communication cost of i-bounding. Note that in this equation, $C(x, N)$ only depends on those $C^*(i)$ whose $i \leq N$. As such, we can recursively derive the optimal x and $C^*(i)$ values for each N through bottom-up dynamic programming, starting from $N = 2$. At each N, we apply the same approach as in unary bounding to derive the optimal x, that is, we take the first derivative in Equation 3 and make it zero.

Though this dynamic programming approach is theoretically sound, it requires N rounds of differential equation solving, which is CPU intensive. For a CPU-restrained mobile device, we propose a simplified and approximate equation as follows. We use the expected number of disagreeing users $\overline{i}$ instead of enumerating all possibilities for this number i in Equation 3. By the iid assumption, $\overline{i} = \lfloor N(1 - P(x)) \rfloor$, and the total probability that bound x fails is $1 - P^N(x)$. As such, Equation 3 can be approximated as

$$C(x, N) \approx N * C_b + R(x) + (1 - P^N(x))C^*(\lfloor N(1 - P(x)) \rfloor).$$

In this equation, $C^*(\lfloor N(1 - P(x)) \rfloor)$ can be further bounded and approximated by considering each user individually. That

is, $C^*(\lfloor N(1-P(x))\rfloor) \leq (C^*-R^*)N(1-P(x))+R^*$, where $C^* = C^*(1)$ and R^* is the $R(x)$ value at the optimal x for C^*. R^* is deducted from C^* because R^* is independent of the number of users and should be counted only once. As such, we have

$$C(x,N) \approx N * C_b + R(x) + N(1-P(x))(1-P^N(x))(C^*-R^*) + R^*. \quad (4)$$

Taking the first derivative of x, omitting the high exponent (N) terms, and making it zero, we have

$$R'(x) = (C^*-R^*)Np(x) \quad (5)$$

The optimal N-bounding is the x solution to Equation 5. We show how this equation is solved in the previous two examples.

Example 5.3: Same as Example 5.1. The x variable of a user (i.e., the $\xi - X_0$ value) follows a uniform distribution in the range of $(0, U)$, i.e., $p(x) = 1/U$, $P(x) = x/U$. The communication cost for the service request is proportional to the area of the bound, i.e., $R(x) = C_r * x^2$. Then Equation 5 is reduced to $2C_r x = (C^* - R^*)N/U$, so $x = \frac{N(C^*-R^*)}{2C_r U}$.

Example 5.4: Same as Example 5.2. The x variable of a user (i.e., the $\xi - X_0$ value) follows a negative exponential distribution, i.e., $p(x) = e^{-\lambda x}/\lambda$, $P(x) = 1 - e^{-\lambda x}/\lambda$. The communication cost for the service request is proportional to the length of the bound, i.e., $R(x) = C_r * x$. Then Equation 5 is reduced to $C_r = (C^* - R^*)Ne^{-\lambda x}/\lambda$, so $x = [\ln (C^* - R^*)N - \ln \lambda C_r]/\lambda$.

Algorithm 4 shows the complete pseudocode of secure progressive bounding for a cluster of users. The initial bound is set to the minimum value of ξ domain, and the bound is progressively incremented by the optimal N-bounding value on disagreeing users. The whole algorithm terminates when no more user disagrees. Note that this algorithm treats all users in $\mathcal{C}$ equally, including the host user u, and none of them reveal their ξ values.

Algorithm 4 Secure Bounding

Input: $\mathcal{C}$: the k-cluster
u: the host user
Output: X: the maximum bound
Procedure:
1: X = minimum value of ξ domain;
2: **while** $\mathcal{C}$ is not empty **do**
3: X = X + N-bounding($|\mathcal{C}|$);
4: remove from $\mathcal{C}$ whose $\xi \leq X$;

VI. Performance Evaluation

In this section, we evaluate the performance of t-connectivity k-clustering and secure bounding algorithms through extensive experiments. The user location dataset we use is a Points of Interest (POI) dataset of California from U.S. Geological Survey [25]. This dataset contains 104,770 POIs of various categories. We normalize all their coordinates so that these POIs fit in a unit square. Throughout our experiments, each POI represents a user who is standing right at its coordinates. The user is equipped with a wireless-enabled device and is capable of communicating with other users who are within the distance threshold δ. In practice, mobile devices have limited resources, so in the experiments each user can connect to at most M peers. As shown later, M essentially controls the density of the WPG, or more specifically, the average degree of the vertices in the WPG. As for the edge weight, we assume that a user can measure the radio signal strength (RSS) of the connected peers and sort them according to RSS. Then the weight of an edge (a, b) is set to the ranking of vertex a in the sorted peer list of vertex b. In the experiments, we adopt a simple RSS model that is reversely correlated to the distance. As such, the weight designates how relatively faraway a is from b. In addition, to ensure a and b are reversible, the weight is set to the minimum of the two: the ranking of a in b's list and the ranking of b in a's list.

As for the k-clustering algorithms, we compare the t-connectivity algorithm ($t - Conn$) with the kNN algorithm. Recall that kNN algorithm is distributed, and it clusters the host vertex with its $k-1$ nearest neighbors that have not yet been clustered in the WPG. For $t - Conn$, we implement both the centralized and distributed protocols. The difference is that, the centralized protocol has the knowledge of the entire WPG, and therefore it clusters all vertices in the WPG when the first user requests for location cloaking, whereas the distributed protocol only clusters those vertices in the smallest valid $t-$connectivity cluster of the host vertex.

As for the bounding algorithms, we compare the secure bounding algorithm with three alternative algorithms. The optimal algorithm (OPT) obtains the tightest possible bound by finding the minimum and maximum coordinates of all users in the same cluster. Obviously, OPT is not practical as it requires each user to expose their coordinates. As such, OPT is merely used as a benchmark. The rest two algorithms, namely, linear bounding and exponential bounding, are also based on progressive bounding, i.e., a hypothetic bound is proposed and progressively increased by X until all users agree. The difference between linear, exponential and secure bounding is how the increment X is computed. The linear algorithm increases the bound by a fixed amount, i.e., X is a constant; the exponential algorithm doubles the bound in each iteration, so X equals to the length of current bound. For these two algorithms, we are yet to decide the initial bound. To ensure a fair comparison, all three algorithms assume a uniform distribution of the coordinates. As such, the initial bound is obviously $N/104770$ where N is the size of the cluster. As for secure bounding, we also need to know the communication cost of subsequent service request. In the experiments, we assume that the service request is a range query on the same POI dataset. As such, the communication cost is (approximately) proportional to area of the bound. With ξ (in this case, the coordinates) following a uniform distribution and the communication cost of service request proportional to the area of the bound, we can adopt the formulae developed in Example 5.3 to compute X, where $U = N/104770$, $C_b = 1$ and $C_r = 1,000$ (i.e., the content of a POI is $1,000$ times larger than a bounding message).

We implement a testbed in Java (JDK1.6) on a desktop PC

Parameter	Symbol	Default Value
# of users		104,770
distance threshold	δ	2×10^{-3}
max # of connected peers	M	10
k-anonymity	k	10
bounding cost	C_b	1
service request cost	C_r	1,000
uniform distribution bound	U	$N/104770$
initial bound	X	$N/104770$
# of user requests	S	$2,000$

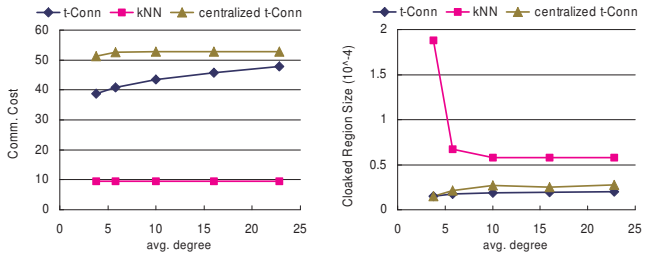
TABLE I
SIMULATION PARAMETER SETTINGS

with Pentium D 3.0GHz CPU and 2GB RAM. The workload consists of S (out of 104,770) users who request location cloaking for range queries. For the k-clustering algorithms, we measure the following two metrics: communication cost (in terms of number of messages) and area of cloaked location, both of which are averaged over the total number of cloaking requests. As we observe that if a user is involved in the k-clustering process, only a single message containing the adjacent vertices as well as the edge weights is sent to the host vertex, the communication cost essentially equals the number of involved users. Furthermore, to eliminate the impact of bounding algorithms on the resulted cloaked location, we use the optimal bounding when k-clustering algorithms are evaluated. For the bounding algorithms, we measure the following three metrics: the communication cost for bounding, the communication cost for service requests, and the CPU time, also averaged over the total number of cloaking requests. Table I summarizes all parameter settings in the experiments.

A. Effect of Average Degree in WPG

In this subsection, we vary M, the maximum number of connected peers for each user to alter the average vertex degree in the WPG. We choose M to increasing exponentially: $4, 8, 16, 32, 64$ and the resulted average vertex degrees are $3.8, 5.8, 10.0, 16.0, 22.8$ respectively. Fig. 9(a) and 9(b) show the measured communication cost and the size of cloaked location (in terms of its area). kNN achieves the lowest communication cost and is almost independent of the degree, which means that the topology has little effect on the number of involved users during kNN clusetering. On the other hand, the centralized t-connectivity algorithm involves all users when the first k-clustering request comes. As such, it serves as an upper bound of the communication cost. The distributed t-connectivity algorithm involves more users than kNN algorithm because it clusters all users in the smallest valid t-connectivity cluster. As the vertex degree increases and the WPG becomes denser, the size of this cluster also increases, but at a moderate rate.

As for the size of cloaked location, the two t-connectivity algorithms significantly outperform kNN — the size is reduced by about $2/3$ and is independent of the average vertex degree. This indicates that t-connectivity effectively clusters vertices in the neighborhood. On the other hand, kNN is much worse because with more location cloaking requests, more users will have been clustered. As such, kNN is no longer able to cluster the host user with the neighboring users because



(a) Avg. Communication Cost (b) Avg. Cloaked Size

Fig. 9. Performance under Various Avg. Degrees

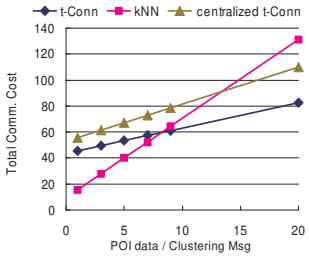
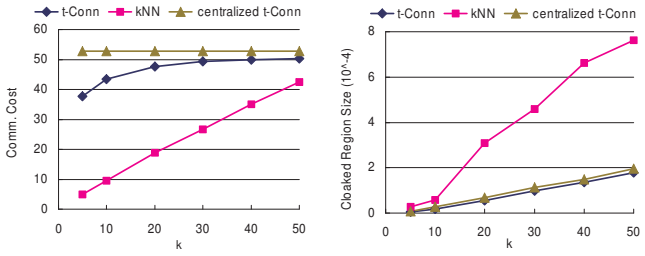


Fig. 10. Overall Communication Cost under Various POI Data Sizes

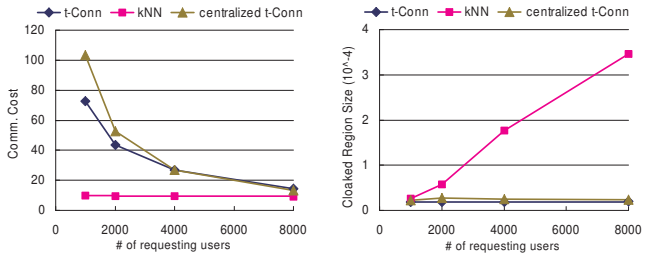
they are already clustered. Instead, the algorithm has to further span the WPG to find $k-1$ un-clustered users, which might be far away. We will discuss this effect in more detail in Section VI-C. Another observation is that centralized $t-Conn$ is even slightly worse than distributed $t-Conn$, especially when the average vertex degree is large. The reasons are two-folded. First, $t-Conn$ aims at minimizing the largest MEW (maximum edge weight) among all clusters, whereas the experimental result only shows the size of the cluster that contains the host user. Second, the centralized $t-Conn$ focuses on the clusters of the entire WPG, whereas the distributed $t-Conn$ focuses on the clusters within the smallest valid t-connectivity cluster. As such, the latter is sensitive to the host users, and will thus cluster in favor of them. More specifically, in this experiment there are only 2% host users (2,000 out of 104,770), most of the clusters by the centralized $t-Conn$ are never requested, that is, the users in these clusters never contribute to the k-clustering, even if they are close to some host users. On the other hand, in the distributed $t-Conn$ all neighboring users can participate in the k-clustering of a host user. This effect is more eminent when the WPG is denser, as there are more neighboring users for a particular host user. Therefore, we can conclude that the distributed $t-Conn$ is robust and efficient in various degree settings.

To understand how the cloaked region size affects the total communication cost, we combine the communication cost of k-clustering and service requests (i.e., a range query on the same POI dataset) and plot the total cost in Fig. 10 for the default M. We vary the data size of each POI object and observe that $t-Conn$ outperforms kNN when the size of a POI is 10 or more than 10 times larger than the size of a clustering message. In practice, a POI object usually contains texts, images and even videos, whereas a clustering message only contains the adjacency list of a user or the ids of users in the cluster so far. As such, the above condition almost always holds, which means that $t-Conn$ almost always achieves a



(a) Avg. Communication Cost (b) Avg. Cloaked Size

Fig. 11. Performance under Various k



(a) Avg. Communication Cost (b) Avg. Cloaked Size

Fig. 12. Performance under Various # of Requesting Users

lower total communication cost.

B. Effect of k

In this subsection, we vary k, the anonymity requirement of the host user. The k values we choose in the experiment are 5, 10, 20, 30, 40, and 50. Fig. 11(a) and 11(b) show the measured communication cost and the size of cloaked location. As expected, the communication cost of centralized $t-Conn$ is independent of k and serves as the upper bound of all algorithms. On the other hand, the cost of distributed $t-Conn$ increases with k, but the increment is relatively slow and saturates after $k = 30$. This indicates that the smallest valid t-connectivity cluster only grows moderately as k increases and it becomes stable for large k values. kNN algorithm, on the other hand, incurs linear cost with regard to k, because it has to find $k-1$ un-clustered users. Therefore, kNN loses its advantage of low communication cost for moderate and large k values.

As for the size of cloaked location, $t - Conn$ greatly outperforms kNN for all k settings. In addition, the size for $t - Conn$ is linear with respect to k, which indicates that the performance of $t-Conn$ is consistent and irrespective of k. On the other hand, the size for kNN algorithm increases sharply with k: at $k = 5$, the size for kNN is only twice as large as the size for $t - Conn$, whereas at $k = 50$, it becomes four times as large. This shows that kNN algorithm deteriorates quickly as k increases. This can be explained by the fact that when k is larger, it becomes more difficult for a host user to find all $k-1$ un-clustered users in the neighborhood. We therefore conclude that $t - Conn$ algorithms are effective and robust regardless of the anonymity requirement.

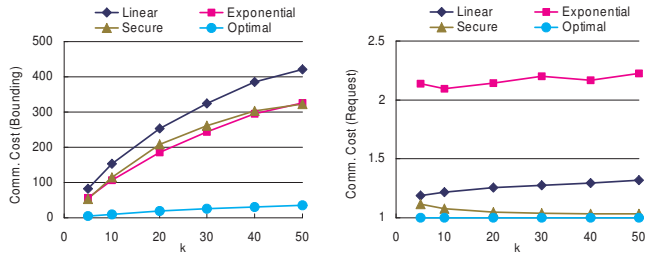
C. Effect of Requesting Users

In this subsection, we vary S, the number of cloaking requests. The S values we choose in the experiment are 1,000, 2,000, 4,000, and 8,000. Fig. 12(a) and 12(b) show the measured communication cost and the size of cloaked location. The communication cost of $t - Conn$ significantly drops as S increases. This is because all users in the smallest valid t-connectivity cluster are clustered and subsequent k-clustering requests on them are directly answered without any communication cost. On the other hand, each time kNN algorithm only forms a cluster of exactly k users, so increasing the number of cloaking requests cannot amortize the communication cost. Among the two $t - Conn$ algorithms, the centralized algorithm drops more quickly as S increases than the distributed algorithm. This is because in the centralized algorithm, once the first cloaking request is completed, all subsequent requests are at no costs. We observe that when S is sufficiently large ($S \geq 4000$), both algorithms achieve almost the same cost. To summarize, the distributed $t-Conn$ is less costly when there are fewer requests, and with more requests the two algorithms cost asymptotically the same.

As for the size of cloaked location, we observe an almost linear increase with S for kNN algorithm. This can be explained by the property of cluster-isolation. Since kNN does not have this property, as more and more users are clustered in the WPG, it becomes increasingly difficult to find $k-1$ un-clustered users in the remaining WPG; and even they can be found, they are far away from the host user. In other words, the performance of kNN deteriorates when many users are requesting location cloaking. On the other hand, the distributed $t - Conn$ has the cluster-isolation property, so the size of cloaked location is not affected by the number of requesting users. We therefore conclude that the distributed $t - Conn$ is effective and robust regardless of the number of requests.

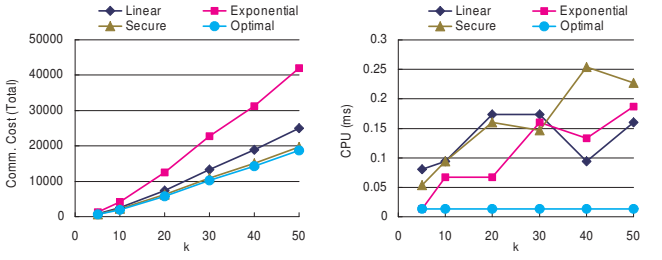
D. Bounding Algorithms

In this subsection, we evaluate the performance of secure bounding algorithm under various k settings against other competitors, namely, the optimal bounding, linear bounding and exponential bounding. As aforementioned, all algorithms except the optimal bounding are progressive algorithms, and they differ in how the new bound X is computed. Fig. 13(a), 13(b) and 13(c) show the communication cost of bounding, service request, and the total, respectively. We observe that the linear algorithm has the highest bounding cost because it is the most conservative — it needs the most iterations to find the bound. The benefit is low subsequent request cost. The exponential algorithm has an opposite performance: it has the lowest bounding cost because it is the most aggressive — it needs the fewest iterations to find the bound. The disadvantage is high request cost, which means that the resulted bound is quite coarse. On the other hand, secure bounding strikes a good balance between the two costs. In particular, as $C_r/C_b = 1,000$ in this experiment, secure bounding tends to sacrifice certain amount of bounding cost to reduce the request cost. As such, in Fig. 13(c) it always achieves the minimum total cost among the three progressive algorithms and very close to the optimal bounding.



(a) Avg. Bounding Cost

(b) Avg. Request Cost (in ratio of the optimal bounding)



(c) Avg. Total Cost

(d) Avg. CPU Time

Fig. 13. Performance under Various # of Requesting Users

In terms of CPU cost (Fig. 13(d)), since closed form formulae have been derived for secure bounding, it costs as little CPU overhead as the other two progressive algorithms. In particular, even for the largest k ($k = 50$), the CPU time is still far less than 1ms, which shows the secure bounding is an efficient and robust bounding algorithm.

VII. Conclusion

In this paper, we investigate the problem of location cloaking without users exposing their accurate locations. We decompose the problem into two subproblems: proximity minimum k-clustering and secure bounding. For the former problem, we propose a t-connectivity k-clustering algorithm that proves to be cluster-isolated. For the latter problem, by developing a cost model, we propose a progressive bounding algorithm that is optimal in terms of the total communication cost. Experimental results consistently show that both algorithms are efficient and robust to various network topology settings, number of requesting users and the anonymity metric k.

As for future work, we plan to design new algorithms for secure bounding with alternative objectives. The progressive bounding algorithm optimizes the communication cost, but a user who disagrees an old bound X and agrees a new bound X' essentially exposes the private ξ value by a bound $[X, X']$. The smaller the increment is from X to X', the smaller is this bound, and thus the more information about ξ is exposed. As such, we need to develop a privacy loss metric and design new secure bounding algorithms based on this metric. We also plan to elaborate the protocols in our distributed k-clustering and secure bounding algorithms to ensure their robustness in undesired scenarios. For example, concurrency control must be considered when two or more users request cloaking almost at the same time. Since a single user can only join one cluster but can participate the clustering process of multiple host users, our protocols must prevent deadlocks while making the best clustering decision. As another example, communication failures during the clustering or bounding process should also be concerned, and a balance must be struck between robustness and efficiency.

References

[1] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proc. of MobiSys*, 2003, pp. 31–42.

[2] B. Gedik and L. Liu, "Location privacy in mobile systems: A personalized anonymization model," in *Proc. of ICDCS*, 2005, pp. 620–629.

[3] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The new casper: Query processing for location services without compromising privacy," in *Proc. of VLDB*, 2006, pp. 763–774.

[4] J. Xu, J. Du, X. Tang, and H. Hu, "Privacy preserving location based queries in mobile environments," Hong Kong Baptist University, Tech. Rep., 2007.

[5] 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. of ICDE*, 2008.

[6] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing location-based identity inference in anonymous spatial queries," *TKDE*, vol. 19, no. 12, pp. 1719–1733, 2007.

[7] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "Prive: Anonymous location-based queries in distributed mobile systems," in *Proc. of WWW '07*, 2007, pp. 371–380.

[8] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A peer-to-peer spatial cloaking algorithm for anonymous location-based services," in *Proc. of ACM GIS*, 2006, pp. 171–178.

[9] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "Mobihide: A mobile peer-to-peer system for anonymous location-based queries," in *Proceedings of the Int. Symposium in Spatial and Temporal Databases (SSTD)*, 2007.

[10] S. Goldwasser, "Multi party computations: past and present," in *Annual ACM Symposium on Principles of Distributed Computing*, 1997, pp. 1–6.

[11] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," *IEEE Transactions on Mobile Computing*, vol. 7, no. 1, pp. 1–18, 2008.

[12] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar, "Preserving user location privacy in mobile data management infrastructures," in *Proceedings of 6th Workshop on Privacy Enhancing Technologies*, 2006.

[13] G. Myles, A. Friday, and N. Davies, "Preserving privacy in environments with location-based applications," *Pervasive Computing*, vol. 2, no. 1, pp. 56–64, 2003.

[14] A. Beresford and F. Stajano, "Location privacy in pervasive computing," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 46–55, 2003.

[15] H. Kido, Y. Yanagisawa, and T. Satoh, "An anonymous communication technique using dummies for location-based services," in *Proc. of the Second International Conference on Pervasive Services (ICPS)*, 2005.

[16] T.-H. You, W.-C. Peng, and W.-C. Lee, "Protect moving trajectories with dummies," in *The International Workshop on Privacy-Aware Location-based Mobile Services*, 2007.

[17] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan, "Private queries in location based services: Anonymizers are not necessary," in *Proc. of SIGMOD*, 2008.

[18] M. Bern and D. Eppstein, *Approximation Algorithms for NP-Hard Problems*. Boston: PWS Publishing Company, 1996.

[19] A. Helmy, "Small worlds in wireless networks," *IEEE Communications Letters*, vol. 7, no. 10, pp. 490–492, 2003.

[20] B. Bollobs and W. F. de la Vega, "The diameter of random regular graphs," *Combinatorica*, vol. 2, no. 2, pp. 125–134, 1982.

[21] A. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, 1982.

[22] ——, "How to generate and exchange secrets," in *Proceedings 27th IEEE Symposium on Foundations of Computer Science*, 1986, pp. 162–167.

[23] O. Goldreich, *The Foundations of Cryptography – Volume 2*, thirteenth ed. Cambridge University Press, 2004.

[24] W. Du, "A study of several specific secure two-party computation problems," Ph.D. dissertation, 2001.

[25] F. Li and G. Kollios, "Real datasets for spatial databases: Road networks and points of interest," http://www.cs.fsu.edu/ lifeifei/SpatialDataset.htm.