2013 IEEE 14th International Conference on Mobile Data Management

A Practical Location Privacy Attack in Proximity Services

Sergio Mascetti Letizia Bertolaja Claudio Bettini
EveryWare Lab, CS Dept., Università degli Studi di Milano
Email: {sergio.mascetti,letizia.bertolaja,claudio.bettini}@unimi.it

***Abstract*—The aim of *proximity services* is to raise alerts based on the distance between moving objects. While distance can be easily computed from the objects' geographical locations, privacy concerns in revealing these locations exist, especially when proximity among users is being computed. *Distance preserving transformations* have been proposed to solve this problem by enabling the service provider to acquire pairwise distances while not acquiring the actual objects positions. It is known that distance preserving transformations do not provide formal privacy guarantees in presence of certain background knowledge but it is still unclear which are the practical conditions that make distance preserving transformations "vulnerable".**

We study these conditions by designing and testing an attack based on public density information and on partial knowledge of distances between users. A clustering-based technique first discovers the approximate position of users located in the largest cities. Then a technique based on trilateration reduces this approximation and discovers the approximate position of the other users. Our experimental results show that partial distance information, like the one exchanged in a friend-finder service, can be sufficient to locate up to 60% of the users in an area smaller than a city.

I. Introduction

Proximity services aim at notifying a user when another user or, more generally, an object of interest to the user happens to be spatially closer than a certain threshold distance. Examples of these services are so called *friend-finder* services. They can be efficiently implemented by a server receiving position information from each moving object and sending an alert when the distance threshold condition is verified.

When users are the considered moving objects and the server is untrusted, the acquisition of exact location information by the server is a privacy threat. The problem of privacy preservation in location based services has been extensively studied in the literature [1], [2], but mostly focusing on services identifying the location of points of interest. Since a proximity service can be provisioned simply based on the distance between the two moving objects, an intuitive way to preserve location privacy is to use fake locations for the objects, with coordinates appropriately chosen in order to retain the correct value or a good approximation for the relative distance. We consider the server provisioning the service as a semi-honest entity, in the sense that it follows the given protocol but it may attempt to infer the locations of the users. A natural question arises: is distance information sufficient for an adversary to find the actual location of a user? Technically, this is equivalent to ask if distance preserving transformations are unsafe. If users are uniformly distributed in the geographical space and no other background knowledge is available, the transformations are actually safe, since the same set of distances can be observed for many different assignments of specific positions to the users. This uniform distribution assumption is implicitly or explicitly made in several papers, as described in the following.

In [3] three secure computation protocols are proposed to notify a user about the proximity of another user. Communications among the user devices occurs through a server; One of the protocols, named *Louis* reveals to the server the proximity information, assuming that this information may not lead to a location privacy breach. This assumption should be probably reconsidered in light of our results. In [4], [5], [6] location privacy is achieved through spatial cloaking and encryption techniques, though these solutions reveal to the server approximate distance information. The *Longitude* protocol [7] allows users a finer control of location privacy with respect to other friends by specifying privacy preferences in terms of spatial granularities. *Longitude* also guarantees complete privacy with respect to the server under the assumption that it has no a priori knowledge on the distribution of users, i.e., when a uniform distribution is assumed. This assumption is indeed related to the fact that the server acquires approximate distance information. We find the assumption that distance information does not reveal the user actual position also in some work on general LBSs [8], [9]. In particular, in [9], spatial cloaking is used to guarantee k-anonymity that is computed using proximity information rather than the precise user locations. The paper proposes both a centralized algorithm and a distributed one. The centralized algorithm assumes a server acquiring proximity information for all users. Revealing proximity instead of exact positions is based on the intuition that information on distance is not sufficient to identify the position. Our findings would suggest that the centralized algorithm may be subject to privacy breaches in case of a semi-honest server.

In the context of privacy preserving data mining, it has been shown that distance preserving transformations can be subject to privacy attacks in presence of prior knowledge of the adversary in terms of input-output pairs and samples from original or similar dataset [10]. In our attack the only background knowledge is the publicly available density distribution. The unsafety of distance preserving transformations in presence of certain kinds of background knowledge is also discussed in

978-0-7695-4973-6/13 $26.00 © 2013 IEEE
DOI 10.1109/MDM.2013.19

the context of secure kNN queries [11]. The authors consider as safe the case in which the adversary observes only the encrypted DB and the distances (called "level 1" attack). Our study shows that even this attack model can lead to privacy breaches when the distribution of the original data is known, as it is the case for population distribution.

Finally, in the context of private spatial join computation, a distance based attack in presence of background knowledge about spatial distribution shows that the position of isolated points (outliers) may be discovered [12]. A defense is proposed based on a spatial transformation that eliminates distances longer than a certain threshold in the transformed space. In principle, friend-finder services may be effective even if limited to short distances (2-3 km); however, our study shows (see Section IV-D) that even ignoring any distance larger than that threshold the attack we propose can still pose privacy threats. In addition, our study does not want to rule out services in which long distances are considered, and identifies under which conditions the adversary actually derives users location.

This paper formalizes a location privacy attack that, based on partial information on the distances between users and public knowledge on the average density of population, aims at approximately localizing the users, independently on their fake position assigned by a privacy preserving algorithm. We achieve this goal by applying a distance-based clustering algorithm, and then matching obtained clusters with geographical regions with similar densities. Population density information drives the mapping process. We have reported preliminary results based on this idea in [13]. In this paper, in addition to formalizing the attack, we significantly enhance its effectiveness by introducing a trilateration technique, which refines the position of other users once the position of a few is determined. We show through an extensive experimental evaluation, considering geographical regions with substantially different density distributions, that the attack is effective in practice and that this technique, not only allows to position many more users, but it also position them with much higher precision.

To summarize, our contributions are as follows:

1) Our study systematically explores the problem of identifying the position of moving objects based only on partial information on their distance and on background knowledge on the density distribution.
2) We provide a formalization of the problem in terms of a privacy attack in presence of background knowledge
3) Our experiments on real geographic data and publicly available population density data show that the attack is effective even with partial distance information as typically acquired by running proximity services.

The rest of the paper is organized as follows. In Section II we formally describe the location re-identification attack. In Section III we describe the specific strategy used to make the attack practically feasible and we report the basic algorithms and several optimizations. In Section IV we report our experimental results, and we conclude the paper in Section V.

II. Problem Modeling

We consider a service provisioned to a set of users by exchanging data through a service provider that only acquires information about the distance between some of the users.

A. Basic definitions

Let U be the set of users. Given a user $u \in U$, his position is denoted by $pos(u)$ that is a point in a discrete bi-dimensional space S. We denote with $d_p(p_1, p_2)$ the distance[1] between two points $p_1, p_2 \in S$.

Given the average density of population in each area at the precision available from public sources, we consider as *background knowledge* BK the information about how many users in U are in each area. More formally, the adversary can compute the function $nu : G \to \mathbb{R}$ returning the number of expected users contained in g for each cell g of a grid[2] G partitioning the spatial domain S. Given $nu()$, the probability that a generic user is located in a cell g is given by the number of users in g divided by the total number of users $|U|$.

Definition 1. *The probability $P[u, g]$ that a user u is located in a cell g of G, based on BK only, is given by: $\frac{nu(g)}{|U|}$*

Note that, since the adversary has no other background knowledge, $P[u, g]$ for a given g is the same for all users.

During the provisioning of the service, the adversary collects data about the distances between users. We call this information the *observation knowledge* OK, and we model it by the function $d_u : U \times U \to \mathbb{R}$ returning for a pair of users u_i and u_j the distance between their positions, i.e., $d_u(u_i, u_j) = d_p(pos(u_i), pos(u_j))$. Note that d_u is a partial function and it is undefined for those pairs of users whose distance in unknown to the adversary.

Example 1. *Consider a friend-finder service provided in a space S composed by a regular grid of 8 possible user positions (see Figure 1(a)). Four cells are defined in S and the probability that a generic user u is located in each cell is: $P[u, g_0] = P[u, g_1] = P[u, g_2] = 1/6$ and $P[u, g_3] = 1/2$. We consider 3 users: Alice (A), Bob (B) and Carl (C). Since Alice is friend of Bob and Carl, while Bob and Carl are not friends, two distances are known to the adversary: $d_u(A, B) = \sqrt{10}$ and $d_u(A, C) = \sqrt{5}$.*

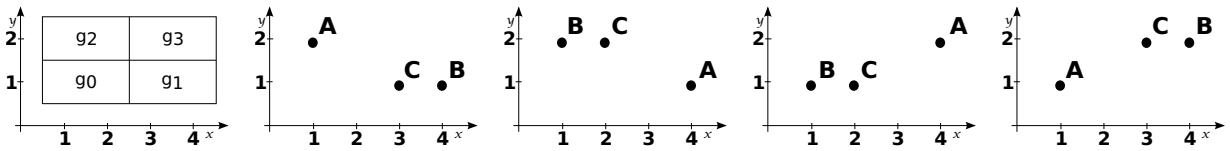
We now show how the adversary can exploit BK and OK to infer the position of the users.

B. Modeling an attack to discover user positions

We first define the concept of *user-to-point mapping* m to capture the intuition of the assignment of a specific geographical position to each of the users. The goal of the adversary is to identify the mappings representing the real position of as much users as possible. Formally $m : U \to S$ is a total function that associates each user with a geographical position.

[1]The distance $d_p()$ can be either the euclidean distance or the geographical distance computed on the geoid.

[2]Here we use a grid, but the more general concept of spatial granularity can be used as well.



(a) Space divided into 4 cells (b) Consistent mapping m_0 (c) Consistent mapping m_1 (d) Consistent mapping m_2 (e) Consistent mapping m_3

Fig. 1. Examples

By using OK, it is possible to restrict the set of *consistent* user-to-point mappings to those that do not violate any distance constraint defined in OK. We denote the set of all consistent user-to-point mappings with M.

Definition 2. *A user-to-point mapping m is a* consistent user-to-point mapping *if, for each $u_i, u_j \in U$ such that $d_u(u_i, u_j)$ is defined, $d_u(u_i, u_j) = d_p(m(u_i), m(u_j))$.*

Example 2. *Let's continue with Example 1 and let's consider a user-to-point mapping m such that: $m(A) = \langle 1, 1\rangle$, $m(B) = \langle 2, 1\rangle$, $m(C) = \langle 3, 1\rangle$. Clearly m is not a consistent user-to-point mapping since*

$$\sqrt{10} = d_u(A, B) \neq d_p(m(A), m(B)) = 1$$

Let's consider the different mapping m_0 shown in Figure 1(b). This mapping is consistent, since $d_u(A, B) = d_p(m(A), m(B))$ and $d_u(A, C) = d_p(m(A), m(C))$. Overall, there are 4 consistent mappings that are shown in Figure 1.

Given a consistent user-to-point mapping, it is possible to compute the probability of that mapping being correct by using the knowledge BK. Intuitively, the probability is computed as the conjunction of the probabilities of each user u being in the cell where u is mapped by m. To formalize this intuition we first need to define the $up(p)$ function that returns, for each point $p \in S$, the cell g of G that contains p. Then, the absolute probability $P[m]$ that a consistent user-to-point mapping m is correct is:

$$P[pos(u_1) \in up(m(u_1)) \wedge \ldots \wedge pos(u_{|U|}) \in up(m(u_{|U|}))] \quad (1)$$

Since, by Definition 1, the probability that a user is located in a given cell is independent from the other users' positions, we can rewrite Equation 1 as shown in Property 1.

Property 1. *The absolute probability $P[m]$ that a consistent user-to-point mapping m is correct is: $\prod_{u \in U} P[u, up(m(u))]$.*

We can now define the probability that a user-to-point mapping m is correct relatively to the set M of all consistent user-to-point mappings. We call this distribution of probability the *relative probability* that a mapping is correct.

Property 2. *The relative probability $P_r[m]$ that a consistent user mapping m is correct is: $\frac{P[m]}{\sum_{m' \in M} P[m']}$*

Example 3. *We continue from Examples 1 and 2. The absolute probability of m_0 is $P[m_0] = P[A, g_2] \cdot P[B, g_1] \cdot P[C, g_1] = 1/6^3$. With analogous computations, the absolute probabilities of the other three consistent mappings are: $P[m_1] = 1/6^3$, $P[m_2] = 3/6^3$ and $P[m_3] = 9/6^3$. Consequently m_3 is the most likely mapping and $P_r[m_3] \simeq 0.65$. As a result the adversary can conclude, with high likelihood, that A is located in $\langle 1, 1\rangle$, B is located in $\langle 4, 2\rangle$ and C is located in $\langle 3, 2\rangle$.*

From Property 2, we can also derive the probability that a user is located in a given area A, as shown in Property 3.

Property 3. *The probability $P[pos(u) \in A]$ that a given user u is located in an area A is given by:*

$$\sum_{m \in M | m(u) \in A} P_r[m]$$

Given the above formalization, we can define an attack as the process of identifying consistent user-to-point mappings and of evaluating their relative probabilities using BK. An attack may be aimed at identifying the exact or approximated (in terms of an area) position of one or more specific users, or aimed at discovering the position of all users.

C. Modeling the attack exploiting users' clustering

The attack process presented in Section II-B has a clear computational limit due to the combinatorial explosion of the number of the mapping functions that are exponential in the number of users. For this reason, the problem of computing the consistent mappings is intractable, and not likely to have a practical solution even for small sets of users.

In order to show that there are practical and still threatening attacks, we now formalize an attack aimed at discovering the area where the user is with a looser approximation. The general idea is to cluster the users according to their distances, hence identifying sets of users close to each other, and then using BK to map clusters to geographical regions. As an example, clusters of appropriate dimensions may correspond to cities, and the adversary may be able to associate a cluster of users to the correct city as their actual approximate position.

In this attack we model an arbitrary geographical region z in S as the set of cells from G that overlap with z. We denote with Z a set of these regions. For example, the regions in Z are the cities where the adversary tries to locate users. The minimum distance between regions is defined as:

$$d_z^{min}(z_1, z_2) = \min_{g_1 \in z_1, g_2 \in z_2} d_g^{min}(g_1, g_2)$$

where $d_g^{min}(g_1, g_2)$ is the minimum distance between two cells g_1 and g_2. The maximum distance $d_z^{max}(z_1, z_2)$ between two regions is defined analogously.

In Section IV, while considering cities as the regions, we show that, given OK, it is possible to identify a set C of

clusters of U that are actually composed by persons located in cities. Note that C is not a partition of U, since we are interested in a set of regions not covering the entire space (e.g., only the largest cities). After the clustering, we need to associate each cluster with the correct region, so that we can derive the geographical position of the cluster. For this purpose, we model the *cluster-to-region* mapping $m : C \rightarrow Z$ that represents an association of each cluster to a region. Conceptually, a cluster-to-region mapping is similar to a user-to-point mapping, with the only theoretical difference that in a cluster-to-region mapping all clusters are mapped to different regions, while in the user-to-point mapping two or more users can be mapped to the same point.

Intuitively, this attack becomes practically feasible when the number of clusters and regions is small, as in the case of the main cities of a country.

Analogously to the case of user-to-point mappings, we are interested in identifying the consistent cluster-to-region mappings, hence excluding the ones inconsistent with available distance information. Intuitively, a cluster-to-region mapping m is *consistent* if for each pair of clusters and each pair of users in the two clusters the distance between the two users is smaller (greater, respectively) than the maximum (minimum, respectively) distance between the two regions *corresponding* to the two clusters. Clearly the consistency condition can only be checked for those users such that $d_u()$ is defined.

Definition 3. *A cluster-to-region mapping m is called* consistent cluster-to-region mapping *if, for each pair of clusters $c_1, c_2 \in C$, and for each pair of users $u_1 \in c_1$, $u_2 \in c_2$ such that $d_u(u_1, u_2)$ is defined, the following holds:*

$$d_z^{min}(m(c_1), m(c_2)) \leq d_u(u_1, u_2) \leq d_z^{max}(m(c_1), m(c_2))$$

Similarly to the case of user-to-point mappings, we want to assign a probability to the consistent cluster-to-region mappings. This probability can be derived from $nu()$. Indeed, extending Definition 1, the probability $P[u, z]$ that a generic user u is located in region z is: $\sum_{g \in z} P[u, g]$.

Since by Definition 1 the probability that a user is in a region z is independent from the probability of other users being in z and each user has the same probability, the probability $P[c, z]$ that all users in a cluster c are in z is given by: $(P[u, z])^{|c|}$.

We can now define the absolute probability of a consistent cluster-to-region mapping.

Property 4. *The* absolute probability $P[m]$ *that, given a clustering C and a generic user u, a consistent cluster-to-region mapping m is correct is:*

$$P[m] = \prod_{c \in C} (P[u, m(c)])^{|c|}$$

Property 5 defines the relative probability of a mapping.

Property 5. *Let C be a clustering and M the set of consistent cluster-to-region mappings. The* relative probability $P_r[m]$ *that a consistent cluster-to-region mapping m is correct is:*

$$P_r[m] = \frac{P[m]}{\sum_{m' \in M} P[m']}$$

III. Techniques

In this section we first show how the attack based on user clustering can be computed in practice. Then, we describe an additional step, based on trilateration, that, as shown in Section IV, significantly improves the attack performances.

A. Computing clusters of users

We now present a clustering technique designed to identify k groups of users that are located in user-dense areas of S. The computation of the clusters of users needs to be based only on the observation knowledge OK, i.e., on the relative distances between some pairs of users. We adopt the *Single Linkage* clustering technique that indeed relies only on relative distances [14]. The idea of this technique is to start with a set containing a single user for each cluster, and then to iteratively merge the two closest clusters. The notion of "closest clusters" is based on the minimum distance between two clusters c_1 and c_2 defined as follows:

$$d_c^{min}(c_1, c_2) = \min_{u_1 \in c_1, u_2 \in c_2 \text{ s.t. } d_u(u_1, u_2) \text{ is defined}} d_u(u_1, u_2)$$

Note that $d_c^{min}(c_1, c_2)$ is undefined if $d_u(u_1, u_2)$ is undefined for all pairs of users $u_1 \in c_1$, $u_2 \in c_2$.

Algorithm 1 $SingleLinkageClustering$

Input: the set U of users, a set Z of sets of cells, the function $du()$, an integer k.
Output: a set C of sets of users.
Procedure:

1: $C = \{\{u_1\}, \{u_2\}, \ldots, \{u_{|U|}\}\}$
2: Order Z in decreasing order with respect to the expected number of users. Let n be the number of users in the k-th element of Z.
3: **while** ($|C| > 1$) AND (exist $c_1, c_2 \in C$ such that $d_c(c_1, c_2)$ is defined) AND (the number of users in the k-th largest cluster of C is less than n) **do**
4: Find clusters c_1 and c_2 in C such as their distance is minimal (closest clusters)
5: $C = C \setminus \{c_1\} \setminus \{c_2\}$
6: $C = C \cup \{c_1 \cup c_2\}$
7: **end while**
8: Remove from C the sets with cardinality smaller than n;
9: **return** C;

The clustering procedure is presented in Algorithm 1. In the initialization phase (Lines 1 and 2), the $|U|$ clusters are created, each one containing a single user, and the set Z of regions is ordered accordingly to the number of expected users. This number is actually derived from background knowledge that includes population density for each cell. Expected number of users can be easily derived from this. The number n of expected users in the k-th most user-populated region in Z is used in the clustering steps of the algorithm.

At each iteration, the two closest clusters are merged (Lines 4 to 6). The termination of the iteration (and consequently of the algorithm) depends on three conditions. The first condition

says to stop when $|C| = 1$, since in this case the users are all into a single cluster, and the algorithm just fails to identify the clusters that are associated with the regions of Z. The second termination condition is satisfied when no distance among any pair of current clusters is known. This also indicates a "failed clustering" unless the third condition is also verified. The third condition denotes a "successful clustering". When it is verified, the algorithm has identified a clustering set C containing k clusters each having a number of users at least as large as n. Intuitively, this condition is aimed at identifying the clusters that correspond to the k most populated regions. Since we are only interested in these clusters, we remove the others before returning C (Line 8).

Example 4. *Figure 2 graphically shows the clusters identified by Algorithm 1 after* 1500 *and* 2500 *iterations in a run with* $16,000$ *users in France. Each dots corresponds to a user, with black dots representing users belonging to a "large" cluster (i.e., having cardinality larger than* n*). After* 1500 *iterations only* 4 *"large" clusters are found, while after* 2500 *iterations the algorithm identifies a total of* 8 *clusters that actually correspond to some of the* 11 *most populated cities of France.*

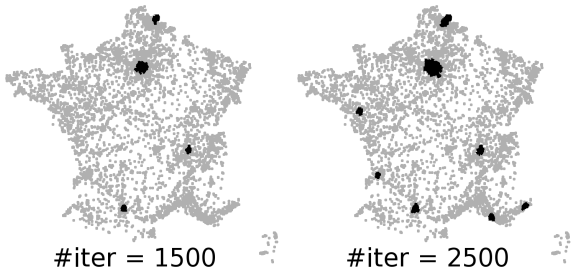


Fig. 2. Example of execution of Algorithm 1

The number of iterations of Algorithm 1 is linear in the number of users. The operation that dominates the temporal complexity of each iteration is the computation of the two closest clusters. Without any optimization, this computation would require to calculate the distance between any pair of clusters. This operation is quadratic in the number of users, making the worst case time complexity of the algorithm $O(|U|^3)$ and the space complexity linear in $|U|$, the space needed to store C.

To improve the time complexity we designed a data structure that stores, for each cluster c: a) the set $c.users$ of users in the cluster, b) the list $c.dist$ of pairs $\langle c', d_c(c, c')\rangle$ for each $c' \in C$ such that $d_c(c, c')$ is defined, and c) the element $c.min$ of $c.dist$ having the smallest value of $d_c(c, c')$. Note that, since users' and clusters' distances are symmetric, it is not necessary to store all pairwise distances between clusters, but only half of them. In the initialization phase it is necessary to compute $d_u()$ for each pair of users and the worst case time complexity of this operation is $O(|U|^2)$. The operation that dominates the complexity of each iteration is to merge $c_1.dist$ and $c_2.dist$ for two clusters c_1 and c_2. This is analogous to compute the union between two sets. Since we define a total order among clusters and we use it to maintain $c_1.dist$ and $c_2.dist$ ordered, the union of $c_1.dist$ and $c_2.dist$ can be computed in a time linear in the sum of the number of users in c_1 and c_2, that is bounded by $|U|$. To summarize, the initialization phase has a worst case time complexity of $O(|U|^2)$. Each iteration has a worst case time complexity of $O(|U|)$ and at most $|U|$ iterations are required, hence the worst case time complexity of the algorithm is $O(|U|^2)$.

B. Computation of consistent mappings

Once clusters of users have been identified, cluster-to-region mappings should be generated and the consistent ones selected. Two problems arise in this process. The first problem is due to what we call "cluster stretching": when the clustering algorithm is run, the large majority of the users in a cluster are located in a region of Z, but few of them are also located slightly outside the region (e.g., in the suburbs of a city). For example see Figure 3 where the rectangle represents the minimum bounding rectangle (MBR) of Paris and the dots (both black and gray) are the users in a cluster. It can be observed that, while the great majority (90%) of dots are within the MBR, few of them are slightly outside.

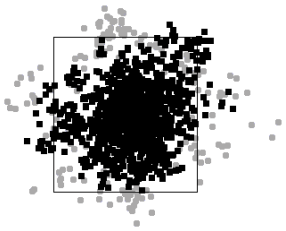


Fig. 3. The "cluster stretching" problem.

The "cluster stretching" problem could lead to erroneously classify a mapping as inconsistent. To face this problem, we introduce a tolerance error factor $\alpha \in (0, 1)$ in the condition to check the consistency (see Definition 3) as follows:

$$\alpha \cdot d_z^{min}(m(c_1), m(c_2)) \leq d_u(u_1, u_2) \leq \frac{1}{\alpha} \cdot d_z^{max}(m(c_1), m(c_2))$$

The intuition is that, when α is close to one, the tolerance is low. For values of α closer to 0, the value of the left part of the inequality gets smaller, while the right part gets larger, hence making the condition easier to satisfy even if a user may be slightly outside the region. In Section IV-B we show the impact of the tolerance error factor and we motivate the choice of the value used in our experiments.

The second problem is related to the fact that the number of possible cluster-to-region mappings is the same as the number of all ordered sequences of $|C|$ distinct elements of Z, i.e., $|Z|!/(|Z| - |C|)!$. Clearly, this leads to a large number of possible mappings, also for small sets Z and C. Since for each possible mapping we need to check if that mapping is consistent, it is necessary to optimize this operation. For

this purpose the minimum and maximum distance between each pair of regions in Z (i.e., $d_z^{min}(z_1, z_2)$ and $d_z^{max}(z_1, z_2)$) are pre-computed off-line. We also pre-compute, for each pair of clusters $c_1, c_2 \in C$, the minimum ($d_c^{min}(c_1, c_2)$) and maximum ($d_c^{max}(c_1, c_2)$) distance between each pair of users $u_1 \in c_1$ and $u_2 \in c_2$. It follows that checking if a mapping is consistent requires evaluating the following two inequalities for each pair of clusters c_1, c_2 of C.

$$\alpha \cdot d_z^{min}(m(c_1), m(c_2)) \leq d_c^{min}(c_1, c_2)$$

$$d_c^{max}(c_1, c_2) \leq \frac{1}{\alpha} \cdot d_z^{max}(m(c_1), m(c_2))$$

By using this optimization, the process of checking if a mapping is consistent has a worst case time complexity of $O(|C|^2)$. Considering that we need to run this computation for each possible mapping, the computational complexity is $O\left(|C|^2 \cdot \frac{|Z|!}{(|Z|-|C|)!}\right)$. Clearly this may be still problematic in general, however, as we show in Section IV, the attack can be effectively computed for values of $|C|$ and $|Z|$ that are sufficient to identify the approximate position of the users located in the most populated cities.

C. Evaluation of probabilities

The last step of the attack process consists in computing the absolute and relative probabilities of each consistent cluster-to-region mapping. This is done accordingly the formulas presented in Section II. We compute the function $nu(g)$ from the distribution of inhabitants of g, which is public information [15], and by assuming that the users of the service are uniformly distributed in the population. Formally, given $inhab(g)$ the number of inhabitants of g and $totPop$ the total population in the spatial domain S: $nu(g) = inhab(g)/totPop \cdot |U|$.

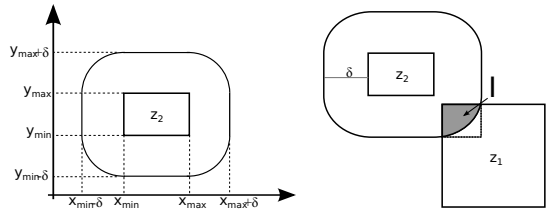
While, in theory, the number of consistent cluster-to-region mappings can be in the same order of all the possible cluster-to-region mappings, in practice these mappings are a small fraction (in our experiments, in which major cities are considered as regions, they are in the order of a few units at most).

A technical difficulty arises in the computation of the absolute probability of a mapping due to the fact that some of the values used in the computations may be non-negative numbers very close to 0 (for example, these values can be in the order of 10^{-10000}). The precision of standard floating point representations (e.g., primitive double type in Java) is not sufficient to manage these values. We solved the problem with non approximated decimal representations and, in particular, with the "BigDecimal" standard Java class, that, however, has a significant negative effect on the computation time.

D. Trilateration

When the cluster-to-region mapping with highest probability is found, the adversary learns the approximate position of the users belonging to each cluster. We now present a technique that can lead the adversary to obtain more precise location information for these users and to also compute the approximate location of the other users. The technique is based on trilateration [16]. It takes as input the approximate locations given by the candidate cluster-to-city mapping with highest relative probability. Our solution addresses two differences with respect to "standard" trilateration: (1) only approximated location information is available so we have to trilaterate with rectangular areas instead of points; (2) approximate location of users belonging to a cluster can be incorrect, due to the "cluster stretching" problem. We tackle this inaccuracy through the "refinement" technique.

(1) To address the difficulty of having only approximated location information, we represent known users' positions as rectangular regions and we operate geometrical intersections to locate users. Intuitively, suppose that a user u_1 is at distance δ from another user u_2 located in a city z_2. This bounds the location of u_1 to the set of points whose minimum distance from z_2 is at most δ. We denote this "extended" region as $ext(z_2, \delta)$, see an example in Figure 4(a). If the (approximate) position user u_1 is unknown, then the adversary learns that u_1 is located inside $ext(z_2, \delta)$. Otherwise, if the adversary already knows that u_1 is located in z_1, it can be deduced that u_1 is located in the intersection between z_1 and $ext(z_2, \delta)$ (e.g., the dark region in Figure 4(b)).



(a) $ext(z', \delta)$ (b) Geometrical intersection

Fig. 4. Computation of intersection

When the above geometrical computation reduces the uncertainty about the position of a user u, this restriction can "propagate" to all the users u' whose distance from u is known. This idea leads to our iterative Algorithm 2 that computes the function r associating a spatial region to each user. In the initialization phase (Lines 1-4), each clustered user is assigned to the corresponding city, while the other users are assigned to the entire space. At each iteration (Lines 6-8) for each pair of users $\langle u_1, u_2 \rangle$ whose distance is known, we compute the intersection between the approximate position of u_1 and $ext(r(u_2), d_u(u_1, u_2))$ where $r(u_2)$ is the approximate position of u_2. The computation terminates when there are no more approximate positions to restrict.

Note that the intersection between the approximate position of u and $ext(r(u_2), d_u(u_1, u_2))$ is not always a rectangle (see again Figure 4(b)). When iterating, this can lead to figures with complex shapes that require a non-constant-time computation of the geometrical intersection function. In order to ease the computation, we first compute the exact intersection and then we bound it with its MBR (dotted lines in Figure 4(b)). In our experiments this optimization leads to a negligible decrease in the effectiveness of the algorithms (in terms of the size of the approximated positions) but it decreases significantly the computation time.

Algorithm 2 $Trilateration$

Input: the set U of users, a set C of clusters, a cluster-to-region mapping $m : C \to Z$, the function d_u.
Output: the function $r(u)$ for each user $u \in U$.
Procedure:

```
for all u ∈ U do
    if u ∈ cluster c ∈ C then r(u) = m(c)
    else r(u) = entire space
end for
while (at least 1 region is reduced) do
    for all pairs of users ⟨u1, u2⟩ such that d_u(u1, u2) is defined do
        r(u1) = MBR(r(u1) ∩ ext(r(u2), d_u(u1, u2)))
    end for
end while
return function r()
```

(2) The "cluster stretching" problem (see Section III-B) can lead to erroneous associations between users and regions. In our experiments we observe that the "cluster stretching" problem can drastically affect the precision of Algorithm 2. Intuitively, this is caused by the iterations of Algorithm 2 that propagate errors. To prevent this, we adopt a solution that improves the precision of the cluster-to-region mapping and hence, as shown in Section IV, drastically increases the precision of our attack. The idea is that, after computing the mapping with highest probability, we "refine" the clustering so that each cluster contains a fraction of the expected number of users in the corresponding city. The refinement of a cluster c takes as input a number n' and the set of users belonging to c: these users are clustered together until a cluster with exactly n' users is found. This cluster represents the refinement of c.

The refinement increases the precision in terms of the correct association of users to a region since, in general, the last users added to a cluster have a higher probability to be outside the region, as shown in Example 5. We use the refinement to restrict the set of users in a cluster to a fraction of the expected users in the corresponding region. We call "refinement factor" the percentage of expected users that are discarded when refining the cluster. For example, with a refinement factor of 20%, a cluster is refined including 80% the expected users in the corresponding region.

Example 5. *Consider Figure 3 again. The expected number of users in Paris is* 1200*, while* 1310 *users have been clustered into c. The figure shows with black dots the* 1200 *users in the refinement of c (with refinement factor of* 0%*), while the gray dots represent the other* 110 *users that are in c but not in its refinement. The percentage of users of c that are located in the MBR is about* 90% *while considering the refinement of c the percentage is about* 94% *and among the* 110 *users in c but not in the refinement the percentage is about* 45%*. This clearly indicates that the first users that are clustered together actually have a much higher probability of being within the MBR.*

IV. Experiments

Our experiments simulate the observation knowledge (OK) that an adversary may obtain by running a friend-finder service in which privacy protection is implemented through a distance preserving transformation. We use as background knowledge (BK) publicly available geographic data and population density. The aim of the attack is to infer the actual position of users (supposed to be hidden to the service) in terms of a region in which they are located. In Section IV-A we present the experimental setting. We evaluate the effectiveness of the attack by first focusing on the clustering technique (Section IV-B) and then showing how trilateration improves the attack (Section IV-C). Finally, in Section IV-D we discuss other experimental results we obtained.

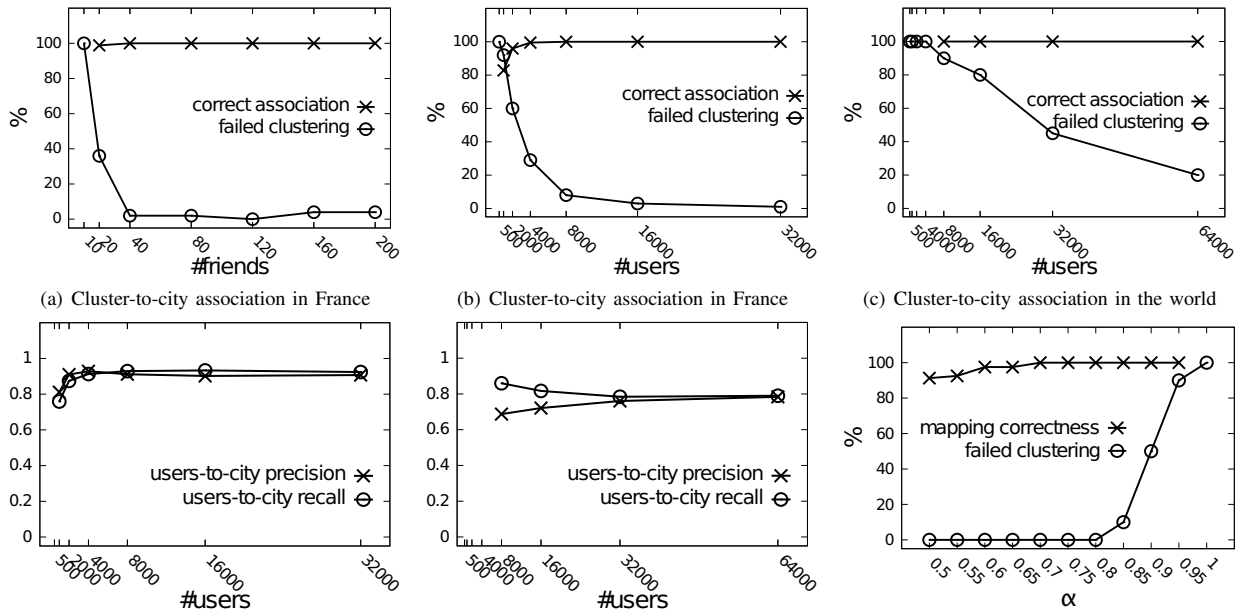
A. Experimental setting

The overall structure of our experimental evaluation is the following: we first simulate the position of some users in a geographical area. Then, we compute the distance between some pairs of users (hence simulating OK) and we use this data to perform the attack. Finally, using the original information about users' position, we evaluate the correctness of the attack.

The simulation of users' position consists in randomly choosing a point in the geographical area according to a probabilistic distribution. We consider three scenarios corresponding to the geographic areas covering Australia, France and the entire world. The intuition behind the choice of the first two scenarios is that the adversary could know, from a number of different sources, the country where users are located. In the world scenario we assume the adversary does not have this knowledge. The choice of using France and Australia scenarios is due to the fact that the two countries have very different average population density, (about 2.8 $inhabitants/km^2$ in Australia, about 116 $inhabitants/km^2$ in France) and also large difference in population density variance (Australia is characterized by large unpopulated regions, while in France the variance of population density is much lower). Differently from what expected, we obtained very similar results for these two scenarios. For this reason, and due to page limit, in the following of this section we only report the results for France.

The probabilistic distribution we adopted is taken from GPWv3 [15], a dataset that provides density information by dividing the world into cells and providing the number of inhabitants for each cell. In our experiments we observed that smaller cells result in more effective attacks while not significantly impacting on the computation time. Hence, we use the cells at the highest available resolution, i.e., 2.5 arc-minutes, that corresponds to a cell edge of about $5km$ at the equator. Since no public information is available at a higher resolution, in the current setup we assume that, within each cell, the population density is uniform. In the following we denote with "#users" the value $|U|$.

Another parameter that has an impact on the attack is the number of distances between users that are known to the adversary. In the friend-finder service, this corresponds to the



(a) Cluster-to-city association in France (b) Cluster-to-city association in France (c) Cluster-to-city association in the world

(d) Users-to-city association in France (e) Users-to-city association in the world (f) Cluster-to-city association in France

Fig. 5. Results obtained without trilateration

mean number of friendships per user ($\#friends$). In order to capture the intuition that users tend to have more friends in a close-by region, 50% of each users' friends are located in an area whose size is 2% of the simulated region. In the case of France this corresponds to a distance of 100km from the user.

The parameters k and $|Z|$ are fixed to 8 and 11, respectively (note that, when the clustering is successful, $|C| = k$). In our experiments (not reported in the following) we observed that while these values already enable powerful attacks, higher values of these two parameters would improve it even more. However, higher values negatively impact on the performances, due to the reasons explained in Section III. Using these two values for k and $|Z|$ and the default values for $\#users$ and $\#friends$, we can compute an attack in about 2 minutes on a 2.26GHz CPU with 4GB of main memory.

We evaluate the attack in three scenarios by varying parameters, showed in Table I, with default values in bold. The value of the tolerance error factor α is set to 0.75, empirically chosen, as will be shown in Section IV-B. The results are computed as the average, as well as minimum and maximum, out of 10 runs.

TABLE I
PARAMETER VALUES

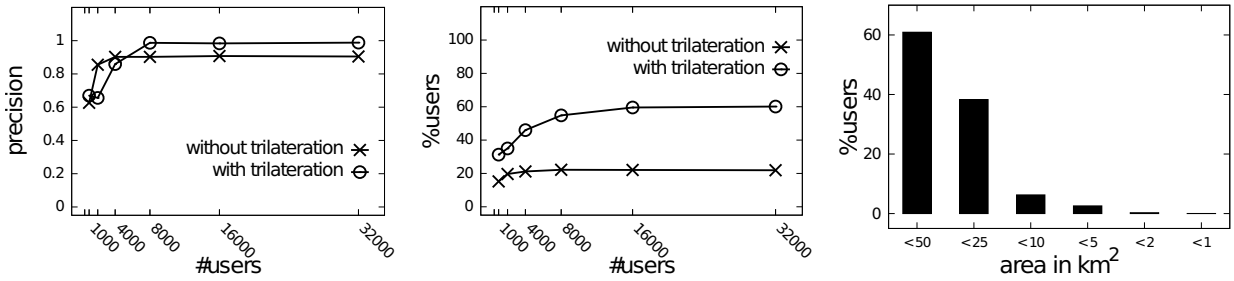
Parameter	Values
#users	500, 1000, 2000, 4000, 8000, **16000**, 32000, 64000
#friends	10, 20, 40, **80**, 120, 160, 200
refinement	0, 20, 40, 60, **80**, 90, 92, 94, 95, 96, 97, 98, 99
α	1, 0.95, 0.90, 0.85, 0.80,**0.75**, 0.7, 0.65, 0.6, 0.55, 0.5
Scenario	Australia, France, World

B. Evaluation of the Attack Based on Clustering Only

The aim of the experimental evaluation of the clustering technique is to assess the attack effectiveness in terms of the "cluster-to-city association" and "users-to-city association".

The "cluster-to-city association" measures the percentage of clusters that are correctly associated to the corresponding city. A cluster is considered correctly assigned to a city if at least 50% of its users are located within the assigned city's MBR. We can observe from Figure 5(a) that, for values of $\#friends$ equal to or larger than 40 in France, the clustering algorithm fails less than 3% of the times while, when it does not fail clustering, the cluster-to-city association is correct 100% of the times. This is an important property of our attack: when the number of users is sufficiently large (in this case we are using the default value of 16000) if the clustering does not fail, then the adversary knows with high likelihood that the cluster-to-city association is correct. This can also be observed in Fig. 5(b) showing that the cluster-to-city association is always correct for large values of $\#users$. Vice versa, for $\#users$ smaller than 4000, we observe that a wrong cluster-to-city association is more likely.

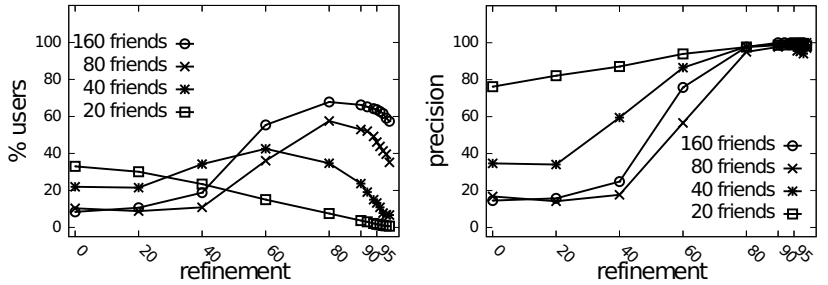
We observe a similar trend in the world scenario (see Figure 5(c)). In this case the algorithm always returns correct cluster-to-city associations when there are at least 8000 users. Compared to the France scenario, the percentage of failed clustering decrease more slowly for larger values of $\#users$. Indeed, the clustering fails about 20% of the times with 64000 users. This is due to the fact that in the world scenario we are simulating a much smaller percentage of the population.



(a) Precision of positioning users in an area smaller than 50 km^2 in France

(b) Users correctly positioned in an area smaller than 50 km^2 in France

(c) Size of the regions where users are correctly positioned



(d) Users correctly positioned in an area smaller than 50 km^2 in France

(e) Precision of positioning users in an area smaller than 50 km^2 in France

Fig. 6. Results of the attack with trilateration

Indeed, with 32000 $\#users$ in France we simulate about the 0.04% of the population, while with 64000 $\#users$ we simulate only the 0.00001% of the world's population.

The "user-to-city association" measures the precision and recall of the association between a user and a city. In this case, the association of a user with a city is correct if the cluster containing that user is associated to the city where the user is actually located. Since, as we observed above, the cluster-to-city association is always correct for sufficiently large values of $\#users$, the user-to-city association helps us understanding the impact of the "clustering stretching" problem. Consider Figure 5(d): for values of $\#users$ equal to or larger than 2000 the average precision is almost not affected by the value of $\#users$ and it is always above 90%. Similar trend and values can be observed for the recall. This means that most of the users that are located in a city are actually reported as being in that city and that, in most of the cases, a user reported being in a city is actually located there. A very similar result can be observed for the world scenario (see Figure 5(e)). The main difference is that we need a minimum of 8000 users, since for smaller values of $\#users$ the clustering always fails.

One experiment is devoted to find a proper value for the tolerance factor α described in Section III-B. As shown in Figure 5(f), the cluster-to-city association often fails for large values of α (above 0.85) since no consistent mapping is found. For values smaller than 0.65, more consistent mappings need to be evaluated, resulting in much higher computation time and higher probabilities of errors in choosing the correct mapping. A good trade-off is found for values of α between 0.7 and 0.8, for which there are no failed clusterings and the cluster-to-city associations are always correct. For these reasons, in our experiments we used a value of $\alpha = 0.75$.

C. Evaluation of the attack with trilateration

The next set of experiments evaluates the effectiveness of the trilateration step, in terms of its correctness and of the percentage of users that it can associate to a "small" area.

Figure 6(a) and 6(b) highlight the improvement of the attack due to trilateration. Figure 6(a) shows that the trilateration technique improves the precision of the attack. This is mainly due to the refinement step (see Section III-D). At the same time, Figure 6(b) shows that, when trilateration is used, it is possible to correctly associate more than 60% of the users to an area smaller than $50km^2$. The attack with trilateration is about three times better along this metric than the attack without trilateration that can only locate the users in the cities. The results in the world scenario (not shown here) are very similar, given that the number of users is sufficiently large.

Figure 6(c) helps us understanding the size of the regions where the attack locates the users. Approximately 40% of the users are correctly associated with a region smaller than $25km^2$. The attack can also discover a more precise location (less than $5km^2$) for about 416 users on average (i.e., 2.6% of the users). A few users (5, on average) can be localized in an area smaller than $1km^2$.

Figure 6(d) and Figure 6(e) show the percentage and the precision, respectively, of users located in an area smaller than $50km^2$ for different values of the refinement factor. From the comparison of the two figures, we can observe that there is a trade-off between precision and the percentage of users whose location is discovered: increasing the refinement with values

higher than 80% makes the precision converge to 1, but the percentage of users whose location is discovered decreases. Experiments show that the threshold value depends on the value $\#friends$: when this value is equal to 20, better results are obtained with a refinement factor of 0, while if $\#friends$ are more than 80 a refinement factor of the 80% is needed.

D. Other experimental results

We run a set of experiments to evaluate the effectiveness of our attack when only short distances (less than $3km$) are known to the adversary. This is the situation that would result when using a defense like the one proposed in [12]. We adapted our attack to this situation by changing the clustering algorithm so that it terminates when the distance between the two closest clusters is larger than 3km. A major difference is due to the fact that in absence of long distances it is not possible to compute if a cluster-to-region mapping is consistent or not. For this reason we skip the consistency check and compute the relative probability for each mapping. This solution incurs in two distinct problems. The first is a computation cost problem: computing the relative probability among all the possible mappings is much more time consuming than first selecting consistent mappings and then computing their probability. For this reason we had to reduce the values of k and $|Z|$ to a maximum of 3 and 4, respectively. The second problem is related to the fact that, by assuming that all mappings are consistent, it is much more frequent that an incorrect mapping has a high relative probability, hence resulting in an incorrect cluster-to-city association. Despite these problems, our results show that the attack is still effective in most of the cases. Using default values for $\#users$ and $\#friends$, $k = 2$ and $|Z| = 4$ in France, our clustering algorithm has always been able to correctly compute the clusters, and the cluster-to-city association always resulted to be correct. The computation time is in the order of 2 minutes. However, we observed that, differently from the case in which even long distances are known to the adversary, increasing the values of k and $|Z|$ can result in worse performance, since wrong cluster-to-city associations are more frequent. For example, by using $k = 3$ and $|Z| = 4$ we obtained 73% of correct cluster-to-city associations. Overall, we can conclude that the application of a defense technique like the one proposed in [12] to our reference scenario (e.g., a friend-finder service) does not protect from our attack, that it still able to identify the correct position of many users. On the other hand this defense poses some challenges to our attack both from the computational point of view, and from its effectiveness, since it reduces the number of possible users whose position can be directly identified.

V. Conclusions

In this paper we investigated how distance information between moving individuals can be used to violate location privacy in presence of background knowledge about population density. In contrast with other kinds of background knowledge that have been considered in the literature we base our attack entirely on public knowledge and on realistic partial distance information as it could be released by a geo-social network friend-finder service. Our experiments show that the knowledge of a relatively low number of distances is sufficient to correctly position the majority of users in an area smaller than 50 km^2 and to correctly identify the position of some users in an area smaller than 1 km^2.

We believe that this study shades a new light on the safety of distance preserving transformations as privacy techniques; at the very least, it provides new important elements to design safer defense techniques considering public knowledge on population density. The major future work is indeed the design of these techniques.

VI. Acknowledgments

The authors wish to thank X. Sean Wang for his insightful comments on a preliminary version of this paper. This work was partially supported by Italian MIUR under grants FIRB-RBFR081L58_002.

References

[1] C. Bettini, S. Jajodia, P. Samarati, and X. S. Wang, *Privacy in Location-Based Applications*, ser. LNCS. Springer, 2009.

[2] G. Ghinita, "Private queries and trajectory anonymization: a dual perspective on location privacy," *Trans. Data Privacy*, 2009.

[3] G. Zhong, I. Goldberg, and U. Hengartner, "Louis, Lester and Pierre: Three protocols for location privacy," in *Privacy Enhancing Technologies*, ser. LNCS. Springer, 2007.

[4] L. Šikšnys, J. R. Thomsen, S. Šaltenis, M. L. Yiu, and O. Andersen, "A location privacy aware friend locator," in *Proc. of the 11th Int. Symp. on Spatial and Temporal Databases*, ser. LNCS. Springer, 2009.

[5] L. Šikšnys, J. R. Thomsen, S. Šaltenis, and M. L. Yiu, "Private and flexible proximity detection in mobile social networks," in *Proc. of the 11th Int. Conf. on Mobile Data Management*. IEEE, 2010.

[6] H. P. Li, H. Hu, and J. Xu, "Nearby friend alert: Location anonymity in mobile geo-social networks," *IEEE Pervasive Computing*, 2012.

[7] S. Mascetti, C. Bettini, and D. Freni, "Longitude: Centralized privacy-preserving computation of users' proximity," in *Proc. of 6th VLDB workshop on Secure Data Management*, ser. LNCS. Springer, 2009.

[8] S. Mascetti, D. Freni, C. Bettini, X. S. Wang, and S. Jajodia, "Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies," *The VLDB Journal*, 2011.

[9] H. Hu and J. Xu, "Non-exposure location anonymity," in *Proc. of the 25th Int. Conf. on Data Engineering*. IEEE, 2009.

[10] K. Liu, C. Giannella, and H. Kargupta, "An attacker's view of distance preserving maps for privacy preserving data mining," in *Proc. of the 10th Eur. Conf. on Principles and Practice of Knowledge Discovery in Databases*. Springer, 2006.

[11] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proc. of the 2009 SIGMOD Int. Conf. on Management of data*. ACM, 2009.

[12] G. Ghinita, C. R. Vicente, N. Shang, and E. Bertino, "Privacy-preserving matching of spatial datasets with protection against background knowledge," in *Proc. of the 18th SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*. ACM, 2010.

[13] S. Mascetti, L. Bertolaja, and C. Bettini, "Location privacy attacks based on distance and density information," in *Proc. of the 20th SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*. ACM, 2012.

[14] R. Sibson, "SLINK: an optimally efficient algorithm for the single-link cluster method," *The Computer Journal*, 1973.

[15] C. U. Center for International Earth Science Information Network (CIESIN) and C. I. de Agricultura Tropical (CIAT), "Gridded population of the world, version 3 (gpwv3)," 2005.

[16] H. L. Groginsky, "Position estimation using only multiple simultaneous range measurements," *IRE Transactions on Aeronautical and Navigational Electronics*, 1959.