X. Wang, et al., "Location Proximity Attacks against Mobile Targets: Analytical Bounds and Attacker Strategies", *23rd European Symposium on Research in Computer Security (ESORICS 2018)*, LNCS vol. 11099, pp. 373-392, 2018.
http://doi.org/10.1007/978-3-319-98989-1
NICS Lab. Publications: https://www.nics.uma.es/publications

Location Proximity Attacks against Mobile Targets: Analytical Bounds and Attacker Strategies

Xueou Wang[1], Xiaolu Hou[2][0000−0002−4512−6921], Ruben Rios[3][0000−0002−6251−4897], Per Hallgren[4,5], Nils Ole Tippenhauer[1][0000−0001−8424−2602], and Martín Ochoa[1,6]

[1] Singapore University of Technology and Design (SUTD),
{xueou_wang,nils_tippenhauer}@sutd.edu.sg
[2] Cyber Security Lab, School of Computer Science and Engineering,
Nanyang Technological University, Singapore
ho0001lu@e.ntu.edu.sg
[3] Computer Science Department, University of Málaga, Spain
ruben@lcc.uma.es
[4] Chalmers University of Technology, Gothenburg, Sweden
[5] Einride AB, Sweden
per.hallgren@einride.tech
[6] Department of Applied Mathematics and Computer Science,
Universidad del Rosario, Bogotá, Colombia
martin.ochoa@urosario.edu.co

Abstract. Location privacy has mostly focused on scenarios where users remain static. However, investigating scenarios where the victims present a particular mobility pattern is more realistic. In this paper, we consider abstract attacks on services that provide location information on other users in the proximity. In that setting, we quantify the required effort of the attacker to localize a particular mobile victim. We prove upper and lower bounds for the effort of an optimal attacker. We experimentally show that a *Linear Jump Strategy* (LJS) practically achieves the upper bounds for almost uniform initial distributions of victims. To improve performance for less uniform distributions known to the attacker, we propose a *Greedy Updating Attack Strategy* (GUAS). Finally, we derive a realistic mobility model from a real-world dataset and discuss the performance of our strategies in that setting.

1 Introduction

Proximity services are a special type of location-based service (LBS) where the user is informed about nearby people of interest and their distance rather than their exact location in an attempt to protect location privacy. To that end, queries are sent to the proximity service including the location of the user, the search radius and possibly some information about the target. Unfortunately, when the exact distance to a user is revealed by the proximity service it is

possible to retrieve the exact location of the user. Examples include claims that Egyptian authorities leveraged dating apps to track down gay users [6], and others attempted to find locations of Tinder users [19]. Consequently, the need to provide rigorous privacy guarantees in proximity services is evident.

In view of these threats, some effort has been devoted to the development of privacy-preserving proximity testing protocols [12,15,8]. These solutions allow two users to learn whether they are within a certain distance of each other but no further information about their location or distance is revealed to the other user. Some of these protocols rely on a trusted third party to handle users' locations while others get rid of this third party and operate in a decentralized way.

In general, these solutions focus on partially static models, where attackers can change their location freely but victims do not. Capturing the behavior of an optimal attacker in this setting when the victim is static is already hard although some progress has been made in recent years [13,9]. However, this situation is quite unrealistic and motivated us to investigate the expected effort for adversaries to localize users that move in a particular mobility pattern.

Although there has been extensive study in mobility models [2], attack strategies [11,19,13], location privacy protection mechanisms [18,8,9] and location privacy quantification metrics [17,16], there is very limited research on location prediction based on sequential spatio-temporal data using a probabilistic approach. Given that location data acquired from an LBS platform are sequential spatio-temporal data, one can obtain information on the moving patterns/behaviors of the user by analyzing the trajectory dataset. Hence, we are interested in quantifying the effort of an *arbitrary attacker* issuing proximity queries in finding a user under certain models. In other words: *how quickly can an attacker locate a user based on queries to the LBS*?

Our main contributions are as follows:

1. Given any attacker strategy, assuming the victim follows a random walk, we establish a formula for calculating the probability of the attacker finding the location of the victim after any number of queries.
2. We give upper and lower bounds on the minimum number of queries an attacker needs to issue to locate the victim with probability $\frac{1}{2}$ (generalizable to other probabilities). In particular, for a search space of size M and assumptions on victim's initial location and mobility, we show that an optimal attacker needs at most $\frac{M}{2}$ queries to locate a victim with probability $\frac{1}{2}$.
3. We implement the *linear jump* strategy from the proof, and show empirically that its effort falls within the theoretical bounds. We find that for non-uniform initial distributions of victims, the strategy performs worse. To address this, we propose a *greedy updating attacker* strategy.
4. We then use the strategies to estimate attacker effort for more general settings (e.g., limited knowledge known by the attacker, other mobility models).
5. We derive a transition matrix representing real-world mobility patterns from a large dataset [21,22], and show how that mobility pattern influences attacker effort compared to previous results.

The rest of this work is organized as follows. In Section 2, we present our mathematical modeling of search spaces and mobility patterns. Section 3 summarizes the problem statement. We present the mathematical analysis for calculating the probability of an attacker locating a victim using location proximity queries in Section 4. In Section 5, we present our applied linear jumping and greedy attacker strategies, and evaluate them empirically in Section 6. Related work and conclusions are discussed in Sections 7 and 8, respectively.

2 Preliminaries

2.1 Search Domain

In our model, users are able to move in a finite space that can be divided into discrete *locations*. The granularity of the locations is limited by the maximum precision of the positioning device or privacy considerations [5].

Space. We consider two cases. In the one-dimensional case, the search space is divided into n locations, each of which has two adjacent locations, left and right, except for the corners. We call this space $\mathcal{S}^n$. For the two-dimensional case, the search space can be divided into $m \times n$ locations. Each point typically has four adjacent locations, except for the corners. We call this space $\mathcal{S}^M$, with $M = mn$.

Time. We assume discrete time steps $k \in \{0, 1, 2, \dots\}$. In each time step, the user $\mathcal{X}$ can move once. The movement of users is represented as a transition in our model. Let $\mathcal{X}_k$ denote the position of user $\mathcal{X}$ at time k.

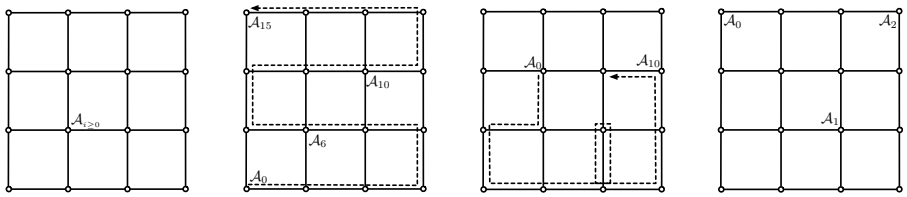
Location. A user $\mathcal{X}$ located in the space $\mathcal{S}^M$ at time k is denoted $\mathcal{X}_k$, with $k \in \mathbb{Z}_{\geq 0}$ and $\mathcal{X}_k \in \{0, 1, \dots, M-1\}$. The location of $\mathcal{X}$ at time 0 is called the *initial location of* $\mathcal{X}$. The set of possible values for $\mathcal{X}_{k+1}$ will be determined by the particular mobility pattern of the user.

2.2 Mobility Models

We now describe some common mobility patterns (see Fig. 1) to give a better intuition of our modeling of search spaces and descriptions of entities moving in the search space. We assume an honest user like Bob follows a realistic mobility pattern where subsequent locations are contiguous to each other. We will use mobility patterns and mobility models interchangeably throughout this paper. Two-dimensional search spaces can be projected to a one-dimensional space.

Static mobility model An entity $\mathcal{X}$ that follows a static mobility model starts at a random initial position r in the search space $\mathcal{S}^M$ and does not move afterwards. For example, a person who stays at home or in the office. With a fixed $r \in \mathcal{S}^M$:

$$\mathcal{X}_{k+1} = \mathcal{X}_k,\ \mathcal{X}_0 = r,\ k \in \mathbb{Z}_{\geq 0}$$



(a) Static (b) Linear (c) Random Walk (d) Random Jump

Fig. 1. Mobility Models

Linear mobility model An entity $\mathcal{X}$ that follows a linear mobility model starts at an arbitrary initial position $\mathcal{X}_0$ within the search space $\mathcal{S}^M$ ($M = mn$ for dimension two and $M = n$ for dimension one) and keeps moving such that the following conditions are satisfied:

1. $\{\mathcal{X}_{iM}, \mathcal{X}_{iM+1}, \ldots, \mathcal{X}_{iM+M-1}\} = \{0, 1, \ldots, M-1\}$ for all $i \in \mathbb{Z}_{\geq 0}$;
2. $\mathcal{X}_{k+1} \in \{\mathcal{X}_k + 1, \mathcal{X}_k - 1\}$ for a one dimensional search space of size n ($k \in \mathbb{Z}_{\geq 0}$).

This model may apply when a person is driving a car on the road.

Random walk mobility model An entity $\mathcal{X}$ that follows a random walk mobility model starts at a random position and decides its next move uniformly at random from all the positions in its vicinity. Hence the following mathematical expressions hold for a one-dimensional search space of size n (similar expressions can be derived for the two-dimensional case).

$$\Pr(\mathcal{X}_{k+1} = \mathcal{X}_k + 1) = \Pr(\mathcal{X}_{k+1} = \mathcal{X}_k - 1) = \frac{1}{2}, \text{ if } 1 \leq \mathcal{X}_k \leq n-2;$$
$$\Pr(\mathcal{X}_{k+1} = 1) = 1, \text{ if } \mathcal{X}_k = 0;$$
$$\Pr(\mathcal{X}_{k+1} = n-2) = 1, \text{ if } \mathcal{X}_k = n-1,$$

where $\Pr(\cdot)$ denotes probability hereinafter. Later, our theoretical probabilistic derivation will start with a random walk mobility model. A person sightseeing or shopping could fit this model.

Random jump An entity $\mathcal{X}$ that follows this strategy can arbitrarily move to any other position. In other words, the next position is sampled freshly from a uniform distribution. An attacker, for example, who can fake his/her location as a means to perform attacks, such as trilateration, could be described using the random jump model.

3 Location Proximity Attacks

Based on the modeling of search spaces and possible mobility patterns from Section 2, we proceed to give a formal mathematical description of the problem

statement for this paper. Afterwards, we will derive general analytical bounds on the expected location effort in dimension one.

3.1 System and Attacker Model

Let Alice ($\mathcal{A}$) be the attacker and Bob ($\mathcal{B}$) be a user whose location is of interest to Alice. Bob uses aLBS that will disclose Bob's presence at location $\mathcal{B}_k$ to other users that claim to be at the same location. Bob (and Alice) can only make one location claim per discrete time step k. Each time k, Bob will move once (and update its location on the LBS), and Alice will thus be able to perform one query to the LBS to verify if Bob is at Alice's claimed location $\mathcal{A}_k$. Alice sends the first query at time $k = 0$, and thus k is the time of the $(k + 1)$-th query.

The goal of Alice is to minimize the number of queries that she needs to send to LBS to be able to verify Bob's location. Conversely, Bob does not have a particular goal except to use the service privately. He is not even aware of being tracked. Alice on the other hand is assumed to have a priori information about Bob's probability distribution obtained from past observations, external sources, geographic features of the terrain or a combination thereof. Later in Section 6.2, we will discuss a real data example, where the attacker can obtain some information from historical trajectory data of a victim.

While we use a third party LBS in this system model for simplicity, similar scenarios could be constructed if Alice and Bob engage in a privacy-preserving proximity protocol that is initiated by Alice and where the inputs of both of them remain private (e.g., the protocol discussed in [8]).

3.2 Problem Statement and Formalization

We now present our problem statement and the underlying formalization.

Problem Statement. We are interested in finding $k_{O,p}$: the number of queries required by an optimal attacker strategy to locate Bob with probability of at least p. Formally, we define $k_{O,p}$ as follows:

$$k_{O,p} := \min_{\mathcal{A}} k_{\mathcal{A},p}, \tag{1}$$

with $k_{\mathcal{A},p}$ being the number of steps required by a specific attacker strategy $\mathcal{A}$ to locate Bob with probability $0 \leq p \leq 1$. We can find that number as follows:

$$k_{\mathcal{A},p} := \min\{k : \Pr(E_k) \geq p\},$$

with $\Pr(E_k)$ being the cumulative probability that Bob was located within k steps, that is, $k + 1$ queries.

Attacker Strategies. For a fixed attacker strategy $\mathcal{A} = \mathcal{A}_0, \mathcal{A}_1, \ldots$, we are interested in the probability of two events: E_k, and F_j. The event E_k is the event that Alice locates Bob within k steps:

$$E_k := \{\exists i \leq k \text{ s.t. } \mathcal{A}_i = \mathcal{B}_i\}$$

F_j is the event that Alice locates Bob exactly at step j:

$$F_j := \{\mathcal{A}_j = \mathcal{B}_j\}.$$

Before we can derive the probabilities of those events ($\Pr(E_k)$ and $\Pr(F_j)$), we now show how to compute the probabilistic locations of Bob.

Probabilistic Locations. Consider a search space $\mathcal{S}^M$. We assume the mobility model of Bob can be described by a transition matrix P where each entry of P is a transition probability p_{ij} representing the probability of Bob moving from location i to location j in one step. Furthermore, we assume the probability of Bob moving from location i to location j is the same at any step. More precisely:

$$\Pr(\mathcal{B}_k = j | \mathcal{B}_{k-1} = i) = p_{ij},\ \forall k \in \mathbb{Z}_{\geq 1} \text{ and } i, j \in \mathcal{S}^M.$$

Thus, it is straightforward to calculate the probability of Bob being at a particular location after k steps by simply taking the kth power of the transition matrix.

Let $B^{(k)}$ ($k \in \mathbb{Z}_{\geq 0}$) be a vector representing the probability of Bob being at each location j ($j \in \{0, 1, \ldots, M-1\}$) after k steps, i.e. $B_j^{(k)} = \Pr(\mathcal{B}_k = j)$. Assume that Bob is at location i after k steps, then $B^{(k+1)}$ ($k \in \mathbb{Z}_{\geq 0}$) can be calculated as follows:

$$B^{(k+1)} = B^{(k)} \cdot P = \overset{\qquad\quad i}{\begin{pmatrix} 0 & \ldots & 1 & \ldots & 0 \end{pmatrix}} \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,M} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ p_{M,1} & p_{M,2} & \cdots & p_{M,M} \end{pmatrix}$$

Upper Bound for $\Pr(E_k)$. By definition,

$$\Pr(E_k) = \Pr(F_0 \cup F_1 \cdots \cup F_k),$$

which gives the following *upper bound* for $\Pr(E_k)$ since the probability of finding Bob at step i is at most equal to the probability of Bob's most likely location at that step:

$$\Pr(E_k) \leq \Pr(F_0) + \Pr(F_1) + \cdots + \Pr(F_k) \leq \sum_{i=0}^{k} \max_j B_j^{(i)}. \tag{2}$$

Note that the above upper bound on $\Pr(E_k)$ holds for any attacker strategy $\mathcal{A}$.

Lower Bound on k. We define

$$k_{lower,p} := \min \left\{ k : \sum_{i=0}^{k} \max_j B_j^{(i)} \geq p \right\}. \tag{3}$$

In view of Equation 2, $k_{lower,p} \leq k_{O,p}$ is a *lower bound* of $k_{O,p}$.

4 Quantifying Attacker Effort

Next we present the theoretical analysis on the minimal number of steps required by an optimal attack strategy to locate a particular victim. First, we derive the formula for calculating $\Pr(E_k)$ and then we consider the case of a victim following a random walk mobility model. We choose a random walk model to give a rigorous mathematical analysis. In more complex moving patterns, the problem may not have analytical or closed solutions because the transition matrix is indefinable or needs to be recursively updated. Our mathematical derivation will be presented for the one-dimensional case only as we can always project two-dimensions to a one-dimensional space.

4.1 Attacker's Effort Computation

Given the transition matrix P and the vector $B^{(0)}$ of initial position probabilities, $B^{(k)} = B^{(0)}P^k$ gives the probabilities of Bob being at each different position after k steps. More precisely, $B_j^{(k)} = \Pr(\mathcal{B}_k = j)$ is the probability of Bob at position j after k steps.

Let $P^i_{j_1 j_2}$ denote the $(j_1, j_2)-$entry of the matrix P^i. It gives the probability of Bob going from position j_1 to position j_2 in i steps, i.e. for any $k \in \mathbb{Z}_{\geq 0}$,

$$\Pr(\mathcal{B}_{i+k} = j_2 | \mathcal{B}_k = j_1) = P^i_{j_1 j_2}.$$

Fixing an attacker strategy $\mathcal{A}$, let $\mathcal{A}_i$ denote the position of Alice at step i. For any positive integers $i_1 < i_2$, the probability of Alice locating Bob at both steps i_1 and i_2 is equal to the probability of Bob being at position $\mathcal{A}_{i_1}$ at step i_1 multiplied by the probability of Bob reaching position $\mathcal{A}_{i_2}$ in $i_2 - i_1$ steps, i.e.

$$\Pr(F_{i_1} \cap F_{i_2}) = \Pr(\mathcal{B}_{i_1} = \mathcal{A}_{i_1} \cap \mathcal{B}_{i_2} = \mathcal{A}_{i_2}) \tag{4}$$

$$= \Pr(\mathcal{B}_{i_1} = \mathcal{A}_{i_1}) \Pr(\mathcal{B}_{i_2} = \mathcal{A}_{i_2} | \mathcal{B}_{i_1} = \mathcal{A}_{i_1}) = B^{(i_1)}_{\mathcal{A}_{i_1}} P^{i_2 - i_1}_{\mathcal{A}_{i_1} \mathcal{A}_{i_2}}. \tag{5}$$

To get a general formula for $\Pr(E_k)$, we first note

$$\begin{aligned}
\Pr(E_k) &= \Pr(F_0 \cup F_1 \cup \cdots \cup F_k) = \Pr((F_0 \cup F_1 \cup \cdots \cup F_{k-1})^c \cap F_k) \\
&= \Pr(F_0) + \Pr(F_0^c \cap F_1) + \Pr((F_0 \cup F_1)^c \cap F_2) + \cdots \\
&\quad + \Pr((F_0 \cup F_1 \cup \cdots \cup F_{k-1})^c \cap F_k) \\
&= \sum_{0 \leq i \leq k} \Pr((F_0 \cup F_1 \cup \cdots \cup F_{i-1})^c \cap F_i),
\end{aligned} \tag{6}$$

where F^c is the complement of F, i.e., F_j^c means Alice does not successfully locates Bob at step j. Our GUAS attacker strategy uses this result for aggregate success computation (see Section 5.2). From probability theory we can also write

$$\begin{aligned}
\Pr(E_k) &= \Pr(F_0 \cup F_1 \cup \cdots \cup F_k) \\
&= \sum_{m=0}^{k} (-1)^{m+1} \left(\sum_{0 \leq i_1 < \cdots < i_m \leq k} \Pr(F_{i_1} \cap \cdots \cap F_{i_m}) \right).
\end{aligned} \tag{7}$$

By combining Equation 4 with Equation 7, we have the following general formula for $\Pr(E_k)$:

$$\sum_{m=0}^{k} B_{\mathcal{A}_m}^{(m)} \left(1 + \sum_{\ell=1}^{k-m} (-1)^{\ell} \sum_{m<i_1<i_2<\cdots<i_\ell\leq k} P_{\mathcal{A}_m\mathcal{A}_{i_1}}^{i_1-m} \cdots P_{\mathcal{A}_{i_{\ell-1}}\mathcal{A}_{i_\ell}}^{i_\ell-i_{\ell-1}}\right)$$
$$= B_{\mathcal{A}_0}^{(0)} \left(1 - \sum_{i=1}^{k} P_{\mathcal{A}_0\mathcal{A}_i}^{i} + \sum_{1\leq i_1<i_2\leq k} P_{\mathcal{A}_0\mathcal{A}_{i_1}}^{i_1} P_{\mathcal{A}_{i_1}\mathcal{A}_{i_2}}^{i_2-i_1} - \cdots\right)$$
$$+ B_{\mathcal{A}_1}^{(1)} \left(1 - \sum_{i=2}^{k} P_{\mathcal{A}_1\mathcal{A}_i}^{i-1} + \sum_{2\leq i_1<i_2\leq k} P_{\mathcal{A}_1\mathcal{A}_{i_1}}^{i_1-1} P_{\mathcal{A}_{i_1}\mathcal{A}_{i_2}}^{i_2-i_1} - \ldots\right) + \cdots + B_{\mathcal{A}_k}^{(k)}. \quad (8)$$

If the attacker strategy is not deterministic, then for one fixed sequence of positions $a_0, a_1, a_2, \ldots, a_k$, $\Pr(E_k|\mathcal{A}_0 = a_0 \cap \mathcal{A}_1 = a_1 \cap \cdots \cap \mathcal{A}_k = a_k)$ is given by

$$\sum_{m=0}^{k} B_{a_m}^{(m)} \left(1 + \sum_{\ell=1}^{k-m} (-1)^{\ell} \sum_{m<i_1<i_2<\cdots<i_\ell\leq k} P_{a_m a_{i_1}}^{i_1-m} P_{a_{i_1} a_{i_2}}^{i_2-i_1} \cdots P_{a_{i_{\ell-1}} a_{i_\ell}}^{i_\ell-i_{\ell-1}}\right),$$

and we have the following formula for $\Pr(E_k)$

$$\sum_{1\leq a_0,a_1,\ldots,a_k\leq n} \Pr(\mathcal{A}_0 = a_0 \cap \cdots \cap \mathcal{A}_k = a_k) \cdot \Pr(E_k|\mathcal{A}_0 = a_0 \cap \cdots \cap \mathcal{A}_k = a_k) \quad (9)$$

Note that when the attacker strategy is deterministic, for one sequence of positions $a_0, a_1, a_2, \ldots, a_k$ we have $\Pr(\mathcal{A}_0 = a_0, \ldots, \mathcal{A}_k = a_k) = 1$ and we get the formula in (8).

For the next subsection, we assume Bob follows a random walk strategy that can be represented as a Markovian process with transition probabilities p_{ij}, consistent with the provisions of the random walk mobility model in Section 2.2. We give our evaluations for the case $p = 0.5$. To simplify the notations, we define

$$k_{\mathcal{A}} := k_{\mathcal{A},0.5} = \min\{k : \Pr(E_k) \geq 0.5\},$$
$$k_O := k_{O,0.5} = \min_{\mathcal{A}} k_{\mathcal{A}},$$
$$k_{lower} := k_{lower,0.5}.$$

Thus if Alice follows strategy $\mathcal{A}$, $k_{\mathcal{A}}$ (resp. $k_{\mathcal{A}} + 1$) is the number of steps (resp. number of queries) needed for Alice to locate Bob with a probability of at least 0.5. k_O (resp. $k_O + 1$) is the minimum number of steps (resp. minimum number of queries) needed for Alice to locate Bob with a probability of at least 0.5 independently of strategy used. In addition, $k_{lower} \leq k_O$ is a lower bound of k_O.

4.2 Random Walk Example

In this section, we derive upper and lower bounds on k_O by analyzing the matrices $B^{(k)}$. We show that "for a search space of size n, $\lfloor \frac{n}{3} \rfloor - 1 \leq k_O \leq \lfloor \frac{n}{2} \rfloor$" (see Corollary 1). To achieve this goal, we first estimate the values of $B_j^{(k)}$. For a one-dimensional search space of size n, Bob follows a transition matrix P with initial position vector $B^{(0)}$:

$$B^{(0)} = \left[\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right], \quad P = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & \ldots & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \ldots & 1 & 0 \end{bmatrix}$$

Then the probabilities of Bob in each position at step i is given by $B^{(i)} = B^{(0)}P^i$:

$$\begin{aligned} B^{(1)} &= \left[\frac{1}{2n}, \frac{3}{2n}, \frac{1}{n}, \ldots, \frac{1}{n}, \frac{3}{2n}, \frac{1}{2n}\right], \\ B^{(2)} &= \left[\frac{3}{4n}, \frac{1}{n}, \frac{5}{4n}, \frac{1}{n}, \ldots, \frac{1}{n}, \frac{5}{4n}, \frac{1}{n}, \frac{3}{4n}\right], \\ B^{(3)} &= \left[\frac{1}{2n}, \frac{11}{8n}, \frac{1}{n}, \frac{9}{8n}, \frac{1}{n}, \ldots, \frac{1}{n}, \frac{9}{8n}, \frac{1}{n}, \frac{11}{8n}, \frac{1}{2n}\right], \\ B^{(4)} &= \left[\frac{11}{16n}, \frac{1}{n}, \frac{5}{4n}, \frac{1}{n}, \frac{17}{16n}, \frac{1}{n}, \ldots, \frac{1}{n}, \frac{17}{16n}, \frac{1}{n}, \frac{5}{4n}, \frac{1}{n}, \frac{11}{16n}\right], \end{aligned} \tag{10}$$

We have the following observation:

Lemma 1 *For $k \in \mathbb{Z}_{\geq 0}$,*

$$\begin{cases} \frac{1}{n} \leq B_j^{(k)} \leq \frac{3}{2n} & 1 \leq j \leq n-2 \\ \frac{1}{2n} \leq B_j^{(k)} \leq \frac{3}{4n} & j = 0, n-1 \end{cases}.$$

Proof. We prove the above claims by mathematical induction. For $B^{(0)}$, the claim is true. We assume it is true for $B^{(k)}$, then for $B^{(k+1)}$

1. $2 \leq j \leq n-3$, $B_j^{(k+1)} = \frac{1}{2}\left(B_{j-1}^{(k)} + B_{j+1}^{(k)}\right)$, by induction hypothesis

$$\frac{1}{n} = \frac{1}{2}\left(\frac{1}{n} + \frac{1}{n}\right) \leq B_j^{(k+1)} \leq \frac{1}{2}\left(\frac{3}{2n} + \frac{3}{2n}\right) = \frac{3}{2n}.$$

2. $j = 1, n-2$, $B_1^{(k+1)} = B_0^{(k)} + \frac{1}{2}B_2^{(k)}$, by induction hypothesis

$$\frac{1}{n} = \frac{1}{2n} + \frac{1}{2}\cdot\frac{1}{n} \leq B_2^{(k+1)} \leq \frac{3}{4n} + \frac{1}{2}\cdot\frac{3}{2n} = \frac{3}{2n}; \quad \frac{1}{n} \leq B_{n-2}^{(k+1)} \leq \frac{3}{2n}.$$

3. $j = 0, n-1$, $B_0^{(k+1)} = B_1^{(k)} \frac{1}{2}$, by induction hypothesis

$$\frac{1}{2n} = \frac{1}{2} \cdot \frac{1}{n} \leq B_1^{(k+1)} \leq \frac{1}{2} \cdot \frac{3}{2n} = \frac{3}{4n}; \quad \frac{1}{2n} \leq B_{n-1}^{(k+1)} \leq \frac{3}{4n}.$$

The above bounds on $B_j^{(k)}$ helps us to get upper and lower bounds on k_O. Next we show a lower bound of k_{lower}, which gives a lower bound for k_O.

Proposition 1

$$k_O \geq k_{lower} \geq \lfloor \frac{n}{3} \rfloor - 1.$$

Proof. By Lemma 1, for any k,

$$\sum_{i=0}^{k} \max_j B_j^{(i)} \leq \frac{3}{2n} \cdot (k+1) = \frac{3k+3}{2n}.$$

If $k < \lfloor \frac{n}{3} \rfloor - 1$, $\sum_{i=0}^{k} \max_j B_j^{(i)} < \frac{1}{2}$. Thus by definition, we must have $k_{lower} \geq \lfloor \frac{n}{3} \rfloor - 1$.

By the definition of k_O, for any given attacker strategy $\mathcal{A}$, $k_O \leq k_{\mathcal{A}}$. Next, we construct a specific attacker strategy $\mathcal{A}_{jp}$ and prove an upper bound for $k_{\mathcal{A}_{jp}}$, to obtain an upper bound for k_O.

Lemma 2 *For any $n \geq 4$, there exists a strategy $\mathcal{A}_{jp}$ such that $k_{\mathcal{A}_{jp}} \leq \lfloor \frac{n}{2} \rfloor$.*

Proof. First we notice that $P_{ij}^{\ell} = 0$ if $j - i > \ell$ due to our construction of the random walk P.

1. Let $n \geq 4$ be even, consider the attacker strategy $\mathcal{A}_{jp}$ with the first $\frac{n}{2}$ positions given by $\mathcal{A}_0 = 0, \mathcal{A}_1 = 2, \mathcal{A}_2 = 4, \ldots, \mathcal{A}_m = 2m, \ldots, \mathcal{A}_{\frac{n}{2}-1} = n-2$. For all $0 \leq i < j \leq \frac{n}{2} - 1$,

   $$\mathcal{A}_j - \mathcal{A}_i = 2(j-i) > j - i.$$

   Then for $0 \leq i < j \leq \frac{n}{2} - 1$,

   $$\begin{aligned} \Pr(F_i \cap F_j) &= \Pr(\mathcal{B}_i = \mathcal{A}_i \text{ and } \mathcal{B}_j = \mathcal{A}_j) = \Pr(\mathcal{B}_i = \mathcal{A}_i) \Pr(\mathcal{B}_j = \mathcal{A}_j | \mathcal{B}_i = \mathcal{A}_i) \\ &= B_{\mathcal{A}_i}^{i} P_{\mathcal{A}_i \mathcal{A}_j}^{j-i} = 0. \end{aligned}$$

   Together with Lemma 1 we have

   $$\Pr\left(E_{\frac{n}{2}-1}\right) = \sum_{i=0}^{\frac{n}{2}-1} \Pr(F_i) = B_0^{(0)} + B_2^{(1)} + \cdots + B_{n-2}^{\left(\frac{n}{2}-1\right)} \geq \sum_{i=0}^{\frac{n}{2}-1} \frac{1}{n} = \frac{1}{n} \frac{n}{2} = \frac{1}{2}.$$

   Hence $k_{\mathcal{A}_{jp}} \leq \frac{n}{2} - 1$.

2. Let $n \geq 5$ be odd, consider the attacker strategy $\mathcal{A}_{jp}$ with the first $\lfloor \frac{n}{2} \rfloor + 1$ positions given by $\mathcal{A}_0 = 0, \mathcal{A}_1 = 2, \mathcal{A}_2 = 4, \ldots, \mathcal{A}_m = 2m, \ldots, \mathcal{A}_{\lfloor \frac{n}{2} \rfloor} = n-1$. Similarly we can prove $\Pr(F_i \cap F_j) = 0$ for $0 \leq i < j \leq \lfloor \frac{n}{2} \rfloor$. Together with Lemma 1 we have

$$\begin{aligned}\Pr\left(E_{\lfloor \frac{n}{2} \rfloor}\right) &= \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} \Pr(F_i) = B_0^{(0)} + B_2^{(1)} + \cdots + B_{n-1}^{\left(\lfloor \frac{n}{2} \rfloor\right)} \\ &\geq \frac{1}{2n} + \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor - 1} \frac{1}{n} = \frac{1}{n}\left(\left\lfloor \frac{n}{2} \right\rfloor + \frac{1}{2}\right) = \frac{1}{2},\end{aligned}$$

and hence $k_{\mathcal{A}_{jp}} \leq \lfloor \frac{n}{2} \rfloor$.

Recall the notation:

$$k_{\mathcal{A}} = \min\{k : \Pr(E_k) \geq \frac{1}{2}\}, \qquad k_O = \min_{\mathcal{A}} k_{\mathcal{A}}$$

Corollary 1 *For a one-dimensional search space of size n, $\lfloor \frac{n}{3} \rfloor - 1 \leq k_O \leq \lfloor \frac{n}{2} \rfloor$.*

We used the bounds in Lemma 1 to approximate $\sum_{i=0}^{k} \max_{0 \leq j \leq n-1} B_j^{(i)}$ in Proposition 1. We also derived the lower bound of k_{lower}, which then gave lower bounds on k_O. From Equation (10), we can see $\max_{0 \leq j \leq n-1} B_j^{(k)}$ decreases when k increases, which means the upper bounds on $B_j^{(k)}$ derived in Lemma 1 can be tightened for larger values of k. Thus we expect the lower bound for k_O in Corollary 1 to be higher than $\lfloor \frac{n}{3} \rfloor - 1$.

Similarly, the upper bound $\lfloor \frac{n}{2} \rfloor$ for k_O was derived through a lower bound on $B_j^{(k)}$. However, from Equation (10) we see that $B_j^{(k)}$ can achieve higher values than the lower bounds, which suggests that the specific attacker strategy $\mathcal{A}_{jp}$ described in Lemma 2, $k_{\mathcal{A}_{jp}}$ is strictly smaller than $\lfloor \frac{n}{2} \rfloor$. These observations motivate the calculations of exact values of $\sum_{i=0}^{k} \max_{0 \leq j \leq n-1} B_j^{(i)}$ and $k_{\mathcal{A}_{jp}}$, which yield tighter bounds on k_O.

5 Attacker Strategies

In the previous section we derived theoretical bounds for the optimal attacker strategy. Unfortunately, the lower bound estimate does not help to find the optimal strategy. Conversely, the upper bound provides us with a constructive strategy, which we briefly discuss in this section and name it the *Linear Jumping Strategy* (LJS). We also propose a greedy strategy, which we call the *Greedy Updating Attack Strategy* (GUAS), which leverages estimations of Bob's positions to determine next queries.

5.1 Linear Jumping Strategy

The linear jumping strategy sequentially selects every second location in B. Given uniform initial distributions and a random walk mobility model of the victim, it is expected to meet the upper bound cost as discussed in the previous section. The strategy was implemented by us for evaluation on different cases. Algorithm 1 in Appendix A summarizes the implementation in pseudocode. The runtime of LJS is linear in n.

5.2 Greedy Updating Attack Strategy

While we could implement an exhaustive search to find optimal strategies for any given setting, that approach is impractical and computationally expensive. In particular, for n locations, we need to evaluate the $k_{\mathcal{A},p}$ for $n^{\lceil \frac{n}{2} \rceil}$ different strategies (n potential choices per step, we expect to be done within $\lceil \frac{n}{2} \rceil$ steps). This is infeasible even for moderate values of n. Instead, we introduce a (locally) greedy strategy which is computationally cheaper and allows us to perform simulations on a larger range of settings. We call it the *Greedy Updating Attack Strategy* (GUAS).

Suppose Alice has some assumed initial location of Bob $\tilde{B}^{(0)}$ (for example, Bob's actual initial location distribution $B^{(0)}$), and transition matrix P. At each time step i, Alice keeps her current estimate of Bob's location as $\tilde{B}^{(i)}$. In particular, Alice can use $\tilde{B}^{(i)}$ to keep track of locations she checked previously, and found to be empty. That means that $\tilde{B}^{(i)}$ depends on the actual choices of the attacker, while $B^{(i)}$ just depends on $B^{(0)}$ and P.

In query $(i+1)$, Alice checks location $\max_j \tilde{B}^{(i)}_j$, that is, the most likely location of Bob at that time[7]. If Alice succeeds, then we are done; if Alice does not find Bob, she updates $\tilde{B}^{(i)}$ by setting the probability of the location checked in the current query to be 0 and re-normalizing $\tilde{B}^{(i)}$ to ensure that $\sum \tilde{B}^i = 1$. Thus, the following values of $\tilde{B}^{(i+1)}$ will be computed under the condition that the victim was not at the location tested in query i and earlier. The algorithm is described as pseudocode in Algorithm 2 in Appendix A. The runtime of the GUAS is quadratic in n: for each of the up to $\lceil \frac{n}{2} \rceil$ queries, we need to find the minimal value of $\tilde{B}^{(i-1)}$ (which is of size n). Thus, we were able to run this strategy for a search space size $n = 2000$ without issues.

In Section 6, we evaluate both LJS and GUAS for a random walk transition pattern and a sparse high-dimensional transition matrix derived from a real dataset. For the real dataset, we also investigate the attacker's performance if $\tilde{B}^{(0)} \neq B^{(0)}$, i.e., attacker has no information about victim's initial location distribution.

[7] Bob's location vector is B^i in Alice's (i+1)-th query, where i starts from 0.

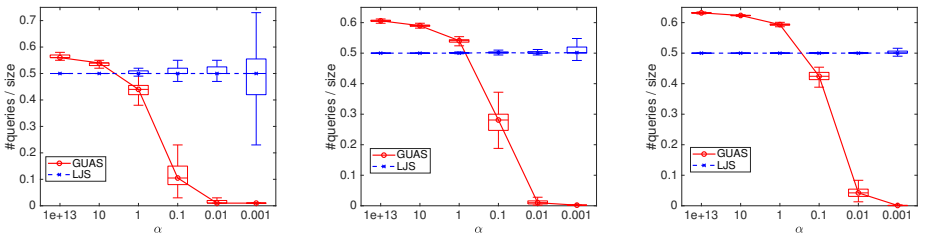
6 Evaluation

Based on the assumption that the attacker is aware of the victims probability distribution for initial location and the transition matrix, we simulated LJS and GUAS for different values of the initial location of Bob and the transition matrix. In addition, we tested our strategy on a transition matrix derived from a real dataset. All experiments were performed on a Core i3-4005U CPU@1.70GHz with 8GB RAM.

6.1 Random Walk Transition Matrix

In this experiment, we used the transition matrix that represents a random walk (i.e., same P as in Section 4.2) and performed simulations with various initial distributions for Bob, $B^{(0)}$, and for different search spaces with sizes varying from 100 to 2000. We assume the attacker knows the initial distribution of the victim.

Specifically, we consider space sizes of $100, 500$ and 2000. We run 100 simulations for each of these sizes and represent the initial location probability distribution for $B^{(0)}$ by a Dirichlet distribution with concentration parameter α. With increasing α values, $B^{(0)}$ is *closer* to being uniformly distributed, so a positive α value of almost zero indicates an initial distribution with very high likelihood in one location, and almost zero in all others. The results represent the effort of the attacker to successfully locate Bob with a probability of 0.5 for GUAS and LJS strategies under various initial distributions for $B^{(0)}$ regardless of the search space size. In Fig. 2 we provide a box-plot representation of some representative results (see Table 3 in Appendix B for full results). The upper and lower borders of the box represent the upper and lower quartiles, and the bar in the box is the median. The upper and lower whiskers represent the maximum and minimum number of queries in all simulations, excluding outliers, which are more than 1.5 times beyond the upper and lower quartiles.



(a) Search space size = 100 (b) Search space size = 500 (c) Search space size = 2000

Fig. 2. GUAS and LJS results for random walk mobility for different search space sizes with success probability of 0.5.

We observed that with a large concentration parameter α, which leads to an *almost* uniform distribution of the initial position, LJS achieves the optimal lower bound, while GUAS is unable to do so. With decreasing concentration parameter value, the initial distribution becomes less uniformly distributed. Assuming that the attacker is aware of the exact initial distribution, GUAS becomes more effective for such non-uniform initial distribution. This is true regardless of the size of the search space, as shown in Fig. 2.

6.2 Real-World Mobility Pattern

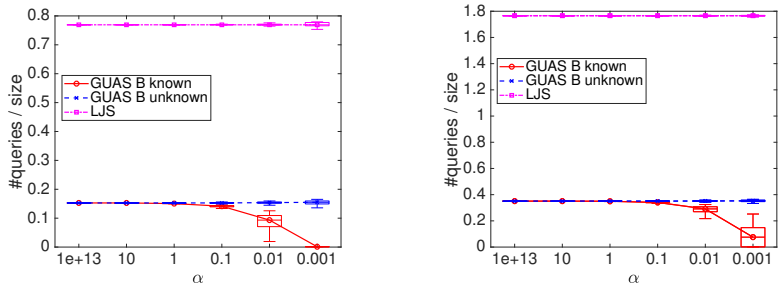
We also performed simulations with a transition matrix we derived from the T-Drive dataset [21,22] released by Microsoft. The dataset comprises GPS trajectories of 10357 taxis between Feb. 2 and Feb. 8 2008 within the city of Beijing, around 15 million points and a total trajectories distance of sums up to 9 million kilometers. The average sampling interval is around 177 seconds with a distance of 623 meters.

In our simulation, we take a subset of data within the third ring of Beijing. We discretize that area in grid cells of 500 meters $\times$ 500 meters, which corresponds to a lat-long unit of around 0.005. This results in 884 different locations. The area is mapped into a one-dimensional space using the following projection:

$$proj(i, j) = j \times n + i,$$

where (i, j) represents the coordinate of a grid cell in lat-long plane, and n is the total number of partitions in longitude. Based on the aforementioned discretization and projection method, we compute the probability of transitioning from one grid cell to another and build up the aggregated transition matrix, P_{taxi}, in the following way. For each taxi, we check which grid cell the taxi is moving to by reading the data in chronological order on a daily basis. Whenever there is a transit from, say location (or grid cell) x to y, we record this transit. After processing and recording all the transits for the first taxi, we can normalize the resulting matrix and build up a transition matrix only for this taxi. We do this for all the taxis, aggregate all the transition matrices and then normalize them to arrive at P_{taxi}.

Results. We simulated different "true" initial distributions for the victim, and checked on both the cases where the attacker does not know and knows about Bob's initial distribution. If the attacker is unaware of the initial distributions for the victim, she assumes a uniform initial distribution. See Fig. 3a and 3b for results calculated over 100 simulations. In these cases, the success is determined by how close the victim's distribution is to uniform. While for larger values of α, the attacker's guess is close enough to not decrease performance significantly. If the victim's initial distribution is less random ($\alpha \leq 0.01$), the attacker without knowledge on the initial distribution will perform worse, while the attacker with knowledge on initial B can leverage this in the GUAS strategy to decrease the required number of guesses.



(a) Performance with p=0.5

(b) Performance with p=0.8

Fig. 3. Attack strategies performance on T-Drive dataset for unknown and known initial distributions with success probabilities of 0.5 and 0.8.

Comparing performance of GUAS and LJS in Fig. 3a and Fig. 3b, we observe that for a less structured/dynamic mobility pattern, which for example could be modeled by some highly sparse transition matrix, GUAS performs better than LJS. Our interpretation of this result is that a realistic transition matrix leads to a set of locations that are significantly more likely than others. For LJS, it does not matter if the attacker knows the initial distribution since LJS does not utilize knowledge on the victim. See Table 1 and 2 in Appendix B for detailed results.

7 Related Work

An extensive body of research has been devoted to describing and understanding mobility models for mobile ad hoc and other types of networks, such as vehicular networks [3,10]. Typically, these papers describe mobility models where mobile nodes are independent of or dependent on each other, namely entity mobility models or group mobility models. The goal is usually to study the behavior of individual entity mobility models and help researchers decide which model is most suitable. Random mobility models are the models of choice of most authors.

Similarly, other authors have studied the *Rendezvous Problem* [20], which consists of finding an optimal strategy for two or more mobile entities who are unaware of each others' location, to meet. This problem and some slight variations of it has attracted much attention from the research community because of its potential application to many engineering problems like the one we are considering in this paper. However, as far as we are concerned, this problem remains open [1,4]. The main difference between rendezvous search problems and the one we are tackling in this paper is that rather than having two entities trying to find each other, here one of the entities is a victim that might not even aware of the existence of the attacker trying to find him/her.

Another line of work studies the probability for n independent entities to meet while they all follow random walk trajectories in dimension two [7,14]. For $n = 2$,

this is related to the special case when Alice chooses a random walk strategy. The difference is that we assume the entities take a step after every fixed length of time interval, while in the aforementioned papers they consider an entity takes a step after a time interval whose length follows a Poisson distribution.

In the realm of location privacy, the effort an attacker needs to locate a victim has been studied in different settings [13,9,16,17]. In [13,9], the focus was on static victim. A particular attacker model was used against moving target in [9]. Although this attacker model was shown to be optimal for a static victim, its efficiency as an attacker model for moving victim was not discussed. This is precisely the focus of our paper.

In [16,17], the focus is on the quantification of users' location privacy by analyzing location-based applications and location-privacy preserving mechanism (LPPM). They assume an obfuscated trace (obtained by an LPPM) of the victim is available to the attacker and the goal of the attacker is to find out the real trace [17] or the real location [16] of the victim. In contrast, in this paper the attacker is only aware of the mobility model of the victim, without any knowledge of the victim's trajectory.

8 Conclusions

In this paper, we have presented a framework to reason about the expected effort of attackers attempting to locate moving targets using proximity testing. We first provide mathematical analysis for asymptotic bounds (on the search space) on the best attacker strategies under a random walk mobility model for the moving victim. We then derive two concrete strategies, and evaluate their performance over a range of parameters. The LJS strategy is found to work well for random walk mobility and close to uniform initial distribution of Bob, while the GUAS strategy requires less queries for less uniform initial distributions of Bob (which are known to the attacker). We then derive a realistic mobility model from a real dataset consisting of spatio-temporal trajectory data, and analyze the performance of our strategies. In that setting, we find that the GUAS strategy consistently requires less than $\frac{N}{6}$ queries ($p = 0.5$), while LJS requires more than $0.75N$, where N is the search space size. Summarizing, we have shown theoretically and practically that (using a strategy suitable to the setting), an attacker is able to locate a victim with 50% probability with at most $\frac{N}{2}$ steps. For example, using GUAS in our Beijing dataset the attacker could localize a victim with 50% probability in 134 steps (6.6 hours), within an area of $0.25km^2$.

Proposed countermeasures. To prevent the demonstrated attack, we propose the following countermeasures: a) The LBS could probabilistically return a wrong result (e.g., with 20% chance), b) the LBS could verify that the sequence of attacker's location claims confirms to some assumed transition matrix P (i.e., the likelihood of a claimed trajectory is above some threshold τ), and c) the LBS could impose limitations on the number of queries or the speed/frequency of queries by the requester. We leave the evaluation of these countermeasures for future work.

Acknowledgements

Xueou was supported by SUTD-ZJU grant ZJUSP1600102. R. Rios was partially funded by the Spanish Ministry of Economy and Competitiveness (TIN2016-79095-C2-1-R, TIN2014-54427-JIN) and the 'Captación de Talento para la Investigación' fellowship from University of Malaga. This work was partly funded by the Swedish Foundation for Strategic Research (SSF) and the Swedish Research Council (VR).

References

1. Alpern, S.: Ten Open Problems in Rendezvous Search, pp. 223–230. Springer New York, New York, NY (2013). https://doi.org/10.1007/978-1-4614-6825-7_14
2. Bai, F., Helmy, A.: A survey of mobility models. Wireless Adhoc Networks, University of Southern California, USA (2004)
3. Camp, T., Boleng, J., Davies, V.: A survey of mobility models for ad hoc network research. Wireless Communications and Mobile Computing **2**(5), 483–502 (2002). https://doi.org/10.1002/wcm.72
4. Chen, L., Bian, K.: The telephone coordination game revisited: From random to deterministic algorithms. IEEE Transactions on Computers **64**(10), 2968–2980 (Oct 2015). https://doi.org/10.1109/TC.2015.2389799
5. Cuellar, J., Ochoa, M., Rios, R.: Indistinguishable regions in geographic privacy. In: Proceedings of the ACM Symposium on Applied Computing (SAC). pp. 1463–1469. SAC '12, ACM, New York, NY, USA (2012). https://doi.org/10.1145/2245276.2232010
6. Culzac, N.: Egypt's police 'using social media and apps like Grindr to trap gay people' (2014), article on The Independent
7. Gaudillière, A.: Collision probability for random trajectories in two dimensions. Stochastic Processes and their Applications **119**(3), 775–810 (2009). https://doi.org/http://dx.doi.org/10.1016/j.spa.2008.04.007
8. Hallgren, P.A., Ochoa, M., Sabelfeld, A.: Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol. In: Proceedings of the Annual Conference on Privacy, Security and Trust (PST). pp. 1–6 (2015). https://doi.org/10.1109/PST.2015.7232947
9. Hallgren, P.A., Ochoa, M., Sabelfeld, A.: Maxpace: Speed-constrained location queries. In: Proceedings of IEEE Conference on Communications and Network Security (CNS). pp. 136–144 (2016). https://doi.org/10.1109/CNS.2016.7860479
10. Harri, J., Filali, F., Bonnet, C.: Mobility models for vehicular ad hoc networks: a survey and taxonomy. IEEE Communications Surveys Tutorials **11**(4), 19–41 (Fourth 2009). https://doi.org/10.1109/SURV.2009.090403
11. Huang, M.S., Narayanan, R.M.: Trilateration-based localization algorithm using the lemoine point formulation. IETE Journal of Research **60**(1), 60–73 (2014)
12. Narayanan, A., Thiagarajan, N., Lakhani, M., Hamburg, M., Boneh, D.: Location privacy via private proximity testing. In: Proceedings of the Network and Distributed System Security Symposium (NDSS) (2011)
13. Polakis, I., Argyros, G., Petsios, T., Sivakorn, S., Keromytis, A.D.: Where's Wally?: Precise user discovery attacks in location proximity services. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS). pp. 817–828 (2015). https://doi.org/10.1145/2810103.2813605

14. Puchala, Z., Rolski, T.: The exact asymptotic of the time to collision. Electronic Journal of Probability **10**, 1359–1380 (2005)
15. Sedenka, J., Gasti, P.: Privacy-preserving distance computation and proximity testing on earth, done right. In: Proceedings of ACM Symposium on Information, Computer and Communications Security (ASIACCS). pp. 99–110 (2014). https://doi.org/10.1145/2590296.2590307
16. Shokri, R., Theodorakopoulos, G., Danezis, G., Hubaux, J.P., Le Boudec, J.Y.: Quantifying location privacy: The case of sporadic location exposure. In: Proceedings of the International Conference on Privacy Enhancing Technologies (PETS). pp. 57–76. Springer-Verlag, Berlin, Heidelberg (2011)
17. Shokri, R., Theodorakopoulos, G., Le Boudec, J.Y., Hubaux, J.P.: Quantifying location privacy. In: Proceedings of the IEEE Symposium on Security and Privacy (S&P). pp. 247–262. IEEE Computer Society, Washington, DC, USA (2011). https://doi.org/10.1109/SP.2011.18
18. Shokri, R., Theodorakopoulos, G., Troncoso, C., Hubaux, J.P., Le Boudec, J.Y.: Protecting location privacy: Optimal strategy against localization attacks. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS). pp. 617–627. ACM, New York, NY, USA (2012). https://doi.org/10.1145/2382196.2382261
19. Veytsman, M.: How I was able to track the location of any Tinder user (Feb 2014), `http://blog.includesecurity.com/2014/02/how-i-was-able-to-track-location-of-any.html`
20. Weber, R.: Optimal symmetric rendezvous search on three locations. Math. Oper. Res. **37**(1), 111–122 (Feb 2012). https://doi.org/10.1287/moor.1110.0528
21. Yuan, J., Zheng, Y., Xie, X., Sun, G.: Driving with knowledge from the physical world. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 316–324. ACM (2011)
22. Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., Huang, Y.: T-drive: driving directions based on taxi trajectories. In: Proceedings of the SIGSPATIAL International conference on advances in geographic information systems. pp. 99–108. ACM (2010)

A Pseudocodes of Algorithms

The *Linear Jump Strategy* (LJS) is given as pseudocode in Algorithm 1, and the *Greedy Updating Attacker Strategy* (GUAS) is provided in Algorithm 2.

B Simulation Results

In Table 1, 2, and 3 we provide the exact simulation results used for plotting Fig. 2 and 3. In these tables, α is the Dirichlet concentration parameter, size represents the number of locations in the search space and "#Q" stands for number of queries. The mean ("#Q.mean"), standard deviation ("#Q.std"), min ("#Q.min") and max ("#Q.max") are calculated over 100 simulations. The value "$\frac{\#\text{Q.mean}}{\text{Size}}$" represents how the strategy performs under different initial position distributions regardless of the search space size.

Algorithm 1: Linear Jumping Strategy (LJS)

Result: Number of queries that Alice needs to perform to locate Bob with probability of at least 0.5.

Set $success = 0$;

while $i < MAX\_QUERIES$ **do**

  $\mathcal{A}_i = 2 * i$; `// linear jumps through B`

  $success = success + (1 - success) * B_{\mathcal{A}_i}^{(i-1)}$;

  if $success \geq 0.5$ **then**

    return i; `// Minimal number of steps found`

  end

  $B_{\mathcal{A}_i}^{(i-1)} = 0$; `// Set this position empty for following calculations`

  $B^{(i-1)} = \text{normalize}(B^{(i-1)})$;

  $B^i = B^{(i-1)} \cdot P$; `// Bob's location probability in next query`

  $i = i + 1$;

end

return ERROR; `// Attacker was unable to locate Bob within MAX_QUERIES`

Algorithm 2: Greedy Updating Attack Strategy (GUAS)

Result: Number of queries that Alice needs to perform to locate Bob with probability of at least 0.5.

Initialize $\tilde{B}$, set $success = 0$;

while $i < MAX\_QUERIES$ **do**

  $\mathcal{A}_i = \max_j \tilde{B}_j^{(i-1)}$; `// select maximum likelihood estimate of B`

  $success = success + (1 - success) * B_{\mathcal{A}_i}^{(i-1)}$;

  if $success \geq 0.5$ **then**

    return i; `// Minimal number of steps found`

  end

  $B_{\mathcal{A}_i}^{(i-1)} = 0$; `// Set this position empty for following calculations`

  $\tilde{B}_{\mathcal{A}_i}^{(i-1)} = 0$; `// Attacker estimates this position empty`

  $B^{(i-1)} = \text{normalize}(B^{(i-1)})$;

  $\tilde{B}^{(i-1)} = \text{normalize}(\tilde{B}^{(i-1)})$;

  $B^i = B^{(i-1)} \cdot P$; `// Bob's actual location probability in next query`

  $\tilde{B}^i = \tilde{B}^{(i-1)} \cdot P$; `// Bob's estimated location probability in next query`

  $i = i + 1$;

end

return ERROR; `// Attacker was unable to locate Bob within MAX_QUERIES`

Table 1. GUAS performance on T-Drive dataset with known and unknown initial distribution for probability of 0.5 and 0.8.

		Unknown						Known				
	α	Size	#Q.mean	$\frac{\#Q.mean}{Size}$	#Q.std	#Q.min	#Q.max	#Q.mean	$\frac{\#Q.mean}{Size}$	#Q.std	#Q.min	#Q.max
p=0.5	1e+13	884	135	0.1527	0	135	135	135	0.1527	0	135	135
	10	884	134.93	0.1526	0.2564	134	135	134.54	0.1522	0.5009	134	135
	1	884	134.83	0.1525	0.5515	134	136	133.24	0.1507	0.6215	132	134
	0.1	884	134.9	0.1526	1.2432	132	138	124.9	0.1413	2.6874	118	129
	0.01	884	135.33	0.1531	3.7526	123	141	78.39	0.0887	21.7004	2	111
	0.001	884	135.25	0.1530	9.028	81	146	3.3	0.0037	7.0245	1	48
p=0.8	1e+13	884	310	0.3507	0	310	310	310	0.3507	0	310	310
	10	884	310.02	0.3507	0.1407	310	311	310	0.3507	0	310	310
	1	884	310.15	0.3508	0.5573	309	311	308.65	0.3492	0.5573	307	310
	0.1	884	310.32	0.3510	1.2624	307	313	300.34	0.3400	2.6370	293	304
	0.01	884	310.75	0.3515	3.7400	298	317	253.65	0.2869	22.1630	167	287
	0.001	884	310.7	0.3515	9.0403	256	322	76.01	0.0860	70.8871	1	223

Table 2. LJS performance on T-Drive dataset for probability of 0.5 and 0.8

	$p = 0.5$						$p = 0.8$				
α	Size	#Q.mean	$\frac{\#Q.mean}{Size}$	#Q.std	#Q.min	#Q.max	#Q.mean	$\frac{\#Q.mean}{Size}$	#Q.std	#Q.min	#Q.max
1e+13	884	680	0.7692	0	680	680	1561	1.7658	0	1561	1561
10	884	679.98	0.7692	0.1407	679	680	1560.96	1.7658	0.1969	1560	1561
1	884	679.85	0.7691	0.4794	678	682	1560.81	1.7656	0.4191	1559	1561
0.1	884	680.19	0.7694	1.0982	677	682	1560.83	1.7656	0.6522	1558	1562
0.01	884	680.48	0.7698	5.2445	641	687	1560.14	1.7649	5.1247	1517	1564
0.001	884	681.43	0.7708	5.0317	666	689	1560.44	1.7652	4.0310	1544	1566

Table 3. GUAS and LJS results for different search space sizes and Bob's initial distributions with success probability of 0.5

		GUAS					LJS				
α	Size	#Q.mean	$\frac{\#Q.mean}{Size}$	#Q.std	#Q.min	#Q.max	#Q.mean	$\frac{\#Q.mean}{Size}$	#Q.std	#Q.min	#Q.max
1e+13	100	56.13	0.5613	0.7740	54	58	50	0.5	0	50	50
10	100	53.79	0.5379	0.6860	52	56	50.04	0.5004	0.1969	50	51
1	100	44	0.44	2.5898	33	50	50.43	0.5043	1.1304	47	53
0.1	100	11.5	0.115	4.5115	3	23	50.49	0.5049	3.2114	32	58
0.01	100	1.42	0.0142	0.7272	1	4	50.96	0.5096	10.5523	5	87
0.001	100	1.00	0.01	0	1	1	49.54	0.4954	20.0668	1	95
1e+13	500	302.68	0.6054	2.0345	298	306	250	0.5	0	250	250
10	500	295.32	0.5906	1.7401	290	299	250.1	0.5002	0.3015	250	251
1	500	269.83	0.5397	3.3183	262	277	250.28	0.5006	0.8538	248	252
0.1	500	136.36	0.2727	22.1622	79	186	250.7	0.5014	2.8727	240	260
0.01	500	6.54	0.0131	5.2655	1	25	249.87	0.4997	16.3130	138	290
0.001	500	1.1	0.0022	0.3015	1	2	261.05	0.5221	43.3281	105	456
1e+13	2000	1263.5	0.6318	2.9763	1255	1270	1000	0.5	0	1000	1000
10	2000	1246.38	0.6232	2.9432	1238	1253	1000.1	0.5001	0.3015	1000	1001
1	2000	1186.99	0.5935	5.5405	1173	1202	1000.39	0.5002	0.8978	999	1004
0.1	2000	846.92	0.4235	36.9484	703	952	1000.4	0.5002	3.0218	981	1007
0.01	2000	86.84	0.0434	34.3602	26	245	1001.38	0.5007	6.9787	977	1028
0.001	2000	2.42	0.0012	1.5646	1	10	1010.6	0.5053	59.4701	731	1275