# TrafficAdaptor: An Adaptive Obfuscation Strategy for Vehicle Location Privacy Against Traffic Flow Aware Attacks

*Chenxi Qiu, †Li Yan, ‡Anna Squicciarini, ⊤Juanjuan Zhao, ⊥Chengzhong Xu, ‡Primal Pappachan
*Department of Computer Science and Engineering, University of North Texas, USA
†Department of Electronic and Information Engineering, Xi'an Jiaotong University, P. R. China
‡College of Information Science and Technology, The Pennsylvania State University, USA
⊤Shenzhen Institute of Advanced Technology, P. R. China
⊥Department of Computer Science, University of Macau, P. R. China

## ABSTRACT

One of the most popular location privacy-preserving mechanisms applied in location-based services (LBS) is *location obfuscation*, where mobile users are allowed to report obfuscated locations instead of their real locations to services. Many existing obfuscation approaches consider mobile users that can move freely over a region. However, this is inadequate for protecting the location privacy of vehicles, as their mobility is restricted by external factors, such as road networks and traffic flows. This auxiliary information about external factors helps an attacker to shrink the search range of vehicles' locations, increasing the risk of location exposure.

In this paper, we propose a vehicle traffic flow aware attack that leverages public traffic flow information to recover a vehicle's real location from obfuscated location. As a countermeasure, we then develop an adaptive strategy to obfuscate a vehicle's location by a "fake" trajectory that follows a realistic traffic flow. The fake trajectory is designed to not only hide the vehicle's real location but also guarantee the quality of service (QoS) of LBS. Our experimental results demonstrate that 1) the new threat model can accurately track vehicles' real locations, which have been obfuscated by two state-of-the-art algorithms, and 2) the proposed obfuscation method can effectively protect vehicles' location privacy under the new threat model without compromising QoS.

## CCS CONCEPTS

• **Security and privacy** → **Privacy protections**; • **Theory of computation** → *Adversary models.*

## KEYWORDS

location privacy, location obfuscation, traffic flow

**ACM Reference Format:**
*Chenxi Qiu, †Li Yan, ‡Anna Squicciarini, ⊤Juanjuan Zhao, ⊥Chengzhong Xu, ‡Primal Pappachan. 2022. TrafficAdaptor: An Adaptive Obfuscation Strategy for Vehicle Location Privacy Against Traffic Flow Aware Attacks. In *The 30th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '22), November 1–4, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3557915.3560938


*SIGSPATIAL '22, November 1–4, 2022, Seattle, WA, USA*

ACM ISBN 978-1-4503-9529-8/22/11…$15.00
https://doi.org/10.1145/3557915.3560938

## 1 INTRODUCTION

With ubiquitous wireless connectivity and continued advances in positioning technologies in mobile/on-board devices, vehicles have been increasingly participating in a variety of location-based services (LBS). Examples range from real-time navigation (e.g., Waze [30]), journalism and crisis response (MediaQ [20]) to commercial transportation systems (e.g., Uber-like platforms [22]). In most LBS applications, a vehicle's location is essential information for the server to provide services such as task assignment and navigation. Vehicles report their locations to the centralized servers at frequent or even regular intervals, raising potential privacy issues. Privacy risks are not only limited to whereabouts of the vehicles, but may also relate to some other sensitive information such as drivers' home/working address, and financial status [27].

Location privacy issues of LBS have been widely acknowledged in the recent years [1, 9, 11, 13, 22–24, 28, 30, 35]. A large body of recent work has focused and developed protection mechanisms by way of *location obfuscation* [1, 9, 22, 24, 30], wherein which mobile users are allowed to report obfuscated locations instead of true locations to servers. Compared with traditional cryptographic techniques [13], obfuscation has been acknowledged to be more suitable for mobile LBS due to its 1) low computational demand for mobile devices [22], 2) high effectiveness in protecting data privacy from server-side eavesdropping [30], and 3) high accessibility for various service providers [1].

Despite these merits, most existing location obfuscation approaches are still designed against simple threat models, where mobile users can move freely in a dimensional (2D) plane [1, 9, 24, 30]. Clearly, a vehicle's mobility is restricted by the road network topology and existing traffic regulations, or traffic flow [33]. Thus, the aforementioned assumptions are unrealistic and cannot adequately protect the privacy of vehicles operating in a real road network. Given the pervasiveness of geo-location and mobility services, an attacker can access public traffic flow information with minimal effort, and use this as auxiliary information to increase the accuracy of location tracking.

In this paper, we formalize the above privacy risks by developing a new threat model, namely *Traffic Flow Aware (TFA) Inference Attack*. Under TFA, an attacker uses publicly available vehicle traffic flow information to recover a target vehicle's real locations from its reported (obfuscated) locations. From the perspective of an attacker, the vehicle's mobility can be modeled by a *hidden Markov model*

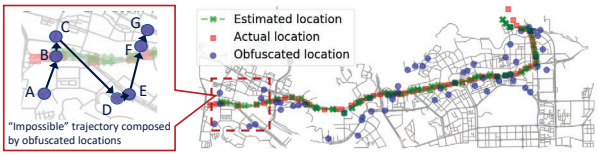



**Figure 1: Location tracking using traffic flow information.**

*(HMM).* In each round, the vehicle's actual and obfuscated locations are considered as a *hidden state* and an *observable state*, respectively. The HMM transition matrix, which can be learnt using the traffic flow information, describes the probabilities of the vehicle traveling between the locations over the map. Given the HMM matrix, the vehicle's real locations can be estimated with a high accuracy using well-developed hidden state inference algorithms (e.g., the Viterbi algorithm [10]). Figure 1 shows an example on how a traffic flow aware attacker can accurately determine a taxicab's locations (in Shenzhen mobility trace dataset [18]) even when the taxicab's location has been obfuscated with a state-of-the-art obfuscation algorithm [9]. When a vehicle obfuscates its location independently the trajectory looks "impossible" as the transition between the many adjacent reported locations in the trajectory is improbable (see the trajectory $\{A, B, C, D, E, F, G\}$ in Figure 1). This, unfortunately, helps an attacker eliminate unlikely trajectories and increases their probability of accurately tracking vehicle's real location.

Based on this threat model, we design an advanced traffic-aware obfuscation strategy, called *FTraj*, in which the traffic flow data is taken in account when obfuscating vehicle's real location. Particularly, instead of obfuscating the vehicle's locations at different rounds independently, a vehicle uses FTraj to generate multiple "fake" trajectories by following traffic flow data, and then randomly selects a fake trajectory and reports its current location as required. This makes it harder for attackers to track the vehicle's location as they cannot eliminate the reported location as impossible.

Besides hiding the real locations, the generated fake trajectory guarantees high QoS, i.e., its deviation from the vehicle's real locations is kept within a certain accuracy range. Note that this is a non-trivial guarantee to offer, since a vehicle's route is undetermined. For instance, a fake trajectory originally close to the vehicle may inevitably deviate far away from the vehicle due to restrictions imposed by real time traffic flows. As a solution, we let vehicles maintain a pool of candidate trajectories, and hence the vehicle can select a trajectory from the pool that achieves high QoS and privacy. The trajectory pool is updated using a form of *natural selection*, i.e., in each round, some candidate trajectories with high privacy guarantees and high QoS survive, while other trajectories die off and are removed from the pool. After the vehicle reports the fake location in the current round, new candidate trajectories will be generated in the following round by starting from the current reported location.

With respect to performance, simulation results based on Shenzhen taxi trajectory records [18] show that: (1) Given vehicles' locations obfuscated by the two state-of-the-art obfuscation methods [22, 24], the new inference algorithm TFA can accurately track vehicles' locations, and on average, and its expected inference error (EIE, reflecting the location privacy level) is 87.28% lower and 62.65% lower than that of the classic Bayesian inference attacks [22, 35] and temporal correlation aware attacks [8, 19], respectively. (2) Our new

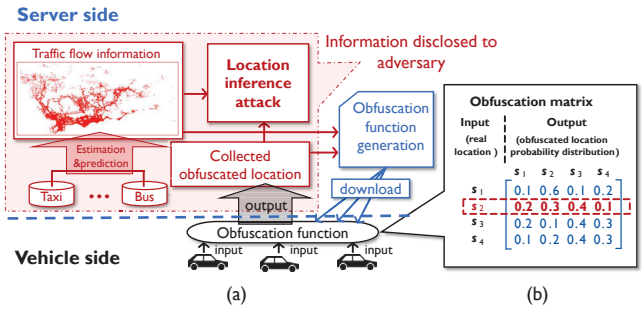



**Figure 2: Location obfuscation framework.**

obfuscation strategy, FTraj, can effectively protect vehicles' location privacy under the attack of TFA, e.g., its EIE is at least 558.34% higher than the two classic obfuscation algorithms in [22, 24].

Our contributions can be summarized as follows:
1) We develop a realistic location inference attack algorithm, TFA, which relies on contextual vehicle traffic flow information to accurately recover vehicles' real locations from obfuscated locations.
2) We develop an attack resistant location obfuscation strategy, FTraj, that obfuscates vehicles' locations via fake trajectories following realistic vehicle traffic flows.
3) We carry out a large set of experiments based on a real-world dataset to test the performance of our strategies. Empirical results demonstrate i) high accuracy of TFA in tracking vehicles of which locations have been obfuscated with state-of-the-art methods, and ii) effectiveness of FTraj in protecting vehicles' location privacy under the attack of TFA.

The rest of the paper is organized as follows: The next section provides some preliminaries. Section 3 introduces our new threat model and Section 4 describes the obfuscation algorithm. Section 5 evaluates the performance of our algorithm. Section 6 presents the related work. Finally, Section 7 makes a conclusion.

## 2 PRELIMINARIES

In Section 2.1, we first introduce our reference location obfuscation framework. In Section 2.2 we present the main notions of privacy and quality loss for single location obfuscation.

### 2.1 Location Obfuscation

Figure 2(a) shows a general location obfuscation framework [1, 9, 22, 24, 30]. We focus on LBS wherein vehicles need to physically move to target locations to complete their assigned services [23, 34]. In these settings, vehicles need to report their current locations to the server before being assigned any service/task. We assume that the server, albeit robust, may suffer from a *passive attack where attackers can eavesdrop on vehicles' reported locations breached by the server* [1, 9, 24, 30].

As a solution, before reporting the location to the server, each privacy-aware vehicle needs to conceal its actual location, via an *obfuscation function*. The obfuscation function takes the vehicle's current true location as input, and returns the probability distribution of the obfuscated location, based on which the vehicle can select an obfuscated location to report. Considering that the server will use the vehicle's obfuscated location to assign a service, in parallel to protecting vehicles' location privacy, the obfuscation function also aims to limit service quality loss (QL) caused by obfuscation.

For the sake of computational tractability, many prior works [1, 9, 22–24, 30] discretize users' location range to a finite set of $K$ locations $\mathcal{S} = \{s_1, ..., s_K\}$. Here, to discretize vehicles' location range in the road network, we partition roads to a finite number of *road segments*. We create a *connection* when a road intersects, furcates, or joins with other roads. These connections divide the road network into a set of *edges*, which only connect with other edges at their endpoints. Each edge can be further partitioned into *road segments* with the same length $\delta$[1]. We assume that attackers aim to identify in which segment the vehicle is located. For simplicity, we use the term "location" instead of "road segment" in the following part.

After location discretization, the obfuscation function can be represented as a stochastic matrix $\mathbf{Z} = \{z_{k,l}\}_{K\times K}$, namely *obfuscation matrix*, where each $z_{k,l}$ denotes the probability of taking $s_l$ as the obfuscated location given the actual location $s_k$. Then, given an exact location $s_k$ as input, the obfuscation function returns a vector $[z_{k,1}, ..., z_{k,K}]$, where each $z_{k,l}$ $(l = 1, ..., K)$ specifies the probability of selecting $s_l$ as the obfuscated location. Figure 2(b) gives an example of obfuscation matrix, where a vehicle's possible locations are $\{s_1, s_2, s_3, s_4\}$. Then, the obfuscation function can be represented as a $(4 \times 4)$-matrix. Suppose that the vehicle's actual location is $s_2$, as indicated by the matrix in Figure 2(b), the probabilities that this vehicle selects $s_1$, $s_2$, $s_3$, and $s_4$ as the obfuscated locations are 0.2, 0.3, 0.4, and 0.1, respectively.

Even though vehicles' location range have been discretized, the obfuscation matrix is still computationally intensive to derive, and beyond the capability of mobile devices at the vehicle-side. For instance, in [22, 23], the obfuscation matrix calculation involves computing millions of decision variables. Therefore, like many existing works [1, 9, 22, 24, 30], we adopt a remote computing framework, where the server first generates/updates the obfuscation matrix periodically, and then each vehicle downloads the function to obfuscate its current location in the coming report [22, 30]. Note that, although the server takes charge of calculating the obfuscation matrix, it cannot obtain vehicles' exact locations since each vehicle selects its obfuscated location in a probabilistic manner [9, 30].

## 2.2 Criteria of Privacy and Quality Loss for Single Location Obfuscation

*2.2.1 Geo-Indistinguishability (GI).* GI corresponds to a generalized version of the well-known concept of *differential privacy* [1, 9]. Differential privacy originally targets publishing aggregate queries with low sensitivity, i.e., changes of a single individual have a relative small impact on the outcome, while GI generalizes this definition to location privacy by requiring that a small change of a single user's location won't affect the distribution of his/her reported location too much.

From the attacker's perspective, we can describe the vehicle's actual and reported locations as two random variables $X$ and $Y$. GI in the road network is formally defined as 2.1 [22]:

**Definition 2.1.** *A obfuscation matrix* $\mathbf{Z}$ *satisfies* $\epsilon$*-GI if* $\forall s_i, s_j, s_k \in \mathcal{S}$,

$$\frac{z_{i,k}}{z_{j,k}} \leq e^{\epsilon d(s_i, s_j)}. \tag{1}$$

*where* $\epsilon$ *is called the privacy budget, used to quantify how much the user's actual location is disclosed according to the reported location and* $d(s_i, s_j)$ *represents the distance from* $s_i$ *to* $s_j$.

Here, higher $\epsilon$ implies more information disclosed and a lower privacy level achieved. $d$ can be Hamming distance, Euclidean distance [1, 35], or traveling distance in the road network [22, 23]. Like [22, 23], we consider $d$ as traveling distance in the road network.

**Limitation of GI for protecting vehicle location privacy**. Most current GI-based obfuscation algorithms still focus on single location obfuscation [1, 22, 23, 32, 35]. While, when a vehicle applies GI-based obfuscation separately in each round, the transition between its adjacent reported locations look "impossible" in the vehicle traffic flows (e.g. the route $\{A, B, C, D, E, F, G\}$ in Figure 1). *This, unfortunately, helps attackers shrink the search range of vehicles' true locations when attackers have the traffic flow information.*

*2.2.2 Expected Inference Error (EIE).* EIE of an obfuscated location $s_l$, also known as the *unconditional expected privacy* [23, 24], is defined by

$$\mathrm{EIE}\,(s_l) = \sum_{\hat{s}} \sum_k f_{X|Y=s_l}(s_k)\, h_{\hat{X}|Y=s_l}(\hat{s}) d_{\hat{s},s_k}, \tag{2}$$

where $h_{\hat{X}|Y=s_l}(\hat{s})$ represents the probability that the attacker estimates $\hat{s}$ as the vehicle's location given the vehicle's reported location is $s_l$. EIE essentially describes the expected distortion from the estimated location $\hat{s}$ to the actual location $s_k$, where higher EIE implies higher privacy level achieved.

*2.2.3 Quality loss (QL).* Since we focus on the LBS wherein vehicles need to physically present at target locations, quality loss highly depends on the accuracy of traveling cost (distance) estimation. Therefore, we define quality loss as the expected estimation error of traveling cost derived by obfuscated locations. We assume that the distribution of the target's location $\Pr(Q = s_q)$ is given $(s_q \in \mathcal{S})$. Then, given the vehicle's true location $s_i$, the quality loss caused by the obfuscated location $s_j$ is can be calculated by [22]:

$$\mathrm{QL}\,(s_j) = \sum_{s_q \in \mathcal{S}} \Pr(Q = s_q) \left| C(s_i, s_q) - C(s_j, s_q) \right| \tag{3}$$

where $C(s, s_q)$ denotes the traveling cost from $s$ to $s_q$. Then, the expected quality loss caused by an obfuscation matrix $\mathbf{Z}$ is calculated by averaging the quality loss of all obfuscated locations $s_j$ and real locations $s_i$, i.e.,

$$\mathrm{QL}\,(\mathbf{Z}) = \sum_{s_i, s_j \in \mathcal{S}} \Pr(X = s_i)\, z_{i,j} \mathrm{QL}\,(s_j). \tag{4}$$

# 3 TRAFFIC FLOW AWARE INFERENCE ATTACKS

In this section, we introduce our new threat model and the **T**raffic **F**low **A**ware (TFA) attack. Table 1 lists the main notations used throughout this paper.

When a vehicle reports its obfuscated location to the server, its true location is hidden from the attacker. However, attackers can still coordinate an inference attack, e.g., Bayesian inference models [22, 23, 30]. We make the following assumptions about the types of information that can be accessed by attackers when coordinating an inference attack:

**Assumption 1: The obfuscation matrix** is transparent to attackers [1, 9, 22–24, 30].

[1]Some segments may be shorter than $\delta$. As $\delta$ is small enough, their impact would be minimal, so we don't consider these segments.

**Table 1: Main notations and their descriptions**

| Symbol | Description |
|---|---|
| $\mathcal{S}$ | Location set $\mathcal{S} = \{s_1, ..., s_K\}$ |
| $t_n$ | Time slot of $n^{\text{th}}$ location report |
| $\mathbf{Z}^{t_n}$ | Obfuscation matrix at time slot $t_n$ |
| $z_{i,j}^{t_n}$ | Probability of taking $s_j$ as the obfuscated location given the true location $s_i$ at $t_n$ |
| $X^t$ | Random variable: vehicle's true location at $t$ |
| $Y^t$ | Random variable: vehicle's obfuscated location at $t$ |
| $\vec{\mathbf{s}}_o^{[t_1,t_n]}$ | Observed locations $\left\{s_o^{t_1}, ..., s_o^{t_n}\right\}$ at $t_1, ..., t_n$ |
| $\vec{\mathbf{s}}_r^{[t_1,t_n]}$ | Real locations $\left\{s_r^{t_1}, ..., s_r^{t_n}\right\}$ at $t_1, ..., t_n$ |
| $\mathbf{P}^{t,t'}$ | Vehicle's transition matrix from round $t$ to $t'$ |
| $p_{i,j}^{t,t'}$ | Vehicle's transition probability from $s_i$ to $s_j$ from round $t$ to $t'$ |

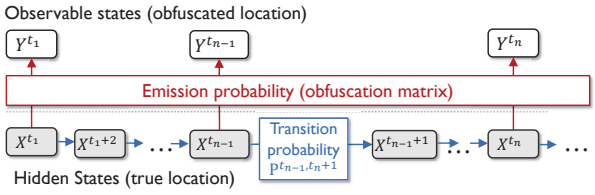



**Figure 3: Hidden Markov Model.**

**Assumption 2: The traffic flow information in the target region** can be accessed by attackers [33, 36].

For modeling purposes, we discretize time in *time slots* $\{1, 2, ...\}$. We consider the case that a vehicle reports its locations $N$ times during its trip, where the report time slots are $\{t_1, ..., t_n, ..., t_N\}$. Note that the interval between adjacent report time slots may differ. We let $\mathbf{Z}^{t_n} = \left\{z_{i,j}^{t_n}\right\}_{K\times K}$ represent the obfuscation matrix at $t_n$ ($n = 1, ..., N$), where each $z_{i,j}^{t_n}$ denotes the probability of selecting location $s_j$ as the obfuscated location given the true location $s_i$ at $t_n$. We let $s_r^{t_n}$ and $s_o^{t_n}$ denote the vehicle's reported location and real location at time $t_n$. The sequence of the vehicle's first $n$ reported locations and the corresponding real locations are represented by $\vec{\mathbf{s}}_o^{[t_1,t_n]} = \left\{s_o^{t_1}, ..., s_o^{t_n}\right\}$ and $\vec{\mathbf{s}}_r^{[t_1,t_n]} = \left\{s_r^{t_1}, ..., s_r^{t_n}\right\}$, respectively.

## 3.1 Hidden Markov Model

From the attacker's perspective, vehicles' reported (obfuscated) locations are observable, while the vehicles' true locations are hidden. Nevertheless, the vehicle's obfuscated location is related to its true location by following a probability distribution determined by the obfuscation matrix $\mathbf{Z}^{t_n}$, which is also visible to the attacker (Assumption 1). On the other hand, due to the constraints of the vehicle traffic flows, the vehicle's true locations in adjacent rounds are spatially correlated. That is, the vehicle's location in each round is dependent on its location in the last round.

Accordingly, a *hidden Markov model (HMM)* offers the attacker a straightforward model to characterize the aforementioned vehicles' mobility features. As Figure 3 shows, we use two random variables $X^t$ and $Y^t$ to denote the vehicle's actual and reported locations at time slot $t$, where $X^t$ and $Y^t$ are considered as its *hidden* and *observable* states, respectively, and $X^t$ is assumed to follow a *Markov process*, i.e., $X^t$ only depends on $X^{t-1}$.

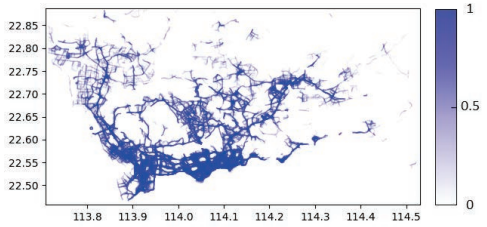



**Figure 4: Heat map of the estimated transition probabilities between the road segments in Shenzhen.**

The parameters of a HMM are of two types, *transition probabilities* and *emission probabilities*:

**1) Emission probabilities** describe the conditional distribution of obfuscated location $Y^{t_n}$ given the vehicle's true location $X^{t_n}$. Note that the emission probabilities are visible by the attacker, since each $\mathbb{P}\mathrm{r}\left(Y^{t_n} = s_j \middle| X^{t_n} = s_i\right) = z_{i,j}^{t_n}$ is essentially an entry of the obfuscation matrix $\mathbf{Z}^{t_n}$.

**2) Transition probabilities** describe the conditional distribution of vehicle's true location at $t$, $X^t$, given the vehicle's true location at $t-1$, $X^{t-1}$. We let the transition matrix $\mathbf{P}^{t-1,t} = \left\{p_{i,j}^{t-1,t}\right\}_{K\times K}$ include the transition probabilities between the vehicle's hidden states from time $t-1$ to $t$, where $p_{i,j}^{t-1,t} = \mathbb{P}\mathrm{r}\left(X^t = s_j | X^{t-1} = s_i\right)$. As introduced in Section 3.2, the transition matrix can be learned via the vehicle traffic flow information, which is available to the attacker (Assumption 2).

## 3.2 Transition Matrix Learning

The attacker can learn the transition matrices of vehicles in HMM from vehicle traffic flow information. Specifically, attackers can obtain traffic data using floating vehicle data, both historic, in the form of trajectory datasets [33], and real-time, in the form of continuous data streams [36]. Traffic datasets usually record the vehicles' coordinates along with timestamps, where time is (or can be) discretized into *slots* (e.g., seconds [18]). The attacker can then calculate the vehicle's transition probability from location $s_i$ to location $s_j$ during the time slot $t$ by

$$p_{i,j}^{t-1,t} = \frac{\#\text{ of vehicles from } s_i \text{ to } s_j \text{ from } t-1 \text{ to } t}{\#\text{ of vehicles in } s_i \text{ at time } t-1}. \quad (5)$$

In Figure 4, we show an example of a heat map of the transition probabilities between *adjacent road segments*. The heat map is created using a historical traffic flow dataset from Shenzhen [18]).

Note that vehicles typically do not report their locations in each time point. Given that the vehicle $n^{\text{th}}$ and $(n+1)^{\text{th}}$ reports are at time slots $t_n$ and $t_{n+1}$, respectively, the corresponding transition matrix for the adjacent reports $s_o^{t_n}$ and $s_o^{t_{n+1}}$, $\mathbf{P}^{t_n,t_{n+1}}$, is calculated by

$$\mathbf{P}^{t_n,t_{n+1}} = \prod_{t=t_n}^{t_{n+1}-1} \mathbf{P}^{t,t+1}. \quad (6)$$

## 3.3 Location Tracking via TFA

Given the emission probabilities (obfuscation matrices $\mathbf{Z}^{t_n}$), the transmission probabilities $\mathbf{P}^{t_1,t_2}$, ..., $\mathbf{P}^{t_{n-1},t_n}$, and the observed sequence $\vec{\mathbf{s}}_o^{[t_1,t_n]}$, the task of the attacker is to derive the maximum likelihood estimate of the vehicle's real trajectory $\vec{\mathbf{s}}_r^{[t_1,t_n]}$.

To infer a vehicle's real trajectory $\vec{\mathbf{s}}_r^{[t_1,t_n]}$, the attacker can use a hidden state sequence inference algorithm, such as the Viterbi

algorithm [10]. Viterbi provides an efficient way of finding the hidden state sequence with the maximum posterior given the observable states in HMM[2]. Given the observed location sequence $\vec{\mathbf{s}}_o^{[t_1,t_n]} = \left\{s_o^{t_1}, ..., s_o^{t_n}\right\}$, the Viterbi algorithm uses a dynamic programming approach to find the most likely real location sequence $\vec{\mathbf{s}}_r^{[t_1,t_n]} = \left\{s_r^{t_1}, ..., s_r^{t_n}\right\}$, namely *the Viterbi path*. We let $v_j^{t_n}$ represent the probability that the vehicle's $n^{\text{th}}$ hidden state is at $s_j$ given the first $n-1$ observations $\left\{s_o^{t_1}, ..., s_o^{t_{n-1}}\right\}$, and passing through the most probable state sequence $\vec{\mathbf{s}}$, i.e.,

$$v_j^{t_n} = \max_{\vec{\mathbf{s}} \in \mathcal{S}^{n-1}} \Pr\left\{ \underbrace{\left\{X^{t_1} ... X^{t_{n-1}}\right\} = \vec{\mathbf{s}}, X^{t_n} = s_j}_{\text{hidden states}} \middle| \underbrace{\left\{Y^{t_1} ... Y^{t_{n-1}}\right\} = \vec{\mathbf{s}}_o^{[t_1,t_{n-1}]}}_{\text{observations}} \right\} \tag{7}$$

The value of each cell $v_j^{t_n}$ in iteration $n$ is computed recursively based on the cells calculated in iteration $n-1$: $v_1^{t_{n-1}}, ..., v_K^{t_{n-1}}$:

$$v_j^{t_n} = \max_{s_k \in \mathcal{S}} v_k^{t_{n-1}} p_{k,j}^{t_{n-1},t_n} z_{j,o^{t_n}}. \tag{8}$$

Finally, the Viterbi path, or the estimated real trajectory, can be retrieved by saving back pointers that remember which state was used in Equ. (8).

We evaluate the ability of TFA in tracking vehciles' locations in Section 5.1. The experimental results show that TFA can track vehicles' locations more accurately compared with the current state of art inference algorithm such as the optimal inference algorithms [23, 35] and temporal correlation aware inference algorithms [7].

## 4 LOCATION PRIVACY PROTECTION VIA FAKE TRAJECTORIES

In this section, we introduce a new location obfuscation strategy, called *FTraj*, to ensure that the generated obfuscated locations follow realistic vehicle traffic flows. The basic idea of FTraj is to let the vehicle generate fake trajectories that follow realistic traffic flows when the vehicle is driving. Then, the vehicle report the current locations of fake trajectories when requested. Since the reported locations are from trajectories following realistic traffic flows, they are hard to be eliminated by TFA.

Note that to limit quality loss, fake trajectories are required to be within a certain range of the vehicle's real locations in each time slot. Such a requirement, however, is hard to satisfy if an insufficient number of fake trajectories is maintained, considering that the vehicle's mobility might be unpredictable (i.e., before the vehicle is assigned a task, its future route is undetermined). For example, a fake trajectory originally close to the vehicle may inevitably deviate far away from the vehicle after the time slot $t$. As a solution, we let the vehicle maintain a sufficiently large pool of fake trajectories, and select one to report when requested. Before introducing the details of the algorithm, we first give the formal definition of *fake trajectory* in Definition 4.1:

**Definition 4.1.** *Given a time interval $[t_a, t_b]$ and the corresponding learned transition matrices $\mathbf{P}^{t_a,t_{a+1}}, ..., \mathbf{P}^{t_b-1,t_b}$, a* fake trajectory $\vec{\mathbf{s}}_f^{[t_a,t_b]}$ *is a sequence of locations $\vec{\mathbf{s}}_f^{[t_a,t_b]} = \left\{s_f^{t_a}, s_f^{t_a+1}, ..., s_f^{t_b}\right\}$ such that 1) each location $s_f^t$ is generated at each time slot $t$ ($t = t_a, t_a+1, ..., t_b$), and 2) and the transition from each $s_f^{t-1}$ to $s_f^t$ follows a realistic traffic flow, i.e., the transition probability from $s_f^{t-1}$ to $s_f^t$, denoted by $p_{s_f^{t-1},s_f^t}^{t-1,t}$ ($t = t_a, t_a+1, ..., t_b-1$), is non-zero.*

[2]Note that other algorithms or simpler heuristic may be used by an attacker. Viterbi is seen as a standard and sound approach for this type of reverse engineering, as such it can be considered as the worst case scenario algorithm.

**Table 2: Main notations and definitions in Section 4.**

| Symbol | Description |
|---|---|
| $\vec{\mathbf{s}}_f^{[t_a,t]}$ | A fake trajectory during the time interval $[t_a, t]$ |
| $s_f^t$ | Location of $\vec{\mathbf{s}}_f^{[t_a,t_b]}$ at time $t$ |
| $\mathcal{F}^{[t_a,t]}$ | Set of fake trajectories during time interval $[t_a, t]$ |
| $O_{fi}^{[t_a,t-1]}$ | Offsprings of the fake trajectory $\vec{\mathbf{s}}_{fi}^{[t_a,t-1]}$ |
| $M$ | The maximum number of fake trajectories in the pool |
| $\Gamma$ | Predetermined constant to limit quality loss caused by each location in fake trajectory |

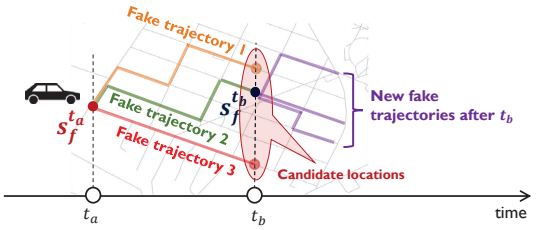



**Figure 5: Basic idea of FTraj.**

Table 2 lists the additional notations used in this section.

### 4.1 Fake Trajectory Pool

When the vehicle starts its trip, the vehicle selects a single location as the starting point of all the fake trajectories. Like [23, 30], the location is selected in a probabilistic manner via an obfuscation matrix $\mathbf{Z}^1$, generated by the server and downloaded by the vehicle.

Subsequently, the vehicle needs to generate/update a *fake trajectory pool* during each consecutive location reports. As opposed to the obfuscation matrix calculated by the server, fake trajectory pools are generated at the vehicle side using local traffic flow information. Figure 5 gives a brief idea: Suppose that the last time the vehicle reports its obfuscated location $s_f^{t_a}$ is at time $t_a$. After $t_a$, the vehicle generates and maintains the fake trajectories $\{1, 2, 3\}$, all of which start from $s_f^{t_a}$ and follow realistic routes. At the next report time $t_b$, the current locations of all the fake trajectories are the candidate locations, from which the vehicle randomly select one to report. In this example, the vehicle selects *trajectory 2*, and reports its current location $s_f^{t_b}$ to the server. After $t_b$, the vehicle clears the existing fake trajectories in the pool and generates a pool of new fake trajectories which all start from $s_f^{t_b}$, until the next report. As the vehicle applies the same strategy during each consecutive report, without losing the generality, next we only consider the fake trajectory generation/update during $[t_a, t_b]$.

**Pool update**. The pool is updated using an algorithm inspired by *natural selection* principles: In each round, trajectories within a certain distance from the current real location will survive and generate a set of new trajectories for the following round, while

other trajectories will die off and be removed from the candidate pool (see the example in Figure 6). Specifically, in each time slot, the pool update is composed of *reproduction* and *elimination*:

**S1. Reproduction** (see Figure 6(a)). We let $\mathcal{F}^{[t_a,t]}$ denote the fake trajectory pool during $[t_a, t]$ ($t_a \le t \le t_b$). For each fake trajectory $\vec{\mathbf{s}}_f^{[t_a,t-1]} = \left\{s_f^{t_a}, ..., s_f^{t-1}\right\} \in \mathcal{F}^{[t_a,t-1]}$, the vehicle extends $\vec{\mathbf{s}}_f^{[t_a,t-1]}$ by adding a new location $s_f^t$ that is reachable by $s_f^{t-1}$, i.e., $p^{t-1,t}_{s_f^{t-1},s_f^t} > 0$. Here, in each time slot $t$, we assume that the transition matrix $\mathbf{P}^{t-1,t}$ is available to the vehicle[3]. In the transition matrix, the vehicle only needs to retrieve the transition probabilities from $s_f^{t-1}$ (the last location in the fake trajectory $\vec{\mathbf{s}}_{f[t_a,t-1]}$) to all other locations $s_1, ..., s_K$, denoted by $\left[p^{t-1,t}_{s_f^{t-1},s_1}, ..., p^{t-1,t}_{s_f^{t-1},s_K}\right]$. Moreover, we enforce a constraint to limit the QL caused by each $s_f^t$:

$$\mathrm{QL}\left(s_f^t\right) \le \Gamma, \ (\mathrm{QL}(\cdot) \text{ is define in Equ. (3)}) \tag{9}$$

where $\Gamma$ is a predetermined constant. Trajectories with the current locations not satisfying Equ (9) will not be considered even it is not the time slot to report the location, since the pool can only maintain a limited number of candidate trajectories and trajectories already deviated far away from the real trajectory is less likely to generate high QoS in the future.

The newly generated fake trajectory $\vec{\mathbf{s}}_f^{[t_a,t]} = \left\{s_f^{t_a}, ..., s_f^t\right\}$ is called an *offspring* of $\vec{\mathbf{s}}_f^{[t_a,t-1]}$. Given the transition matrix $\mathbf{P}^{t-1,t}$ and the constant $\Gamma$, we can find the set of possible locations to generate an offspring of $\vec{\mathbf{s}}_f^{[t_a,t-1]}$,

$$\mathcal{R}\left(\vec{\mathbf{s}}_f^{[t_a,t-1]}\right) = \left\{s_j \in \mathcal{S} \,\middle|\, p^{t-1,t}_{f^{t-1},j} > 0, \ \mathrm{QL}(s_j) \le \Gamma\right\}. \tag{10}$$

**S2. Elimination** (see Figure 6(b)). Due to tractability limitations, the vehicle may not add all the locations in $\mathcal{R}\left(\vec{\mathbf{s}}_f^{[t_a,t-1]}\right)$ to the current fake trajectories; as the number of fake trajectories might increase exponentially over time. Therefore, after adding the locations to the existing fake trajectories, the vehicle needs to remove fake trajectories with low privacy guarantees (measured by EIE in Equ. (2)) and high quality loss. Specifically, we define the *fitness value* of the current location $s_f^l$ of the fake trajectory $\vec{\mathbf{s}}_f^{[t_a,t]}$

$$\mathrm{Fit}\left(s_f^l\right) = \alpha_\mathrm{e}\mathrm{EIE}\left(s_f^t\right) - \alpha_\mathrm{q}\mathrm{QL}\left(s_f^t\right), \tag{11}$$

to evaluate both EIE and quality loss caused by its current location $s_f^t$, where $\alpha_\mathrm{e}$ and $\alpha_\mathrm{q}$ denote the weights of EIE and quality loss in the fitness value, respectively. The vehicle only maintains the fake trajectories with top $M$ fitness values in the pool, where $M$ is the pool capacity. The trajectory pool at each time $t$ can be represented by

$$\mathcal{F}^{[t_a,t]} = \left\{\vec{\mathbf{s}}_{f_1}^{[t_a,t]}, ..., \vec{\mathbf{s}}_{f_M}^{[t_a,t]}\right\}, \tag{12}$$

where each $\vec{\mathbf{s}}_{f_i}^{[t_a,t]} = \left\{s_{f_i}^{t_a}, ..., s_{f_i}^t\right\}$ $(i = 1, ..., M)$.

[3]The vehicle can obtain the transition matrix either by using historical data [33], or by downloading it from the server.

## 4.2 Pool Update Algorithm

In each time slot $t$, we let the vehicle store the location information of each fake trajectory $\vec{\mathbf{s}}_{f_i}^{[t_a,t]}$ as a node or triple, Figure 6(c) shows: $\left(s_{f_i}^t, s_{f_i}^{t-1}, \mathrm{Fit}\left(s_{f_i}^t\right)\right)$, including (i) $\vec{\mathbf{s}}_{f_i}^{[t_a,t]}$'s current location $s_{f_i}^t$, (ii) its previous location $s_{f_i}^{t-1}$, and (iii) its fitness value $\mathrm{Fit}\left(s_{f_i}^t\right)$. The nodes of different trajectories at round $t$ are stored in a *min heap* $\mathbf{q}^t$, which is efficient in finding the top $M$ elements from a set [5]. $\mathbf{q}^t$ has two features: (F1) the top node of $\mathbf{q}^t$ has the minimum fitness value; (F2) two types of operations can be conducted on $\mathbf{q}^t$: *push* (i.e., to insert a new node to $\mathbf{q}^t$) and *pop* (i.e., to remove the top node from $\mathbf{q}^t$). The time complexity of both types of operations is $O(\log M)$. *Both pop and push operations won't change feature F1 of* $\mathbf{q}^t$.

*Algorithm 1* shows the pseudo code of the pool update: $\mathbf{q}^t$ is initialized by empty (line 1). If it is the first time slot after that the vehicle reports the location, i.e., $t = t_a + 1$ (line 2), FTraj randomly picks up $M$ locations around the current true location: $s_{f_1}^{t_a}$, ..., $s_{f_M}^{t_a}$ (line 3), calculates their fitness values $\mathrm{Fit}\left(s_{f_1}^{t_a}\right)$, ..., $\mathrm{Fit}\left(s_{f_M}^{t_a}\right)$, and pushes the corresponding nodes $\left(s_{f_i}^{t_a}, \phi, \mathrm{Fit}\left(s_{f_i}^{t_a}\right)\right)$ $(i = 1, ..., M)$ sequentially onto $\mathbf{q}^{t_a}$ (line 4 – 6).

After the first time slot, the candidate pool $\mathcal{F}^{[t_a,t]}$ is updated from $\mathcal{F}^{[t_a,t-1]}$ via two steps: *reproduction* and *elimination*:

**S1. Reproduction** (line 8–12). Using the transition matrix, the vehicle extends each candidate trajectory $\vec{\mathbf{s}}_{f_i}^{[t_a,t-1]} \in \mathcal{F}^{[t_a,t-1]}$ to each of the locations in $\mathcal{R}\left(\vec{\mathbf{s}}_{f_i}^{[t_a,t-1]}\right)$, i.e., the set of locations that can be reached by $\vec{\mathbf{s}}_{f_i}^{[t_a,t-1]}$. We can then obtain the set of $\vec{\mathbf{s}}_{f_i}^{[t_a,t-1]}$'s offsprings):

$$O_{f_i}^{[t_a,t-1]} = \left\{\left\{\vec{\mathbf{s}}_{f_i}^{[t_a,t-1]}, s_j\right\} \,\middle|\, s_j \in \mathcal{R}\left(\vec{\mathbf{s}}_{f_i}^{[t_a,t-1]}\right)\right\}. \tag{13}$$

**S2. Elimination** (line 13–22). Reproduction generates a set of new fake trajectories $\mathcal{F}'^{[t_a,t]} = \cup_{i=1}^{M} O_{f_i}^{[t_a,t-1]}$. To eliminate the the trajectories with low fitness values, the vehicle ranks all the trajectories in $\mathcal{F}'^{[t_a,t]}$ by their fitness values (defined by Equ. (11)) and only keeps the top $M$ ones.

Specifically, for each location $s_j \in \mathcal{R}\left(\vec{\mathbf{s}}_{f_i}^{[t_a,t-1]}\right)$, FTraj calculates its fitness value $\mathrm{Fit}(s_j)$ (line 16). The first $M$ nodes are pushed directly onto $\mathbf{q}^t$ (line 18). Once $\mathbf{q}^t$ reaches its capacity, i.e., $|\mathbf{q}^t| = M$, FTraj first checks whether the top node in $\mathbf{q}^t$ has higher fitness value than the new node $q_{\mathrm{new}}^t$ with $\mathrm{Fit}(s_j)$ (line 20): If NO, $q_{\mathrm{new}}^t$ won't be pushed onto $\mathbf{q}^t$, since $q_{\mathrm{new}}^t$ has a lower fitness value than any of the $M$ nodes in $\mathbf{q}^t$; otherwise, $q_{\mathrm{top}}^t$ will be popped off and $q_{\mathrm{new}}^t = \left(s_{f_{j'}^{t-1}}, s_j, \mathrm{Fit}(s_j)\right)$ will be pushed onto $\mathbf{q}^t$. $q_{\mathrm{top}}^t$ cannot be one of the $M$ highest nodes, since $q_{\mathrm{top}}^t$ has a lower fitness value than $q_{\mathrm{new}}^t$'s and the other $M-1$ nodes in $\mathbf{q}^t$ (based on feature F1).

**Time complexity**. Suppose that there are $U$ locations in each $\mathcal{R}\left(\vec{\mathbf{s}}_{f_i}^{[t_a,t-1]}\right)$. To obtain $\mathbf{q}^t$, it takes up to $UM$ push/pop operations, which amounts to $O(UM \log M)$ operations. As both $M$ and $U$ are not large in practical ($M = 100$ and $U \le 300$ in the experiment in Section 5), such a computation load is acceptable to mobile devices like smartphones.

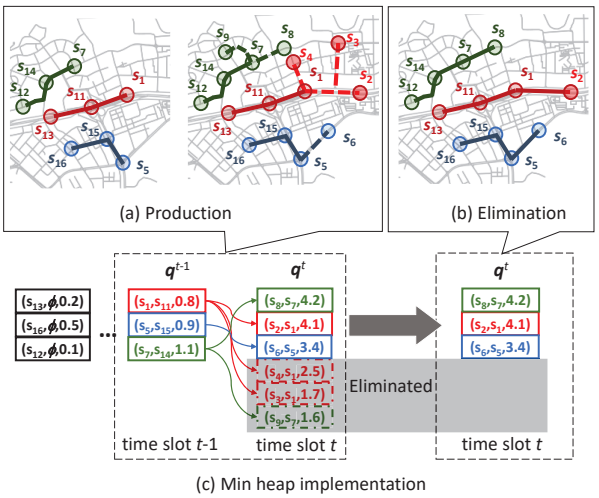



**Figure 6: Example of fake trajectory maintenance: In round $t$, there are three fake trajectories $\vec{\mathbf{s}}_{f_1} = \{s_{12}, s_{14}, s_7\}$, $\vec{\mathbf{s}}_{f_2} = \{s_{13}, s_{11}, s_1\}$, and $\vec{\mathbf{s}}_{f_3} = \{s_{16}, s_{15}, s_5\}$.**
**(a) Reproduction: $\vec{\mathbf{s}}_{f_2}$ has three possible locations to move to: $s_2$, $s_3$, and $s_4$, and hence $\vec{\mathbf{s}}_{f_2}$ has three offsprings: $\{\vec{\mathbf{s}}_{f_2}, s_2\}$, $\{\vec{\mathbf{s}}_{f_2}, s_3\}$, and $\{\vec{\mathbf{s}}_{f_2}, s_4\}$. Similarly, $\vec{\mathbf{s}}_{f_1}$ has two offsprings: $\{\vec{\mathbf{s}}_{f_1}, s_8\}$ and $\{\vec{\mathbf{s}}_{f_1}, s_9\}$; $\vec{\mathbf{s}}_{f_3}$ has one offspring: $\{\vec{\mathbf{s}}_{f_3}, s_6\}$.**
**(b) Elimination: Among the 6 offsprings generated at time slot $t_n$, 3 offsprings with lower fitness values $\{\vec{\mathbf{s}}_{f_1}, s_9\}$, $\{\vec{\mathbf{s}}_{f_2}, s_3\}$, and $\{\vec{\mathbf{s}}_{f_2}, s_4\}$ are eliminated from the pool.**

## 4.3 Obfuscated Location Selection

At time slot $t$, we use $\mathcal{S}^t_{\text{can}}$ denote the set of current locations of the fake trajectories in the vehicles' trajectory pool, including the vehicle's real location, i.e.,

$$\mathcal{S}^t_{\text{can}} = \left\{ s^t_{f_i} \in \mathcal{S} \,\middle|\, \vec{\mathbf{s}}^{[t_a,t]}_{f_i} \in \mathcal{F}^{[t_a,t]} \right\} \cup s^t_r \tag{14}$$

When the vehicle reports the location at $t_b$, it randomly selects one of the locations in $\mathcal{S}^{t_b}_{\text{can}}$ as the obfuscated location to report. The first objective of location selection is to satisfy $\epsilon$-GI (Definition 2.1), i.e., the attacker's perspective, any pair of locations in $\mathcal{S}^{t_b}_{\text{can}}$ are hard to be distinguished given the vehicle's reported locations. We let the decision variable $z_{i,j}$ ($s_i, s_j \in \mathcal{S}^{t_b}_{\text{can}}$) represent the probability of selecting location $s_j$ given the real location $s_i$ and let $\mathbf{Z} = \{z_{i,j}\}_{(M+1)\times(M+1)}$ represent the *location obfuscation matrix*. According to Definition 2.1, for each pair of $s_i, s_j \in \mathcal{S}^{t_b}_{\text{can}}$, the $\epsilon$-GI constraint $z_{i,k} \le e^{\epsilon d(s_i,s_j)} z_{j,k}$ needs to be satisfied for each obfuscated location $s_k \in \mathcal{S}^{t_b}_{\text{can}}$.

Besides GI, the location obfuscation matrix $\mathbf{Z}$ aims to achieve the highest expected fitness value, calculated by $\sum_{i,j} \Pr(X = s_i)\, z_{i,j} \text{Fit}(s_j)$. Finally, the derivation of the optimal $\mathbf{Z}$ can be formulated as a linear programming (LP) problem:

$$\max \quad \sum_{i,j} \Pr(X = s_i)\, z_{i,j} \text{Fit}(s_j) \tag{15}$$

$$\text{s.t.} \quad z_{i,k} \le e^{\epsilon d(s_i,s_j)} z_{j,k}, \forall i, j, k. \tag{16}$$

$$\sum_{k=1}^{M} z_{i,k} = 1, \forall i, \; 0 \le z_{i,k} \le 1, \forall i, k. \tag{17}$$

where Equ. (16) represents the GI constraints, and Equ. (17) indicates that, for each real trajectory $\vec{\mathbf{s}}^{[t_a,t]}_{f_i}$, the sum probability of selecting all candidate locations in $\mathcal{S}^t_{\text{can}}$ is equal to 1 [25]. The problem formulated by Equ. (15)-(17) can be solved using the LP standard algorithms, like the simplex method or the interior point algorithm [15].

**Algorithm 1:** Fake trajectory pool update.

**Input** : $t, \mathbf{q}^1, ..., \mathbf{q}^{t-1}, s^{ta}_f$
**Output** : $\mathbf{q}^t$

$\mathbf{q}^t$ is initialized by empty;
**if** $t = t_a + 1$ **then**
  Randomly pick up $M$ locations around $s^{ta}_f$: $s^{ta}_{f_1}, ..., s^{ta}_{f_M}$;
  **for** *each* $s^{ta}_{f_i}$ $(i = 1, ..., M)$ **do**
    Calculate $s^{ta}_{f_i}$'s fitness value Fit $\left(s^{ta}_{f_i}\right)$ using Equ. (11);
    $\mathbf{q}^t$.push($\left(s^{ta}_{f_i}, \phi, \text{Fit}\left(s^{ta}_{f_i}\right)\right)$);
**else**
  // S1: Reproduction
  **for** *each* $\left(s^t_{f_i}, s^{t-1}_{f_{i'}}, \text{Fit}\left(s^t_{f_i}\right)\right)$ *in* $\mathbf{q}^{t-1}$ **do**
    Initialize the location set $\mathcal{R}\left(\vec{\mathbf{s}}^{[t_a,t-1]}_{f_i}\right)$ by an empty set;
    **for** *each* $s_j \in \mathcal{S}$ *with* $p^{t-1,t}_{f_i^{t-1},j} > 0$ **do**
      Add $s_j$ to $\mathcal{R}\left(\vec{\mathbf{s}}^{[t_a,t-1]}_{f_i}\right)$;
  // S2: Elimination
  **for** *each* $\left(s^t_{f_i}, s^{t-1}_{f_{i'}}, \text{Fit}\left(s^t_{f_i}\right)\right)$ *in* $\mathbf{q}^{t-1}$ **do**
    **for** $\forall s_j \in \mathcal{R}\left(\vec{\mathbf{s}}^{[t_a,t-1]}_{f_i}\right)$ **do**
      Calculate $s_j$'s fitness value Fit $(s_j)$ using Equ. (11);
      **if** $|\mathbf{q}^t| < M$ **then**
        $\mathbf{q}^t$.push($\left(s^{t-1}_{f_{j'}}, s_j, \text{Fit}\left(s_j\right)\right)$);
      **else**
        **if** $\mathbf{q}^t$.top() *has higher fitness value than* Fit $(s_j)$ **then**
          $\mathbf{q}^t$.pop();
          $\mathbf{q}^t$.push($s^{t-1}_{f_{j'}}, s_j$, Fit $(s_j)$);
**return** $\mathbf{q}^t$;

# 5 EXPERIMENTAL VALIDATION

In this section, we evaluate the performance of the **T**raffic **F**low **A**ware (TFA) inference attack in Section 5.1 and our new location obfuscation algorithm (FTraj) in Section 5.2[4]. We carry out an extensive evaluation using a real dataset, containing the GPS records of around 28,000 vehicles. The benchmarks in our experiment, including both inference attacks and obfuscation algorithms, are listed as follows:

**A) Inference attack algorithm**. In Section 5.1, we compare our inference attack algorithm with the following two benchmarks:
(i) *Bayesian inference attack* (**Bayes**) [22, 35]. According to the vehicle's reported locations, Bayes first derives the posterior of a target vehicle's real location via Bayes' theorem. It then estimates the vehicle's real location by finding the location $\hat{s} \in \mathcal{S}$ that minimizes the EIE, i.e.,

$$\hat{s} = \arg\min_{s_r \in \mathcal{S}} \sum_{s_k \in \mathcal{S}} \Pr\left(Y^{t_n} = s_k | X^{t_n} = s_l\right) d(s_r, s_k). \tag{18}$$

(ii) ***Temporal Correlation Aware (TCA)*** *inference attack* [7], where

[4]The research artifacts are available to download from [29].

dependencies between vehicles' reported locations at consecutive time points are considered. TCA assumes that vehicles' mobility follows a Markov process, but without using the vehicle traffic flow information. We let TCA estimate the vehicles' locations using HMM, where the transition matrix of the Markov chain is learnt based on the vehicles' reported locations using the maximum likelihood estimation.

**B) Obfuscation algorithms**. In Section 5.2, we compare our location obfuscation algorithm with the following state-of-the-art algorithms, both of which obfuscate vehicles' locations independently per round:

(i) *Planar Laplacian noise* (**Laplace**) [1], where the obfuscation probabilities are calculated by $\mathbb{P}\mathrm{r}\left(Y^{t_n} = s_j | X^{t_n} = s_i\right) \propto e^{-\epsilon \frac{d(s_j, s_i)}{L_{\max}}}$. Here, $\epsilon$ is the *privacy budget* defined in Equ. (1), i.e., higher $\epsilon$ implies more information to be disclosed. $L_{\max}$ is the maximum distance between any two locations in the target region.

(ii) ***LP** based **O**bfuscation* (**LPO**) [22], which follows a linear programming framework. The objective is to minimize the quality loss of a single vehicle with the $\epsilon$-GI constraints being satisfied (defined in Equ. (1)). LPO, as compared to Laplace model, also accounts for network-constrained mobility features.

We measure the following two metrics:

(i) *Privacy level*, measured by EIE (Equ. (2)).

(ii) *Quality loss (estimation error of traveling distance)*, defined as the expected distortion of estimated traveling distance by the server (Equ. (4)). Here, we assume the tasks are uniformly distributed over the location set $\mathcal{S}$.

**Dataset**. The dataset used for our experiments includes the mobility records of vehicles from Jan 1, 2015 to Dec 31, 2015, including:

(1) *Taxicab Dataset*, which records the status (e.g., timestamp, GPS position, velocity, occupancy) of 15,610 taxicabs.

(2) *Dada Car Dataset*, provided by the Dada Car corporation (a customized transit service similar to UberPool). It records the status (e.g., timestamp, position, velocity) of 12,386 customized transit service vehicles.

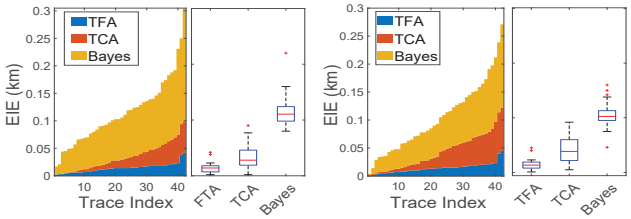
(3) *Road Map.* The road map of Shenzhen is obtained from OpenStreetMap [21]. According to the municipal information of Shenzhen [18], we use a bounding box with coordinate ($lat = 22.4450, lon = 113.7130$) as the south-west corner, and coordinate ($lat = 22.8844, lon = 114.5270$) as the north-east corner, which covers an area of around 2,926km$^2$, to crop the road map data.

We utilized a 117 TB Hadoop Distributed File System (HDFS) [2] on a cluster consisting of 51 nodes to efficiently manage these datasets. Each node is equipped with 28 cores and 64 GB RAM. All data processing and analysis is accomplished with Apache Spark [3], which is a fast in-memory cluster computing system, deployed over the Hadoop cluster.

## 5.1 Trajectory Inference Algorithms

We first test the accuracy of our trajectory inference algorithm TFA. In what follows, we set the parameters $\epsilon = 100$/km, $\Gamma = 1km$, and $M = 100$ in TFA. Using the vehicles' historical traffic records in Shenzhen [18], we can train the HMM transition matrices of TFA over time by Equ. (5). Figure 4 shows a heat map of the transition probabilities between the adjacent road segments in Shenzhen.

We pick up 42 taxicab traces from the dataset and consider them as the trajectories of the target vehicles. Figure 7(a)(b) compare the




(a) Obfuscated by Laplace (b) Obfuscated by LPO

**Figure 7: Comparison of EIE between TFA, TCA, and Bayes.**

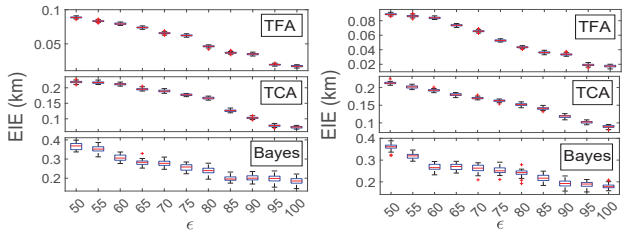
EIE of TFA, TCA, and Bayes when the vehicles' locations are obfuscated by Laplace and LPO, respectively. Without loss of generality, we sort the vehicle traces according to the EIE achieved by Bayes on the left side of the figure. On the right side, we show the box plot (variability) of the three algorithms' EIE. From both Figure 7(a)(b), we can find that TFA, as a traffic flow aware inference attack, has significant higher inference accuracy than Bayes and TCA. When applying Laplace and LPO for location obfuscation, the EIE of TFA is 87.97% and 86.58% lower than that of Bayes', and 58.45% and 66.85% lower than that of TCA, respectively. Compared with Bayes, TFA can eliminate more "impossible trajectories" by considering the temporal correlation of vehicles' consecutive reported locations. Although both TFA and TCA consider the dependency of vehicles' reported locations, TFA additionally considers the restrictions of traffic flows on vehicles' mobility. Therefore, TFA can achieve lower EIE by further shrinking the searching range of the vehicle's real trajectory. Our results, as shown, also demonstrate that classic location obfuscation methods, like Laplace and LPO, are insufficient to protect vehicles' location privacy from the inference attack.

Next, we measure how $\epsilon$ impacts the EIE of TFA, TCA, and Bayes. Recall that the privacy level in $\epsilon$-GI is quantified by the parameter $\epsilon$. The results are reported in Figure 8(a)(b), where $\epsilon$ is changed from 50 to 100. The target vehicles' locations are obfuscated by Laplace and LPO in Figure 8(a)(b), respectively. Not surprisingly, EIE decreases with the increase of $\epsilon$ in both figures, since higher $\epsilon$ allows less deviation from obfuscated locations to real locations. However, the impact of $\epsilon$ on TFA and TCA is not as significant as it is on Bayes, i.e. when an obfuscated location has a higher deviation from the real location, it has less correlation with the last reported location, and hence it is more likely to be eliminated by both TFA and TCA.

## 5.2 Location Obfuscation Algorithms

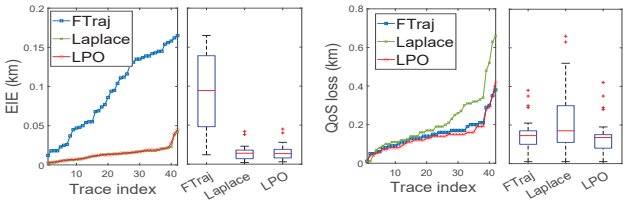
We now test the performance of our obfuscation algorithm **FTraj** in terms of both privacy and quality loss, by comparing it with two benchmarks: **Laplace** and **LPO**. We use our own TFA as the inference algorithm to estimate vehicles' locations.

Figure 9(a) compares the EIE of the three obfuscation algorithms in the 42 taxicab traces. Without loss of generality, we sort the vehicle traces according to the EIE achieved by FTraj on the left side of the figure. On the right side, we show the box plot (variability) of the three algorithms' EIE. The figure demonstrates that FTraj outperforms Laplace and LPO in terms of EIE: On average, the EIE of FTraj is 560.55% and 558.34% higher than that of Laplace and LPO, respectively. As introduced in Section 4, FTraj can offer stronger privacy guarantees (EIE) because it obfuscates vehicles' locations via fake trajectories that follow realistic traffic flows. Hence, the




(a) Obfuscated by Laplace (b) Obfuscated by LPO

**Figure 8: EIE of TFA, TCA, and Bayes with different $\epsilon$.**




(a) Expected inference error (EIE) (b) Quality loss

**Figure 9: Comparison of different obfuscation algorithms.**
***Vehicle trace indices are sorted by the quality loss caused by FTraj.**

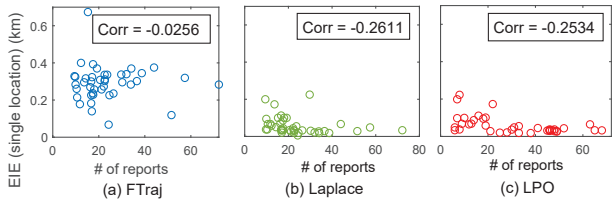
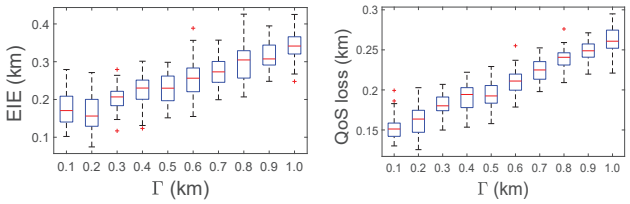



**Figure 10: Comparison of the correlation between EIE and report frequency.**

obfuscated locations generated by FTraj are hard to distinguish from the vehicles' real locations by TFA. Figure 9(b) compares the quality loss of the three algorithms. As shown, Laplace > FTraj ≈ LPO in terms of quality loss. On average, the quality loss caused by FTraj is 28.94% lower than the quality loss of Laplace, and 9.47% higher than that of LPO. Laplace has higher quality loss than both FTraj and LPO as it simply considers users' mobility on a 2D plane, in which the sensitivity of quality loss to location obfuscation is different than in a road network. Due to the restriction of the road network, a small location deviation on the 2D plane may lead to a significant difference in estimated traveling distance over roads (e.g., when a vehicle has to take a detour to reach a nearby destination). Therefore, obfuscated locations generated by Laplace are more likely to generate a high traveling cost. Although LPO has a slightly lower quality loss than FTraj, LPO minimizes quality loss at the expense of privacy (EIE), as demonstrated in Figure 9(a).

We note that in the dataset, the number of vehicles' location reports in any given time interval varies across different traces, as vehicles report their locations with different frequencies. Therefore, it is interesting to check how the report frequency impacts the EIE when FTraj, Laplace, and LPO are applied to obfuscate vehicles' locations. Figure 10(a)(b)(c) depicts the correlation between EIE and the report frequency when vehicles' locations are obfuscated by the three algorithms. The figure indicates that (1) the EIE of both Laplace and LPO is negatively correlated with the vehicles' report frequency, while (2) no significant correlation can be found




(a) Varying Γ (b) Varying Γ

**Figure 11: EIE of FTraj given different Γ.**

between FTraj's EIE and report frequency. Observation (1) stems from the fact that when the report frequency is higher, the hidden states of adjacent reports are more likely to be closely correlated, and accordingly the attacker is more likely to recover the vehicles' real locations (hidden states) by eliminating unlikely obfuscated locations. As for (2), as fake trajectories generated by FTraj follow realistic traffic flows, the attacker cannot eliminate obfuscated locations no matter how closely each pair of adjacent locations in fake trajectories is correlated. As such, the privacy level achieved by our approach will not degrade even when vehicles report their locations frequently.

Finally, we evaluate the impact of the parameter Γ on both EIE and quality loss, where Γ is a constant to limit the quality loss (defined in Equ. (9)). Figure 11(a)(b) show the change of EIE and quality loss with Γ increased from 0.1km to 1.0km, respectively. The figure implies that both EIE and quality loss of FTraj increase with the increase of Γ. This is because that higher Γ allows more space to select obfuscated locations for each vehicle, leading to a higher privacy level. By contrast, higher Γ introduces errors to the vehicles' reported locations, which may cause higher quality loss. Therefore, it is important to consider the trade off between these metrics, and balance them according to the users' preference.

## 6 RELATED WORK

**Sporadic location privacy**. The discussion of location privacy criteria can date back to almost two decades ago, when Gruteser and Grunwald [14] first introduced the notion of *location k-anonymity* based on Sweeney's well-known concept of *k-anonymity* for data privacy [26]. This notion has been extended to obfuscate location by means of *l-diversity*, i.e., a user's location cannot be distinguished with other $l-1$ locations [35]. However, *l*-diversity is hard to achieve in many applications as it assumes dummy locations are equally likely to be the real location from the attacker's view [1].

In recent years, two practical privacy notions have been proposed based on statistical quantification of attack resilience, e.g., *EIE* [24], and *GI* [1]. On the basis of EIE and GI, a large body of location obfuscation strategies have been proposed to achieve either of these two privacy criteria (e.g., [1, 24, 30]) or their combination [35]. As location obfuscation inevitably introduce errors to users' reported locations, leading to quality loss in LBS, a key issue has been discussed in those works is how to trade-off QoS and privacy. Many existing works follow a global optimization framework: given the privacy (measured by EIE/GI) or QoS constraints, an optimization model is formulated to maximize QoS or privacy respectively [9, 24, 30].

Different with our work, the above approaches based EIE and GI are still based on isolated obfuscated location with no consideration of spatiotemporal correlation between obfuscated locations.

**Spatiotemporal Location Privacy**. A variety of privacy protection location inference algorithms have been proposed to date. These algorithms focus on spatiotemporal correlation of users' reported locations, either from a single user at different time points (e.g., trajectory) [4, 7, 8, 16, 19, 32] or from multiple users [6, 17]. Many of those works are based on the assumption that users' mobility follows a Markov process [8, 19], i.e., users' current locations depend on the locations attained in the previous round. For instance, Liao *et. al* [19] applied a hierarchical Markov model to learn and infer a mobile user's trajectory based on the places and temporal patterns the user visited. Given uncertain locations of moving objects, Emrich *et. al* [8] proposed a modified matrix computation method to compute the probability of a user appearing in certain region during certain time period. Recently, research works closer to ours have proposed privacy criteria and solutions that account for statistical features of users' location data [6, 7, 12, 31]. For example, under the assumption that attackers use Markov models to describe users' mobility, Cao *et al.* [7] defined a criterion to quantify the privacy level that existing methods can achieve. Cao et al. [6] extended the notion of DP to a new criteria to protect *spatiotemporal event privacy* and provided a framework to calculate the privacy loss of a given location privacy protection mechanism. By counting for the temporal correlations in location data, Xiao et al. [31] proposed a new definition, called $\delta$-location set based DP, and presented a planar isotropic mechanism for location obfuscation. While elegant, all these formulations are designed for general moving objects and yet do not consider vehicles' mobility constraints. In practice, vehicles are bound by traffic regulations and road traffic. Therefore, they fall short in regard to their applicability in vehicles' location privacy protection.

## 7 CONCLUSIONS

In this paper, we demonstrated how popular obfuscation models fail to protect against traffic aware attacks. We developed a new location inference attack, namely TFA, that leverages traffic information to accurately recover vehicles' trajectories from their obfuscated locations. As a countermeasure, we proposed a robust location obfuscation algorithm, called FTraj, that protects against context-aware location inference attacks. Trace-driven simulation results show that (1) the current obfuscation algorithms are insufficient to address the vulnerability of vehicles when TFA is applied, and (2) FTraj can effectively protect vehicles' location privacy under the attack of TFA.

We envision several promising directions to continue this research. First, our current work accounts only for a single vehicle, without considering the temporal correlations between multiple vehicles. Also, our framework can be extended to general LBS applications, where service utilities can be defined in different ways. Finally, we will consider different threat models where the information disclosed to attackers is in different formats (e.g., smartphones' accelerometer and gyroscope).

## 8 ACKNOWLEDGEMENTS

This work was partly supported by U.S. NSF grants CNS-2029881, CNS-2029976, National Natural Science Foundation of China (No.62103325), Shaanxi High-level Talent Program (No.2021QCYRC4-26), and Industrial application research project of Shenzhen for undertaking the national key project of China (No. CJGJZD20210408091600002).

## REFERENCES

[1] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. 2013. Geo-indistinguishability: Differential Privacy for Location-based Systems. In *Proc. of ACM CCS*. 901–914.
[2] Apache Hadoop 2.7.3 2022. hadoop.apache.org. Accessed: 2022-06-01.
[3] Apache Spark 1.5.2 2022. spark.apache.org. Accessed: 2022-06-01.
[4] Qasim Arain and et al. 2018. Location Monitoring Approach: Multiple Mix-Zones with Location Privacy Protection Based on Traffic Flow over Road Networks. *Multimedia Tools Appl.* 77, 5 (mar 2018), 5563–5607.
[5] Harsh Bhasin. 2015. *Algorithms: Design and Analysis.* Oxford Univ Press.
[6] Y. Cao, Y. Xiao, L. Xiong, and L. Bai. 2019. PriSTE: From Location Privacy to Spatiotemporal Event Privacy. In *Proc. of IEEE ICDE*. 1606–1609.
[7] Y. Cao, M. Yoshikawa, Y. Xiao, and L. Xiong. 2017. Quantifying Differential Privacy under Temporal Correlations. In *Proc. of IEEE ICDE*. 821–832.
[8] T. Emrich, H. Kriegel, N. Mamoulis, M. Renz, and A. Zufle. 2012. Querying Uncertain Spatio-Temporal Data. In *Proc. of IEEE ICDE*. 354–365.
[9] K. Fawaz and K. G. Shin. 2014. Location Privacy Protection for Smartphone Users. In *Proc. of ACM CCS* (Scottsdale, Arizona, USA). 239–250.
[10] G. D. Forney. 1973. The viterbi algorithm. *Proc. of the IEEE* 61, 3 (1973), 268–278.
[11] B. Gedik and L. Liu. 2005. Location Privacy in Mobile Systems: A Personalized Anonymization Model. In *Proc. of IEEE ICDCS*. 620–629.
[12] G. Ghinita, M. L. Damiani, C. Silvestri, and E. Bertino. 2009. Preventing Velocity-Based Linkage Attacks in Location-Aware Applications. In *Proc. of ACM SIGSPATIAL*. 246–255.
[13] G. Ghinita and et al. 2008. Private Queries in Location Based Services: Anonymizers Are Not Necessary. In *Proc. of ACM SIGMOD*. 121–132.
[14] M. Gruteser and D. Grunwald. 2003. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proc. of ACM MobiSys*.
[15] Frederick S. Hillier. 2008. *Linear and Nonlinear Programming*. Stanford University.
[16] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W. Ma. 2008. Mining User Similarity Based on Location History. In *Proc. of SIGSPATIAL*. Article 34, 10 pages.
[17] W. Li, H. Chen, W. Ku, and X. Qin. 2017. Scalable Spatiotemporal Crowdsourcing for Smart Cities Based on Particle Filtering. In *Proc. of ACM SIGSPATIAL*.
[18] Yanhua Li and et al. 2015. Growing the Charging Station Network for Electric Vehicles with Trajectory Data Analytics. In *Proc. of ICDE*.
[19] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. 2007. Learning and inferring transportation routines. *Artificial Intelligence* 171, 5 (2007), 311 – 331.
[20] MediaQ 2022. MediaQ. https://imsc.usc.edu/platforms/mediaq/. Accessed: 2022-06-01.
[21] OpenStreetMap 2022. https://www.openstreetmap.org/. Accessed: 2022-06-01.
[22] C. Qiu and et al. 2020. Location Privacy Protection in Vehicle-Based Spatial Crowdsourcing via Geo-Indistinguishability. *IEEE TMC* (2020), 1–1.
[23] C. Qiu and et al. 2020. Time-Efficient Geo-Obfuscation to Protect Worker Location Privacy over Road Networks in Spatial Crowdsourcing. In *Proc. of ACM CIKM*.
[24] R. Shokri, G. Theodorakopoulos, C. Troncoso, J. Hubaux, and J. L. Boudec. 2012. Protecting Location Privacy: Optimal Strategy Against Localization Attacks. In *Proc. of ACM CCS*. 617–627.
[25] Daniel W. Stroock. 2010. *Probability Theory: An Analytic View* (2nd ed.). Cambridge University Press.
[26] L. Sweeney. 2002. Achieving K-anonymity Privacy Protection Using Generalization and Suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* 10, 5 (2002).
[27] H. To, G. Ghinita, L. Fan, and C. Shahabi. 2017. Differentially Private Location Protection for Worker Datasets in Spatial Crowdsourcing. *IEEE TMC* (2017), 934–949.
[28] H. To, G. Ghinita, and C. Shahabi. 2014. A Framework for Protecting Worker Location Privacy in Spatial Crowdsourcing. *VLDB Endow.* 7, 10 (June 2014).
[29] Vehicle traffic flow aware attack 2022. https://github.com/chenxiq1986/vehicle-traffic-flow-aware-attack. Accessed: 2022-06-01.
[30] L. Wang, D. Yang, X. Han, T. Wang, D. Zhang, and X. Ma. 2017. Location Privacy-Preserving Task Allocation for Mobile Crowdsensing with Differential Geo-Obfuscation. In *Proc. of ACM WWW*. 627–636.
[31] Y. Xiao and L. Xiong. 2015. Protecting Locations with Differential Privacy under Temporal Correlations. In *Proc. of CCS*. 1298–1309. https://doi.org/10.1145/2810103.2813640
[32] F. Xu, Z. Tu, Y. Li, P. Zhang, X. Fu, and D. Jin. 2017. Trajectory Recovery From Ash: User Privacy Is NOT Preserved in Aggregated Mobility Data. In *Proc. of ACM WWW*. 1241–1250.
[33] L. Yan, H. Shen, J. Zhao, C. Xu, F. Luo, and C. Qiu. 2017. CatCharger: Deploying wireless charging lanes in a metropolitan road network through categorization and clustering of vehicle traffic. In *Proc. of IEEE INFOCOM*. 1–9.
[34] Yelp 2022. https://www.yelp.com/. Accessed: 2022-06-01.
[35] L. Yu, L. Liu, and C. Pu. 2017. Dynamic Differential Location Privacy with Personalized Error Bounds. In *Proc. of IEEE NDSS*.
[36] Y. Zhu, Z. Li, H. Zhu, M. Li, and Q. Zhang. 2013. A Compressive Sensing Approach to Urban Traffic Estimation with Probe Vehicles. *IEEE TMC* 12 (11 2013), 2289–2302. https://doi.org/10.1109/TMC.2012.205