Differential Privacy for Distributed Traffic Monitoring in Smart Cities (Full Version)

Marcus Gelderie[1], Maximilian Luff[1] and Lukas Brodschelm[1]
[1]*Aalen University of Applied Sciences, Beethovenstr. 1,73430 Aalen, Germany*
{firstname.lastname}@hs-aalen.de

Keywords: Differential Privacy,Smart City,Traffic Monitoring

Abstract: We study differential privacy in the context of gathering real-time congestion of entire *routes* in smart cities. Gathering this data is a distributed task that poses unique algorithmic and privacy challenges. We introduce a model of distributed traffic monitoring and define a notion of adjacency for this setting that allows us to employ differential privacy under continual observation. We then introduce and analyze three algorithms that ensure ε differential privacy in this context. First we introduce two algorithms that are built on top of existing algorithmic foundations, and show how they are suboptimal in terms of noise or complexity. We focus, in particular, on whether algorithms can be deployed in our distributed setting. Next, we introduce a novel hybrid scheme that aims to bridge between the first two approaches, retaining an improved computational complexity and a decent noise level. We simulate this algorithm and demonstrate its performance in terms of noise.

1 INTRODUCTION

Smart cities are an ongoing trend in large urban areas, where communities seek to leverage data analytics in order to optimize aspects of their infrastructure. One prominent example of this is smart traffic management (Gade, 2019; Bhardwaj et al., 2022). The goal is to minimize congestion and reduce overall point-to-point travel time. Solutions in this context rely on live data to predict movements and react accordingly. This usually requires tracking vehicles as they move about the city to discern movement patterns that span large parts of or even the entire city (for example, see (Djahel et al., 2015; Khanna et al., 2019; Rizwan et al., 2016)).

From a privacy perspective, however, tracking individual citizens day and night, possibly storing this data at a central location, is, of course, a nightmare. Local legislation (e.g. the GDPR in the EU) may even prohibit some of those solutions, threatening the adoption of modern traffic management. Legal risks aside, massive data collection at a centralized location poses significant risks from a information security perspective (Gracias et al., 2023). Ultimately, cities need solutions that minimize the sensitivity of the data that is stored and reduce the privacy risks to affected individuals.

Differential Privacy (DP) (Dwork et al., 2006; Dwork, 2006) is a well-known tool to design algorithms that give quantifiable privacy guarantees. Much research has gone into developing DP algorithms for various statistical tasks, such as counting, summing, top-k queries and the like (Dwork et al., 2010; Dwork et al., 2015; Chan et al., 2011; Henzinger et al., 2023) (see also *Related Work* below). DP has also been applied to traffic and vehicle data analysis in the past (Hassan et al., 2019; Ma et al., 2019; Zhou et al., 2018; Li et al., 2018; Sun et al., 2021). However the monitoring of city-scale point-to-point traffic movements centrally has, to our knowledge, not been considered before, even though it has been identified as a relevant research topic (Hassan et al., 2019).

We consider the task of monitoring movements of individual vehicles at locally distinct points throughout a city and aggregating that data into a central statistic that captures the number of vehicles traveling along a set of routes within the city limits. We propose three different algorithms that provide ε DP in this setting and compare their relative merits. Specifically, we show how there appears to be a trade-off between the noise incurred by DP and the complexity of the algorithms that run centrally and at local tracking points.

Our contributions are as follows:

- We propose an architecture for distributed traffic monitoring that is generic and applicable in multiple scenarios.

- We develop three DP algorithms for this architecture and analyze their relative noise levels.
- We provide an analysis of the algorithmic properties of our three algorithms, where the input is the size of the city, and the length of the monitoring period. We pay particular attention to whether algorithms can be deployed in a distributed fashion at the various tracking locations throughout the city.

Our three algorithms are designed to showcase the engineering trade-offs that impact the noise level and algorithmic properties. Much depends on the input format and the notion of adjacency that is considered.

For example, counting, as a primitive, has been studied extensively (Dwork et al., 2010) form a DP perspective. The "binary tree technique" introduced in (Dwork et al., 2010) and later studied in (Chan et al., 2011; Henzinger et al., 2023) yields $\log(T)$ noise, where T is the duration of the counting task. But this technique, as we explain, cannot easily be ported to our setting. Instead, the *naive* option of sampling noise per time-step (cf. algorithm 3) outperforms any attempt to port the binary tree technique to our setting in terms of noise. However, it cannot be meaningfully deployed in a distributed way.

Next we introduce algorithm 4, which has a noise bound linear in the number of tracked routes. Note the number of routes itself is exponential in the duration T of tracking, i.e. $R \leq V^{T+1}$, where V is the number of vertices. However, the resulting algorithm can be deployed in a distributed way. Its runtime is also linear in the number of routes, both if deployed centrally or in a distributed way.

Finally, we propose a third probabilistic hybrid scheme (cf. algorithm 5) that bridges between these two approaches. We analyze and simulate this third approach and find that it strikes a balance between both the "naive" approach (algorithm 3) and the "noise per route" approach (i.e. algorithm 4) in both runtime and noise. It is also easily deployable in a distributed way.

Related Work Differential Privacy was introduced for static databases (Dwork et al., 2006; **?**). Subsequently, several algorithms giving $(\varepsilon, 0)$- and (ε, δ)-differentially private queries were introduced (see also (Dwork et al., 2014)). However, since many applications require the continual release of statistics, the notion of continual observation was introduced and has since been studied extensively (Dwork et al., 2010; Chan et al., 2011; Henzinger et al., 2023). As a part of this, the notion of adjacency of input sequences, specifically user-level and event-level privacy, was introduced. User-level privacy requires the continual mechanism to be DP-private independent of how often a individual participates in the accumulated statistical data. This continual counting mechanism has since been improved (Dwork et al., 2015).

Besides counting events, histogram queries are of interest and have been studied. For instance, (Henzinger et al., 2023) considered an intermediate DP histogram in order to continuously release differentially private max-sum, top-k-, and histogram queries. Therefore a changed stream definition was used that modeled each time-step t as a vector of dimensionality d. This parameter is defined by the number of individual partitions of the intermediate histogram. For queries like max-sum or sum-select upper and lower bounds on the accuracy have been established (Jain et al., 2023).

Furthermore, there has been research into settings where the sensitivity between histogram queries is limited but the domain from which the items for the histogram are sampled is unknown (Cardoso and Rogers, 2022). There is also some prior work to calculate dynamic sliding windows to minimize the error bounds of such algorithms (Chen et al., 2023).

However, none of these works fit our use case of continuously releasing traffic statistics that track vehicles through a city. In this setting, counters are not monotonic and many of those algorithms cannot be applied. A dynamically generated sliding window size is not feasible, since we have to reliable get updated counts for different routes each phase. Lastly, our adjacency notion differs significantly from the previous definition (cf. section 3), which presents a significant challenge to porting previous algorithms.

Although differential privacy is a well researched field, smart cities as a possible application pose many distinct challenges (Yao et al., 2023; Husnoo et al., 2021). Privacy is one of the main factors that should be kept in mind when designing smart city systems (Kumar et al., 2022). And (Qu et al., 2019) already stated that smart mobility in particular is one of the main factors for security concerns, since the attacker is able to learn locations of any single individual. There are related works studying DP in scenarios like vehicle-2-X communications in electric-charging or vehicle trajectory estimation, such as (Li et al., 2018; Ma et al., 2019). Those works do not focus on vehicle tracking for the purpose of optimizing city traffic. (Sun et al., 2021) study traffic volume measurement and present an estimator that is DP under certain conditions. The focus is on estimating traffic volume for a given set of locations. In this paper, we study traffic statistics for specific *routes* (ordered sequences of locations) within a city.

2 PRELIMINARIES

Notation Given a set X, we write X^* for all finite (possibly empty) sequences of elements of X. Given $s \in X^*$ we write $s = s_1 \cdots s_l$ with the understanding that $s_i \in X$ for $1 \leq i \leq l$. The length of s is $|s| = l$. A *prefix* is a sequence $s|_i = s_1 \cdots s_i$ where $0 \leq i \leq |s|$. If $i = 0$, then $s|_0 = \phi$ is the empty sequence of length 0. If s' is a prefix of s, we write $s' \preceq s$. The concatenation of sequences $s = s_1 \cdots s_l$ and $s' = s'_1 \cdots s'_{l'}$ is written as $s \,\|\, s' = s_1 \cdots s_l s'_1 \cdots s'_{l'}$.

If we consider intervals of naturals numbers, we sometimes write $[a,b]_{\mathbb{N}} \overset{\text{def}}{=} [a,b] \cap \mathbb{N}$. We may drop the subscript when it is obvious that we refer to intervals of naturals numbers.

Differential Privacy Differential privacy (DP) was introduced by (Dwork et al., 2006; Dwork, 2006) for single databases (see (Dwork et al., 2014) for a comprehensive introduction) and later adapted to the continual setting (Dwork et al., 2010). Here one considers "adjacent" sequences of input. Several notions of adjacency exist (in particular *event level* and *user level* adjacency). We postpone the precise definition of *adjacency* to section 3, where we will discuss why existing definitions of adjacency do not fit our use-case well and propose a more tailored definition. The following definition of DP under continual observation is adapted[1] from (Dwork et al., 2010):

Definition 1. Let $\varepsilon > 0$. Let $\mathcal{A}$ be a randomized algorithm that on input sequence $s_1, \ldots, s_l$, it produces an output sequence $\mathcal{A}(s) \in \Sigma^l$ of the same length.

$\mathcal{A}$ provides ε-*differential privacy*, if for all *adjacent* input streams s, s' and all $S \subseteq \Sigma^*$

$$\Pr[\mathcal{A}(s) \in S] \leq \exp(\varepsilon) \Pr[\mathcal{A}(s') \in S]$$

We sometimes refer to $\mathcal{A}$ as the *curator*.

The following useful "post-processing" theorem allows us to round outputs to the nearest integer without loosing DP. We thus consider only algorithms producing real numbers in this paper.

Theorem 1 (see (Dwork et al., 2014)). *If $\mathcal{A}$ provides ε-differential privacy and $\mathcal{B}$ is any randomized mapping defined on* $\mathsf{range}(\mathcal{A})$, *then $\mathcal{B} \circ \mathcal{A}$ provides ε-differential privacy.*

[1]Different from the original definition, we do not consider internal states and pan-privacy. We are only interested in differentially private outputs.

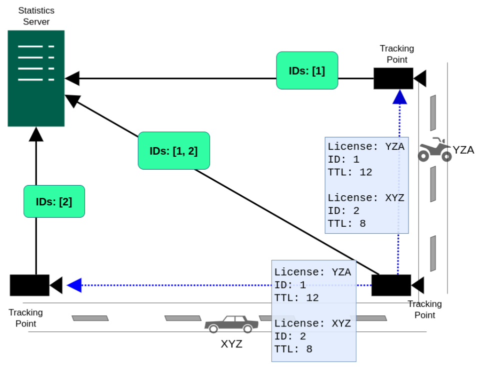


Figure 1: System architecture

We say an algorithm $\mathcal{A}$ has (α, β) error, if for any input sequence s of length L, the maximal difference between the true statistic $f(s)$ and the computed statistic $\mathcal{A}(s)$ is below α with probability at least $1 - \beta$.

$$\Pr\left[\max_{1 \leq t \leq L} \|f(s)_t - \mathcal{A}(s)_t\|_\infty \leq \alpha\right] \geq 1 - \beta$$

Throughout this paper, $\mathsf{Lap}(b)$ refers to the Laplace distribution of scale $b > 0$ with PDF $p(x) = (2b)^{-1} \exp(-|x| \cdot b^{-1})$. We have that

$$p(x) \leq \exp\left(\frac{1}{b}\right) \cdot p(x \pm 1) \qquad (1)$$

3 PROBLEM STATEMENT AND ARCHITECTURE

We consider traffic monitoring in a smart city context. Our goal is to compute the number of of vehicles travelling along a certain route in a given time-period. To this end, vehicles are recorded at specific *tracking points* in the city, such as at traffic lights. This information is aggregated centrally into one statistic about the number of vehicles traveling along a specific route. The overall situation is depicted in fig. 1.

Tracking vehicles requires two pieces of auxiliary data. Once a vehicle is detected, it is assigned a *unique ID* u from a countable set $\mathcal{U}$ of possible IDs along with a *time-to-live (TTL)* that is always initialized to a fixed constant $T \in \mathbb{N}$. Vehicles are recognized via some sort of identifying information (e.g. license plate). For privacy reasons, we hide this information behind a randomized unique ID. The TTL is decremented at each tracking point. Once it reaches zero, the vehicle is no longer tracked. If the same vehicle continues to move through the city, it will instead by assigned a new unique ID $u' \neq u$ and a new TTL. In fact, the new unique ID will satisfy the

```
global : $v \in V$ ID of this tracking point
1 procedure ProcessVehicle
      input : $I$ identifying information
2     $u, t \leftarrow$ Lookup($I$)
3     if $u = \bot$ or $t = 0$ : // new vehicle
4         $u \leftarrow$ SelectNextFreeID()
5         $t \leftarrow T$
6     Report($u, v$)
7     forall $v' \in \mathrm{Adj}(v)$
8         Transmit($I, u, t-1$)
9 end
```

Algorithm 1: Track a single vehicle at a tracking point.

```
1 procedure GatherReports
      input : $u$ unique ID
      input : $v$ location of tracking point
2     $route \leftarrow$ Lookup($u$)
3     $newroute \leftarrow route \,\|\, v$
4     Update($u, newroute$)
      // update statistic
5     Decrement($route$)
6     Increment($newroute$)
7 end
```

Algorithm 2: Gathers reports made by tracking points in algorithm 1.

stronger property that it is distinct from *any* previously used ID. We will elaborate on why this is necessary further below.

We model the city as a directed graph $\mathcal{G} = (V, E)$. The vertices correspond to the tracking points. While it is somewhat arbitrary what a tracking point is in practice, it is perhaps easiest to simply think of tracking points as traffic lights in the context of this paper. Given $\mathcal{G}$, a *route* is a path $r = r_1 \cdots r_m$ through the graph: $(r_i, r_{i+1}) \in E$ for $1 \leq i < m$. Repetitions $r_j = r_i$ for $i \neq j$ are possible. Note that we may assume $m \leq T$, because we never track vehicles along more than T tracking points. Thus the set $\mathcal{R}$ of all possible routes along which a vehicle can be tracked is finite. We write $\mathcal{R}_{\max} = \{r \in \mathcal{R} \mid |r| = T\}$ for the maximal length routes in $\mathcal{R}$. Note $\mathcal{R}$ is prefix-closed.

The overall process (without any concern for differential privacy) is specified in algorithms 1 and 2. While algorithm 1 is executed at every tracking point, algorithm 2 is executed centrally (the gray server in fig. 1). Tracking points report the unique ID of a vehicle and transmit its TTL to neighboring locations $\mathrm{Adj}(v) = \{v' \in V \mid (v, v') \in E\}$. In subsequent DP algorithms, additional data may be transmitted between tracking points (such as "ghost cars" carrying noise). This is explained in section 4.

"Identifying information" is whatever the detection system requires to recognize and identify a specific vehicle. In practice, this could be a license plate, or an identifier stored on an RFID tag. It is possible to re-use the identifying information as a unique ID (see section 4 for how to ensure such a finite set fulfils or requirements for $\mathcal{U}$). We opt for the presented approach to allow for the option to obfuscate license plates before transmitting PII to a central server.

Discussion: The masking of license plates by unique IDs already provides some privacy, but this is difficult to quantify. A vehicle on a very low-traffic route may still be identifiable. DP provides more robust and quantifiable guarantees in this situation.

In practice, it is possible that vehicles take a long time to travel from v to v', even for adjacent $(v, v') \in E$ (e.g. because the driver stops to buy coffee). In such situations, the resulting sequence of locations $v_1, \ldots, v_t$ for that ID will contain "gaps" – time periods, where the vehicle is not tracked anywhere.

In the remainder of this paper, we assume that every vehicle is recorded at every time index until it stops moving or its TTL elapses – i.e. that there are no "gaps" in recording. This is no limitation, because such gaps will not change how our algorithms work. However, accounting for these corner-cases in the mathematical notation below is tedious while adding little value.

In a similar vein, we assume that all tracking points report their data simultaneously. Again, this is not a realistic assumption. But similar to our "no gaps" assumption, it simplifies proofs while not affecting the way our algorithms work.

4 DIFFERENTIAL PRIVACY IN TRAFFIC MONITORING

In this section we propose three different algorithms that ensure differentially private traffic monitoring. The algorithms work on input sequences of different types. However, all such input sequences are (partial) functions from unique IDs to some domain $\mathcal{D}$ (either $\mathcal{R}$ or V). Specifically, all algorithms studied in this paper process sequences of the form $s = (s_1, \ldots, s_L)$, where for each $1 \leq i \leq L$

$$s_i = (s_{i,u})_{u \in \mathcal{U}} \in \mathcal{D}^{\mathcal{U}}$$

for some domain $\mathcal{D}$. Note we sometimes use vector notation $(s_u)_{u \in \mathcal{U}}$ with $s_u \in \mathcal{D}$ instead of functional notation $s(u) \in \mathcal{D}$. We stress that these vectors or functions can be partial. In such cases, we write $s_u = \bot$ if $u \notin \mathsf{dom}(s)$ and require that $\bot \notin \mathcal{D}$.

Differential privacy in continual settings (i.e. where sequences of events are processed) has been studied before (e.g. (Dwork et al., 2010; Jain et al., 2023; Henzinger et al., 2023; Cardoso and Rogers, 2022; Chan et al., 2011)). However, our case is subtly different. To illustrate, we recall the following definition of differential privacy from (Dwork et al., 2010):

Definition 2 (Adjacency)**.** Let $\mathcal{X}$ be some set of events, $L \in \mathbb{N}$ and $s = (s_1, \dots, s_L)$, $s' = (s'_1, \dots, s'_L) \in \mathcal{X}^L$. Then s, s' are *adjacent* if there exists some subset $I \subseteq \{1, \dots, L\}$ and $x, x' \in \mathcal{X}$, such that $s'_i = s_i$ for all $i \notin I$ and $s'_i = x'$ for all $i \in I$ if $s_i = x$.

This definition does not fit our case well, because it is restricted to two *fixed* symbols x, x' that are being exchanged. Simply removing a single tracking point is clearly not enough to hide the presence of a given individual. Instead, we would like to remove a specific unique ID from the entire sequence (or alter its route), which requires changing the value of *several* functions $s \in \mathcal{D}^{\mathcal{U}}$ at *one* point $u \in \mathcal{U}$. We therefore introduce a slightly adapted notion of adjacency. Given any function $f\colon A \to B$, $a \in A$ and $b \in B$, write $f[a/b]$ for the function $f[a/b](x) = f(x)$ for all $x \neq a$ and $f[a/b](a) = b$. We now define:

Definition 3 (ID Adjacency)**.** Let $\mathcal{X} = \mathcal{D}^{\mathcal{U}}$, $L \in \mathbb{N}$ and $s = (s_1, \dots, s_L)$, $s' = (s'_1, \dots, s'_L) \in \mathcal{X}^L$. Then s, s' are *ID adjacent* if there exists some subset $I \subseteq \{1, \dots, L\}$, $u \in \mathcal{U}$ and $d \in \mathcal{D} \cup \{\perp\}$, such that $s'_i = s_i$ for all $i \notin I$ and $s'_i = s_i[u/d]$ for all $i \in I$.

Note that $d = \perp$ is possible, effectively dropping (some) occurrences of u in the sequence.

In this paper we study only ID adjacency. Therefore, whenever we use the word "adjacent" in what follows, we mean ID adjacency.

We can now understand why we assume that IDs from $\mathcal{U}$ are never reassigned. This assumption allows us to define adjacency without considering corner cases, such as whether the occurrences of an ID in s and s' overlap, and without defining what it means for repeated occurrences of the same ID to be "causally connected". In practice, any set of IDs can be made infinite with no risk of reassigning by taking the Cartesian product with the set of all timestamps.

Remark 1. In this paper we consider algorithms that process input sequences of unbounded length. Because of the bounded TTL, differences in two ID adjacent sequences will always affect at most T distinct time steps of the execution of the algorithm

4.1 Route Counting

In this section we study an approach that processes (sequences of) mappings of IDs to routes and simply

```
1 procedure ProcessRoutes
      input : vector $v = (r_u)_{u \in \mathcal{U}}$
2     $c \leftarrow 0 \in \mathbb{R}^{\mathcal{R}}$
3     for $r \in \mathcal{R}$
4         $\mu \leftarrow \mathsf{Lap}(2 \cdot \varepsilon^{-1} \cdot T)$
5         $c_r \leftarrow m_r(v) + \mu$
      output: $c$
6 end
```

Algorithm 3: Counting routes.

counts the number of IDs per route. In this setting, the actual tracking of IDs along routes (cf. algorithm 2) is done as a preprocessing step before the curator is even fed the actual data: In this situation, the curator works as a post-processing step.

In the event that the input to the curator is a (partial) mapping of unique IDs to routes, the statistics function is particularly simple: Given $v \in \mathcal{R}^{\mathcal{U}}$ and $r \in \mathcal{R}$ write $m_r(v) = |\{u \in \mathcal{U} \mid v_u = r\}|$ for the number of IDs that v maps to r. If we enumerate $\mathcal{R} = \{r_1, \dots, r_d\}$, we can write the statistics function $f\colon (\mathcal{R}^{\mathcal{U}})^* \to (\mathbb{N}^d)^*$ as

$$f(s_1, \dots, s_L) = \bar{m}(s_1), \dots, \bar{m}(s_L) \in \left(\mathbb{N}^d\right)^L$$

where $\bar{m}(s_i) = (m_{r_1}(s_i), \dots, m_{r_d}(s_i)) \in \mathbb{N}^d$, $1 \leq i \leq L$. To simplify notation, we write $f^s_{t,r} \overset{\text{def}}{=} ((f(s))_t)_r$ for $1 \leq t \leq L$ and $r \in \mathcal{R}$ in the remainder of this paper.

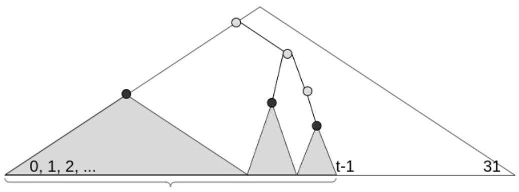
The simplest solution to create differential privacy in this setting is to add noise per reported route. Our setting might seem similar to event counting (Dwork et al., 2010), and so one might expect that the $\log^2(T)$ noise bound from the binary tree mechanism introduced in (Dwork et al., 2010) carries over to our setting. This seems *not* to be the case! We will elaborate on the reasons at the end of this subsection.

For our first algorithm, we therefore rely on the straightforward method of adding independent noise per time step. This is shown in algorithm 3.

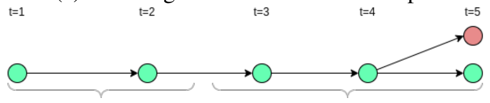
Theorem 2. *Algorithm 3 gives* ε *differential privacy with* $(\varepsilon^{-1} \cdot 2T \ln(\frac{LR}{\beta}), \beta)$ *error on input sequences of length at most L tracking* $R = |\mathcal{R}|$ *routes.*

Proof. Let s, s' be two ID adjacent sequences of length L differing in ID u. Let w.l.o.g. u enter the sequence at time $t = 1$ and let ρ be the route that u takes in s. Let likewise ρ' be the route that u takes in s'. The output $\mathcal{A}(s)$ can be seen a random matrix $(A^s_{t,r})_{1 \leq t \leq L, r \in \mathcal{R}}$, where $A^s_{t,r} = f^s_{t,r} + \mu_{t,r}$ for $\mu_{t,r} \sim \mathsf{Lap}(2T \cdot \varepsilon^{-1})$. The RVs $\mu_{t,r}$ are mutually independent and the PDF of this matrix is

$$p_s(z) = \prod_{t=1}^{L} \prod_{r \in \mathcal{R}} p(z_{t,r} - f^s_{t,r}) \qquad (*)$$



(a) Covering an initial set of time-steps.



(b) Two routes covered by the same set of segments.

Figure 2: Binary Tree Technique and Route Counting.

where p is the $\mathsf{Lap}(2T \cdot \varepsilon^{-1})$ PDF.

Note that for every $1 \le t \le T$ (recall u enters at $t = 1$ and $|\rho| \le T$) we have $\sum_{r \in \mathcal{R}} |f^s_{t,r} - f^{s'}_{t,r}| \le 2$, because there are at most two routes per time-step that differ (where u drives in s and where u drives in s'). If u does not participate at all in either s or s', then the sum is even bounded by 1, and we could work with half the noise. Hence for all $1 \le t \le T$ by eq. (1)

$$\prod_{r \in \mathcal{R}} p(z_{t,r} - f^s_{t,r}) \le \left(\exp\left(\frac{\varepsilon}{2T}\right)\right)^2 \prod_{r \in \mathcal{R}} p(z_{t,r} - f^{s'}_{t,r})$$

and for any $T < t \le L$ we get $\prod_{r \in \mathcal{R}} p(z_{t,r} - f^s_{t,r}) = \prod_{r \in \mathcal{R}} p(z_{t,r} - f^{s'}_{t,r})$. Together we get a factor of $\exp(\varepsilon)$ as desired.

For every $\mu \sim \mathsf{Lap}(b)$ and $\lambda > 0$ we have $\Pr[|\mu| > \lambda \cdot b] = \exp(-\lambda)$ (this is straightforward to verify, e.g. by integrating the PDF of the Laplace distribution). Hence per time-step and route we have $\Pr[|\mu| > -\ln(\eta)\frac{2T}{\varepsilon}] \le \eta$ for any $\eta \in (0,1)$. Let $\eta = \frac{\beta}{LR}$. A union bound over the entire sequence of length L and all routes gives the result. □

For event-counting, (Dwork et al., 2010) introduced a *binary tree mechanism*, that maintains T noise values in a binary tree, but only applies $\log(T)$ of those values to each count. The resulting noise depends only logarithmically on T. However, this technique is not portable to our scenario. The reason is subtle but showcases some of the unique challenges of applying DP to traffic monitoring.

We very briefly sketch the binary tree technique to present a self-contained paper (see (Dwork et al., 2010; Henzinger et al., 2023; Dwork et al., 2014) for a detailed treatment): To implement a counter of events over some sequence of length T, construct a complete binary tree of height $\lceil \log(T) \rceil$ and draw Laplace noise for each node in that tree. Given a leaf $0 \le t \le T-1$, take the sum of the $\log(T)$ noise values along the path from t to the root and add the result μ_t to the counter at time $t+1$. The reason this technique provides ε DP with Laplace noise scaled to only $\log(T) \cdot \varepsilon^{-1}$ lies in the crucial fact that given any initial segment $\{0, \ldots, t-1\}$ of leafs, one can find a set of at most $\log(T)$ nodes in the tree, whose subtrees (a) cover *exactly* the set $\{0, \ldots, t-1\}$ of leafs and (b) are mutually disjoint (see fig. 2a). The fact that only $\log(T)$ such segments are needed is important: Only $\log(T)$ noise values must change to account for a missing (or added) event.

In our case, at any time t and for any non-maximal route r, we do not know which way a given ID will travel. Consider fig. 2b, where two routes (green and red) have the same prefix, but diverge after four steps. The last subtree (indicated by second brace below) spans across the point of divergence. If the count corresponding to this subtree is altered, it will affect *two* routes, not one.

Whenever we use the binary tree technique in our setting, we must ensure that we can find $\log(T)$ noise values that will affect only *one* route of maximal length. There are ways to use multiple trees and achieve ε DP; but they have worse noise properties than algorithm 3 (e.g. maintain one tree per maximal route r^*; for route r at time t, add the noise μ_t for *every* tree $r^* \succeq r$).

4.2 Location Tracking

In the previous section we studied algorithms that process vectors of routes, indexed by IDs from $\mathcal{U}$. The actual tracking of vehicles to map IDs to routes was not part of the curator. In this and the next subsection, we study two algorithms that process sequences of vectors of *locations* – formally vertices in V. The inputs are now sequences of partial functions $s = (s_u) \in V^{\mathcal{U}}$.

To compute the same statistic, $f\colon (V^{\mathcal{U}})^* \to (\mathbb{N}^{\mathcal{R}})^*$ must now track IDs across time-steps. Formally, let s be a sequence of length L, let $1 \le k \le L$ and $r \in \mathcal{R}$. Define

$$c_r(s|_k) = \Big|\{u \in \mathcal{U} \mid \exists t \in [0, T-1]_{\mathbb{N}_0} : \\ s_{k-t}(u) \cdots s_k(u) = r \wedge \forall t' < k-t \colon s_{t'}(u) = \bot\}\Big|$$

Note that we bound the suffix-length by T and require that u does not occur in $s_1, \ldots, s_k$ prior to time $k-t$. That is, the route $r = s_{k-t}(u) \cdots s_k(u)$ is maximal for u at time k. Note also that we implicitly require that $s_j(u) \neq \bot$ for $k-t \le j \le k$ (recall our assumption from section 3 that tracking contains no gaps).

Now the statistics function is given by:

$$f(s) = (c_r(s|_1))_{r \in \mathcal{R}}, \cdots, (c_r(s|_L))_{r \in \mathcal{R}} \in \left(\mathbb{N}^{\mathcal{R}}\right)^L$$

```
global : $R$ database holding routes per ID
global : $L$ database IDs and associated noise
procedure ProcessLocs
    input : $f \in V^{\mathcal{U}}$
    $c \leftarrow 0 \in \mathbb{R}^{\mathcal{R}}$
    forall $\langle u_F, \mu, r \rangle \in L$
        decompose $r = v \,\|\, r'$ with $v \in V$
        if $|r'| = 0$ :
            Drop$(L, u_F)$
        else
            Update$(L, u_F, \mu, r')$
        $c_r \leftarrow c_r + \mu$
    forall $v \in V$
        forall routes $r = v \,\|\, r' \in \mathcal{R}$ starting at $v$
            $\mu \leftarrow \mathsf{Lap}\left(\frac{2}{\varepsilon}\right)$
            $u_F \leftarrow$ SelectNextFreeID()
            Insert$(L, u_F, \mu, r')$
            $c_v \leftarrow c_v + \mu$
        // the actual counting
        forall $u \in f^{-1}(v)$
            // $r = \phi$ if $u$ new ID
            $r \leftarrow$ Lookup$(R, u)$
            $r \leftarrow r \,\|\, v$
            $c_r \leftarrow c_r + 1$
            $r \leftarrow$ UpdateOrInsert$(R, u, r)$
    output: c
end
```

Algorithm 4: Location tracking with Per-Route Noise

Algorithm 4 computes f and adds noise. To this end, at each time t and for every location $v \in V$, noise μ is sampled from $\mathsf{Lap}(2 \cdot \varepsilon^{-1})$ for *each route* r that begins in v. This noise is then assigned an ID and is then sent along that route. A TTL is not required, because it is implicit in the length of the route.

Theorem 3. *Algorithm 4 provides ε-differential privacy with* $\left(2 \cdot \varepsilon^{-1} \cdot \sqrt{8} \cdot \left(R \cdot \ln(2) - \ln\left(\frac{\beta}{2LR}\right)\right), \beta\right)$ *error, where L is the length of the sequence and* $R = |\mathcal{R}|$.

Proof. Denote the algorithm by $\mathcal{A}$. Let s, s' be two ID adjacent sequences of length L differing in ID u. Let w.l.o.g. u enter the sequence at time $t = 1$ and let ρ be the route that u takes in s. The output $\mathcal{A}(s)$ can be seen as a sequence of random vectors $z^s_1, \dots, z^s_L$ where each $z^s_t = (z^s_{t,r})_{r \in \mathcal{R}}$. The random variables $z^s_{t,r}$ and $z^{s'}_{t,r}$ are i.i.d. whenever $t > |\rho|$, or r does not correspond to $\rho|_t$. We therefore consider only $z^s_{t,r}$ for values of $1 \le t \le |\rho|$ and $r = \rho|_t$. We may therefore drop the redundant subscript r in the remainder of this proof, writing z^s_t instead of $z^s_{t,r}$.

There is precisely one noise value sampled at time $t = 1$ for route ρ and this noise value progresses along the route prefixes at times $t = 1, \dots, |\rho|$. Thus

$$\Pr[z^s_t \le x_t] = \Pr[\mu \le x_t - f^s_t]$$
$$= \Pr\left[\mu \le x_t - (f^{s'}_t \pm 1)\right] \le \exp\left(\frac{\varepsilon}{2}\right) \Pr\left[z^{s'}_t \le x_t\right]$$

The same reasoning gives an identical probability bound for the route ρ' that u takes in s'. Since the two noise values are independent, the claim follows by multiplication of probabilities.

Computing the noise bound is more subtle. Using a Chernoff bound, one can show[2] similar to, e.g. (Dwork et al., 2014; Henzinger et al., 2023):

Proposition 4. *Let* $X_1, \dots, X_n \sim \mathsf{Lap}(b)$ *be i.i.d. RVs,* $b > 0$. *Write* $Y = \sum_{i=1}^n X_i$, *then* $\forall \beta \in (0,1)$: $\Pr\left[|Y| > \sqrt{2}b\left(n\ln(2) - \ln\left(\frac{\beta}{2}\right)\right)\right] \le \beta$

When $R_v = |\{r \in \mathcal{R} \mid \text{first}(r) = v\}|$ noise values are sampled at some time t, their values affect the count of route v at time t. At subsequent time steps, the number $n_{t,r}$ of those noise values affecting the count for any extension of $v \preceq r$ can only be lower (the noise "fans out" across all possible routes starting in v). Hence, $n_{t,r} \le R_v \le R$ for all t and all $r \in \mathcal{R}$. Thus, using the above proposition we have:

$$\Pr\left[|z_{t,r} - f_{t,r}| > \frac{\sqrt{8} \cdot \left(R \cdot \ln(2) - \ln\left(\frac{\beta'}{2}\right)\right)}{\varepsilon}\right] \le \beta'$$

for any $\beta \in (0,1)$ and any t, r. Setting $\beta' = \frac{\beta}{LR}$ and taking a union bound proves the claim. □

Remark 2. Algorithm 4 can be adapted to run at the tracking points in a distributed fashion. Each tracking point $v \in V$ then requires access to the set $\mathcal{R}_{\max}(v)$ of maximal routes starting in v (or needs to re-compute this set at every time-step on input $\mathcal{G}$). This spreads out the runtime across all points $v \in V$, but it remains exponential in $\mathcal{G}$ at each tracking point.

4.3 Location Tracking – A Hybrid Approach

In this subsection, we study a hybrid and probabilistic approach that bridges between algorithms 3 and 4. Where algorithm 4 sampled noise per route at each time-step t and then sends that noise along its corresponding route, we now use a random number $n \in \mathbb{N}_0$ of noise values that we sample at each location $v \in V$ at each time-step t. For each of those n noise values, a neighbor $v' \in \text{Adj}(v)$ of v is chosen uniformly

[2] A proof can be found in the appendix.

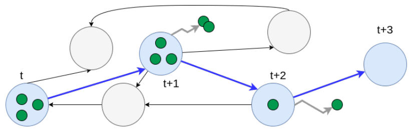


Figure 3: A number $n = 3$ of ghost cars traverse $\mathcal{G}$ starting at time t. The last car leaves route r at time $t+2$.

at random and the noise is passed along to v'. During step $t+1$, that noise value is again sent on to another neighbor $v'' \in \mathrm{Adj}(v')$ and so on, until T steps have elapsed. In essence, the n noise values behave like n "ghost cars" that perform a random walk on $\mathcal{G}$. This situation is depicted in fig. 3.

Since the noise values travel along a random path, it is possible that a given route is without noise at some time-step t. This destroys differential privacy, if left untreated[3]. However, this event is detectable and can be solved by falling back to algorithm 3.

Algorithm 5 implements this idea. Note that the magnitudes of the parameter b of the Laplace distribution for the two kinds of noise differ substantially. This algorithm, as defined, is based on the assumption that no car takes a route shorter than T. It can be adapted to work for the general case, by adding dedicated ghost cars per possible route length $1,\dots,T$.

Theorem 5. *Algorithm 5 provides ε differential privacy, provided the adjacent sequences differ in a route of length T.*

For the proof we require the following lemma[4]. For $p\colon \mathbb{R} \to \mathbb{R}$, let $p^{(1)} = p$ and $p^{(k)} = p * p^{(k-1)}$ (where $*$ is the convolution), $k \geq 2$. It is well known that if p is the PDF of k i.i.d. random variables, then $p^{(k)}$ is the PDF of their sum.

Lemma 6. *Let $b > 0$ and $p(x)$ be the* $\mathsf{Lap}(b)$ *PDF. Then* $\exp(b) \cdot p^{(k)}(x) \leq p^{(k)}(x+1) \leq \exp\left(\frac{1}{b}\right) \cdot p^{(k)}(x)$ *for all $x \in \mathbb{R}$ and $1 \leq k \in \mathbb{N}$.*

Proof of Theorem 5. Let s, s' be two adjacent sequences of length L. Let u be the differing ID and let ρ be the route u takes in s, ρ' be the route u takes in s'. We assume $|\rho| = |\rho'| = T$. The case where u does not occur in either s or s' is again treated in the same way.

Write the output of $\mathcal{A}(s)$ as a sequence of random vectors $z_1^s, \dots, z_L^s$, where for each $1 \leq t \leq L$ and each $r \in \mathcal{R}$ we write $z_{t,r}^s$ for the component at $r \in \mathcal{R}$. Suppose u enters at time t_0 and leaves at time t_1. Then the random variables $z_{t,r}^s$ and $z_{t,r}^{s'}$ are i.i.d. whenever $t \notin [t_0, t_1]$ or whenever r is not a prefix of ρ or ρ'. It is thus sufficient to consider the case where $t \in [t_0, t_1]$

```
   global : $R$ database holding routes per ID
   global : $L$ database IDs and associated noise
1  procedure ProcessLocs
     input : $f \in V^{\mathcal{U}}$
2    $c \leftarrow 0 \in \mathbb{R}^{\mathcal{R}}$
3    covered $\leftarrow (\bot, \dots, \bot)$
4    forall $\langle u_F, \mu, r, l \rangle \in L$
5      if $l = 0$ :
6        Drop$(L, u_F)$
7        continue
8      $v \leftarrow$ Uniform[Adj(last$(r)$)]
9      $r' \leftarrow r \| v$
10     Update$(L, u_F, \mu, , l-1)$
11     $c_{r'} \leftarrow c_{r'} + \mu$
12     covered$_{r'} \leftarrow \checkmark$
13   forall $v \in V$
14     while continue with probability $p$
15       $\mu \leftarrow$ Lap$\left(\frac{2}{\varepsilon}\right)$
16       $l \leftarrow T$
17       $u_F \leftarrow$ SelectFreeID()
18       Insert$(L, u_F, \mu, v, l)$
19       $c_v \leftarrow c_v + \mu$
20       covered$_v \leftarrow \checkmark$
       // the actual counting
21     forall $u \in f^{-1}(v)$
         // $r = \phi$ if $u$ is new ID
22       $r \leftarrow$ Lookup$(R, u)$
23       $r \leftarrow r \| v$
24       $r \leftarrow$ UpdateOrInsert$(R, u, r)$
25       $c_r \leftarrow c_r + 1$
     // fallback: noise per
       time-step
26   forall $r \in \mathcal{R}$ with covered$_r \neq \checkmark$
27     covered$_r = \checkmark$
28     $c_r \leftarrow c_r +$ Lap$\left(\frac{2T}{\varepsilon}\right)$
     output: c
29 end
```

Algorithm 5: Location tracking – Hybrid Approach

and $r \preceq \rho$. We focus on ρ first. The proof for ρ' is identical.

To simplify notation, we let w.l.o.g. $t_0 = 1$. Moreover, since we only consider prefixes of ρ, it is sufficient to drop all references to r from our notation and refer only to time index t, which then uniquely determines the prefix $\rho|_t$. That is, we write z_t^s for $z_{t,\rho|_t}^s$. We also drop the superscript s whenever we do not need to distinguish between the output of $\mathcal{A}$ on s or s'.

Noise values travel along prefixes of ρ until, at some time t, they follow along different path (in line 8 of algorithm 5). Otherwise, the noise value moves on to the next longer prefix of length $t+1$. In this event,

[3]We do obtain (ε, δ) differential privacy for $\delta > 0$ in this case, but δ is prohibitively large.

[4]A proof can be found in the appendix.

we say the noise value is *alive* (at time $t+1$).

Note that for $t' \leq t$, the random variables z_t and $z_{t'}$ are *not* independent. In fact, the event that a noise variable μ is alive at t implies that is alive at t'. But during execution of the algorithm and for any route, there exists a time-index $\tau \in [0,T]$, such that some noise is alive at τ, but no noise is alive at $\tau+1$. Note that it is possible that τ corresponds to the length of the route, in which case noise remained alive for the entire duration of the route. In the other extreme, $\tau = 0$ means that no noise was sampled.

Writing $z^s \leq x$ for $z_i^s \leq x_i$, $1 \leq i \leq |\rho|$, we now have

$$\Pr[z^s \leq x] = \sum_{i=0}^{|\rho|} \Pr[\tau = i] \cdot \prod_{j=i+1}^{|\rho|} \Pr\left[z_j^s \leq x_j\right] \cdot \Pr\left[z_1^s \leq x_1, \ldots, z_i^s \leq x_i \mid \tau = i\right] \quad (2)$$

because for $j > \tau$, the values of $z_{j,}$ are independent (new noise is sampled in each step). Given $\tau = i$:

$$\begin{aligned}\prod_{j=i+1}^{|\rho|} \Pr\left[z_j^s \leq x_j\right] &= \prod_{j=i+1}^{|\rho|} \Pr\left[\mu \leq x_j - f_j^s\right] \\ &\leq \exp\left(\frac{\varepsilon \cdot (|\rho| - i)}{2T}\right) \prod_{j=i+1}^{|\rho|} \Pr\left[\mu \leq x_j - f_j^{s'}\right] \\ &\leq \exp\left(\frac{\varepsilon}{2}\right) \prod_{j=i+1}^{|\rho|} \Pr\left[\mu \leq x_j - f_j^{s'}\right]\end{aligned}$$

As for the second term in eq. (2): For any fixed i, in the event $\tau = i$, denote by $N_j = \{\mu_1, \ldots, \mu_{m_j}\}$ the m_j noise values that are still alive at time $1 \leq j \leq i$. Clearly $N_1 \supseteq N_2 \supseteq \cdots \supseteq N_i \neq \emptyset$. So $\mu_1, \ldots \mu_{m_1}$ is an enumeration of all noise values affecting $z_1, \ldots z_i$ and $\mu_1, \ldots, \mu_{m_i}$ (note $m_i \leq m_1$) is the subset of those that are still alive at time i.

$$\begin{aligned}&\Pr\left[z_1^s \leq x_1, \ldots, z_i^s \leq x_i \mid \tau = i\right] \\ &= \Pr\left[\sum_{k=1}^{m_1} \mu_k \leq x_1 - f_1^s, \ldots, \sum_{k=1}^{m_i} \mu_k \leq x_i - f_i^s\right]\end{aligned}$$

For $\sigma_i \overset{\text{def}}{=} \sum_{k=1}^{m_i} \mu_k$ we get by lemma 6:

$$\Pr\left[\sigma_i \leq x_i - f_i^s \pm 1\right] \leq \exp\left(\frac{\varepsilon}{2}\right) \Pr\left[\sigma_i \leq x_i - f_i^{s'}\right]$$

Moreover, in this event all remaining sums also increase or decrease by 1. By $|\rho| = T \geq \tau$ the term $f_i^s - f_i^{s'}$ is constant in i. The claim follows by applying the same calculation to ρ'. □

Note that algorithm 5 can be implemented in a distributed fashion: by transmitting the noise values for the ghost cars to neighboring tracking-points and reporting them to the central server as part of their regular location reports. The tracking-points would *not* provide DP by themselves, because they still have to transmit an ID, which necessarily differs between two adjacent sequences of inputs.

The noise bound for algorithm 5 is difficult to state and prove, because it is an amalgamation of the noise bounds for a sum of Laplacians and the noise bound for algorithm 3. We instead simulate the algorithm in the following section.

5 COMPARISON OF APPROACHES

5.1 Simulation

It is clear that algorithm 3 has better noise properties than algorithm 4. But when we consider algorithm 5, the picture is not so clear. This algorithm shares elements with algorithm 4, but spawns less noise on the first hops of a route (depending on p). On the other hand, it periodically falls back to algorithm 3. It is also not clear whether the ghost cars spawned in algorithm 5, which produce less noise per car, will survive long enough to reduce the overall noise. Note that if too many such cars are spawned, their combined noise might dominate the overall noise value and we converge to algorithm 4. To investigate this, we implemented the hybrid approach and algorithm 3 using `numpy` (Harris et al., 2020). We then compared these two approaches by simulating each one for a total of $m = 10000$ times using evenly spaced values for ε between 0.1 and 1.0. We seeded our RNG with a pseudo-random seed chosen as the SHA-256 hash of the string "ICISSP'24 Simulation" to produce reproducible yet unbiased results. The code for our evaluation can be found here. Moreover, we chose multiple values for p between 0.6 and 0.99. However, the value of T and the probability of a noise value staying alive along a route are somewhat more difficult to select and model.

The probability of a noise values being alive at any given time clearly depends on the route and specifically on the degrees of the vertices along that route. We approximated the probability of selecting the next neighbor for a ghost car by $\frac{1}{d}$, where d is the degree of the graph. Clearly d depends heavily on the city and will vary in general. However, we posit that most crossings have ≈ 4 ingoing edges, rarely more. Moreover, it is often not permitted to "U-turn". We thus chose to set $d = 3$. However, as we will see next, setting $d = 4$ gives similar results.

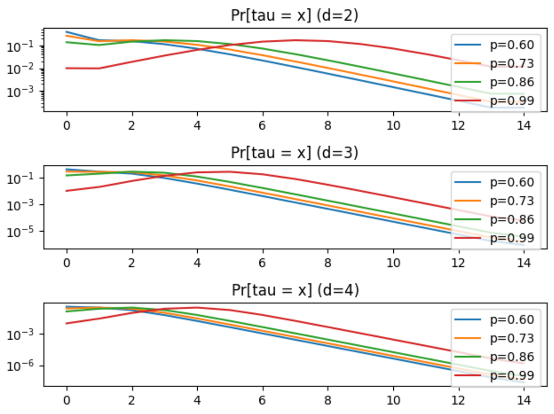


Figure 4: $\Pr[\tau = i]$ for various degrees and values of i.

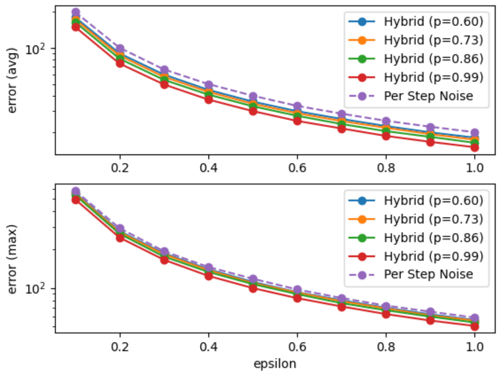


Figure 5: Noise values ($d = 3$).

To find a meaningful value for T, we need to see how long noise will remain alive on average. Recall τ is the RV denoting the last position along a route where noise is still alive. Writing $\delta(v)$ for the degree of $v \in V$, it can be shown[5]:

Proposition 7. *Let $T \in \mathbb{N}$, and $\mathcal{G}$ a graph. Fix a route $r \in \mathcal{R}$ and write $q_i = \prod_{j=1}^{i-1} \frac{1}{\delta(r_i)}$ for $1 \le i \le T$. Define $q_{T+1} \overset{def}{=} 0$. Then*

$$\Pr[\tau = i] = \begin{cases} (1-p) & i = 0 \\ \frac{(1-p)p(q_i - q_{i+1})}{(1-p(1-q_{i+1}))(1-p(1-q_i))} & 1 \le i \le T \end{cases}$$

Using this result, we plotted the probabilities of $\Pr[\tau = i]$ for various values of i. The results can be seen in fig. 4. We see that for low and (in our opinion) plausible values of d, noise rarely survives beyond $t = 8$. However, we also see that if routes offer only two outgoing edges per tracking point, the survival rate of noise is much higher. Because we believe $d = 2$ to be a fringe-case, we chose $T = 10$. We also see from the figure that $d = 3$ and $d = 4$ do not differ much in terms of where the probability mass is mostly concentrated (around $t = 2$ to $t = 6$).

We then performed measurements using $T = 10$, $d = 3$ and plotted the maximum and average[6] noise for various values of p and ε (note that parameter p does not affect algorithm 3). The results are shown in table 1 and plotted in fig. 5.

We can see that the hybrid approach outperforms algorithm 3 both in terms of maximum and in terms of average noise. The benefit is present over the entire range of ε values. Algorithm 3 requires roughly 1.33× the average noise of the hybrid approach and roughly 1.17× the maximum noise over the entire range of values for ε that were simulated.

Note that increasing values of p achieve increasingly better results. This is somewhat surprising, because large values of p also imply a large number of ghost cars which then carry noise. It seems that with the given parameters, $p = 0.99$ is small enough for the benefits to outweigh the costs.

5.2 Computational Complexity

We close this section by comparing the algorithmic properties of the three algorithms. We consider two deployment scenarios:

Centralized The algorithm runs completely in the context of the curator. The tracking points behave as is algorithm 1.

Distributed Part of the algorithm is executed at the tracking points. These parts differ between algorithms. The tracking-points execute these parts in addition to algorithm 1.

We begin by defining how we measure algorithmic complexity, which is traditionally measured in terms of input size. In our case, this might mean the input per time step, the overall design parameters (e.g. the graph $\mathcal{G}$ or the duration T, or both). We will focus on the design parameters, disregarding the input per time-step.

The input per time-step is misleading: Consider algorithm 3 which has inputs of the form $v \in \mathcal{R}^{\mathcal{U}}$. It runs linearly in this vector. However note the vector first needs to be built from the reports by individual tracking points (cf. section 3). On the other hand, algorithms 4 and 5 compute the same vector internally from an input proportional to the number P of IDs currently moving through the in the city. Ultimately, both algorithms need to process all routes, which is a function of the number T and the graph $\mathcal{G}$. Hence, we

[5]See the appendix for a proof.

[6]The average is computed for the *absolute* values. The expected noise of (sums of) Laplacians centered at 0 is 0. The absolute values give a more meaningful picture.

Table 1: Noise Values ($d = 3$)
(a) Hybrid (Algorithm 5)

p	ε									
	0.1000	0.2000	0.3000	0.4000	0.5000	0.6000	0.7000	0.8000	0.9000	1.0000
					Average Noise					
0.6	179.76	90.1546	60.187	44.9128	35.9133	29.8941	25.7821	22.5026	19.9598	18.0358
0.73	173.044	86.6364	57.6857	43.294	34.4829	28.7725	24.7632	21.7792	19.2939	17.3447
0.86	164.259	81.5817	54.4553	41.058	32.8701	27.3812	23.3966	20.4218	18.3088	16.3498
0.99	149.67	74.3359	49.8462	37.3506	29.9638	24.8269	21.5065	18.6528	16.6512	15.0286
					Max Noise					
0.6	558.502	278.99	187.068	140.083	111.4	92.5214	79.889	69.8763	61.9741	55.9016
0.73	550.002	274.61	183.93	137.93	109.379	91.9244	78.5733	69.035	60.8245	55.3188
0.86	537.232	266.589	176.419	133.537	107.124	89.3853	75.8848	66.7421	59.4957	53.4784
0.99	497.295	248.684	166.418	124.294	99.7349	83.0822	71.5099	62.2997	55.4173	50.2797

(b) Algorithm 3

	ε									
	0.1000	0.2000	0.3000	0.4000	0.5000	0.6000	0.7000	0.8000	0.9000	1.0000
avg	199.072	100.465	66.4164	50.1661	40.165	33.098	28.4976	24.8917	22.2226	20.0027
max	580.695	294.562	194.252	145.865	118.074	97.1609	83.2701	72.5793	65.0409	58.3871

measure runtime not in the input per time-step, but in the overall design parameters $\mathcal{G} = (V, E)$, T, and (in the case of algorithm 5) p. Usually we will use the proxy variable $|\mathcal{R}| \leq \sum_{i=1}^{T} |V|^i \leq |V|^{T+1}$.

We will treat sampling a Laplacian as constant cost. Likewise, we will treat all database-lookups as constant costs (which, in the average case, can be achieved using an appropriately sized hash-table).

Finally, it is easy to see that both algorithms 3 and 4 are asymptotically linear in the number $|\mathcal{R}|$ of routes. From an asymptotic perspective, the algorithms perform almost equally well. In our opinion, this unfairly hides the fact that algorithm 4 needs to iterate over $\mathcal{R}$ twice (or perform twice the work per loop iteration) compared with algorithm 3. We therefore do not consider asymptotic complexity, but rather count loop iterations in terms of the parameters laid out above.

Centralized Algorithm 3 run linearly in $|\mathcal{R}|$. If we add computing the statistic from per-vehicle reports to its runtime, then it runs in time $|\mathcal{R}| + P$ (where again P is the number of participants in the system at the given time-step). Algorithm 3 requires no storage. Again, if we add the task of computing the statistic from per-vehicle reports, some sort of mapping of ID to route-prefixes is needed and we require storage on the order of P.

Algorithm 4 runs in time $|\mathcal{R}_{\max}| + |\mathcal{R}| + P$, where the first term is to spawn one ghost-car per route, the second is to add the noise to the route-counts and to propagate the ghost-cars, and the third is to compute the actual noise-counts from per-vehicle reports. Storage is required to store both the ghost cars in the system and the ID-to-route mapping per participant. This means storage on the order of $P + |\mathcal{R}|$ is needed.

Finally, algorithm 5 runs in time $P + |\mathcal{R}| + \frac{T \cdot |V| \cdot p}{1-p}$. Note that $\frac{p}{1-p}$ is the expected number of ghost-cars spawned for each $|V|$. They remain in the system for T rounds. These cars need to be propagated in every step. Additionally, the P reports by “real” need to be processed. Finally, we need to check for each route, if noise was added to it and fall back to algorithm 3 otherwise. Space is required to store the route (and possibly noise) for all real and ghost cars, meaning storage on the order of $P + \frac{|V| \cdot T \cdot p}{1-p}$.

Distributed Only algorithms 4 and 5 can be implemented in a distributed fashion. Algorithm 4 would offload creating ghost noise per $r \in \mathcal{R}_{\max}$ to each $v \in V$. The tracking points forward the ghost cars to each other just like they do for real cars (cf. algorithm 1). The runtime per tracking point $v \in V$ then is $P_v + |\mathcal{R}_{\max}(v)|$, where P_v is the number of participants at v and $\mathcal{R}_{\max}(v)$ is the set of maximal length routes beginning in v. Storage is required only to store the set $\mathcal{R}_{\max}(v)$. The curator now reduces to algorithm 2 and runs in time $P + |\mathcal{R}|$.

Algorithm 5 similarly offloads the generation of ghost cars to the tracking points. These now run in time at most $\frac{T \cdot |V| \cdot p}{1-p} + P_v$ each. They might run significantly faster on average, depending on the structure of $\mathcal{G}$: The $\frac{T \cdot |V| \cdot p}{1-p}$ ghost cars will distribute over $\mathcal{G}$ in general, but not necessarily uniformly (vertices receive ghost cars proportional to their in-degree in every time-step). The tracking points need only store a representation of $\mathrm{Adj}(v)$, which is required for algorithm 1 anyway. The curator is as above.

Note that the overall storage now seems much lower. This is misleading, because the actual storing of noise to ID bindings happens at the network link level, which witnesses an increased load as a result of the decentralized deployment.

Discussion We see that while algorithm 4 is optimal in terms of noise, it has worst complexity in the centralized setting and the distributed setting. Algorithm 3 on the, other hand, cannot be implemented in a distributed fashion and incurs a large amount of noise. We can see that algorithm 5 strikes a balance between both algorithms in terms of runtime and space – both in the centralized and distributed settings – provided p is chosen appropriately. The previous subsection showed that it can outperform algorithm 3 in terms of noise to some extent.

6 CONCLUSION

We have introduced a model for decentralized traffic monitoring in the smart city based on vehicle tracking. We then introduced a notion of adjacency that fits this model and permits us to study ε DP in this context. Building on that, we presented three algorithms that each achieve ε DP in our setting. Each algorithm has unique advantages and disadvantages in terms of noise, runtime, and their ability to be deployed in a distributed fashion. Together, they showcase the various engineering tradeoffs that a practioner might encounter when applying this model to a specific city.

Our first algorithm is simple to implement, yet incurs high noise as it requires Laplace noise scaled to T. Moreover, it cannot be run in a distributed fashion. Remarkably, despite the superficially similar setting, the well-known binary tree technique cannot be ported to this setting. We next showed that a dependence on T in terms of noise is altogether unnecessary. The resulting algorithm can run in a distributed fashion, but incurs high runtime overhead. Our third algorithm is a hybrid approach that strikes a balance between the first two algorithms. Its noise bound is difficult to compute due to its hybrid and probabilistic nature. Yet it is reasonably efficient in a distributed setting. Moreover, when simulated, it outperforms the first algorithm in terms of noise.

We believe that our results show the merit of hybridizing DP algorithms in a probabilistic way, specifically when working in the presented context of traffic monitoring. While the resulting algorithms are more difficult to analyze, they perform reasonably well. We believe that future work can improve this situation. For example, hybrid schemes could adapt to the topology of the graph, covering routes that have a higher probability of "loosing" a ghost with a higher number of the same. Finally, the benefits of adopting (ε, δ) DP, where $\delta > 0$, may hold significant improvements in terms of noise at the cost of a modest imbalance in the privacy guarantees.

ACKNOWLEDGEMENTS

This work was done as part of a project funded by the German Federal Office for Information Security under project funding reference number 01MO23006.

REFERENCES

Bhardwaj, V., Rasamsetti, Y., and Valsan, V. (2022). Traffic control system for smart city using image processing. *AI and IoT for Smart City applications*, pages 83–99.

Cardoso, A. R. and Rogers, R. (2022). Differentially private histograms under continual observation: Streaming selection into the unknown. In *International Conference on Artificial Intelligence and Statistics*, pages 2397–2419. PMLR.

Chan, T.-H. H., Shi, E., and Song, D. (2011). Private and continual release of statistics. *ACM Transactions on Information and System Security (TISSEC)*, 14(3):1–24.

Chen, Q., Ni, Z., Zhu, X., and Xia, P. (2023). Differential privacy histogram publishing method based on dynamic sliding window. *Frontiers of Computer Science*, 17(4):174809.

Djahel, S., Doolan, R., Muntean, G.-M., and Murphy, J. (2015). A communications-oriented perspective on traffic management systems for smart cities: Challenges and innovative approaches. *IEEE Communications Surveys & Tutorials*, 17(1):125–151.

Dwork, C. (2006). Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, pages 265–284. Springer.

Dwork, C., Naor, M., Pitassi, T., and Rothblum, G. N. (2010). Differential privacy under continual observation. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 715–724, New York, NY, USA. Association for Computing Machinery.

Dwork, C., Naor, M., Reingold, O., and Rothblum, G. N. (2015). Pure differential privacy for rectangle queries via private partitions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 735–751. Springer.

Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407.

Gade, D. (2019). Ict based smart traffic management system “ismart” for smart cities. *International Journal of Recent Technology and Engineering*, 8(3):1000–1006.

Gracias, J. S., Parnell, G. S., Specking, E., Pohl, E. A., and Buchanan, R. (2023). Smart cities—a structured literature review. *Smart Cities*, 6(4):1719–1743.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Hassan, M. U., Rehmani, M. H., and Chen, J. (2019). Differential privacy techniques for cyber physical systems: a survey. *IEEE Communications Surveys & Tutorials*, 22(1):746–789.

Henzinger, M., Sricharan, A., and Steiner, T. A. (2023). Differentially private data structures under continual observation for histograms and related queries. *arXiv preprint arXiv:2302.11341*.

Husnoo, M. A., Anwar, A., Chakrabortty, R. K., Doss, R., and Ryan, M. J. (2021). Differential privacy for iot-enabled critical infrastructure: A comprehensive survey. *IEEE Access*, 9:153276–153304.

Jain, P., Raskhodnikova, S., Sivakumar, S., and Smith, A. (2023). The price of differential privacy under continual observation. In *International Conference on Machine Learning*, pages 14654–14678. PMLR.

Khanna, A., Goyal, R., Verma, M., and Joshi, D. (2019). Intelligent traffic management system for smart cities. In Singh, P. K., Paprzycki, M., Bhargava, B., Chhabra, J. K., Kaushal, N. C., and Kumar, Y., editors, *Futuristic Trends in Network and Communication Technologies*, pages 152–164, Singapore. Springer Singapore.

Kumar, A., Upadhyay, A., Mishra, N., Nath, S., Yadav, K. R., and Sharma, G. (2022). Privacy and security concerns in edge computing-based smart cities. In *Robotics and AI for Cybersecurity and Critical Infrastructure in Smart Cities*, pages 89–110. Springer.

Li, Y., Zhang, P., and Wang, Y. (2018). The location privacy protection of electric vehicles with differential privacy in v2g networks. *Energies*, 11(10):2625.

Ma, Z., Zhang, T., Liu, X., Li, X., and Ren, K. (2019). Real-time privacy-preserving data release over vehicle trajectory. *IEEE transactions on vehicular technology*, 68(8):8091–8102.

Qu, Y., Nosouhi, M. R., Cui, L., and Yu, S. (2019). Privacy preservation in smart cities. In *Smart cities cybersecurity and privacy*, pages 75–88. Elsevier.

Rizwan, P., Suresh, K., and Babu, M. R. (2016). Real-time smart traffic management system for smart cities by using internet of things and big data. In *2016 International Conference on Emerging Technological Trends (ICETT)*, pages 1–7.

Sun, Y.-E., Huang, H., Yang, W., Chen, S., and Du, Y. (2021). Toward differential privacy for traffic measurement in vehicular cyber-physical systems. *IEEE Transactions on Industrial Informatics*, 18(6):4078–4087.

Yao, A., Li, G., Li, X., Jiang, F., Xu, J., and Liu, X. (2023). Differential privacy in edge computing-based smart city applications: Security issues, solutions and future directions. *Array*, page 100293.

Zhou, Z., Qiao, Y., Zhu, L., Guan, J., Liu, Y., and Xu, C. (2018). Differential privacy-guaranteed trajectory community identification over vehicle ad-hoc networks. *Internet Technology Letters*, 1(3):e9.

APPENDIX

Proofs Section 4

Proof of Proposition 4

Proposition. *Let $X_1, \dots, X_n \sim \mathsf{Lap}(b)$ be i.i.d. random variables, where $b > 0$. Denote their sum by $Y = \sum_{i=1}^n X_i$. Then for all $\beta \in (0,1)$:*

$$\Pr\left[|Y| > \sqrt{2}b\left(n\ln(2) - \ln\left(\frac{\beta}{2}\right)\right)\right] \leq \beta$$

The proof uses the well-known Chernoff bound, which is a common tool in analysing noise levels of DP algorithms (Dwork et al., 2014; Henzinger et al., 2023).

Lemma. *Let $X_1, \dots, X_n \sim \mathsf{Lap}(b)$ be i.i.d. random variables, where $b > 0$. Denote their sum by $Y = \sum_{i=1}^n X_i$. Then for all $\alpha \in (0, b^{-1})$ and $\lambda > 0$:*

$$\Pr[|Y| > \lambda] \leq 2\exp\left(-\alpha\lambda + n \cdot \ln\left(\frac{1}{1-\alpha^2 b^2}\right)\right)$$

Proof. We use the well-known fact that the moment-generating function $\mathbb{E}[\exp(\alpha \cdot X)] = \frac{1}{1-\alpha^2 b^2}$ for any random variable $X \sim \mathsf{Lap}(b)$, whenever $|\alpha| < b^{-1}$. Using a Chernoff bound, we get for all $\alpha \in (0, b^{-1})$:

$$\begin{aligned}\Pr[Y > \lambda] &\leq \exp(-\alpha\lambda) \cdot \mathbb{E}[\exp(\alpha \cdot Y)] \\ &= \exp(-\alpha\lambda) \cdot \mathbb{E}[\exp(\alpha \cdot X_1)]^n \\ &= \exp(-\alpha\lambda) \cdot \left(\frac{1}{1-\alpha^2 b^2}\right)^n \\ &= \exp\left(-\alpha\lambda + n \cdot \ln\left(\frac{1}{1-\alpha^2 b^2}\right)\right)\end{aligned}$$

By replacing α by $-\alpha$ we obtain an identical probability bound for the event $Y < -\lambda$. □

If we denote the exponential in the above lemma by β and solve for λ, we obtain:

Corollary. *Let $b, \alpha, X_1, \dots, X_n$ and Y be as above. Let $\beta \in (0,1)$. Then*

$$\Pr\left[|Y| > \frac{n\ln\left(\frac{1}{1-\alpha^2 b^2}\right) - \ln\left(\frac{\beta}{2}\right)}{\alpha}\right] \leq \beta$$

Proposition 4 now follows by setting $\alpha = \frac{1}{\sqrt{2} \cdot b}$. □

Proof of Lemma 6

Let $p \colon \mathbb{R} \to \mathbb{R}$. For $2 \leq k \in \mathbb{N}$ write

$$p^{(k)} = \underbrace{p * \cdots * p}_{k \text{ factors}}$$

for the k-fold convolution of p with itself. For $k = 1$ we let $p^{(k)} = p$.

Lemma. *Let $b > 0$ and let $p(x) = (2b)^{-1}\exp(-|x| \cdot b^{-1})$ be the $\mathsf{Lap}(b)$ PDF. Then we have both*

$$p^{(k)}(x) \leq \exp\left(\frac{1}{b}\right) \cdot p^{(k)}(x+1)$$

and

$$p^{(k)}(x+1) \leq \exp\left(\frac{1}{b}\right) \cdot p^{(k)}(x)$$

for all $x \in \mathbb{R}$ and $1 \leq k \in \mathbb{N}$.

Proof. Let first $k \geq 2$. We have by associativity of the convolution operator that $p^{(k)} = p * p^{(k-1)}$ and so:

$$\begin{aligned}&p^{(k)}(x+1) \\ &= \int_{-\infty}^{\infty} p(x+1-\tau) \cdot p^{(k-1)}(\tau)\, d\tau \\ &= \int_{-\infty}^{\infty} \frac{1}{2b}\exp\left(-\frac{|x+1-\tau|}{b}\right) \cdot p^{(k-1)}(\tau)\, d\tau \\ &\geq \int_{-\infty}^{\infty} \frac{1}{2b}\exp\left(-\frac{|x-\tau|+1}{b}\right) \cdot p^{(k-1)}(\tau)\, d\tau \\ &= \exp\left(-\frac{1}{b}\right) \cdot \int_{-\infty}^{\infty} p(x-\tau) \cdot p^{(k-1)}(\tau)\, d\tau\end{aligned}$$

Moreover

$$\begin{aligned}&p^{(k)}(x-1) \\ &= \int_{-\infty}^{\infty} \frac{1}{2b}\exp\left(-\frac{|x-1-\tau|}{b}\right) \cdot p^{(k-1)}(\tau)\, d\tau \\ &\geq \int_{-\infty}^{\infty} \frac{1}{2b}\exp\left(-\frac{|x-\tau|+1}{b}\right) \cdot p^{(k-1)}(\tau)\, d\tau \\ &= \exp\left(-\frac{1}{b}\right) \cdot \int_{-\infty}^{\infty} p(x-\tau) \cdot p^{(k-1)}(\tau)\, d\tau\end{aligned}$$

The case $k = 1$ is a well-known property of the Laplace distribution, but can be proven in the same way by working with p directly. □

Proof of Proposition 7

Proposition. *Let $T \in \mathbb{N}$, and $\mathcal{G}$ a graph. Fix a route $r \in \mathcal{R}$ and write $q_i = \prod_{j=1}^{i-1} \frac{1}{\delta(r_i)}$ for $1 \leq i \leq T$. Define $q_{T+1} \overset{def}{=} 0$. Then*

$$\Pr[\tau = i] = \begin{cases} (1-p) & i = 0 \\ \frac{(1-p)p(q_i - q_{i+1})}{(1-p(1-q_{i+1}))(1-p(1-q_i))} & 1 \leq i \leq T \end{cases}$$

Proof. Let N denote the number of ghost cars. Then

$$\Pr[\tau = i] = \sum_{n=0}^{\infty} \Pr[N = n] \Pr[\tau = i \mid N = n]$$

For $i = 0$ we obviously have

$$\Pr[\tau = 0 \mid N = n] = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases}$$

and so

$$\Pr[\tau = i] = \begin{cases} (1-p) & i = 0 \\ \sum_{n=1}^{\infty} \Pr[N = n] \Pr[\tau = i \mid N = n] & i > 0 \end{cases}$$

Given $N = n > 0$, noise survives to $i > 0$ iff there are $k = 1, \ldots, n$ of those cars that arrive at i but all leave the route at r. Write $q_i = \prod_{j=1}^{i-1} \frac{1}{\delta(r_i)}$. Then q_i is the probability for any given ghost car to still be alive at i (note that given $N > 0$ this probability is 1 for $i = 1$ as implied by the empty product). Then for all $1 < i \leq T$ and $n > 0$:

$$\begin{aligned} &\Pr[\tau = i \mid N = n] \\ =&\sum_{k=1}^{n} \binom{n}{k} q_i^k \cdot \left(1 - \frac{1}{\delta(r_i)}\right)^k (1-q_i)^{n-k} \\ =&\sum_{k=1}^{n} \binom{n}{k} (q_i - q_{i+1})^k (1-q_i)^{n-k} \qquad (\star) \end{aligned}$$

Note that if $i = T$, the term $(1 - \frac{1}{\delta(r_T)})^k$ must be replaced by 1, because the ghost cars will not proceed further with probability 1 due to their elapsing TTL. This is where our convention that $q_{T+1} = 0$ comes into play, which covers this case in $(\star)$.

The sum may be extended to $k = 0$ and simplified

$$\begin{aligned} &\sum_{k=1}^{n} \binom{n}{k} (q_i - q_{i+1})^k (1-q_i)^{n-k} \\ =&((q_i - q_{i+1}) + (1-q_i))^n - (1-q_i)^n \\ =&(1-q_{i+1})^n - (1-q_i)^n \end{aligned}$$

Taken together this gives for $i > 0$

$$\begin{aligned} &\sum_{n=1}^{\infty} \Pr[N = n] \Pr[\tau = i \mid N = n] \\ =&(1-p)\left(\sum_{n=1}^{\infty} (p(1-q_{i+1}))^n - \sum_{n=1}^{\infty} (p(1-q_i))^n\right) \\ =&(1-p)\left(\frac{1}{1-p(1-q_{i+1})} - \frac{1}{1-p(1-q_i)}\right) \end{aligned}$$

Note the corrective -1 summands that are required to make the sums begin at $n = 0$ cancel each other out.

From this on arrives via straightforward algebraic manipulations at:

$$(1-p)p\frac{q_i - q_{i+1}}{(1-p(1-q_{i+1}))(1-p(1-q_i))}$$

□