

# Differential Privacy for Distributed Traffic Monitoring in Smart Cities

Marcus Gelderie<sup>a</sup>, Maximilian Luff and Lukas Brodschelm  
Aalen Univeristy 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  $\epsilon$  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 moving about the city to discern movement patterns that span large parts of (or even the entire) city (e.g., 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 an 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) 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  $\epsilon$  DP in this setting and compare their relative merits. Specifically, we show how there appears to be a trade-off between the noise incurred and the complexity of the algorithms that run centrally or in a distributed way.

Our contributions are as follows: I) We propose a generic architecture for distributed traffic monitoring that is applicable in multiple scenarios. II) We develop three DP algorithms for this architecture and analyze their relative noise levels. III) We provide an

<sup>a</sup>  <https://orcid.org/0009-0003-0291-3911>

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 support a distributed deployment at the various locations across 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) from 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 cannot easily be ported to our setting. Instead, the *naive* option of sampling noise per time-step (cf. alg. 1) 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 alg. 2, 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. alg. 3) that bridges between these two approaches. We analyze and simulate this third approach and find that it strikes a balance between both the “naive” (alg. 1) and the “noise per route” approach (alg. 2) in both runtime and noise. It is also easily deployable in a distributed way.

**Related Work.** Differential Privacy (DP) was introduced first introduced in (Dwork et al., 2006). Subsequently, several algorithms giving  $(\epsilon, 0)$ - and  $(\epsilon, \delta)$ -DP 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. 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 reliably get updated counts for different routes each phase. Lastly, our adjacency notion differs significantly from the previous definitions (cf. sec. 3), presenting a 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.** For a set  $X$ , write  $X^*$  for all finite (possibly empty) sequences of elements of  $X$ . For  $s \in X^*$ , write  $s = s_1 \cdots s_l$  where  $s_i \in X$ ,  $1 \leq i \leq l$ . The length of  $s$  is  $|s| = l$ . A *prefix* is a sequence  $s|_i = s_1 \cdots s_i$ , where  $0 \leq i \leq |s|$ . The concatenation  $s = s_1 \cdots s_l$  and  $s' = s'_1 \cdots s'_l$  is written as  $s \parallel s' = s_1 \cdots s_l s'_1 \cdots s'_l$ . We

write  $[a, b]_{\mathbb{N}} \stackrel{\text{def}}{=} [a, b] \cap \mathbb{N}$  and drop the subscript when the need for natural numbers is clear.

**Differential Privacy.** Differential privacy (DP) was introduced by (Dwork et al., 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 sec. 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<sup>1</sup> from (Dwork et al., 2010):

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

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

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

The following useful “post-processing” theorem allows rounding 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  $\epsilon$ -DP and  $\mathcal{B}$  is any randomized mapping defined on  $\text{range}(\mathcal{A})$ , then  $\mathcal{B} \circ \mathcal{A}$  provides  $\epsilon$ -DP.*

As usual,  $\mathcal{A}$  has  $(\alpha, \beta)$  error, if for any  $s$  of length  $L$ , the maximal difference (in the  $\infty$ -norm) between the true statistic  $f(s)$  and the computed statistic  $\mathcal{A}(s)$  is below  $\alpha$  with probability at least  $1 - \beta$ .

The Laplace distribution  $\text{Lap}(b)$  of scale  $b > 0$  has PDF  $p(x) = (2b)^{-1} \exp(-|x| \cdot b^{-1})$  and satisfies  $p(x) \leq \exp(\frac{1}{b}) \cdot p(x \pm 1)$ .

### 3 PROBLEM STATEMENT

We consider traffic monitoring in a smart city context. Our goal is to compute the number 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.

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

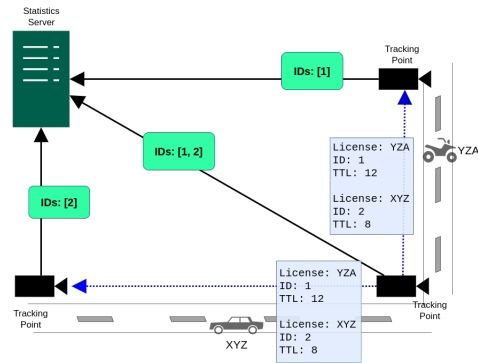


Figure 1: System architecture.

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 and a *time-to-live (TTL)* that is initialized to a constant  $T \in \mathbb{N}$ . Vehicles are recognized via some form 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 be assigned a new unique ID  $u' \neq u$  and a new TTL. In fact, the new unique ID will satisfy the 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. A *route* is a path  $r = r_1 \dots r_m$  through the graph:  $(r_i, r_{i+1}) \in E$  for  $1 \leq i < m$  (repetitions are possible). Note  $m \leq T$ , because of the TTL. We write  $\mathcal{R}$  for the (finite) set of all routes and  $\mathcal{R}_{\max} = \{r \in \mathcal{R} \mid |r| = T\}$ .

For each observed vehicle, the tracking points transmit its TTL to neighboring locations  $\text{Adj}(v) = \{v' \in V \mid (v, v') \in E\}$ , so that the same vehicle is always assigned the same unique ID (within its TTL). In this way, each tracking point reports a sequence of observed IDs to the central statistics server (see fig. 1). Using the resulting pairs of ID and location, the server builds a statistic of vehicles per route.

**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. if the driver stops for coffee). The resulting sequence for that ID will contain “gaps”: intermittent time-steps where the vehicle is not recorded. 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. there are no ‘‘gaps’’. This is no limitation, because such gaps will not change how our algorithms work. But 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 our algorithms.

## 4 DP IN TRAFFIC MONITORING

In this section we propose three different algorithms that ensure DP 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  $s = (s_1, \dots, s_L)$ , 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  $s_u = \perp$  if  $u \notin \text{dom}(s)$  and require that  $\perp \notin \mathcal{D}$ .

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 DP 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  $f: A \rightarrow B$ ,  $a \in A$  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 counts the number of IDs per route. The actual tracking of IDs along routes is done as a preprocessing step, before the curator is fed the actual data: 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  $f: (\mathcal{R}^{\mathcal{U}})^* \rightarrow (\mathbb{N}^d)^*$  as  $f(s_1, \dots, s_L) = \bar{m}(s_1), \dots, \bar{m}(s_L) \in (\mathbb{N}^d)^L$  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  $f_{i,r}^s \stackrel{\text{def}}{=} ((f(s))_i)_r$  for  $1 \leq i \leq L$  and  $r \in \mathcal{R}$  in the remainder of this paper.

The simplest solution to create DP 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! Due to space constraints, we refer the reader to the full version of this paper (Gelderie et al., ) for a justification of this claim. As a result, our first algorithm, relies on the straightforward method of adding independent noise per time step. This is

```

input : vector  $v = (r_u)_{u \in \mathcal{U}}$ 
1  $c \leftarrow 0 \in \mathbb{R}^{\mathcal{R}}$ 
2 for  $r \in \mathcal{R}$ 
3    $c_r \leftarrow m_r(v) + \text{Lap}(2 \cdot \epsilon^{-1} \cdot T)$ 
output:  $c$ 
    
```

Algorithm 1: DP on pre-aggregated routes.

```

global :  $R$  database holding routes per ID
global :  $L$  database IDs and associated noise
input :  $f \in V^{\mathcal{U}}$ 
1  $c \leftarrow 0 \in \mathbb{R}^{\mathcal{R}}$ 
2 forall  $\langle u_F, \mu, r \rangle \in L$ 
3   decompose  $r = v \parallel r'$  with  $v \in V$ 
4   if  $|r'| = 0$  : DROP( $L, u_F$ )
5   else UPDATE( $L, u_F, \mu, r'$ )
6    $c_r \leftarrow c_r + \mu$ 
7 forall  $v \in V$ 
8   forall routes  $r = v \parallel r' \in \mathcal{R}$  starting at  $v$ 
9      $\mu \leftarrow \text{Lap}(\frac{2}{\epsilon})$ 
10    INSERT( $L, \text{NEXTFREEID}(), \mu, r'$ )
11     $c_v \leftarrow c_v + \mu$ 
12    forall  $u \in f^{-1}(v)$ 
13       $r \leftarrow \text{LOOKUP}(R, u) \parallel v$ 
14       $c_r \leftarrow c_r + 1$ 
15       $r \leftarrow \text{UPDATEORINSERT}(R, u, r)$ 
output:  $c$ 
    
```

Algorithm 2: Tracking ID+location pairs with perroute noise.

shown in alg. 1.

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

We omit this an all further proofs due to space constraints, and refer the reader to the full version of this paper (Gelderie et al., ).

## 4.2 Location Tracking

Now we study two algorithms that process sequences of vectors of *locations* (elements of  $V$ ). The inputs are now sequences of partial functions  $s = (s_u) \in V^{\mathcal{U}}$ .

To compute the same statistic,  $f: (V^{\mathcal{U}})^* \rightarrow (\mathbb{N}^{\mathcal{R}})^*$  now tracks IDs across time-steps. Formally, let  $s$  be a sequence of length  $L$ , let  $1 \leq k \leq L$  and  $r \in \mathcal{R}$ . Let  $c_r(s|_k) = |\{u \in \mathcal{U} \mid \exists t \in [0, T-1]_{\mathbb{N}_0} : s_{k-t}(u) \cdots s_k(u) = r \wedge \forall t' < k-t : s_{t'}(u) = \perp\}|$ . We require that  $u$  does not occur in  $s|_k$  prior to time  $k-t$  (so, given  $k$ , the value for  $t$  is maximal). The statistics function is  $f(s) = (c_r(s|_1))_{r \in \mathcal{R}}, \dots, (c_r(s|_L))_{r \in \mathcal{R}}$ .

Alg. 2 computes  $f$  and adds noise. To this end, at each time  $t$  and for every location  $v \in V$ , noise  $\mu$  is

sampled from  $\text{Lap}(2 \cdot \epsilon^{-1})$  for *each route*  $r$  that begins in  $v$ . This noise is assigned an ID and sent along that route. The route length replaces the TTL.

**Theorem 3.** *Alg. 2 provides  $\epsilon$ -DP with  $(2 \cdot \epsilon^{-1} \cdot \sqrt{8} \cdot (R \cdot \ln(2) - \ln(\frac{\beta}{2LR})), \beta)$  error, where  $L$  is the length of the sequence and  $R = |\mathcal{R}|$ .*

**Remark 2.** Alg. 2 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 A Hybrid Approach

In this subsection, we study a hybrid and probabilistic approach that bridges between algs. 1 and 2. Where alg. 2 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 at random and the noise is passed along to  $v'$ . During step  $t+1$ , that noise value is again sent on to another neighbor  $v'' \in \text{Adj}(v')$  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  $\mathcal{G}$ . 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 DP, if left untreated. However, this event is detectable and can be solved by falling back to alg. 1.

Alg. 3 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 4.** *Alg. 3 provides  $\epsilon$ -DP, provided the adjacent sequences differ in a route of length  $T$ .*

Alg. 3 can be implemented in a distributed way, by transmitting the ghost cars to neighboring tracking-points and reporting them alongside the regular location reports. The tracking-points would *not* provide DP by themselves, because the transmitted IDs differ between adjacent sequences.

The noise bound 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 alg. 1. We instead simulate the algorithm below.

```

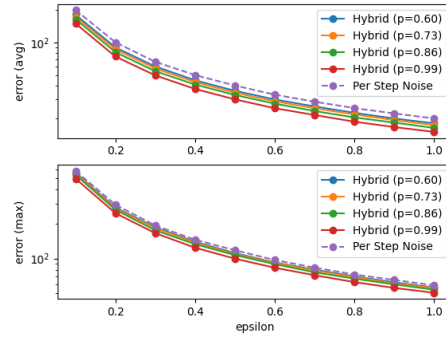
global :  $R$  database holding routes per ID
global :  $L$  database IDs and associated noise
input :  $f \in V^{\mathcal{U}}$ 
1  $c \leftarrow 0 \in \mathbb{R}^{\mathcal{R}}$ 
2  $\text{covered} \leftarrow (\perp, \dots, \perp)$ 
3 forall  $\langle u_F, \mu, r, l \rangle \in L$ 
4   if  $l = 0$  : DROP( $L, u_F$ ) and continue
5    $v \leftarrow \text{Uniform}[\text{Adj}(\text{last}(r))]$ 
6    $r' \leftarrow r \parallel v$ 
7   UPDATE( $L, u_F, \mu, r', l - 1$ )
8    $c_{r'} \leftarrow c_{r'} + \mu$  and  $\text{covered}_{r'} \leftarrow \checkmark$ 
9 forall  $v \in V$ 
10  while continue with probability  $p$ 
11     $\mu \leftarrow \text{Lap}(\frac{2}{\epsilon})$ 
12    INSERT( $L, \text{NEXTFREEID}(), \mu, v, T$ )
13     $c_v \leftarrow c_v + \mu$ 
14     $\text{covered}_v \leftarrow \checkmark$ 
15  forall  $u \in f^{-1}(v)$ 
16     $r \leftarrow \text{LOOKUP}(R, u) \parallel v$ 
17     $r \leftarrow \text{UPDATEORINSERT}(R, u, r)$ 
18     $c_r \leftarrow c_r + 1$ 
19 forall  $r \in \mathcal{R}$  with  $\text{covered}_r \neq \checkmark$ 
20    $c_r \leftarrow c_r + \text{Lap}(\frac{2T}{\epsilon})$ 
output:  $c$ 
    
```

Algorithm 3: DP for location tracking – Hybrid Approach.

## 5 COMPARISON

### 5.1 Simulation

It is clear that alg. 1 has better noise properties than alg. 2. But when we consider alg. 3, the picture is not so clear. This algorithm shares elements with alg. 2, but spawns less noise on the first hops of a route (depending on  $p$ ). On the other hand, it periodically falls back to alg. 1. It is also not clear whether the ghost cars spawned in alg. 3, 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 alg. 2. To investigate this, we implemented the hybrid approach and alg. 1. We then compared these two approaches by simulating each one for a total of  $m = 10000$  times using evenly spaced values for  $\epsilon$  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. We chose multiple values for  $p$  between 0.6 and 0.99.


 Figure 2: Noise values ( $d = 3$ ).

Choosing an appropriate  $T$  and studying the probability of a noise value staying alive along a route pose a challenge. One can compute  $\Pr[\tau = i]$ , if one assumes a constant out-degree for every vertex in  $\mathcal{G}$ . Doing so, one finds that noise cars are rarely alive after time  $t = 8$ . Moreover, there is only a small difference between out-degree 2 and 3. We omit the details and refer the reader to the full version of this paper (Gelderie et al., ).

We then performed measurements using  $T = 10$ ,  $d = 3$  and plotted the maximum and average absolute noise for various values of  $p$  and  $\epsilon$  (note that parameter  $p$  does not affect alg. 1). The results are shown in fig. 2. A complete table of results can be found in (Gelderie et al., ). We can see that the hybrid approach outperforms alg. 1 in terms of both maximum and average noise. The benefit is present over the entire range of  $\epsilon$  values. Alg. 1 requires  $\approx 1.33 \times$  the average noise of the hybrid approach and  $\approx 1.17 \times$  the maximum noise. With increasing  $p$ , the hybrid approach perform better. This is surprising: Large  $p$  imply a large number of ghost cars carrying 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: In the *centralized* scenario, the algorithm runs completely in the context of the curator. The tracking behave as described in sec. 3. In the *distributed* scenario, a part of the algorithm is executed at the tracking points. These parts differ between algorithms. 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 alg. 1 which has inputs of the form  $v \in \mathcal{R}^u$ . It runs linearly in this vector. However note the vector first needs to be built from the reports by individual tracking points (cf. sec. 3). On the other hand, algs. 2 and 3 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 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 alg. 3)  $p$ . 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.

Finally, it is easy to see that both algs. 1 and 2 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 alg. 2 needs to iterate over  $\mathcal{R}$  twice (or perform twice the work per loop iteration) compared with alg. 1. We thus count loop iterations in terms of the parameters laid out above, rather than use asymptotic complexity.

**Centralized.** Alg. 1 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). Alg. 1 requires no storage; with computing the statistic from per-vehicle reports, some mapping of ID to route-prefixes is needed and we require storage on the order of  $P$ .

Alg. 2 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, alg. 3 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 alg. 1 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 algs. 2 and 3 can be implemented in a distributed fashion. Alg. 2 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, much like real cars. 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 runs in time  $P + |\mathcal{R}|$ .

Alg. 3 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  $\text{Adj}(v)$ . The curator is as above. While overall storage seems lower, the actual noise-to-ID binding now reside on network link buffers.

**Discussion.** We see that while alg. 2 is optimal in terms of noise, it has worst complexity in the centralized setting and the distributed setting. Alg. 1 on the other hand cannot be implemented in a distributed fashion and incurs a large amount of noise. We can see that alg. 3 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 alg. 1 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  $\epsilon$  DP in this context. Building on that, we presented three algorithms that each achieve  $\epsilon$  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 practitioner 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 striking a balance be-

tween the first two algorithms. It is reasonably efficient in a distributed setting and outperforms the first algorithm in terms of noise when simulated.

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  $(\epsilon, \delta)$  DP, where  $\delta > 0$ , may hold significant improvements in terms of noise at the cost of a modest imbalance in the privacy guarantees.

## 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 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., 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.
- Gelderie, M., Luff, M., and Brodschlem, L. Differential privacy for distributed traffic monitoring in smart cities (full version).
- 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.
- 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., Chakraborty, 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 *Futuristic Trends in Network and Communication Technologies*, pages 152–164. 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*.
- 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.