Private and Decentralized Location Sharing for Congestion-Aware Routing

Matthew Tsao
Stanford University
mwtsao@stanford.edu

Kaidi Yang
Stanford University
kaidi.yang@stanford.edu

Karthik Gopalakrishnan
Stanford University
gkarthik@stanford.edu

Marco Pavone
Stanford University
pavone@stanford.edu

March 1, 2022

Abstract

Data-driven methodologies offer promising ways to improve the efficiency, safety, and adaptability of modern and future mobility systems. While these methodologies offer many exciting upsides, they also introduce new challenges, particularly in the realm of user privacy. Specifically, data-driven algorithms often require user data, and the way this data is collected can pose privacy risks to end users. Centralized data sharing systems where a single entity (mobility service provider or transit network operator) collects and manages user data have a central point of failure. Users have to trust that this entity will not sell or use their data to infer sensitive private information. Unfortunately, in practice many advertising companies offer to buy such data for the sake of targeted advertisements.

With this as motivation, we study the problem of using location data for congestion-aware routing in a privacy-preserving way. Rather than having users report their location to a central operator, we present a protocol in which users participate in a decentralized and privacy-preserving computation to estimate travel times for the roads in the network in a way that no individuals’ location is ever observed by any other party. The protocol uses the Laplace mechanism in conjunction with secure multi-party computation to ensure that it is cryptographically secure and that its output is differentially private.

A natural question is if privacy necessitates degradation in accuracy or system performance. We show that if a road has sufficiently high capacity, then the travel time estimated by our protocol is provably close to the travel time estimated by the ground truth. This result highlights an interesting complementarity between differential privacy and volume delay functions. There are settings where differential privacy necessitates poor accuracy in traffic estimation, and there are settings where delay functions are very sensitive to errors. Remarkably, these settings are disjoint, and thus differential privacy and volume delay functions cover for each others’ weaknesses to produce an overall protocol with provable accuracy guarantees. We validate this insight in practice through numerical experiments which show that the protocol provides privacy guarantees with minimal overhead to system performance.
1 Introduction

Big Data and data-driven methodologies have shown promise in improving the efficiency, safety and adaptability of mobility services. However, certain types of data sharing can also lead to privacy risks for users. In this paper we focus on merits and risks of sharing location data. We discuss how location data is useful for determining congestion levels for the sake of vehicle routing, and we discuss user privacy risks involved with location sharing. With this as motivation we show how a protocol for decentralized location sharing can mitigate privacy risks while retaining some of the merits of location information for routing.

Location information is helpful for congestion-aware vehicle routing because congestion levels of a road can be estimated from the number of vehicles on the road. Importantly, the congestion level only depends on aggregate location information; what matters is the number of vehicles on a road, not which particular users (i.e., user identification number) are on the road. This suggests that aggregation procedures can be used to protect individual user location while still providing the location information needed for congestion routing.

Repeated exposure to location sharing can lead to privacy risks for users. Users often only provide a small amount of their location data to each app that they interact with. However, since users are often assigned a unique identification number based on their device or their personal information (name, birthday, etc.), if a user interacts with many apps, the data they share with these apps can be stitched together to form a more complete picture of the user’s routines, behaviors, preferences, etc. User privacy in such settings thus requires trust that the apps won’t share user data with other entities. However in practice, advertising companies offer to buy user data from app operators to build user profiles for the sake of targeted advertising. As a result of this centralized process, even though users only share small amounts of their location data to each entity they interact with, a single entity ends up with a large amount (likely more than the user is comfortable with) of their location data (see Fig. 1).

1.1 Statement of Contributions

Motivated by this observation, in this paper we propose a decentralized location sharing protocol where users on the road will periodically compute and announce the traffic counts (e.g., approximate number of vehicles traveling on each road) of the transportation network in a decentralized and privacy-preserving manner. Since only the total number of vehicles on each road is announced, the location of individual users is not discernible by observers, which is contrary to a centralized location sharing setting where users give their individual location data directly to apps. With this protocol, user privacy does not rely on a trusting apps or data collectors, and there are no single points of failure.

Furthermore, assuming the roads in the network are sufficiently large, we can prove that the travel time estimates produced by the protocol will be close to the estimates produced by the ground truth with high probability. This result showcases an interesting complementarity between differential privacy and volume delay functions used in travel time estimation. In low traffic situations, differential privacy constraints lead to poor accuracy for traffic count estimation. However, volume delay functions are insensitive for small inputs and can thus tolerate the poor accuracy. On the other hand, delay functions are very sensitive in high traffic situations, and differential privacy can provide high accuracy in these settings. Thus when a delay function is composed with
a differentially private mechanism, the two compensate for the others’ weaknesses to yield accurate and private travel time estimates. We corroborate this insight using numerical experiments which show that the protocol enables privacy-preserving congestion routing with minimal overhead to the travel time of users.

1.2 Related Work

Privacy research in transportation mainly focuses on location privacy, whereby the aim is to prevent untrusted entities from learning geographic locations or location sequences of an individual [1]. A number of privacy-preserving approaches have been proposed for various location-based applications, e.g., trajectory publishing, mobile crowdsensing, traffic control, etc. From the methodological perspective, these approaches are often implemented through spatial cloaking [2], differential privacy [3], and Secure Multi-Party Computation (MPC) [4].

Spatial cloaking-based approaches rely on aggregation in which users report coarse information instead of their exact locations. These approaches are often based on k-anonymity [5], where a mobility dataset is divided into equivalent classes based on data attributes (e.g., geological regions, time, etc.) so that each class contains at least k records [6, 7]. These k-anonymity-based approaches can guarantee that every record in the dataset is indistinguishable from at least k-1 other records. However, k-anonymity is generally considered to be a weak privacy guarantee. Furthermore, due to coarse data aggregation, spatial cloaking-based approaches can lead to low data accuracy.

Differential privacy-based approaches provide a sound privacy guarantee by producing randomized responses to queries, whereby two datasets that differ in only one entry produce statistically
indistinguishable responses \[8\]. In other words, differential privacy ensures that an adversary with arbitrary background information (e.g., query responses, other entries) cannot infer individual entries with high confidence. Existing research for location data either probabilistically generates obfuscated locations from a user’s true location \[9\] \[10\] or adds noises to the number of users within each equivalent class \[11\] \[12\] \[13\] \[14\] \[15\]. However, differential privacy-based approaches can suffer from two drawbacks. First, due to randomization, there is a trade-off between the accuracy of the response and the level of privacy. Second, most existing research requires a trusted data collector to generate random responses, which does not fit our decentralized setting in this paper.

Secure MPC serves as an excellent technique for decentralized settings, whereby several players jointly compute a function over their data while keeping these data private. Existing secure MPC-based research proposes traffic monitoring and control approaches that keep users’ location data confidential, based on secret sharing \[16\] \[17\], homomorphic encryption \[18\] \[19\], and blockchain \[20\]. Secure MPC can ensure accuracy since no noises are added to protect location privacy. However, Secure MPC can suffer from high computational overhead due to encryption, and the computation results might leak private information (See Remark 4 for more details).

1.3 Organization

This paper is organized as follows. In Section 2 we present a model for the transportation system, and specify both the system objective and privacy requirements. We present a decentralized and privacy-preserving protocol for congestion routing in Section 3 along with all of the statistical and cryptographic tools used by the protocol. In Section 4 we prove that if the roads in the transportation network are sufficiently large, then the protocol provides privacy with minimal overhead to total system travel time in congestion routing. We evaluate our protocol in numerical experiments and present the results in Section 5. We summarize our work and identify important areas for future work in Section 6.

2 Model

In this section we describe the transportation network model, the objective for the users’ distributed algorithm to estimate traffic counts, and the privacy requirements for the algorithm.

2.1 Transportation Network

The transportation network is represented as a directed graph \(G := (V, E)\) where edges \(E\) represent roads and vertices \(V\) represent road intersections. We use \(n := |V|\) and \(m := |E|\) to denote the number of vertices and edges in the graph respectively. The concepts of traffic flow, traffic counts, and travel times are essential to this work, so we will describe them here.

**Definition 1 (Traffic Flow).** For a given road \(e \in E\), its traffic flow \(x_e\) measures the number of vehicles that enter the link per second.

**Definition 2 (Travel Time).** Each edge \(e \in E\) has an associated volume delay function \(f_e : \mathbb{R} \rightarrow \mathbb{R}\) where \(f_e(x)\) is the estimated travel time of the road \(e\) if there are \(x\) vehicles entering the road per second (i.e., the traffic flow).
Definition 3 (Traffic Counts). For a given road $e \in E$, its traffic count $s_e$ is the number of vehicles currently on the road. At steady state the traffic count is equal to the traffic flow multiplied by the travel time. Specifically, $s_e = x_e f_e(x_e)$.

Throughout this paper we make the following natural assumption on the volume delay functions.

Assumption 1 (Properties of Volume Delay Functions). We assume that for each road $e \in E$, $f_e$ is a positive, non-decreasing and differentiable function on $\mathbb{R}_+$.

Remark 1. The Bureau of Public Roads (BPR) function $f_{\text{BPR}}(x) := 1 + 0.15 \left( \frac{x}{c} \right)^4$ is a commonly used volume delay function which satisfies Assumption 1. Namely, it is a degree 4 polynomial with positive coefficients.

Definition 4 (Estimating Travel time from Traffic Counts). For each road $e \in E$ we can derive a function $\tau_e$ that estimates travel time based on traffic counts. From Definition 3 we know that for any road $e \in E$ we have $s_e = x_e f_e(x_e)$ at steady state. Under Assumption 1, the function $F_e(x) := x f_e(x)$ is invertible since $f_e$ is non-decreasing. Thus the flow can be written as a function of traffic counts as $x_e = F_e^{-1}(s_e)$. The travel time can thus be estimated from traffic counts through the function $\tau_e(s) := f_e(F_e^{-1}(s))$ so that $\tau_e(s_e)$ is the travel time estimated from the traffic count.

Road capacities are a concept that will be important to our methodology and results, which we define as follows:

Definition 5 ($\delta$-capacity). For $\delta > 0$, the $\delta$-capacity of a road $e$, denoted $c_{e,\delta}$, is the largest value so that for all $x \leq c_{e,\delta}$ we have $f_e(x) \leq (1 + \delta) f_e(0)$.

2.2 Users and Traffic Counts

At any given time $t$, let $N(t)$ denote the number of users currently traveling in the transportation network. For $1 \leq i \leq N(t)$, the state of user $i$ at time $t$ is given by $s(t, i) \in \{0, 1\}^m$ specifies which road the user is on:

$$s_e(t, i) = 1 \ [\text{user } i \text{ is on road } e].$$

Note that exactly one entry of $s(t, i)$ is 1 and all others are 0. The traffic counts at time $t$, denoted $s(t) \in \mathbb{N}^m$ counts the total number of user vehicles on each road

$$s(t) := \sum_{i=1}^{N(t)} s(t, i).$$

so that $s_e(t)$ specifies the number of users traveling on road $e$ at time $t$.

2.3 Communication Model

In this work we assume that users can communicate with one another through private channels. Concretely, this means that for any pair $i, j$ of users, user $i$ can send a message that user $j$ will receive, but no other user can receive the message. This can be done using public key cryptography systems, for example.
2.4 System Objective

The goal of the system is to periodically broadcast estimated travel times for all roads in the network for the sake of congestion-aware routing. The accuracy of travel time estimates will be measured by mean absolute percentage error (MAPE) as defined in Definition 6.

**Definition 6** (Mean Absolute Percentage Error (MAPE)). Suppose $T$ is a (possibly randomized) estimator for a positive target value $t^*$. Then the mean absolute percentage error (MAPE) of $T$ is given by

$$\mathbb{E}_T \left[ \frac{|T - t^*|}{t^*} \right]$$

where the expectation is taken over the randomness in $T$.

The following remark explains how the traffic counts are valuable to this effort.

**Remark 2** (Travel times from traffic counts). The function $\tau_e$ from Definition 4 can be used to compute an estimated travel time $\tau_e(s_e(t))$ of road $e$ at time $t$ from the traffic counts.

With Remark 2 in mind, the system’s goal is to compute and announce the traffic counts of the system every $\Delta t$ minutes. Concretely, for each $k \in \mathbb{N}$, at time $k\Delta t$ the $N(k\Delta t)$ traveling users must compute an approximation to $s(k\Delta t)$ in a distributed and privacy-preserving way where privacy is defined according to Definition 7.

**Definition 7** (Privacy-Preserving Mechanism). A mechanism is $\epsilon$-privacy preserving if it is $\epsilon$-differentially private and can be computed in a distributed setting in a way that is cryptographically secure against semi-honest adversaries.

The precise definitions for cryptographic security, semi-honest adversaries and differential privacy are presented and motivated in the next section.

**Remark 3** (On the choice of location data for travel time estimation). In this work we use location data to estimate travel times in a transportation network. This is done by first estimating the traffic flow from traffic counts, and then estimating travel time from traffic flow. One natural alternative is to have users share both their location and speed. In this alternative approach, the location would specify which road the user is on and the average speed reported on a road could be used to estimate the its travel time. We opted not to use speed information for two main reasons. The first is due to privacy requirements. As we will discuss and motivate in Section 2.5.2, differential privacy is an important property that we want our method to have. Due to the properties of differential privacy, there are effective ways to compute counts (such as the number of users on a given road) but no clear way to compute an average of user data (such as average speed) in a differentially private way. See Remark 5 for more details. The second is for ease of deployment. Requiring only location data makes the approach easy to implement since many drivers carry a mobile phone with GPS, whereas accurate speed measurements may require more sophisticated hardware.

2.5 Privacy Requirements

To ensure user privacy, there are two requirements we impose on a desired protocol for the computation of traffic counts: cryptographic security and differential privacy.
2.5.1 Cryptographic Security

Cryptographic security pertains to settings where a group of agents, each with private data, would like to compute a joint function of everyone’s data without any agent needing to reveal its private data to other agents. In our setting, at time $k\Delta t$ the $N(k\Delta t)$ users traveling in the network are agents, where $s(t, i)$ is the private data of the $i$th user, and the desired function is the sum of everyone’s private data. We make the following standard assumption on user behavior:

**Assumption 2** (Semi-honest users). We assume that all users are semi-honest\[^1\], which means they will follow the protocol but may try to do additional computation to learn the secret data of other users.

The definition of cryptographic privacy measures privacy by comparing protocols to an ideal computation model which is defined below.

**Definition 8** (Ideal Computation Model). In the ideal computation model, there are $n$ agents $a_1, ..., a_n$ with private data $x_1, ..., x_n$ wanting to compute $f(x_1, ..., x_n)$. Each agent sends its private data to a trusted third party which uses the private data to compute $f(x_1, ..., x_n)$ and sends this value back to all of the agents.

However, since trusted third parties cannot be assumed to exist, the ideal computation model cannot be implemented in a trustless and decentralized setting. Still, this model serves as a gold standard, and cryptographically secure protocols are required to provide the same level of security as this ideal model.

**Definition 9** (Cryptographic Security). A protocol between $n$ agents $a_1, ..., a_n$ with private data $x_1, ..., x_n$ wanting to compute $f(x_1, ..., x_n)$ is cryptographically secure if no probabilistic polynomial time agent learns anything more about other agents’ data than they would have learned in the ideal computation model.

In other words, a protocol is cryptographically secure if no computationally efficient agent learns more from interacting with the protocol than they would from interacting with the Ideal computation model.

**Remark 4.** We emphasize that this does not mean that agents learn nothing about other agents’ data. This is illustrated by a simple three agent example $a_1, a_2, a_3$ with private data $x_1, x_2, x_3$ and query function $f(x_1, x_2, x_3) = x_1 + x_2 + x_3$. In this example, by learning $f(x_1, x_2, x_3)$, $a_1$ learns the sum of the other agents’ data: $x_2 + x_3 = f(x_1, x_2, x_3) - x_1$.

In light of Remark\[^4\] it is more accurate to say cryptographically secure protocols reveal nothing about other agents’ data beyond the value of the output.

Cryptographic security is necessary for user privacy, since we certainly don’t want users to be able to determine the location of certain individuals through our protocol.

Unfortunately, for the application of user location data, cryptographic security alone is not enough to ensure user privacy, which we illustrate in the following example.

---

\[^1\]Semi-honest adversaries, honest-but-curious adversaries, and passive adversaries are equivalent and used interchangeably in the cryptography literature.
Example 1 (Insufficiency of Cryptographic Privacy for Sparse Data). If Alice is an early bird and wakes up to run errands in the city before anyone else wakes up, then in the morning \( N(t) = 1 \) since Alice is the only person in the network, and therefore we have \( s(t) = s(t, Alice) \), hence the traffic counts reveal Alice’s location information. While \( s(t) \) does not explicitly label the single traveler in the system as Alice, this information can be inferred if the traveler begins and ends its route at Alice’s house. More generally, cryptographic security does not provide user privacy in sparse data settings. Even when there are multiple users active in the network, side information attacks can be used to associate trajectories in sparse datasets to certain individuals [21].

This motivates the second privacy requirement we enforce in this paper, which is differential privacy.

2.5.2 Differential Privacy

With Example 1 in mind, to protect the privacy of users like Alice, the output of a privacy-preserving protocol should not depend too much on the data of any single user. One way to ensure this is through differential privacy. To quantify the influence of a single user, we first introduce the concept of adjacent datasets.

Definition 10 (Adjacent Datasets). Two datasets \( D_1, D_2 \) are adjacent if \( D_1 \) contains at most one datapoint that is not in \( D_2 \) and \( D_2 \) contains at most one datapoint that is not in \( D_1 \). Concretely, \( D_1, D_2 \) are adjacent if \(|D_1 \setminus D_2| \leq 1 \) and \(|D_2 \setminus D_1| \leq 1 \).

In our setting, a dataset \( D_1 = \{s(t, i)\}_{i=1}^{N(t)} \) would be the locations of the \( N(t) \) users who are traveling within the transit network at time \( t \). The dataset \( D_2 \) obtained from \( D_1 \) by modifying the location of one user, who we will call Alice, would be adjacent to \( D_1 \) since \( D_1 \setminus D_2 \) contains only the datapoint corresponding to Alice’s original location, and \( D_2 \setminus D_1 \) contains only Alice’s newly modified location. One sufficient way to ensure that a mechanism does not depend too much on any single users’ data is to demand that the mechanism behaves similarly on adjacent datasets. This is the approach taken by differential privacy which is defined below.

Definition 11 (Differentially Private Mechanism). For \( \epsilon > 0 \), a \( \epsilon \)-differentially private mechanism \( M : \mathcal{D} \to \mathcal{X} \) is a randomized function mapping datasets into an output space \( \mathcal{X} \) so that for any event \( E \subset \mathcal{X} \) and any adjacent datasets \( D_1, D_2 \), we have

\[
P[M(D_1) \in E] \leq e^\epsilon P[M(D_2) \in E]
\]

To understand why differential privacy gives us the desired privacy we seek, first note that for any two adjacent datasets \( D_1, D_2 \), the distributions of \( M(D_1), M(D_2) \) are very similar. More specifically, the total variation distance between the distributions of \( M(D_1), M(D_2) \) is at most \( \epsilon \).

Because of this, no hypothesis test can determine from the output of the mechanism whether its input was \( D_1 \) or \( D_2 \) with success probability better than \( \frac{1}{2} + \frac{1}{2e^\epsilon} \), which is barely better than random guessing for small \( \epsilon \). This result holds even if the hypothesis test is given knowledge of all datapoints in \( D_1 \cap D_2 \).

Now suppose \( D_1 \) is a dataset that contains Alice’s location, and \( D_2 \) is obtained from \( D_1 \) by modifying Alice’s location arbitrarily. If an observer were able to accurately infer Alice’s location based on the output of a \( \epsilon \)-differentially private mechanism, then it would be able to reliably
distinguish between the inputs $D_1$ and $D_2$. However, since differential privacy makes such a task statistically impossible, by contraposition it is statistically impossible for an observer to accurately infer Alice’s location based on the mechanism’s output. Hence differential privacy ensures privacy of Alice’s data.

The following remark describes a general methodology for achieving differential privacy.

**Remark 5** (Query sensitivity and the required noise level). Dwork’s pioneering work [8] proposes adding noise to queries in order to achieve differential privacy. Given a data set $D$ and a query $f$, the mechanism $D \mapsto f(D) + Z$ is differentially private so long as $Z$ is a random variable with sufficiently large variance. Specifically, to achieve $\epsilon$-differential privacy, the variance of $Z$ should be at least $L_f^2/\epsilon^2$ where $L_f$ is the sensitivity of the function $f$, which is defined as

$$L_f := \sup_{\text{Adjacent datasets } D_1, D_2} f(D_1) - f(D_2).$$

Revisiting Remark 3, the role of sensitivity in differential privacy is a main reason why we chose to estimate travel time using counts rather than with average speed. The sensitivity of counting functions is 1 since the modification of a single data point can change the count by at most 1, however the sensitivity of an average is unbounded since a large change to a single data point in a data set can lead to a large change in the average. As such, counting functions are much more compatible with the concept of differential privacy than averages are.

## 3 Methodology

In this section we describe our distributed protocol to enable users to approximately compute $s(t)$ in a privacy-preserving way. We will be using a Laplace Mechanism, which is described in Section 3.1 to ensure that the protocol is differentially private and will use secure multi-party computation, which is described in Section 3.2 to achieve cryptographic security. Using these tools, we present our privacy-preserving travel time estimation protocol in Section 3.3

### 3.1 Differential Privacy via the Laplace Mechanism

As previously mentioned, our goal is to compute a differentially private approximation to $s(k\Delta t)$ for every $k \in \mathbb{N}$. For this we will use the Laplace Mechanism [8], which produces a differentially private estimate $S(k\Delta t)$ to $s(k\Delta t)$ based on the following rule

$$S(k\Delta t) := s(k\Delta t) + Z$$

where $Z \in \mathbb{R}^m$ has independent and identically distributed entries according to the Laplace distribution with mean 0 and scale parameter $1/\epsilon$ defined below.

**Definition 12** (Laplace Distribution). The Laplace Distribution with mean 0 and scale parameter $1/\epsilon$ is a probability distribution over $\mathbb{R}$ denoted as $\mathcal{L}_\epsilon$ with probability density function given by

$$\mathcal{L}_\epsilon(z) := \frac{\epsilon}{2} e^{-|z|/\epsilon} \text{ for } z \in \mathbb{R}. \quad (1)$$

We use the Laplace mechanism because it provides a differentially private approximation with the minimum possible mean absolute error [22].
Fact 1. The mechanism $s_e(t) \mapsto S_e(t)$ is $2\epsilon$-differentially private.

Since we are interested in a decentralized and trustless computation model, the following remark shows that care must be taken in the computation of $S(k \Delta t)$, particularly pertaining to the computation of $Z$, in order for differential privacy to be achieved.

Remark 6 (Z must remain hidden). It is essential to the differential privacy of $S(k \Delta t)$ that no observer learns the value of $Z$. Since $S(k \Delta t)$ is announced as the output of the protocol, if in addition $Z$ is known by an observer then that observer can reconstruct $s(k \Delta t)$ by computing $S(k \Delta t) - Z$. In this case, the computation of $S(k \Delta t)$ is not cryptographically secure because the observer learns more about the user data than $S(k \Delta t)$ since in particular it learns the value of $s(k \Delta t)$.

In light of Remark 6 in the next subsection we discuss a cryptographic technique called secret sharing which we will use for a cryptographically secure computation of $S(k \Delta t)$.

3.2 Secure MPC via Secret Sharing

In this section we review a cryptographic tool known as secret sharing and discuss how different variants of it can be used to enable cryptographically secure arithmetic operations on private data. We describe how cryptographically secure addition can be performed on private data in Section 3.2.1 using Additive Secret Sharing, and how cryptographically secure multiplication can be performed on private data in Section 3.2.2 using Shamir Secret Sharing.

3.2.1 Secure Multi-Party Addition via Additive Secret Sharing

Suppose there are $N$ agents $a_1, ..., a_N$ and someone wants to share a secret value $x \in \mathbb{N}$ with the agents so that the $N$ agents can reconstruct $x$ if they work together, but no group of fewer than $N$ agents can reconstruct the secret. This can be done using Additive Secret Sharing.

In Additive Secret Sharing, a large prime integer $p$ is first chosen. The shares $s_1, s_2, ..., s_{n-1}$ are all chosen independently and uniformly at random from the set $\{0, 1, 2, ..., p - 1\}$ and the final share is determined by $s_n := x - \sum_{i=1}^{N-1} s_i \mod p$. Finally, $s_i$ is given to $a_i$ for each $1 \leq i \leq N$.

First, note that the $N$ agents can reconstruct $x$ by simply adding all of their shares together since by construction we have $\sum_{i=1}^{N} s_i = x$.

Next note that any group of strictly fewer than $N$ agents cannot reconstruct the secret. A straightforward calculation shows that for any strict subset $S \subset [N]$, the distribution of $\{s_i\}_{i \in S}$ does not depend on $x$, and therefore $\{s_i\}_{i \in S}$ provides no information on the value of $x$.

Example 2 (Secure Multi-Party Addition). One valuable application of Additive Secret Sharing is cryptographically secure computation of the sum of agents’ private data. Given $N$ agents $a_1, ..., a_N$ with private data $x_1, ..., x_N$, their objective is to compute $x := \sum_{i=1}^{N} x_i$. For each $1 \leq i \leq N$, $a_i$ shares $x_i$ via Additive Secret Sharing by producing shares $s_{1,i}, s_{2,i}, ..., s_{N,i}$ where $s_{j,i}$ is given to $a_j$. At the end of this process, $a_i$ has received $\{s_{i,j}\}_{j=1}^{N}$ and can compute $s_i := \sum_{j=1}^{N} s_{i,j}$. The important observation here is that $\{s_{i,j}\}_{j=1}^{N}$ are additive secret shares for $x$. Hence the agents can share the values $\{s_{i,j}\}_{i=1}^{N}$ with one another and compute $x$ via

$$
\sum_{i=1}^{N} s_i = \sum_{i=1}^{N} \sum_{j=1}^{N} s_{i,j} = \sum_{j=1}^{N} \sum_{i=1}^{N} s_{i,j} = \sum_{j=1}^{N} x_j = x.
$$
3.2.2 Secure Multi-Party Multiplication via Shamir Secret Sharing

Shamir Secret Sharing [23] offers a more general $k$-of-$N$ method for secret sharing. In a setting with $N$ agents $a_1, ..., a_N$ and a secret $x$ to be shared among them, a $k$-of-$N$ secret sharing scheme assigns shares $s_1, ..., s_N$ to the agents so that any subset of $k$ agents can recover $x$, but no subset of fewer than $k$ agents can recover $x$. Note that Additive Secret Sharing is a $N$-of-$N$ scheme.

Shamir’s Secret Sharing is based on the fact that a $k - 1$ degree polynomial is uniquely determined from $k$ evaluations. As in the Additive Secret Sharing setting, a large prime $p$ is chosen. To share a secret value $x$, the sharer generates a random $k - 1$ degree polynomial

$$X(z) = x + \sum_{i=1}^{k-1} C_i z^i$$

where $C_1, C_2, ..., C_{k-1}$ are independent and uniformly distributed over $\{0, 1, ..., p-1\}$. The share given to $a_i$ is $s_i := X(i) \mod p$. By construction, the shares and coefficients satisfy the following linear relationship

$$\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{bmatrix} = \begin{bmatrix} X(1) \\ X(2) \\ \vdots \\ X(N) \end{bmatrix} = \begin{bmatrix} 1^0 & 1^2 & \ldots & 1^{k-1} \\ 2^0 & 2^2 & \ldots & 2^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ N^0 & N^2 & \ldots & N^{k-1} \end{bmatrix} \begin{bmatrix} x \\ C_1 \\ \vdots \\ C_{k-1} \end{bmatrix} \mod p$$

where $V_{N,k}$ is the $N \times k$ Vandermonde matrix.

Since $X$ is a degree $k - 1$ polynomial, any group of $k$ agents can solve a linear system to obtain the values of $x, C_1, ..., C_{k-1}$ and thus recover the secret.

However, for any subset $S \subset [N]$ with $|S| < k$, a careful calculation shows that the distribution of $\{X(i)\}_{i \in S}$ does not depend on $x$ and hence $\{X(i)\}_{i \in S}$ provides no information on the value of $x$.

Example 3 (Secure Multi-Party Multiplication). Shamir Secret Sharing and Additive Secret Sharing can be used to perform cryptographically secure multiplication. Given $N$ agents $a_1, ..., a_N$ with additive secret shares $\{s_i\}_{i=1}^N, \{s'_i\}_{i=1}^N$ for the values $x, y$ respectively so that $x = \sum_{i=1}^n s_i$ and $y = \sum_{i=1}^n s'_i$, the goal is for the agents to compute the product $xy$ in a cryptographically secure way.

The computation of $xy$ will require one round of communication. In this communication round, for each $1 \leq i \leq N$, $a_i$ performs Shamir secret sharing for its values $s_i$ and $s'_i$. Specifically, it generates two random polynomials $X_i, Y_i$ of degree $\frac{N-1}{2}$ so that $X_i(0) = s_i$ and $Y_i(0) = s'_i$ and then sends $X_i(j), Y_i(j)$ to agent $j$ for each $1 \leq j \leq N$.

After the communication, $a_i$ obtains $\{X_j(i), Y_j(i)\}_{i=1}^N$. From these it computes $X(i) := \sum_{j=1}^N X_j(i)$ and $Y(i) := \sum_{j=1}^N Y_j(i)$. Note that $\{X(i)\}_{i=1}^N$ are Shamir shares for the polynomial $X := \sum_{j=1}^N X_j$ and similarly $\{Y(i)\}_{i=1}^N$ are Shamir shares for the polynomial $Y := \sum_{j=1}^N Y_j$. Since the polynomials $\{X_j\}_{j=1}^N, \{Y_j\}_{j=1}^N$ all have degree at most $\frac{N-1}{2}$, the polynomials $X, Y$ also have degree at most $\frac{N-1}{2}$. Thus if we define the polynomial $H(z) := X(z)Y(z)$, then $H$ has degree at most $N - 1$. 
Now $a_i$ computes $X(i)Y(i)$. By definition, $\{X(i)Y(i)\}_{i=1}^N$ are Shamir shares for the polynomial $H$. Noting that

$$H(0) = X(0)Y(0) = \left( \sum_{j=1}^N X_j(0) \right) \left( \sum_{j=1}^N Y_j(0) \right) = \left( \sum_{j=1}^N s_j \right) \left( \sum_{j=1}^N s'_j \right) = xy,$$

we have a degree $N - 1$ polynomial $H$ whose constant term is the desired value $xy$. Furthermore, the $N$ agents know the value of $H$ at $1, 2, ..., N$ and hence can solve a linear system to obtain the coefficients of $H$ and thus obtain $xy$.

The Shamir shares $\{X(i)Y(i)\}_{i=1}^N$ can be converted into Additive shares $\{\theta_i\}_{i=1}^N$ by setting $\theta_i := \lambda_i X(i)Y(i)$ where $\lambda_i$ is the $(1, i)$ entry of $V_{N,N}^{-1}$.

**Remark 7** (Honest Majority Regime). We would like to mention that the Secure Multi-Party Multiplication scheme we describe in this section requires an honest majority assumption on user behavior. This means that the scheme requires at least $\frac{N+1}{2}$ users to be fully honest, meaning that they will follow the protocol and will not collude in any way with any other users. The remaining $\frac{N-1}{2}$ users are assumed to be semi-honest. This is a stronger condition than Assumption 2 which only requires that all users are semi-honest. There exist Secure Multi-Party Multiplication schemes for the semi-honest setting based on Beaver Triples [24], but the scheme we presented in this section is more computationally efficient, and in practice honest majority may not be an unreasonable assumption.

### 3.3 Cryptographically secure and Differentially Private Estimation of Travel Times

In this section we show how the differential privacy and secret sharing tools we have discussed can be used to construct a privacy-preserving and decentralized protocol for travel time estimation, which can then be used to perform congestion routing as per Remark 2. The protocol is described in Algorithm 1.

In order to satisfy the privacy requirements as stated in Definition 7, our protocol must be both differentially private and cryptographically secure. Recall from Section 3.1 that we will use the Laplace mechanism to obtain a differentially private estimate $S_e(k\Delta t)$ to the traffic counts, which is defined below

$$S(k\Delta t) = s(k\Delta t) + Z$$

where $Z$ is an i.i.d. vector of $L_e$ distributed random variables. We thus need to compute $S(k\Delta t)$ in a decentralized and cryptographically secure way.

In this section we demonstrate how to compute one entry of the vector $S(k\Delta t)$, i.e., $s_e(k\Delta t) + Z_e$ for an edge $e \in E$. The computation of the entire vector $S(k\Delta t)$ is obtained by parallelizing the computation of all entries.

We begin by choosing a large prime integer $p$. Inverse transform sampling is a method which can transform a uniform random variable into a random variable with a desired distribution. Using this method we can compute $Z_e = F^{-1}(U_e)$ where $U_e$ is uniformly distributed over $\{0, 1, ..., p - 1\}$ and $F^{-1}$ is a scaled version of the cumulative distribution function of $L_e$. Concretely, $F^{-1}$ is given
by

\[
F^{-1}(u) = \begin{cases} 
\frac{1}{2} \ln \left( \frac{2u}{p} \right) & \text{for } u \leq \frac{p}{2}, \\
-\frac{1}{2} \ln \left( 2 \left( 1 - \frac{u}{p} \right) \right) & \text{if } u > \frac{p}{2}.
\end{cases}
\]

To make the sampling of \( Z_e \) more computationally efficient, we will approximate \( F^{-1} \) with a degree \( d \) polynomial \( P_{\epsilon,d} \). A larger degree leads to a more accurate approximation of the Laplace distribution but comes at a computational cost.

Thus approximately computing \( S_e(k\Delta t) \) amounts to computing \( s_e(k\Delta t) + P_{\epsilon,d}(U_e) \) where \( U_e \) is required to be uniformly distributed over \( \{0, 1, ..., p-1\} \). First, additive shares \( \{\alpha_1, ..., \alpha_{N(k\Delta t)}\} \) for \( s_e(k\Delta t) \) can be computed using Secure Multi-Party Addition as is done in Example 2. Next, note that if \( Y_1, Y_2, ..., Y_{N(k\Delta t)} \) are independent and uniformly random on \( \{0, 1, ..., p-1\} \), then \( \sum_{i=1}^{N(k\Delta t)} Y_i \mod p \) is also uniformly distributed. Therefore user \( i \) will draw a random value \( U_{e,i} \) so that the value \( U_e := \sum_{i=1}^{N(k\Delta t)} U_{e,i} \) will be uniformly distributed. Using Secure Multi-Party Addition, the users can obtain additive shares \( \{\beta_{1,1}, ..., \beta_{1,N(k\Delta t)}\} \) for \( U_e \). The users will need to compute \( P_{\epsilon,d}(U_e) \). Since \( P_{\epsilon,d} \) is a polynomial of degree \( d \), the users will need additive shares for \( U_{e,1}^2, U_{e,1}^3, ... U_{e,1}^d \) for the computation of \( P_{\epsilon,d}(U_e) \). The users can obtain such shares through Secure Multi-Party Multiplication by multiplying \( U_e \) with itself using Shamir and Additive Secret Sharing as described in Example 3. Using this method the users obtain additive shares \( \{\beta_{z,1}, ..., \beta_{z,N(k\Delta t)}\} \) for \( \{U_{e,z}\}_{z=0}^d \) respectively.

Letting \( c_0, ..., c_d \) be the coefficients of \( P_{\epsilon,d} \) so that \( P_{\epsilon,d}(u) = \sum_{z=0}^{d} c_z u^z \), the users can now construct additive shares for \( s_e(k\Delta t) + P_{\epsilon,d}(U_e) \) by taking linear combinations of previously computed shares. Explicitly, the shares

\[
\left\{ \alpha_i + \sum_{z=0}^{d} c_z \beta_{z,i} \right\}_{i=1}^{N(k\Delta t)}
\]

are additive shares for \( s_e(k\Delta t) + P_{\epsilon,d}(U_e) \) since by construction we have

\[
\sum_{i=1}^{N(k\Delta t)} \left( \alpha_i + \sum_{z=0}^{d} c_z \beta_{z,i} \right) = \left( \sum_{i=1}^{N(k\Delta t)} \alpha_i \right) + \sum_{z=0}^{d} c_z \left( \sum_{i=1}^{N(k\Delta t)} \beta_{z,i} \right)
\]

\[
= s_e(k\Delta t) + \sum_{z=0}^{d} c_z U_e^z
\]

\[
= s_e(k\Delta t) + P_{\epsilon,d}(U_e).
\]

Algorithm 1 describes the procedure that each user performs to enable the decentralized and privacy-preserving computation of \( S(k\Delta t) \).

### 4 Analysis: Accuracy of travel times based on \( S(k\Delta t) \)

In the previous section we presented a decentralized and privacy-preserving protocol for computing \( S(k\Delta t) \), which is a differentially private estimate of the traffic counts \( s(k\Delta t) \) at time \( k\Delta t \) (i.e., the \( k \)th timestep). Since the traffic counts are useful for travel time estimation through volume
Algorithm 1: Private and Distributed Traffic Count Estimation

1 Parameters: Large prime number \( p \), approximate inverse CDF \( P_{z,d}(u) := \sum_{z=0}^{d} c_z u^z \) for the Laplace distribution;
2 Inputs: Location information \( s(k\Delta t, i) \) for user \( i \);
3 Output: Estimated traffic counts \( S(k\Delta t) \);
4 for \( e \in E \) do
5 Using Additive Secret Sharing, obtain a share \( \alpha_i \) of \( s(k\Delta t) = \sum_{j=1}^{N(k\Delta t)} s(k\Delta t, j) \) through Secure Multi-Party Addition;
6 Draw \( U_{e,i} \) uniformly at random over \( \{0, 1, ..., p-1\} \);
7 Using Additive Secret Sharing, obtain a share \( \beta_{1,i} \) of \( U_{e} := \sum_{j=1}^{N(k\Delta t)} U_{e,j} \) through Secure Multi-Party Addition;
8 for \( 1 \leq z \leq d \) do
9 Using Shamir and Additive Secret Sharing, obtain a share \( \beta_{z,i} \) of \( U_{z,e} \) through Secure Multi-Party multiplication;
10 Compute \( \theta_i := \alpha_i + \sum_{z=0}^{d} c_z \beta_{z,i} \) and send \( \theta_i \) to all other users;
11 \( S_e(k\Delta t) \leftarrow \sum_{j=1}^{N(k\Delta t)} \theta_j \);
12 Return \( S(k\Delta t) \);

delay functions, one natural question is how the travel time estimates obtained from \( S(k\Delta t) \) will differ from those obtained from the ground truth traffic counts \( s(k\Delta t) \). In this section we show that if the roads in the transportation network \( G \) have sufficiently large capacity, then the travel time estimates obtained from \( S(k\Delta t) \) will be close to those obtained had we used the non-privacy-preserving ground truth value \( s(k\Delta t) \).

We first discuss the accuracy limitations of the Laplace mechanism for estimating the traffic counts in Section 4.1. Next, in Section 4.2 we show how the properties of volume delay functions mitigate the accuracy limitations of the Laplace mechanism so that accurate and private travel time estimates are obtained by composing the two together.

4.1 Accuracy of the Laplace Mechanism

Recall that \( S_e(k\Delta t) = s_e(k\Delta t) + Z_e \) is a differentially private estimate of the traffic count on road \( e \) at time \( t \), which is represented by \( s_e(k\Delta t) \).

The mean absolute percentage error (MAPE) of \( S_e(k\Delta t) \) as an estimate for \( s_e(k\Delta t) \) is given by

\[
\frac{\mathbb{E} \left[ \| S_e(k\Delta t) - s_e(k\Delta t) \| \right]}{s_e(k\Delta t)} = \frac{\mathbb{E} \left[ \| Z_e \| \right]}{s_e(k\Delta t)} = \frac{1}{\epsilon s_e(k\Delta t)}.
\]

From this we can make two conclusions. When there is a lot of traffic on \( e \), meaning that \( s_e(k\Delta t) \) is much larger than \( \frac{1}{\epsilon} \), then \( \epsilon s_e(k\Delta t) \) will be large and hence \( S_e(k\Delta t) \) will have a small MAPE. However, if \( s_e(k\Delta t) \) is small, then \( S_e(k\Delta t) \) will have a large MAPE.

This shows us that the Laplace Mechanism has poor accuracy when reporting small values. In fact, this is true for all differentially private mechanisms since the Laplace Mechanism has the
minimum mean absolute error among all differentially private mechanisms \cite{22}. This observation is consistent with Example 1 in that sparse data and small values pose the most difficulty in privacy-preserving efforts.

Fortunately, this bad news does not end our hopes for achieving both accuracy and privacy in travel time estimation. Even if $S_e(k\Delta t)$ may not always be a good estimate for $s_e(k\Delta t)$, recall that our ultimate objective is travel time estimation, so we are interested in how well $\tau_e(S_e(k\Delta t))$ approximates $\tau_e(s_e(k\Delta t))$. Recall Definition 4 for a description of $\tau_e$. Next we will show how properties of delay functions can enable accurate travel time estimates even if traffic count estimation is poor.

4.2 Protocol accuracy for travel time estimation

In this section we show that if a road $e \in E$ has sufficiently large capacity, then the travel time estimates computed from $S_e(k\Delta t)$ are close to the travel time estimates computed from the ground truth $s_e(k\Delta t)$. Mathematically, this means that $f_e(S_e(k\Delta t))$ is a good estimate for $f_e(s_e(k\Delta t))$.

The key insight behind our result lies in the complementary qualities of differential privacy and volume delay functions. As we saw in Section 4.1, the Laplace mechanism has good accuracy when reporting large values, but poor accuracy for reporting small values. Volume delay functions on the other hand, are very sensitive when the input is large, but very insensitive when the input is small. When composing a volume delay function with a Laplace mechanism, the complementary qualities manifest in two ways. When the traffic $s_e(k\Delta t)$ is larger than a road’s capacity, the volume delay function is very sensitive. Fortunately, in this case the high accuracy of the Laplace mechanism ensures that the traffic count is estimated accurately, leading to accurate traffic flow estimation, which leads to accurate travel time estimation. On the other hand, when the traffic is below the road’s capacity, the Laplace mechanism has poor accuracy, however the delay function is very insensitive in this regime and is able to tolerate large estimation error, leading to accurate travel time estimation. Thus the Laplace mechanism and volume delay function cover each others’ weaknesses to enable accurate travel time estimation for any level of traffic.

We formalize this insight through Theorem 1 which, given desired privacy and accuracy levels $\epsilon$ and $\delta$ respectively, provides conditions under which $\tau_e(S_e(k\Delta t))$ will be close to $\tau_e(s_e(k\Delta t))$ with high probability. The condition is determined by the road’s $\delta$-critical traffic count, which is defined below.

**Definition 13** (\(\delta\)-critical traffic count). The $\delta$-critical traffic count of a road $e \in E$ is the number of vehicles on the road in steady state so that the travel time is exactly $1 + \delta$ times as large as its free-flow travel time. Mathematically, the $\delta$-critical capacity of $e$ is

$$F_e(c_e,\delta) = c_e,\delta f_e(c_e,\delta) = (1 + \delta)c_e,\delta f_e(0).$$

With this setup in place, we now present Theorem 1

**Theorem 1** (Accuracy of Travel Time Estimates). Let $\epsilon, \delta \geq 0$ specify the desired privacy and accuracy levels respectively and let $p \in [0, 1]$ represent the failure probability. For a road $e \in E$, if $f_e$ satisfies Assumption 7 and

$$(1 + \delta)c_e,\delta f_e(0) \geq \frac{1}{\epsilon} \left( \frac{1}{\delta} + 1 \right) \log \frac{1}{p},$$

...
where $c_{e,\delta}$ is the $\delta$-capacity of $e$, then for any value of $s_e(k\Delta t) \in \mathbb{N}$, the following condition is satisfied with probability at least $1 - p$:

$$\frac{|\tau_e(S_e(k\Delta t)) - \tau_e(s_e(k\Delta t))|}{\tau_e(s_e(k\Delta t))} \leq \delta.$$  

See Appendix A for a proof of Theorem 1. Next, we will discuss whether the condition required by Theorem 1 is satisfied in practice by roads in real transit networks.

### 4.3 Discussion

One natural and immediate question is whether the requirement on the $\delta$-critical traffic count of roads in Theorems 1 are satisfied by real road networks. To answer this question we first discuss parameter choices. To ensure a meaningful privacy guarantee for each timestep, $\epsilon$ should be significantly smaller than 1. For this reason we focus on applications where $\epsilon = 0.2$.

With $\epsilon = 0.2$, $\delta = 0.1$ and $p = 0.1$, the condition in Theorem 1 requires that a road’s $\delta$-critical count be at least 127 cars. For such roads, the Theorem states that the estimated travel time will be within 10 percent of the ground truth with probability at least 90 percent. To see whether such a requirement is reasonable, we studied a real world transportation network. The $\delta$-critical capacities of all roads in the Sioux Falls transportation network are plotted in Figure 2. The figure shows that more than 80 percent of the roads in the Sioux Falls network have $\delta$-critical counts above 127. Thus the conditions required by Theorem 1 are realistic for most roads.

Care must also be taken in choosing the value of $\Delta t$, i.e. the amount of time between updates to the estimated congestion state. Specifically, $\Delta t$ should be chosen so that it is similar to the typical travel time of a road in the network. If $\Delta t$ is much smaller than the typical travel time on a road, then sample average approximation can be used to denoise $S(k\Delta t)$ to get a better estimate of $s(k\Delta t)$. While better estimation is usually good, it is also synonymous with less privacy. To illustrate this concept, suppose that for timesteps 1, 2, 3, ..., $N$ the ground truth congestion state for a given road $e$ is constant, meaning that there is a value $s_e$ so that $s_e(k\Delta t) = s_e$ for all $1 \leq k \leq N$. Now for each $k$, $S_e(k\Delta t)$ is an unbiased estimator for $s_e$ with variance $\frac{2}{N\epsilon^2}$. This variance is necessary to ensure differential privacy. However, note that $\frac{1}{N} \sum_{k'=1}^{N} S_e(k'\Delta t)$ is also an unbiased estimator for $s_e$ but now has variance $\frac{2}{N\epsilon^2}$. This decrease in variance leads to worse privacy guarantees. For this reason, $\Delta t$ should be chosen similar to the travel time of a road so that $N$ can never get too large.

### 5 Numerical Experiments

We evaluate the performance of our protocol in the Sioux Falls road network setting. The purpose of these numerical experiments is to compare the travel time experienced by vehicles when they are routed using (a) latency estimates derived from our protocol and (b) latency estimates derived from the ground truth traffic counts. In particular we want to quantify the extent of travel time degradation incurred by the use of our protocol’s privacy-preserving mechanisms. Thus, these experiments help us test the practical implications of our edge-level theoretical results when viewed in the context of the entire network. In Section 5.1, we present details about the data sources, road network, traffic demand, and the experimental setup. In Section 5.2, we study the impact of our
protocol on route choices and travel time. These simulations suggest that the price for privacy may be negligible or even zero, thereby strengthening the case for conducting real-world field studies.

5.1 Setup

The properties of the Sioux Falls road network as well as the typical user demands is obtained from the Transportation Network Test Problems (TNTP) dataset. The Sioux Falls road network consists of 24 nodes and 76 edges. Each edge is characterized by a maximum speed, free flow throughput (i.e., vehicles per hour), and length of the segment. Additionally, the dataset also reports the steady state traffic demand between 528 OD pairs.

We conduct two types of experiments: private routing and non-private routing. In both types of experiments we simulate traffic flow on the network at a time resolution of $\Delta t = 10$ seconds. At every time step, we draw new demand from a Poisson distribution, with the mean demand proportional to the steady state demand reported in the dataset. Thus, at each time step, we draw a random number of vehicles with a corresponding origin and destination. For each vehicle, we identify a shortest travel-time route between the origin and destination node. In the private routing experiment, the shortest path is computed using the most recent travel time estimates produced by our protocol. In the non-private routing experiment, the shortest path is computed using the ground truth travel time which depends on the ground truth traffic counts. In both types of experiments, the simulated movement of the vehicles on the road network is determined by the ground truth traffic counts on that link.

The duration of the simulation is 2 hours, with additional buffer time in the end for vehicles already in transit to complete their trips. The vehicles are assumed to use our protocol to update edge latency estimates every $\Delta t = 2$ minutes to minimize the number of reports that a vehicle makes from the same edge (See Section 4.3). In our experiments, we consider $\epsilon$ values of 0.01 and 0.1. We will discuss how our protocol would perform for other values of $\epsilon$ in Section 5.2. We consider three vehicle demand profiles. In the baseline scenario, about 60,120 vehicles are expected to join the road network every hour. We also consider a low demand scenario, where all the OD Poisson parameters are decreased by 50% and a high demand scenario, where all the OD Poisson parameters are increased by 50%. To gain some more insight into the degree of strain this demand induces on the road network, we refer the reader to Table 1. Here, we report the rate of vehicles being added to the network for each scenario as well as the minimum, maximum, and average edge utilization during the simulation period. The edge utilization is defined as the term $x/c$ from Remark 1. Utilization greater than 1 indicates congestion on a link. The baseline demand profile results in several congested links – representing a realistic scenario for testing our protocol. Additionally, as expected, increasing the demand rate results in a higher edge utilization and more congested links.

5.2 Results

We evaluate the impact of privacy noise on the routes and travel times of vehicles in the network. Table 2 presents several performance measures for $\epsilon = 0.01$ under the three demand profiles. Note that the choice of $\epsilon = 0.01$ is overly conservative. Typical values of $\epsilon$ that are chosen in practice are larger than 0.1 depending on the application. Nevertheless, we consider this relatively stringent privacy requirement to understand the performance limits our protocol.
| Scenarios | Rate (vehicles/hr) | Min edge utilization | Max edge utilization | Average edge utilization |
|-----------|-------------------|----------------------|----------------------|-------------------------|
| Baseline  | 60,100            | 0.05                 | 1.15                 | 0.52                    |
| Low       | 30,050            | 0.00                 | 0.90                 | 0.28                    |
| High      | 90,150            | 0.07                 | 1.47                 | 0.74                    |

Table 1: Demand scenarios used in our simulation

Figure 2: Distribution of $\delta$-critical counts for all edges in the network for $\delta = 0.1$. The dashed blue line denotes the threshold above which the edge travel time estimate will have an error less than 10%, 90% of the time to give a privacy level of $\epsilon = 0.2$. 
| Performance measure                          | Low demand | Baseline demand | High demand |
|---------------------------------------------|------------|-----------------|-------------|
| Travel time (sec)                           | 546.5      | 591.9           | 678.2       |
| Travel time with our protocol (sec)         | 549.6      | 599.9           | 691.4       |
| Increase in travel time (sec)               | 3.1        | 8.0             | 13.2        |
| Increase in travel time (%)                | 0.6        | 1.3             | 1.9         |
| Cars with no change in route (%)           | 90.9       | 88.3            | 87.1        |
| Cars with no increase in travel time (%)   | 65.9       | 41.3            | 20.6        |

Table 2: Performance measures for \( \epsilon = 0.01 \).

| Performance measure                          | Low demand | Baseline demand | High demand |
|---------------------------------------------|------------|-----------------|-------------|
| Travel time (sec)                           | 546.5      | 591.9           | 678.2       |
| Travel time with our protocol (sec)         | 546.5      | 592.2           | 677.4       |
| Increase in travel time (sec)               | 0.01       | 0.22            | -0.7        |
| Increase in travel time (%)                | 0          | 0.04            | -0.1        |
| Cars with no change in route (%)           | 98.4       | 97.5            | 94.4        |
| Cars with no increase in travel time (%)   | 90.7       | 67.9            | 38.6        |

Table 3: Performance measures for \( \epsilon = 0.1 \).

From Table 2, we first observe that the increase in average travel time for a vehicle is only 8 seconds in the baseline case. This corresponds to a 1.3% increase in travel time. For the low and high demand scenarios, the increase in average travel times are 3.1 sec (0.6%) and 13.2 sec (1.9%) respectively. Vehicles will experience a low additional travel time if the routing decisions for vehicles from our protocol closely resembles what they would have done without any privacy noise. In other words, if the shortest paths on the graph with noisy edge latency is the same (or very similar) to the shortest path on a graph with accurate edge latency, the vehicles will experience very little additional travel time. Our results confirm that this is indeed the case – the routes chosen by almost 90% of the vehicles are unchanged due to our protocol. Finally, as we compare the three demand scenarios, we observe that higher demand results in greater congestion and subsequently higher travel times. Furthermore, when demand is higher, the choice of an appropriate route for each vehicle is even more critical to minimize. Working with noisy estimates of link latency, during periods of high congestion can thus lead to incorrect routing choices. This is consistent with our observation that the route of 90.9% of the cars are unchanged by our protocol when the demand is low, but only 87% of the routes remain unchanged when the demand is high.

Next, we study the performance of our protocol with a lower privacy setting of \( \epsilon = 0.1 \). Note that we expect that the performance of our protocol converges to the non-private setting as \( \epsilon \rightarrow \infty \). Interestingly, our results show that even with \( \epsilon = 0.1 \), the performance of our protocol becomes indistinguishable to the non-private setting. The performance of our protocol for \( \epsilon = 0.1 \) is shown in Table 3. We observe that the increase in travel time is nearly 0 for all the three demand scenarios. In fact, the randomness in the edge latency can also result in marginal improvements in travel time in some settings (reflected as a negative increase in travel time). Not surprisingly, the routes chosen by nearly all the cars also is unaffected by our privacy preserving protocol. The results from varying \( \epsilon \) are very encouraging – for a reasonable privacy requirement, we are able to get privacy for ‘free’ with no loss in system performance. Although not shown for brevity, our experiments
indicated that cars observe no increase in average travel time for all values of $\epsilon > 0.1$.

The results from these experiments are significantly better than what is guaranteed by Theorem 1. As we discussed in Section 4.3, Theorem 1 promises that the estimated travel time on each link will be within 10 percent of the ground truth at least 90 percent of the time when $\epsilon = 0.2$. Our numerical results in Table 2 show that even when $\epsilon = 0.01$ (i.e., 20 times as noisy as $\epsilon = 0.2$), our protocol only introduces an overhead of 8 percent in the baseline case and 13 percent in a high demand case. We believe this is because real world road networks have redundancy. By this we mean that there are many near-optimal paths from an origin to a destination, so it is very likely that at least one of these paths has an accurately estimated travel time. Theorem 1 looks only at the edge level and is thus does not exploit favorable network topologies. On a positive note, Theorem 1 is general in the sense that it can be applied to a network with any topology and any demand structure.

6 Conclusion

In this paper we propose a decentralized and privacy-preserving protocol for users of a transportation network to compute travel time estimates based on their location data for the sake of congestion-aware routing. In most current data-sharing schemes, users give their individual data directly to apps in exchange for services. Since this data is associated with the users’ identity, users’ schedules, habits, preferences and other private information can be inferred through repeated interactions with many apps. Contrary to this, the protocol proposed in this paper is both differentially private and cryptographically secure, meaning that only the aggregate effect of traffic on travel time is obtainable from the protocol, and users’ individual location data cannot be inferred by other parties. We also show that for large roads, it is possible to estimate travel time both accurately and privately. This is due to complementary qualities of differential privacy and delay functions. We evaluated the performance of the protocol through simulation in the Sioux Fall transportation network and showed that the protocol incurs minimal performance overhead in practice while providing a principled privacy guarantee.

There are many interesting and important directions for future work. The first direction is related to finding a more refined definition for privacy. Travel time estimation in the literature is often based on flow or average speed of vehicles on the road. However, we chose to estimate travel times based on the number of vehicles on a road due to compatibility with differential privacy. More specifically, it is not possible to compute flow or average speed in both an accurate and differentially private way. Thus while differential privacy is a general and powerful concept, it is perhaps too restrictive for some common mobility applications such as flow or speed estimation. Developing a more specialized notion of privacy for mobility applications could enable more algorithmic possibilities while retaining meaningful privacy guarantees. The second direction is related to adoption rate. In this paper we implicitly assume that all vehicles in the network are willing to participate in the protocol, though technically a uniform and known adoption rate would be sufficient. While we believe this is a reasonable assumption in an era of connected vehicles, developing a protocol that is agnostic to participation rate would provide robustness. A third direction is related to other applications. Developing decentralized and privacy-preserving pricing for roads and for mobility services would be an interesting direction.
References

[1] A. R. Beresford and F. Stajano, “Location privacy in pervasive computing,” *IEEE Pervasive computing*, vol. 2, no. 1, pp. 46–55, 2003.

[2] C.-Y. Chow, M. F. Mokbel, and X. Liu, “Spatial cloaking for anonymous location-based services in mobile peer-to-peer environments,” *GeoInformatica*, vol. 15, no. 2, pp. 351–380, 2011.

[3] C. Dwork, “Differential privacy: A survey of results,” in *International conference on theory and applications of models of computation*, pp. 1–19, Springer, 2008.

[4] O. Goldreich, S. Micali, and A. Wigderson, “How to play any mental game or A completeness theorem for protocols with honest majority,” in *STOC 1987, New York, New York, USA* (A. V. Aho, ed.), pp. 218–229, ACM, 1987.

[5] L. Sweeney, “k-anonymity: A model for protecting privacy,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[6] M. Ghasemzadeh, B. C. Fung, R. Chen, and A. Awasthi, “Anonymizing trajectory data for passenger flow analysis,” *Transportation research part C: emerging technologies*, vol. 39, pp. 63–79, 2014.

[7] B. Y. He and J. Y. Chow, “Optimal privacy control for transport network data sharing,” *Transportation Research Part C: Emerging Technologies*, vol. 113, 2020.

[8] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Theory of Cryptography Conference*, vol. 3876, pp. 265–284, Springer, 2006.

[9] Z. Wang, J. Hu, R. Lv, J. Wei, Q. Wang, D. Yang, and H. Qi, “Personalized privacy-preserving task allocation for mobile crowdsensing,” *IEEE Transactions on Mobile Computing*, vol. 18, no. 6, pp. 1330–1341, 2018.

[10] K. Yan, G. Lu, G. Luo, X. Zheng, L. Tian, and A. M. V. V. Sai, “Location privacy-aware task bidding and assignment for mobile crowd-sensing,” *IEEE Access*, vol. 7, pp. 131929–131943, 2019.

[11] Y. Gong, C. Zhang, Y. Fang, and J. Sun, “Protecting location privacy for task allocation in ad hoc mobile cloud computing,” *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 110–121, 2015.

[12] M. E. Gursoy, L. Liu, S. Truex, and L. Yu, “Differentially private and utility preserving publication of trajectory data,” *IEEE Transactions on Mobile Computing*, vol. 18, no. 10, pp. 2315–2329, 2018.

[13] K. Al-Hussaeni, B. C. Fung, F. Iqbal, G. G. Dagher, and E. G. Park, “SafePath: Differentially-private publishing of passenger trajectories in transportation systems,” *Computer Networks*, vol. 143, pp. 126–139, 2018.

[14] R. Dong, W. Krichene, A. M. Bayen, and S. S. Sastry, “Differential privacy of populations in routing games,” in *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 2798–2803, IEEE, 2015.
[15] Y. Li, D. Yang, and X. Hu, “A differential privacy-based privacy-preserving data publishing algorithm for transit smart card data,” Transportation Research Part C: Emerging Technologies, vol. 115, p. 102634, 2020.

[16] M. Wernke, F. Dürr, and K. Rothermel, “Pshare: Ensuring location privacy in non-trusted systems through multi-secret sharing,” Pervasive and Mobile Computing, vol. 9, no. 3, pp. 339–352, 2013.

[17] W. Ren and S. Tang, “Egeoindis: An effective and efficient location privacy protection framework in traffic density detection,” Vehicular Communications, vol. 21, p. 100187, 2020.

[18] T. Li, L. Lin, and S. Gong, “Autompc: Efficient multi-party computation for secure and privacy-preserving cooperative control of connected autonomous vehicles,” in SafeAI@ AAAI, 2019.

[19] J. Zhang, F. Yang, Z. Ma, Z. Wang, X. Liu, and J. Ma, “A decentralized location privacy-preserving spatial crowdsourcing for internet of vehicles,” IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 4, pp. 2299–2313, 2020.

[20] S. Zou, J. Xi, G. Xu, M. Zhang, and Y. Lu, “Crowdhb: A decentralized location privacy-preserving crowdsensing system based on a hybrid blockchain network,” IEEE Internet of Things Journal, 2021.

[21] V. Pandurangan, “Riding with the stars: Passenger privacy in the nyc taxicab dataset.” Available Online, 2014.

[22] Q. Geng and P. Viswanath, “The optimal mechanism in differential privacy,” in 2014 IEEE International Symposium on Information Theory, 2014.

[23] A. Shamir, “How to share a secret,” Commun. ACM, vol. 22, no. 11, pp. 612–613, 1979.

[24] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, “Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits,” in Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings (J. Crampton, S. Jajodia, and K. Mayes, eds.), vol. 8134 of Lecture Notes in Computer Science, pp. 1–18, Springer, 2013.
A  Proof of Theorem 1

We prove Theorem 1 by proving the following Lemma:

**Lemma 1.** For any $s \in \mathbb{R}_+$, the following inequality

$$|\tau_e(s + Z) - \tau_e(s)| \leq \delta \over \tau_e(s)$$

is satisfied with probability at least $1 - p$ when $Z \sim \mathcal{L}_\epsilon$.

The Theorem follows by applying the result with $s = s_e(k\Delta t)$. We will prove Lemma 1 by considering two exhaustive cases:

Case 1: $s < \log \frac{1}{p\epsilon}$,

Case 2: $s \geq \log \frac{1}{p\epsilon}$.

### A.1 Proving Lemma 1 in Case 1

In Case 1 we have $s < \log \frac{1}{p\epsilon}$. By the condition in Theorem 1, $F_e(c_{e,\delta}) \geq \frac{1}{\epsilon} \left( \frac{1}{\delta} + 1 \right) \log \frac{1}{p}$, and thus $s \leq F_e(c_{e,\delta})$.

Next, note that

$$\mathbb{P}[s + Z \geq F_e(c_{e,\delta})] = \mathbb{P}[Z \geq F_e(c_{e,\delta}) - s]$$

$$\leq \mathbb{P}[Z \geq \left( \frac{1}{\epsilon} \left( \frac{1}{\delta} + 1 \right) \log \frac{1}{p} \right) - \log \frac{1}{p \epsilon}]$$

$$= \mathbb{P}[Z \geq \log \frac{1}{p \epsilon}]$$

$$\overset{(a)}{=} \frac{1}{2} \exp \left( -\epsilon \frac{\log \frac{1}{p}}{\epsilon} \right)$$

$$= \frac{p}{2},$$

where $(a)$ is due to the formula for the cumulative distribution function for the Laplace distribution. Therefore, with probability at least $1 - \frac{p}{2}$, both $s, s + Z$ are less than $F_e(c_{e,\delta})$. For the remainder of the Case 1 discussion we will condition on the high probability event that both $s, s + Z$ are less than $F_e(c_{e,\delta})$.

By Assumption 1 we know that $f_e$ is non-decreasing, which means that $F_e$ is non-decreasing and invertible. From this we can conclude that $F_e^{-1}(s) \leq c_{e,\delta}$ and $F_e^{-1}(s + Z) \leq c_{e,\delta}$. Since $f_e$ is non-decreasing this means

$$f_e(0) \leq f_e(s), f_e(s + Z) \leq f_e(c_{e,\delta}).$$

From this we can deduce that

$$|f_e(s + Z) - f_e(s)| \leq f_e(c_{e,\delta}) - f_e(0)$$
\[ (1 + \delta) f_e(0) - f_e(0) = \delta f_e(0) \leq \delta f_e(s), \]

which establishes Lemma 1 in Case 1.

### A.2 Proving Lemma 1 in Case 2

For Case 2 we have \( s \geq \log \frac{1}{p} \epsilon \delta \). Our analysis for this case will involve \( \frac{d}{dy} F^{-1}_e(y) \) and \( \frac{d}{dy} \tau_e(y) \), so we will compute them using chain rule:

\[
1 = \frac{d}{dy} \frac{d}{dy} F^{-1}_e(y) = F'_e(F^{-1}_e(y)) \left( \frac{d}{dy} F^{-1}_e(y) \right)
\]

\[
\Rightarrow \frac{d}{dy} F^{-1}_e(y) = \frac{1}{F'_e(F^{-1}_e(y))}.
\]

Using this, we can now compute \( \frac{d}{dy} \tau_e(y) \) using chain rule:

\[
\frac{d}{dy} \tau_e(y) = \frac{d}{dy} \frac{d}{dy} f_e(F^{-1}_e(y)) = f'_e(F^{-1}_e(y)) \left( \frac{d}{dy} F^{-1}_e(y) \right)
\]

\[
= \frac{f'_e(F^{-1}_e(y))}{F'_e(F^{-1}_e(y))}
\]

Defining \( x_y := F^{-1}_e(y) \), we see that

\[
\frac{d}{dy} \tau_e(y) = \frac{f'_e(x_y)}{F'_e(x_y)} = \frac{f'_e(x_y)}{x_y f'_e(x_y) + f_e(x_y)}.
\]

We make an observation that will be useful later:

**Observation 1.** Since \( f_e \) is non-negative under Assumption 1, we have \( \frac{d}{dy} \tau_e(y) \leq \frac{1}{x_y} \).

**Observation 2.** \( t_e(y) = \frac{y}{x_y} \). This is because since \( t_e(y) = f_e(F^{-1}_e(y)) \), we have \( F^{-1}_e(y) f_e(F^{-1}_e(y)) = F_e(F^{-1}_e(y)) = y \). Therefore \( t_e(y) = \frac{y}{F^{-1}_e(y)} = \frac{y}{x_y} \).

With this setup in hand, we are now ready to prove the Lemma. First note that

\[
P \left[ |Z| \geq \delta s \right] \overset{(a)}{=} \exp \left( -\epsilon \delta s \right) \leq \exp \left( -\epsilon \delta \frac{\log \frac{1}{p}}{\epsilon \delta} \right)
\]

24
where (a) is due to the fact that $|Z|$ has the exponential distribution with parameter $\frac{1}{\epsilon}$, and this distribution has the cumulative distribution function \( P(|Z| \geq t) = e^{-\epsilon t}1[t \geq 0] \). Thus with probability at least \( 1 - p \), we have $|Z| \leq \delta s$. For the remainder of the Case 2 discussion we will condition on the high probability event that $|Z| \leq \delta s$.

By the fundamental theorem of calculus,

$$|\tau_e(s + Z) - \tau_e(s)| = \left| \int_s^{s+Z} \left( \frac{d}{dy} \tau_e(y) \right) dy \right|$$

$$\leq \int_s^{s+Z} \left| \frac{d}{dy} \tau_e(y) \right| dy$$

(a) \leq \int_s^{s+Z} \frac{1}{|x_y|} dy$$

(b) \leq \int_s^{s+Z} \frac{1}{x_{\min(s,s+Z)}} dy$$

(c) \leq \int_s^{s+Z} \frac{1}{x_{(1-\delta)s}} dy$$

= \frac{|Z|}{x_{(1-\delta)s}}$$

\leq \frac{\delta s}{x_{(1-\delta)s}}$$

= \frac{\delta}{1 - \delta} \frac{(1 - \delta)s}{x_{(1-\delta)s}}$$

(d) = \frac{\delta}{1 - \delta} \tau_e((1 - \delta)s)$$

(e) \leq \frac{\delta}{1 - \delta} \tau_e(s)$$

where (a) is due to Observation 1. Since $x_y$ was defined to be $F_e^{-1}(y)$, (b) is due to the fact that $F_e$ is increasing, therefore $x_y$ is an increasing function of $y$ and hence $\frac{1}{x_y}$ is a decreasing function of $y$. (c) is because $\min(s, s + Z) \geq (1 - \delta)s$ since we are in the event that $|Z| \leq \delta s$. (d) is due to Observation 2 and (e) is because $t_e$ is an increasing function, since it is $f_e$ composed with $F_e^{-1}$, which are both increasing. Since $\frac{\delta}{1 - \delta} = \delta + \frac{\delta^2}{1 - \delta}$, we have

$$|\tau_e(s + Z) - \tau_e(s)| \leq (\delta + O(\delta^2)) \tau_e(s)$$

which proves Lemma 1 in Case 2.