Trust but Verify: Cryptographic Data Privacy for Mobility Management

Matthew Tsao
Stanford University
mwtsao@stanford.edu

Kaidi Yang
Stanford University
kaidi.yang@stanford.edu

Stephen Zoepf
Lacuna Technologies
stephen.zoepf@lacuna.ai

Marco Pavone
Stanford University
pavone@stanford.edu

July 2, 2021

Abstract

The era of Big Data has brought with it a richer understanding of user behavior through massive data sets, which can help organizations optimize the quality of their services. In the context of transportation research, mobility data can provide Municipal Authorities (MA) with insights on how to operate, regulate, or improve the transportation network. Mobility data, however, may contain sensitive information about end users and trade secrets of Mobility Providers (MP). Due to this data privacy concern, MPs may be reluctant to contribute their datasets to MA. Using ideas from cryptography, we propose an interactive protocol between a MA and a MP in which MA obtains insights from mobility data without MP having to reveal its trade secrets or sensitive data of its users. This is accomplished in two steps: a commitment step, and a computation step. In the first step, Merkle commitments and aggregated traffic measurements are used to generate a cryptographic commitment. In the second step, MP extracts insights from the data and sends them to MA. Using the commitment and zero-knowledge proofs, MA can certify that the information received from MP is accurate, without needing to directly inspect the mobility data. We also present a differentially private version of the protocol that is suitable for the large query regime. The protocol is verifiable for both MA and MP in the sense that dishonesty from one party can be detected by the other. The protocol can be readily extended to the more general setting with multiple MPs via secure multi-party computation.

Contents

1   Introduction

The rise of mobility as a service, smart vehicles and smart cities is revolutionizing transportation industries all over the world. Mobility management, which entails operation, regulation, and innovation of transportation systems, can leverage mobility data to improve the efficiency, safety, accessibility, and adaptability of transportation systems far beyond what was previously achievable. The analysis and sharing of mobility data, however, introduces two key concerns. The first concern is data privacy; sharing mobility data can introduce privacy risks to end users that comprise the datasets. The second concern is credibility; in situations where data is not shared, how can the correctness of numerical studies be verified? These concerns motivate the need for data analysis tools for transportation systems which are both *privacy preserving* and *verifiable*.

The *data privacy* issue in transportation is a consequence of the trade-off between data availability and data privacy. While user data can be used to inform infrastructure improvement, equity and green initiatives, the data may contain sensitive user information and trade secrets of mobility providers. As a result, end users and mobility providers may be reluctant to share their data with city authorities. Cities have recently begun mandating micromobility providers to share detailed trajectory data of all trips, arguing that the data is needed to enforce equity or environmental objectives. Some mobility providers argued that while names and other directly identifiable information may not be included in the data, trajectory data can still reveal schedules, routines and habits of the city's inhabitants. The mobility providers' concern over the release of anonymized data is justified. [DMNS06] showed that any attempt to release anonymized data either fails to provide anonymity, or there are low-sensitivity attributes of the original dataset that cannot be determined from the published version. In general, anonymization is increasingly easily defeated by the very techniques that are being developed for many legitimate applications of big data [PCA14]. Such disputes highlight the need for privacy-preserving data analysis tools in transportation.

A communication scheme between a sender and a receiver is *verifiable* if it enables the receiver to determine whether the message or report it receives is an accurate representation of the truth. When the objectives of mobility providers and policy makers are not aligned, one party may benefit from misreporting data or other information, giving rise to verifiability issues in transportation. An example of this is Greyball software [Isa17]. Mobility providers developed Greyball software to deny service or display misleading information to targeted users. It was originally developed to protect their drivers from oppressive authorities in foreign countries, by misreporting driver location to accounts that were believed to belong to the oppressive authorities. However, mobility providers also used Greyball to hide their activity from authorities in the United States when their operations were scrutinized. Another example of verifiability issues is third party wage calculation apps [Szy21]. Drivers, frustrated by instances of being underpaid, created an app to confirm whether the pay was consistent with the length and duration of each trip. Such incidents highlight the need for verifiable data analysis tools in transportation.

1.1   Statement of Contributions

In this paper we propose a protocol between a Municipal Authority and a Mobility Provider that enables the Mobility Provider to send insights from its data to the Municipal Authority in a privacy-preserving and verifiable manner. In contrast to *non-interactive* data sharing mechanisms (which are currently used by most municipalities) where a Municipal Authority is provided an aggregated

and anonymized version of the data to analyze, our proposed protocol is an *interactive* mechanism where a Municipal Authority sends queries and Mobility Providers give responses. By sharing responses to queries rather than the entire dataset, interactive mechanisms circumvent the data anonymization challenges faced by non-interactive approaches [DMNS06, PCA14].

Our proposed protocol, depicted in Figure 1, has three main steps. In the first step, the Mobility Provider uses its data to produce a data identifier which it sends to the Municipal Authority. The Municipal Authority can then send its data query to the Mobility Provider in the second step. In the third step, the Mobility Provider sends its response along with a zero knowledge proof. The Municipal Authority can use the zero knowledge proof to check that the response is consistent with the identifier, i.e., the response was computed from the same data that was used to create the identifier. If the Municipal Authority has multiple queries, steps 2 and 3 are repeated. The protocol is computationally efficient due to zk-SNARK implementations [GWC19] for zero knowledge proofs.

The protocol uses cryptographic commitments and aggregated traffic measurements to ensure that the identifier is properly computed from the true mobility data. In particular, any deviation from the protocol by one party can be detected by the other, making the protocol strategyproof for both parties. Given that the identifier is properly computed, the zero knowledge proof then enables the Municipal Authority to verify the correctness of the response without needing to directly inspect the mobility data. Since the Municipal Authority never needs to inspect the mobility data, the protocol is privacy-preserving.

The protocol can be extended to the more general case of multiple Mobility Providers, each with a piece of the total mobility data. This is done by including a secure multiparty computation in step 3 of the protocol. Answering a large number of queries with our protocol can lead to privacy issues since it was shown in [DN03] that a dataset can be reconstructed from many accurate statistical measurements. To address this concern, we generalize the protocol to enable differentially private responses from the Mobility Provider in large query regimes.

1.2   Organization

This paper is organized as follows. The remainder of the introduction discusses academic work related to privacy and verifiability in transportation networks. In Section 2 we introduce a mathematical model of transportation networks and use it to formulate the data privacy problem for Mobility Management. We provide a high level intuitive description of our proposed protocol in Section 3. In Section 4 we introduce cryptographic tools that are needed to reason about the efficacy of our protocol. In Section 5 we provide a full technical description of our protocol. We discuss some of the technical nuances of the protocol and their implications in Section 6. In Section 7 we present a differentially private version of the protocol that is suitable for the large query regime. Finally, we summarize our work and identify important areas for future research in Section 8.

1.3   Related Work

Within the academic literature, this work is related to the following five fields: misbehavior detection in cooperative intelligent transportation networks, blockchain technologies in transportation systems, data privacy in transportation systems, differential privacy, and secure multiparty computation. We briefly discuss how this work complements ideas from these fields.

Cooperative intelligent transportation networks (cITS) aim to provide benefits to the safety, efficiency, and adaptability of transportation networks by having individual vehicles share their
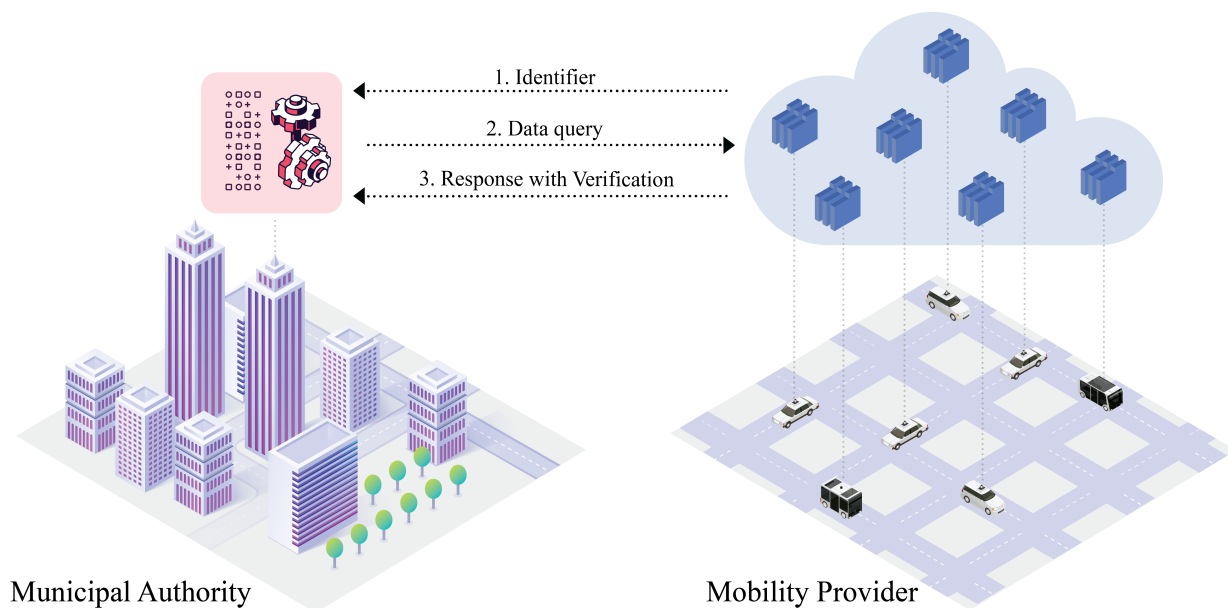
Figure 1: The Mobility Provider can answer the Municipal Authority's data-related mobility queries in a verifiable way *without needing to share the data*. The absence of data sharing in the protocol reduces the chance that a malicious third party intercepts and uses the data for nefarious privacy-invasive purposes.

information. As with all decentralized systems, security and robustness against malicious agents is essential for practical deployment. As such, misbehavior detection in cITS have been studied extensively [vdHDLK19]. Misbehavior detection techniques often rely on honest agents acting as referees, and are able to detect misbehavior in the honest majority setting. Watchdog is one such protocol [MGLB00, HRM10] which uses peer-to-peer refereeing. The protocol uses a public key infrastructure (PKI) to assign a persisting identity to each node in the network, and derives a reputation for each node based on its historical behavior. Our objective in this work is also detection of misbehavior, but in a different setting. In our setting, while the mobility network is comprised of many agents (customers and drivers), there is a single entity (the Mobility Provider, e.g., a ridehailing service) who is responsible for the storage and analysis of trip data. As such, the concept of honest majority does not apply to our setting. Furthermore, [HRM10] does not address the issue of data privacy; indeed, PKIs can often expose the users' identities, especially if an attacker cross-references the network traffic with other traffic records.

Blockchain technologies provide transparency, security, and accountability in a decentralized setting. Bitcoin, the first blockchain network provides the aforementioned transparency, security, and accountability in an electronic cash system [Nak08], and subsequent blockchains such as Ethereum provide these benefits to a wider range of applications through smart contracts [But14]. There has been extensive research in the application of blockchain to transportation [MZN+21, AGMS20], mostly in logistics, supply chain, and road traffic management. The transparency provided by blockchain, however, can be problematic from a privacy perspective. Some works hide user/vehicle identity behind pseudonyms [LLW+18],[LOL+20], but this is far from anonymous. In the Bitcoin

network, which also uses psudonyms, it was shown that it is possible to link all addresses pertaining to a single user [MPJ$^+$13]. In many cases, the users can then be identified from their addresses either by re-identification attacks or "Know Your Customer" regulations. In fact, such identification techniques have been commercialized in companies like Elliptic[1] which aim to identify individuals that are conducting criminal activity on blockchains (e.g., fraud, money laundering). In this work we also want a protocol which is resilient to strategic behavior (i.e., verifiability) and provides privacy for users. Furthermore, the protocol should involve only a few agents (a municipal authority and a mobility provider), so a blockchain setting is not entirely appropriate.

Privacy in intelligent transportation systems is often implemented by using non-interactive anonymization (e.g., data aggregation), cryptographic tools or differential privacy. Providing anonymity in Non-interactive data analysis mechanisms is challenging [DMNS06, PCA14] and thus data aggregation alone is often not enough to provide privacy. From the cryptography side, to address the lack of anonymity provided by blockchains like Bitcoin and Ethereum, zero knowledge proofs [GMR89] were deployed in blockchains like Zcash [SCG$^+$14] to provide fully confidential transactions. In the context of transportation, zero knowledge proofs have been proposed for privacy-preserving vehicle authentication to EV charging services [GAC20], and privacy-preserving driver authentication to customers in ridehailing applications [LMGN20]. These privacy-preserving authentication systems rely on a trusted third party to distribute and manage certificates.

Differential privacy is an interactive mechanism for data privacy which uses randomized responses to hide user-specific information [DMNS06]. For any query, the data collector provides a randomized response, where two datasets which differ in only one entry produce statistically indistinguishable outputs. Due to this randomization, there is a trade-off between the accuracy of the response and the level of privacy provided. Randomization is necessary to preserve privacy in the large query regime as demonstrated by [DN03] which showed that a dataset can be reconstructed from many accurate statistical measurements. The standard model of differential privacy, however, relies on a *trusted* data collector to apply the appropriate randomized response to queries. This is problematic in situations where the data collector is not trusted. A local model of differential privacy where users perturb their data before sending it to the data collector has received significant attention due to trust concerns [KLN$^+$11]. However mobility providers often record exact details about user trips, making local differential privacy unsuitable for current mobility applications.[2] Instead, we believe cryptographic techniques can be used to address trust concerns. There are also more general concerns about trust; downstream applications of data queries can lead to conflicts of interest and encourage strategic behavior.

Secure Multi-Party Computation (MPC) is a technique whereby several players, each possessing private data, can jointly compute a function on their collective data without any player having to reveal their data to other players [GMW87]. MPC achieves confidentiality by applying Shamir's Secret Sharing [Sha79] to inputs and intermediate results. In its base form, MPC is secure against honest-but-curious adversaries, which follow the protocol, but may try to do additional calculations to learn the private data of other players. In general, security against active malicious adversaries, which deviate from the protocol arbitrarily, requires a trusted third party to perform verified secret sharing [CGMA85]. In verified secret sharing, the trusted third party creates initial cryptographic commitments for each player's private data. The commitments do not leak any information about the data, and allows honest players to detect misbehavior using zero knowledge proofs. MPC is a

---

[1]Elliptic is a trusted partner of the Coinbase cryptocurrency exchange. `https://www.elliptic.co/`
[2]See Remark 12.

very promising tool for our problem, but a trusted third party able to eliminate strategic behavior does not yet exist in the transportation industry, therefore a key objective of this work is to develop mechanisms to defend against strategic behavior.

In Summary - Our goal in this work is to develop a protocol that enables a mobility provider to share insights from its data to a municipal authority in a privacy-preserving and verifiable manner. Existing work in accountability and misbehavior detection focus on networks with many agents and rely on honest majority. Such assumptions, however, are not realistic for interactions between a municipal authority and a few mobility providers. We thus turn our attention to differential privacy and secure multiparty computation which provide data privacy but require honesty of participating parties. To address this, we develop mechanisms based on cryptography and aggregated roadside measurements to detect dishonest behavior.

2   Model & Problem Description

In this section we present a model for a city's transportation network and formulate a data Privacy for Mobility Management (PMM) problem. Section 2.1 introduces a mathematical representation of a city's transportation network along with the demand and mobility providers. In Section 2.2 we formalize the notion of data privacy using secure multi-party computation, and introduce assumptions on user behavior that we will need to construct verifiable protocols. We then formally introduce the PMM problem and describe several transportation problems that can be formulated in the PMM framework.

2.1   Transportation Network Model

Transportation Network - Consider the transportation network of a city, which we represent as a directed graph $G = (V, E, f)$ where vertices represent street intersections and edges represent roads. For each road $e \in E$ we use an increasing differentiable convex function $f_e : \mathbb{R}_+ \to \mathbb{R}_+$ to denote the travel cost[3] of the road as a function of the number of vehicles on the road. We will use $n := |V|$ and $m := |E|$ to denote the total number of vertices and edges in G respectively. Time is represented in discrete timesteps of size Δt. The operation horizon is comprised of $T + 1$ timesteps as $\mathcal{T} := \{0, \Delta t, 2\Delta t, ..., T\Delta t\}$.

Mobility Provider - A Mobility Provider (MP) is responsible for serving the transportation demand. It does so by choosing a routing x of its vehicles within the transportation network. The routing must satisfy multi-commodity network flow constraints (see Sections A and B for explicit descriptions of these constraints) and the MP will choose a feasible flow that maximizes its utility function J_{MP}. Some examples of MPs are ridehailing companies, bus companies, train companies, and micromobility (i.e., bikes & scooters) companies.

Transportation Demand Data - The MP's demand data is a list of completed trips $\Lambda := \{\lambda_1, ..., \lambda_q\}$, where λ_i contains the following basic metadata about the ith trip:

Pickup location, Dropoff location, Request time, Match time[4], Pickup time, Dropoff time,

---

[3]This may depend on travel time, distance, and emissions.
[4]the time at which the user is matched to a driver.

Driver wage, Trip fare, Trip trajectory[5], Properties of the service vehicle.

For locations $i, j \in V$ and a timestep t, we use $\Lambda(i, j, t)$ to denote the number of users in the data set who request transit from location i to location j at time t.

Remark 1 (Multiple Mobility Providers)**.** We can consider settings where there are multiple mobility providers, $MP_1, MP_2, ..., MP_\ell$, where Λ_j is the demand data of MP_j. The demand data set for the whole city is thus $\Lambda = \cup_{j=1}^{\ell} \Lambda_j$.

Ridehailing Periods - For MPs that operate ridehailing services, a ridehailing vehicle's trajectory is often divided into three different periods (with Period 0 often ignored):

Period 0: The vehicle is not online with a platform. The driver may be using the vehicle personally.

Period 1: The vehicle is vacant and has not yet been assigned to a rider.

Period 2: The vehicle is vacant, but it has been assigned to a rider, and is en route to pickup.

Period 3: The vehicle is driving a rider from its pickup location to its dropoff location.

2.2 Objective: Privacy for Mobility Management (PMM)

In the data Privacy for Mobility Management (PMM) problem, a Municipal Authority (MA) wants to compute a function $g(\Lambda)$ on the travel demand, where $g(\Lambda)$ is some property of Λ that can inform MA on how to improve public policies. There are two main obstacles to address: privacy and verifiability.

Privacy issues arise since trip information may contain sensitive customer information as well as trade secrets of Mobility Providers (MP). For this reason MPs may be reluctant to contribute their data for MA's computation of $g(\Lambda)$. This motivates the following notion of privacy:

Definition 1 (Privacy in Multi-Party Computation)**.** Suppose $MP_1, ...MP_\ell$ serve the demands $\Lambda_1, ..., \Lambda_\ell$ respectively, and we denote $\Lambda = \cup_{i=1}^{\ell} \Lambda_i$. We say a protocol for computing $g(\Lambda)$ between a MA and several MPs is privacy preserving if

1. MA learns nothing about Λ beyond the value of $g(\Lambda)$.

2. For any pair $i \neq j$, MP_i learns nothing about Λ_j beyond the value of $g(\Lambda)$.

Verifiability issues arise if there is incentive misalignment between the players. In particular, if the MA or a MP can increase their utility by deviating from the protocol, then the computation of $g(\Lambda)$ may be inaccurate. To address this issue, we need the protocol to be verifiable, as described by Definition 2. The following assumption is necessary to ensure accurate reporting of demand (See Appendix D for more details):

Assumption 1 (Strategic Behavior)**.** *We assume in this work that drivers and customers of the transportation network will behave honestly (by this we mean they will always follow the protocol), but MA and MPs may act strategically to maximize their own utility functions.*

---

[5]The vehicle's trajectory from the time the vehicle is matched to the rider until the time the rider is dropped off at their destination.

Definition 2 (Verifiable Protocol)**.** A protocol for computing $g(\Lambda)$ is verifiable under Assumption 1 if:

1. Any deviation from the protocol by the MA can be detected by the MPs provided that all riders and drivers act honestly (i.e., follow the protocol).

2. Any deviation from the protocol by an MP can be detected by the MA provided that all riders and drivers act honestly.

Our objective in this paper is to present a PMM protocol, which is defined below.

Definition 3 (PMM Protocol)**.** A PMM protocol between a MA and $MP_1, ...MP_\ell$ can, given any function g, compute $g(\Lambda)$ for MA while ensuring privacy and verifiability as described by Definitions 1 and 2 respectively.

Remark 2 (Admissible Queries and Differential Privacy)**.** While a PMM protocol hides all information about Λ beyond the value of $g(\Lambda)$, $g(\Lambda)$ itself may contain sensitive information about Λ. The extreme case would be if g is the identity function, i.e., $g(\Lambda) = \Lambda$. In such a case, the MPs should reject the request to protect the privacy of its customers. More generally, MPs should reject functions g if $g(\Lambda)$ is highly correlated with sensitive information in Λ. The precise details as to which functions g are deemed acceptable queries must be decided upon beforehand by MA and the MPs together.

Differential privacy mechanisms provide a principled way to address the sensitivity of g by having MPs include noise in the computation of $g(\Lambda)$. If the noise distribution is chosen according to both the desired privacy level and the sensitivity of g to its inputs, then the output is differentially private. Note that this privacy is not for free; the noise reduces the accuracy of the output. The precise choice of noise distribution is important for both the privacy and accuracy of this method, so ensuring that the randomization step is conducted properly in the face of strategic MAs and MPs is essential. This can be done with a combination of coinflipping protocols and secure multi-party computation, which we describe in Section 7.

We now present some important social decision making problems that can be formulated within the PMM framework.

2.2.1   Regulation Compliance for Mobility Providers

Suppose MA wants to check whether a MP is operating within a set of regulations $\rho_1, ..., \rho_k$. The metadata contained within each trip includes request time, match time, pickup time, dropoff time, and trip trajectory, which can be used to check regulation compliance. If we define the function $\rho_i(\Lambda)$ to be 1 if and only if regulation i is satisfied, and 0 otherwise, then regulation compliance can be determined from the function $g(\Lambda) := \prod_{t=1}^{k} \rho_t(\Lambda)$. Below are some examples of regulations that can be enforced using trip metadata.

Example 1 (Waiting Time Equity)**.** MP is not discriminating against certain requests due to the pickup or droppoff locations. Specifically, the difference in average waiting time among different regions should not exceed a specified regulatory threshold.

Example 2 (Congestion Contribution Limit)**.** The contribution of MP vehicles (in Period 2 or 3) to congestion should not exceed a specified regulatory threshold.

Example 3 (Accurate Reporting of Period 2 Miles)**.** A ridehailing driver's pay per mile/minute depends on which period they are in. In particular, the earning rate for period 2 is often greater than that of period 1. For this reason, mobility providers are incentivized to report period 2 activity as period 1 activity. To protect ridehailing drivers, accurate reporting of period 2 activity should be enforced.

Example 4 (Emissions Limit)**.** The collective emission rate of MP vehicles in Phases 2 and 3. should not exceed a specified regulatory threshold. MP emissions can be computed from the metadata of served trips, in particular the trajectory and vehicle make and model.

See Appendix C for further details on formulating the above examples within the PMM framework.

2.2.2   Transportation Infrastructure Development Projects

Transportation Infrastructure Improvment Projects - A Municipal Authority (MA) measures the efficiency of the current transportation network via a concave social welfare function $J_{\mathrm{MA}}(x)$. The MA wants to make improvements to the network G through infrastructure improvement projects. Below are some examples of such projects.

Example 5 (Building new roads)**.** The MA builds new roads E_{new} so the set of roads is now $E \cup E_{\mathrm{new}}$, i.e., G now has more edges. The impact of additional roads on system utility depends on the demand Λ.

Example 6 (Building Train tracks)**.** The MA builds new train routes. Train routes differ from roads in that the travel time is independent of the number of passengers, i.e., there is no congestion effect.

Example 7 (Adding lanes to existing roads)**.** The MA adds more lanes to some roads $E' \subset E$. As a consequence, the shape of f_e will change for each $e \in E'$. The impact of additional lanes on utility depends on the demand Λ.

Example 8 (Adjusting Speed limits)**.** Similar to adding more lanes, adjusting the speed limit of a road will change its delay function.

Evaluation of Projects - We measure the utility of a project using a Social Optimization Problem (SOP). An infrastructure improvement project θ makes changes to the transit network, so let G_θ denote the transit network obtained by implementing θ. The routing problem $\mathrm{ROUTE}(\theta, \Lambda)$ associated with θ is the optimal way to serve requests in G_θ as measured by MP's objective function J_{MP}. Letting $S_{\theta,\Lambda}$ be the set of flows satisfying multi-commodity network flow constraints[6] for the graph G_θ and demand Λ, $\mathrm{ROUTE}(\theta, \Lambda)$ is given by

$$\max J_{\mathrm{MP}}(x) \qquad\qquad\qquad (\mathrm{ROUTE}(\theta, \Lambda))$$
$$\text{s.t. } x \in S_{\theta,\Lambda}.$$

Definition 4 (The Infrastructure Development Selection Problem)**.** Suppose there are k infrastructure improvement projects $\Theta := \{\theta_1, \theta_2, ..., \theta_k\}$ available, but the city only has the budget for

---

[6]See Sections A and B for time-varying and steady state formulations respectively.

one project. The city will want to implement the project that yields the most utility, which is determined by the following optimization problem.

$$\operatorname*{argmax}_{1 \leq i \leq k} J_{\text{MA}} \left(\operatorname*{argmax}_{x \in S_{\theta_i, \Lambda}} J_{\text{MP}}(x) \right). \tag{SOP(Θ, Λ)}$$

In the context of PMM, the function g associated with the infrastructure development selection problem is $g(\Lambda) := \text{SOP}(\Theta, \Lambda)$.

2.2.3 Congestion Pricing

Some ridehailing services allow drivers to choose the route they take when delivering customers. When individual drivers prioritize minimizing their own travel time and disregard the negative externalities they place on other travelers, the resulting user equilibrium can experience significantly more congestion than the social optimum. In these cases, the total travel time of the user equilibrium is larger than that of the social optimum. This gap, known as the price of anarchy, is well studied in the congestion games literature.

Congestion pricing addresses this issue by using road tolls to incentivize self-interested drivers to choose routes so that the total travel time of all users is minimized. The desired road tolls depend on the demand Λ, so MA would need help from MPs to compute the prices. Congestion pricing can be formulated in the PMM framework through the query function g_{cp} described in (2).

When the travel cost is the same as travel time, the prices can be obtained from the following optimization problem

$$\max \sum_{e \in E} x_e f_e(x_e) \tag{1}$$
$$\text{s.t. } x \in S_\Lambda$$

where S_Λ is the set of flows that satisfy Multi-Commodity Network flow constraints[7] for the transit network G and demand Λ. The objective measures the sum of the travel times of all requests in Λ. The desired prices are then given by:

$$g_{\text{cp}}(\Lambda) := \left\{ x_e^* f_e'(x_e^*) \right\}_{e \in E} \text{ where } x^* \text{ solves (1).} \tag{2}$$

See Appendix F for more details on congestion pricing.

3 A high level description of the protocol

We focus our discussion on the case where there is one MP. The protocol we will present can be generalized to the multiple MP setting through secure Multiparty Computation [GMW87]. The simplest way for MA to obtain $g(\Lambda)$ is via a *non-interactive* protocol where MP sends Λ to MA. MA could then compute $g(\Lambda)$ and any other attributes of Λ that it wants to know. This simple procedure, however, does not satisfy data privacy, since MA now has full access to the demand Λ.

To address this concern, one could use an *interactive* protocol where MA sends a description of the function g to MP, MP then computes $g(\Lambda)$ and sends it to MA. This protocol does not require

---

[7]See footnote 6.

MP to share the demand Λ. The problem with this approach is that there is no way for MA to check whether MP computed $g(\Lambda)$ properly, i.e., this approach is not verifiable. This is problematic if there is an incentive for MP to act strategically, e.g., if MP wants to maximize its own revenue, rather than social utility.

In this paper we present a verifiable interactive protocol, which allows MA to check whether or not the message it receives from MP is in fact $g(\Lambda)$. This will result in a protocol where MA is able to obtain $g(\Lambda)$ without requiring MP to reveal any information about Λ beyond the value of $g(\Lambda)$.

First, we describe a non-confidential way to compute $g(\Lambda)$. We will discuss how to make it confidential in the next paragraph. MP will send a commitment $\sigma = \mathsf{MCommit}(\Lambda, r)$ of Λ to MA. This commitment will enable MA to certify that the result given to it by MP is computed using the true demand Λ. The commitment is confidential, meaning it reveals nothing about Λ, and is binding, meaning that it will be inconsistent with any other demand $\Lambda' \neq \Lambda$. Now suppose MP computes a message $z = g(\Lambda)$. To convince MA that the calculation is correct, MP will construct a witness $w := (\Lambda, r)$. When MA receives the message z and witness w, it will compute $C(\sigma, z, w)$, where C is an evaluation algorithm. $C(\sigma, z, w)$ will be `True` if

1. Rider Witness and Aggregated Roadside Audit checks are satisfied. (σ was reported honestly)

2. $\mathsf{MCommit}(\Lambda, r) = \sigma$. ($\Lambda$ is the demand that was used to compute σ).

3. $g(\Lambda) = z$ (g was evaluated properly.)

If any of these conditions are not met, $C(\sigma, z, w)$ will be `False`. Finally, MA will accept the message z only if $C(\sigma, z, w) = $ `True`.

The approach presented in the previous paragraph is not privacy-preserving because the witness w being sent from MP to MA includes the demand Λ. Fortunately, we can use zero knowledge proofs to obtain privacy. Given an arithmetic circuit C (which in our case is the evaluation algorithm C), it is possible for one entity (the prover) to convince another entity (the verifier) that it knows an input z, w so that $C(\sigma, z, w) = $ `True` without revealing what w is. This is done by constructing a zero knowledge proof π from (z, w) and sending (z, π) to the verifier instead of sending (z, w). MA can then check whether π is a valid proof for z. The proof π is zero knowledge in the sense that it is computationally intractable to deduce anything about w from π, aside from the fact $C(\sigma, z, w) = $ `True`. For our application, the prover will be MP who is trying to convince the verifier, which is MA, that it computed $g(\Lambda)$ correctly.

This protocol requires MP to send a commitment of the true demand data to MA. This is problematic if MP has incentive to be dishonest, i.e., provide a commitment corresponding to a different dataset. To ensure this does not happen, our protocol uses a Rider Witness incentive to prevent MP from underreporting demand, and Aggregated Roadside Audits to prevent MP from overreporting demand. These two mechanisms establish the verifiability of the protocol, since, as seen in first requirement of C, MA will reject the message if either of these mechanisms detect dishonesty.

In Summary - We present a verifiable interactive protocol. First, MP sends a commitment of the demand to MA, which ensures that the report is computed using the true demand. The correctness of this commitment is enforced by Rider Witness and Aggregated Roadside Audits. MA then announces the function g that it wants to evaluate. MP computes a message $z \leftarrow g(\Lambda)$ and constructs a witness w to the correctness of z. Since w in general contains sensitive information, it cannot be used directly to convince MA to accept the message z. MP computes a zero knowledge

proof π of the correctness of z from w, and sends the message z and proof π to MA. MA accepts z if π is a valid zero knowledge proof for z.

Implementation - To implement our protocol we will use several tools from cryptography. The commitment σ is implemented as a Merkle commitment. For computing zero knowledge proofs, we will be using PLONK, which is a zero knowledge proof system in which the construction and verification of proofs is computationally efficient. In the next section we present these cryptographic tools.

4 Cryptographic Tools

In this section we introduce existing cryptographic tools that will be used in the protocol. The contents of this section are discussed in greater detail in [NBF$^+$16, BS20, GMR89, SCG$^+$14, GWC19]. Throughout this paper, we use $r||x$ to denote the concatenation of r and x.

4.1 Cryptographic Hash Functions

Definition 5 (Cryptographic Hash Functions). A function H is a d-bit cryptographic hash function if it is a mapping from binary strings of arbitrary length to $\{0,1\}^d$ and has the following properties:

1. It is deterministic.

2. It is efficient to compute.

3. H is *collision resistant* - For sufficiently large d, it is computationally intractable to find distinct inputs x_1, x_2 so that $H(x_1) = H(x_2)$.

4. H is *hiding* - If r is a sufficiently long random string (256 bits is often sufficient), then it is computationally intractable to deduce anything about x by observing $H(r||x)$.

Property 3 is called collision resistance and enables the hash function to be used as a digital fingerprint. Indeed, since it is unlikely that two files will have the same hash value, $H(x)$ can serve as a unique identifier for x. We refer the interested reader to [NBF$^+$16] for further details on cryptographic hash functions.

SHA256 is a widely used collision resistant hash function which has extensive applications including but not limited to establishing secure communication channels, computing checksums for online downloads, and computing proof-of-work in Bitcoin.

4.2 Cryptographic Commitments

Cryptographic commitment schemes are tamper-proof communication protocols between two parties: a sender and a receiver. In a commitment scheme, the sender chooses (i.e., commits to) a message. At a later time, the sender reveals the message to the receiver, and the receiver can be sure that the message it received is the same as the original message chosen by the sender.

Intuition - We can think of a commitment scheme as follows: A sender places a message into a box and locks the box with a key. The sender then gives the locked box to the receiver. Once the sender has given the box away, the sender can no longer change the message inside the box. At

this point, the receiver, who does not have the key, cannot open the box to read the message. At a later time, the sender can give the key to the receiver, allowing the receiver to read the message.

A commitment scheme is specified by a message space $\mathcal{M}$, nonce space $\mathcal{R}$, commitment space $\mathcal{X}$, a commitment function $\mathsf{commit} : \mathcal{M} \times \mathcal{R} \to \mathcal{X}$ and a verification function $\mathsf{verify} : \mathcal{M} \times \mathcal{R} \times \mathcal{X} \to \{0, 1\}$. Creating a commitment to a message $m \in \mathcal{M}$ happens in two steps:

1. *Commitment Step* - The sender computes $\sigma := \mathsf{commit}(m, r)$ for some $r \in \mathcal{R}$ and gives σ to the receiver.

2. *Reveal Step* - At some later time, the sender gives m, r to the receiver who accepts m as the original message if and only if $\mathsf{verify}(m, r, \sigma) := \mathbb{1}_{[\mathsf{commit}(m,r)=\sigma]}$ evaluates to 1.

A secure commitment scheme has two important properties:

1. *Binding* - If σ is a commitment to a value m, it is computationally intractable to find m', r' so that $m' \neq m$ and $\mathsf{commit}(m', r') = \sigma$. Hence σ binds the committer to the value m.

2. *Hiding* - It is computationally intractable to learn anything about m from σ.

Cryptographic hash functions can be used to build secure commitment schemes. To do this, given a cryptographic hash function H, we define

$$\mathsf{commit}(m, r) := H(r||m) \text{ and } \mathsf{verify}(m, r, \sigma) := \mathbb{1}_{[H(r||m)=\sigma]}.$$

The security of this commitment scheme comes from the properties of H. The binding property of this commitment scheme follows directly from collision resistance of H. Furthermore, if r is chosen uniformly at random from $\mathcal{R}$, then the commitment scheme is hiding due to the hiding property of H.

4.3   Merkle Trees

A Merkle tree is a data structure that is used to create commitments to a collection of items $M := \{m_0, ..., m_{q-1}\}$. A Merkle Tree has two main features:

1. The root of the tree contains a hiding commitment to the entire collection M.

2. The root can also serve as a commitment to each item $m \in M$. Furthermore, the proof that m is a leaf of the tree reveals nothing about the other items in M and has length $O(\log q)$, where q is the total number of leaves in the tree.

A Merkle tree can be constructed from a cryptographic hash function. Concretely, given a cryptographic hash function H and a collection of items $m_0, ..., m_{q-1}$, construct a binary tree with these items as the leaves.

The leaves of the Merkle Tree are the zeroth level $h_{0,0}, h_{0,1}, ..., h_{0,q-1}$, where $h_{0,i} = m_i$. The next level has the same number of nodes $h_{1,0}, ..., h_{1,q-1}$ defined by $h_{1,i} = H(r_i||m_i)$ where r_i is a random nonce. Level k where $k \geq 2$, has half as many nodes as level $k - 1$, defined by $h_{k,i} = H(h_{k-1,2i}||h_{k-1,2i+1})$. Figure 2 illustrates an example of a Merkle tree. In total there are

$\ell_t + 1$ levels where $\ell_q := \lceil \log_2 q \rceil + 1$. With this notation, $h_{\ell_q,0}$ is the value at the root of the Merkle Tree.

The root of a Merkle tree $h_{\ell_q,0}$ is a commitment to the entire collection due to collision resistance of H. To commit to the data M, the committer will generate $r_0, ..., r_{q-1}$, compute the Merkle Tree, and announce $h_{\ell_q,0}$. In the reveal step, the committer can announce $\{(m_i, r_i)\}_{i=0}^{q-1}$, and anyone can then compute the resulting Merkle tree and confirm that the root is equal to $h_{\ell_q,0}$.

4.3.1 Merkle Proofs

The root also serves as a commitment to each $m_i \in M$. Suppose someone who knows m_i wants a proof that m_i is a leaf in the Merkle tree. A proof $\pi(m_i)$ can be constructed from the Merkle tree. Furthermore, this proof reveals nothing about the other items $\{m_j\}_{j \neq i}$.

Define $x_0, x_1, ... x_{\ell_q}$ recursively as:

$$x_0 := i$$
$$x_j := \left\lfloor \frac{x_{j-1}}{2} \right\rfloor \text{ for } 1 \leq j \leq \ell_q.$$

With this notation, $\left\{ h_{j,x_j} \right\}_{j=0}^{\ell_q}$ is the path from m_i to the root of the Merkle Tree. The Merkle proof for m_i is denoted as $\pi(m_i)$ and is given by

$$\pi(m_i) := \{r_i\} \cup \left\{ \text{sibling}(h_{j,x_j}) \right\}_{j=1}^{\ell_q}, \text{ where}$$
$$\text{sibling}(h_{i,j}) := \begin{cases} h_{i,j+1} & \text{if } j \text{ is even,} \\ h_{i,j-1} & \text{if } j \text{ is odd.} \end{cases}$$

See Appendix G for details on the binding and hiding properties of Merkle commitments, and how to verify the correctness of Merkle proofs.

Definition 6 (Merkle Commitment). Given a data set $M = \{m_1, ..., m_t\}$ and a set of random nonce values $r = \{r_1, ..., r_t\}$, we use $\mathsf{MCommit}(M, r)$ to denote the root of the Merkle Tree constructed from the data M and random nonces r.

We refer the interested reader to Section 8.9 of [BS20] for more details on Merkle Trees.

4.4 Digital Signatures

A digital signature scheme is comprised by three functions: $\mathsf{Gen}, \mathsf{sign}, \mathsf{verify}$. $\mathsf{Gen}()$ is a random function that produces valid public and private key pairs $(\mathsf{pk}, \mathsf{sk})$. Given a message, a signature is produced using the secret key via $\sigma = \mathsf{sign}(\mathsf{sk}, m)$. The authenticity of the signature is checked using the public key via $\mathsf{verify}(\mathsf{pk}, m, \sigma)$. A secure digital signature scheme has two properties:

1. *Correctness* - For a valid key pair $(\mathsf{pk}, \mathsf{sk})$ obtained from $\mathsf{Gen}()$ and any message m, we have $\mathsf{verify}(\mathsf{pk}, m, \mathsf{sign}(\mathsf{sk}, m)) = \texttt{True}$.

2. *Secure* - Given a public key pk, if the corresponding secret key sk is unknown, then it is computationally intractable to forge a signature on any message. Specifically, if sk has never been used to sign a message m', then without knowledge of sk, it is computationally intractable to find (m', σ') so that $\mathsf{verify}(\mathsf{pk}, m', \sigma') = \texttt{True}$.

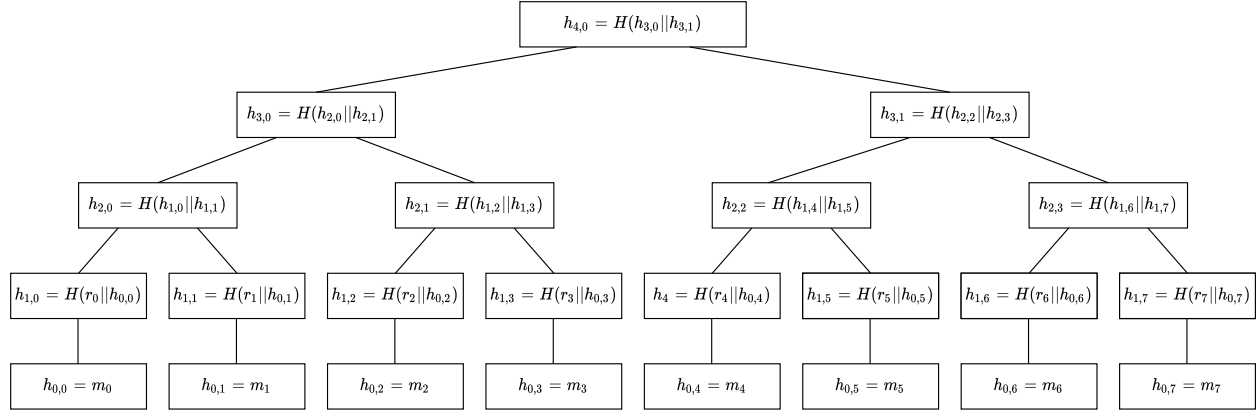We refer the interested reader to Section 13 of [BS20] for more details on digital signatures.

Figure 2: An example of a Merkle tree containing 8 items. Each item m_i is a leaf node and has one parent which is $H(r_i||m_i)$, where r_i is a random hiding nonce. All other internal nodes are computed by applying H to the concatenation of its children.

4.5   Public Key Encryption

A public key encryption scheme is specified by three functions: a key generation function, an encryption function E, and a decryption function D. In a public key encryption scheme, each user has a public key and private key denoted $(\mathsf{pk}, \mathsf{sk})$, produced by the key generation function. As the name suggests, the public key pk is known to everyone, while each secret key sk is known only by its owner. Encryption is done using public keys, and decryption is done using secret keys. To send a message m to Bob, one would encrypt m using Bob's public key via $c = E(\mathsf{pk}_{\mathrm{Bob}}, m)$. Then Bob would decrypt the message via $D(\mathsf{sk}_{\mathrm{Bob}}, c)$. A secure Public Key Encryption scheme has two properties:

1. *Correctness* - For every valid key pair $(\mathsf{pk}, \mathsf{sk})$ and any message m, we have $m = D(\mathsf{sk}, E(\mathsf{pk}, m))$, i.e., the intended recipient receives the correct message upon decryption.

2. *Secure* - For a public key pk and any message m, if the corresponding secret key sk is not known, then it is computationally intractable to deduce anything about m from the ciphertext $E(\mathsf{pk}, m)$.

The appeal of public key encryption is that users do not have to have a shared common key in order to send encrypted messages to one another. We refer the interested reader to Part II of [BS20] for more details on Public Key Encryption.

4.6   Zero Knowledge Proofs

A zero knowledge proof for a mathematical problem is a technique whereby one party (the prover) can convince another party (the verifier) that it knows a solution w to the problem without revealing any information about w other than the fact that it is a solution. Before discussing zero knowledge proofs further, we must first introduce proof systems.

Definition 7 (Proof System)**.** Consider an arithmetic circuit $C : \mathcal{X} \times \mathcal{W} \to \{0, 1\}$, and the following optimization problem: For a fixed $x \in \mathcal{X}$, find a $w \in \mathcal{W}$ so that $C(x, w) = 0$. Here x is part of the problem statement, and w is a solution candidate. Consider a tuple of functions (S, V, P) where

1. S is a *preprocessing function* that takes as input C, x and outputs public parameters pp.

2. P is a *prover function* that takes as input pp, x, w and produces a proof π.

3. V is a *verification function* that takes as input pp, x, π and outputs either 0 or 1 corresponding to whether the proof π is invalid or valid respectively.

The tuple (S, V, P) is a proof system for C if it satisfies the following properties:

1. *Completeness* - If $C(x, w) = 0$, then $V(\mathsf{pp}, x, P(\mathsf{pp}, x, w))$ should evaluate to 1; i.e., the verifier should accept proofs constructed from valid solutions w.

2. *Proof of Knowledge* - If $V(\mathsf{pp}, x, \pi) = 1$, then whoever constructed π must have known a w satisfying $C(x, w) = 0$.

With this definition in hand, we can now define zero knowledge proof systems.

Definition 8 (Zero Knowledge Proof Systems)**.** Consider a proof system (S, V, P) for the problem of finding w so that $C(x, w) = 0$. (S, V, P) is a zero knowledge proof system if it is computationally intractable to learn anything about w from $\pi := P(\mathsf{pp}, x, w)$. If this is the case, then π is a zero knowledge proof.

Zero knowledge proofs were first proposed by [GMR89], but the prover and verifier functions were not optimized to be computationally efficient. In the next section, we present zk-SNARKs, which are computationally efficient zero knowledge proof systems.

4.7 zk-SNARKs

In this section we introduce Succinct Non-interactive Arguments of Knowledge (SNARK). SNARKs are proof systems where proofs are short, and both the construction and verification of proofs are computationally efficient.

Definition 9 (Succinct Non-interactive Argument of Knowledge (SNARK))**.** Consider the problem of finding $w \in \mathcal{W}$ so that $C(x, w) = 0$, where C is an arithmetic circuit with n logic gates. A proof system (S, V, P) is a SNARK if

1. The runtime of the prover P is $\widetilde{O}(n)$,

2. The length of a proof computed by P is $O(\log n)$,

3. The runtime of the verifier V is $O(\log n)$.

Definition 10 (zk-SNARK)**.** If a SNARK (S, V, P) is also a zero knowledge proof system, then it is a zk-SNARK.

The Zcash cryptocurrency, which provides fully confidential transactions, was the first setting where zk-SNARKs have been used in the field [SCG+14]. zk-SNARKs have also been deployed in the zk-rollup procedure which increases the transaction throughput of the Ethereum blockchain [But16].

PLONK [GWC19] is a recently developed zk-SNARK that we will use for PMM.

5   The Protocol

In this section we present our protocol for the PMM problem described in Section 2.2. For clarity and simplicity of exposition we will focus on the case where there is one Mobility Provider. The single MP case can be extended to the multiple MP case via secure multi-party computation [GMW87]. We present the protocol, which is illustrated in Figure 3, in Section 5.1. In Section 5.2 we discuss mechanisms used to ensure verifiability of the protocol.

5.1   Protocol Description

The protocol entails 6 stages:

Stage 0: (Data Collection) MP serves the demand Λ and builds a Merkle Tree T_Λ of the demand it serves. MP publishes the root of T_Λ, which is denoted as $\sigma := \mathsf{MCommit}(\Lambda, r)$ so that MA, all riders and all drivers have access to σ. Here r is the set of nonces used to make the commitment confidential.

Stage 1: (Integrity Checks) MA instantiates Rider Witness and Aggregated Roadside Audits to ensure that σ was computed using the true demand Λ. The description of these mechanisms can be found in Section 5.2.

Stage 2: (Message Specifications) MA specifies to MP the function g it wants to compute.

Stage 3: (zk-SNARK Construction) MA constructs an evaluation circuit C for the function g. σ, z are public parameters of C, and the input to C is a witness of the form $w = (\Lambda_w, r_w, c_w)$, where r_w is a set of nonces, Λ_w is a demand matrix, and c_w is an optional input that may depend on g (See Remark 3). C does the following:

1. Checks whether the Rider Witness and Aggregated Roadside Audit tests are satisfied (This checks that σ was reported honestly),

2. Checks whether $\mathsf{MCommit}(\Lambda_w, r_w) = \sigma$ (This determines whether the provided demand Λ_w is the same as the demand that created σ),

3. Checks whether $g(\Lambda_w) = z$ (This checks that the message z is computed properly from Λ_w).

C will return `True` if and only if all of those checks pass. Now, using PLONK [GWC19], MA will create a zk-SNARK (S, V, P) for C. It sends $C, (S, V, P), g$ to MP.

Stage 4: (Function Evaluation) If the request g is not a privacy-invasive function,[8] MP will compute a message $z = g(\Lambda)$ and construct a witness $w := (\Lambda, r, c_w)$ to the correctness of z.

Stage 5: (Creating a Zero Knowledge Proof) MP uses the zk-SNARK's prover function P to construct a proof $\pi := P(\sigma, z, w)$ that certifies the calculation of z. MP sends z, π to MA.
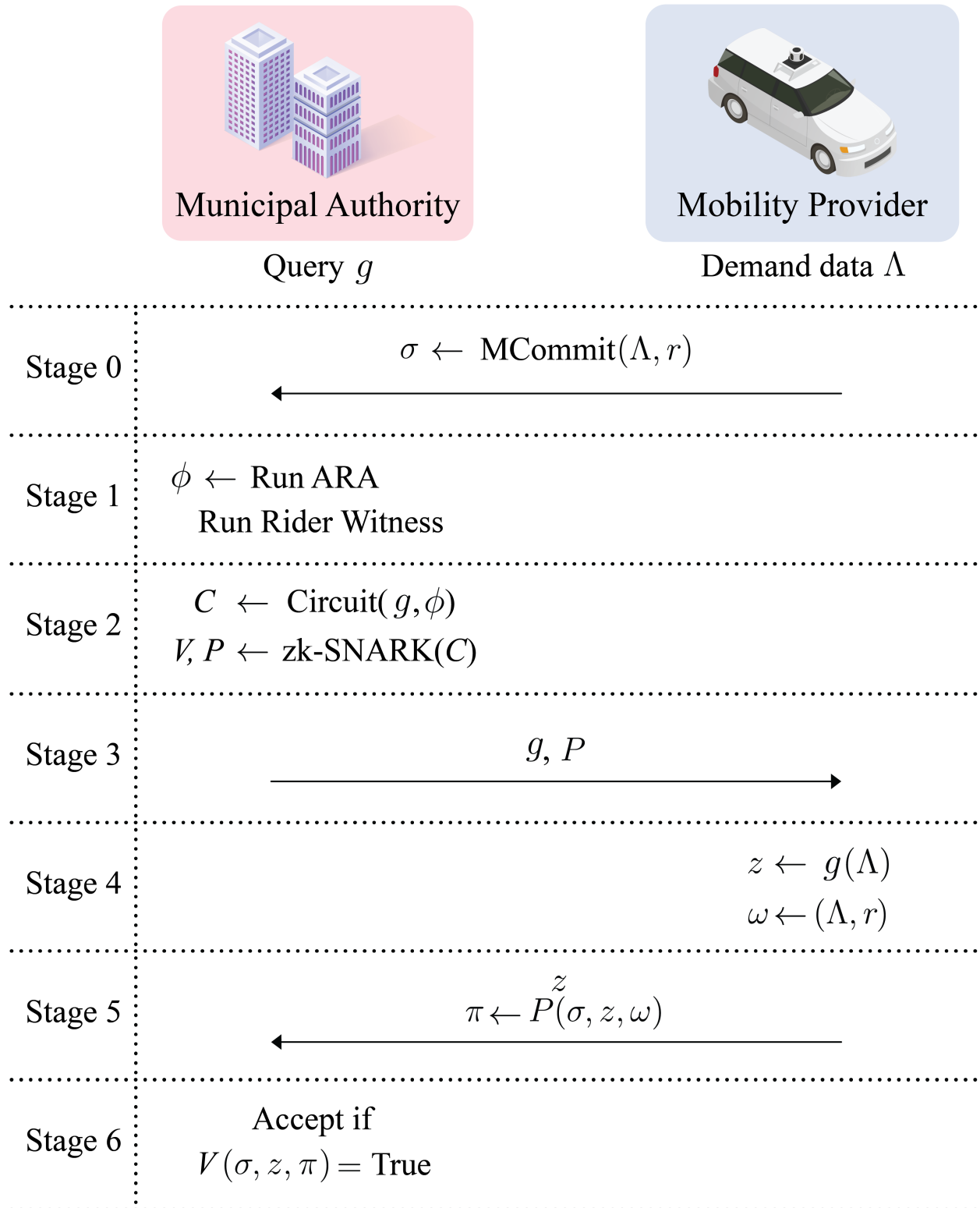
---

[8]See Remark 2.

Figure 3: A block diagram of the communication between MA and MP.

Stage 6: (zk-SNARK Verification) MA uses the zk-SNARK's verification function $V(\sigma, z, \pi)$ to check whether MP is giving a properly computed message. If this is the case, MA accepts the message z.

Remark 3 (Verifying solutions to convex optimization problems)**.** If $g(\Lambda_w)$ is the solution to a convex optimization problem parameterized by Λ_w, (e.g., $g(\Lambda_w) = \mathrm{SOP}(\Theta, \Lambda_w)$ or congestion pricing $g_{\mathrm{cp}}(\Lambda_w)$), then computing $g(\Lambda_w)$ in the verification step may be expensive and cause C to be a very large circuit. Fortunately, this can be avoided by leveraging the structure of convex problems. If $z = g(\Lambda_w)$, we can include the optimal primal and dual variables associated with z in the optional input c_w. This way, checking the optimality of z can be done by checking that c_w satisfy the KKT conditions rather than needing to re-solve the problem.

5.2 Ensuring accuracy of σ

The protocol presented in the previous section requires MP to share a commitment to the true demand Λ. However, scenarios exist where the MP may face direct or indirect incentives to misreport demand, such as per-ride fees, congestion charges, or other regulations that may constrain MP operations. In this section we present mechanisms to ensure that MP submits a commitment $\sigma = \mathsf{MCommit}(\Lambda, r)$ corresponding to the true demand Λ rather than a commitment $\sigma' = \mathsf{MCommit}(\Lambda', r)$ corresponding to some other demand Λ'. Specifically, we present Rider Witness and Aggregated Roadside Audits which detect underreporting and overreporting of demand respectively, which is visualized in Figure 4.
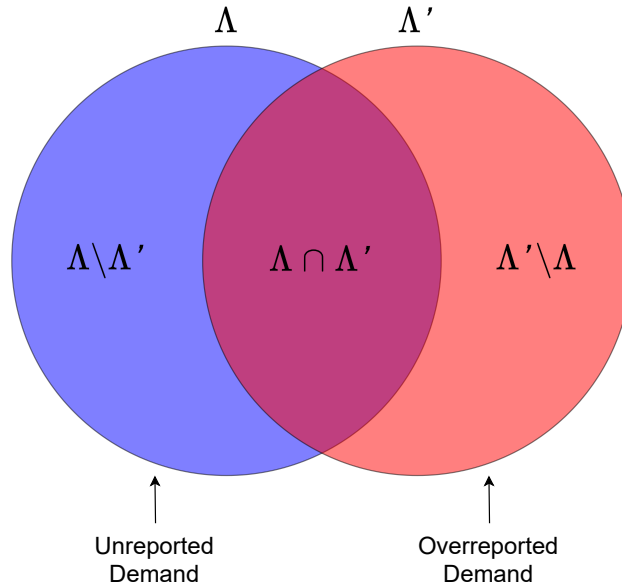
Figure 4: A reported demand Λ' differs from Λ when either $\Lambda \setminus \Lambda'$ or $\Lambda' \setminus \Lambda$ is non-empty. Under Assumption 1, Rider Witness can detect if $\Lambda \setminus \Lambda'$ (blue) is non-empty, i.e., demand has been underreported. Given that $\Lambda \setminus \Lambda'$ is empty, Aggregated Roadside Audits can detect if $\Lambda' \setminus \Lambda$ (red) is non-empty, i.e., demand has been overreported. Thus together, these mechanisms can detect whether σ is a commitment to the true demand Λ.

The Rider Witness mechanism described in Section 5.2.1 prevents MP from omitting real trips from its commitment. Under the Rider Witness mechanism, each rider is given a receipt for their trip signed by MP. By signing a receipt, the trip is recognized as genuine by MP. Since $\sigma' = \mathsf{MCommit}(\Lambda', r)$ is a Merkle commitment, for each $\lambda' \in \Lambda'$, MP can provide a proof that λ' is included in the calculation of σ'. Conversely, if $\lambda \notin \Lambda'$, MP is unable to forge a valid proof to claim that λ is included in the calculation of σ'. Therefore if there exists a genuine trip $\lambda \in \Lambda$ that is not included in Λ', then that rider can report its receipt to MA. MP cannot provide a proof that λ was included, and since the receipt of λ is signed by MP, this is evidence that MP omitted a genuine trip from σ'. If this happens, MP is fined, and the reporting rider is rewarded.

The Aggregated Roadside Audit mechanism described in Section 5.2.2 prevents MP from adding fictitious trips into its commitment. Due to Rider Witness, MP will not omit genuine trips, so $\sigma' = \mathsf{MCommit}(\Lambda', r)$ where $\Lambda \subseteq \Lambda'$. Recall that the trip metadata includes the trajectory. If Λ' contains fictitious trips, then the road usage reported by Λ' will be greater than what happens in reality. Thus if MA measures the number of passenger carrying vehicles that traverse each road, then it will be able to detect if MP has included fictitious trips. However, auditing every road can lead to privacy violations. Therefore, the audits are aggregated so that MA obtains the total volume of passenger carrying traffic in the entire network, but not the per-road traffic information.

5.2.1 Rider Witness: Detecting underreported demand

In this section, we present a Rider Witness mechanism to detect omission or tampering of the demand Λ. Concretely, if a MP sends to MA a Merkle commitment $\sigma' = \mathsf{MCommit}(\Lambda', r)$ which underreports demand, i.e., $\Lambda \setminus \Lambda'$ is non-empty, then Rider Witness will enable MA to detect this. MA can impose fines or other penalties when such detection occurs to deter MP from underreporting the demand.

Rider Witness Incentive Mechanism - At the beginning of Stage 0 (Data Collection) of the protocol, MP constructs a public key and private key pair $(\mathsf{pk}_{\mathrm{mp}}, \mathsf{sk}_{\mathrm{mp}})$ to use for digital signatures. The payment process is as follows: When the ith customer is delivered to their destination, the customer will send a random nonce r_i to MP. MP will respond with a receipt $(H(r_i||\lambda_i), \sigma_i)$, where $\sigma_i := \mathsf{sign}(\mathsf{sk}_{\mathrm{mp}}, H(r_i||\lambda_i))$ is a digital signature certifying that MP recognizes λ_i as an official ride. Here H is SHA256, so that $H(r_i||\lambda_i)$ is a cryptographic commitment to the trip λ_i. The customer is required to pay the trip fare only if $\mathsf{verify}(\mathsf{pk}_{\mathrm{mp}}, H(r_i||\lambda_i), \sigma_i) = \mathtt{True}$, i.e., they received a valid receipt.

Definition 11 (Rider Witness Test)**.** Given a commitment σ' reported by MP to MA, each rider who was served by MP requests a Merkle proof that their ride is included in the computation of σ'. If there exists a valid[9] ride receipt $(H(r_i||\lambda_i), \sigma_i)$ for which MP cannot provide a Merkle proof, then the customer associated with λ_i will report $(H(r_i||\lambda_i), \sigma_i)$ to MA. MA checks if σ_i is a valid signature for $H(r_i||\lambda_i)$, and if so, directly asks MP for a Merkle Proof that λ_i is included in the computation of σ'. If MP is unable to provide the proof, then σ' fails the Rider Witness Test.

Observation 1 (Efficacy of Rider Witness)**.** *Under Assumption 1, if MP submits a commitment $\sigma' = \mathsf{MCommit}(\Lambda', r)$ which omits a ride, i.e., $\Lambda \setminus \Lambda'$ is non-empty, then σ' will fail the Rider Witness Test.*

---

[9]In the sense that $\mathsf{verify}(\mathsf{pk}_{\mathrm{mp}}, H(r_i||\lambda_i), \sigma_i) = \mathtt{True}$.

Proof of Observation 1. If $\Lambda \not\subseteq \Lambda'$, then there exists some λ_i which is in Λ but not Λ'. Suppose Alice was the rider served by ride λ_i. Forging a proof that $\lambda_i \in \Lambda'$ requires finding a hash collision for the hash function used in the Merkle commitment. Since $\mathsf{MCommit}$ is implemented using a cryptographic hash function (e.g., SHA256), it is computationally intractable to find a hash collision, and thus MP will be unable to forge a valid proof that $\lambda_i \in \Lambda'$.

If MP does not provide Alice a valid proof within a reasonable amount of time (e.g., several hours), Alice can then report $(H(r_i||\lambda_i), \sigma_i)$ to MA. This reporting does not compromise Alice's privacy due to the hiding property of cryptographic hash functions. MA will check whether $\mathsf{verify}(\mathsf{pk}_{\mathrm{mp}}, H(r_i||\lambda_i), \sigma_i) = \texttt{True}$, and if so, means that λ_i is recognized as a genuine trip by MP. MA will directly ask MP for a Merkle proof that $H(r_i||\lambda_i) \in T_\Lambda$. Since MP cannot provide a valid proof, this is evidence that a genuine trip was omitted in the computation of σ', and hence σ' will fail the Rider Witness test. $\qquad\square$

In the event that MP's commitment σ' fails the Rider Witness Test, MA can fine MP \$$N$. The purpose of this fine is to deter MP from omitting trips. The \$$N$ can then be given to Alice to encourage customers to report their receipts if they do not receive a Merkle proof from MP. If the fine is chosen large enough, then MP is incentivized to include all trips to avoid paying the fine.

Remark 4 (Tamperproof Property)**.** We note that Rider Witness also prevents the MP from altering the data associated with genuine rides. If MP makes changes to $\lambda_i \in \Lambda$ resulting in some λ_i', then by collision resistance of H, it is computationally infeasible to find r' so that $H(r_i||\lambda_i) = H(r_i'||\lambda_i')$. If such a change is made, then $H(r_i'||\lambda_i')$ is included into the computation of σ' instead of $H(r_i||\lambda_i)$. This means $(H(r_i||\lambda_i), \sigma_i)$ becomes a valid witness that data tampering has occurred.

Remark 5 (Receipts are Unforgeable)**.** Note that it is not possible for a rider to report a fake ride $\lambda' \notin \Lambda$ to MA to make \$$N$. This is because the corresponding signature σ' cannot be forged without knowing MP's secret key $\mathsf{sk}_{\mathrm{mp}}$. Therefore, assuming $\mathsf{sk}_{\mathrm{mp}}$ is only known to MP, only genuine trips can be reported.

Remark 6 (Honesty of riders)**.** The Rider Witness protocol assumes that riders are honest, i.e., they will not collude with MP by accepting invalid receipts.

5.2.2   Aggregated Roadside Audits: Detecting overreported demand

In this section we present an Aggregated Roadside Audit (ARA) mechanism to detect overreporting of demand. Concretely, if MP announces a commitment $\sigma' = \mathsf{MCommit}(\Lambda', r)$, where Λ' is a strict superset of Λ (i.e., $\Lambda' \setminus \Lambda$ is non-empty), then ARA will enable MA to detect this. Thus between ARA and Rider Witness, MA can detect if MP commits to a demand that is not Λ.

Aggregated Roadside Audits - Due to the Rider Witness mechanism, we can assume that MP submits a commitment σ' computed from Λ' satisfying $\Lambda \subseteq \Lambda'$, i.e., Λ' is a superset of Λ. For an edge $e \in E$ and a demand Λ, define

$$\varphi(e, \Lambda) := \sum_{\lambda \in \Lambda} \mathbb{1}_{[\lambda \text{ traverses } e]} \tag{3}$$

to be the number of trips that traversed e during passenger pickup (Period 2) or passenger delivery (Period 3). Since trip route is provided in the trip metadata, $\varphi(e, \Lambda)$ can be computed from Λ.

Definition 12 (ARA Test). The Aggregated Roadside Audit places a sensor on every road to conduct an audit on each road $e \in E$ to measure $\varphi(e, \Lambda)$. These values are then aggregated as $\phi := \sum_{e \in E} \varphi(e, \Lambda)$. A witness $w = (\Lambda_w, r_w, c_w)$ passes the ARA test if and only if

$$\sum_{e \in E} \varphi(e, \Lambda_w) = \phi. \tag{ARA}$$

Observation 2 (Efficacy of Aggregated Roadside Audits). *Under Assumption 1, if MP submits a commitment $\sigma' = \mathsf{MCommit}(\Lambda', r)$ to a strict superset of the demand, i.e., $\Lambda \subset \Lambda'$, then any proof submitted by MP will either be inconsistent with σ' or will fail the ARA test. Hence MP cannot overreport demand.*

Proof of Observation 2. Suppose Λ' is a strict superset of Λ, which means that there exists some $\lambda' \in \Lambda' \setminus \Lambda$. Then there must exist some $e' \in E$ for which $\varphi(e', \Lambda') > \varphi(e', \Lambda)$. In particular, any edge in the trip route of λ' will satisfy this condition. With the inclusion of the ARA test, MP is unable to provide a valid witness for MA's evaluation algorithm C (and as a consequence, will be unable to produce a valid zero knowledge proof) for the following reason:

1. $\mathsf{MCommit}$ is a collision-resistant function (since it is built using a cryptographic hash function H), so because $\sigma' = \mathsf{MCommit}(\Lambda', r)$, it is computationally intractable for MP to find $\Lambda'' \neq \Lambda'$ and nonce values r'' so that $\mathsf{MCommit}(\Lambda'', r'') = \sigma'$. Therefore, in order to satisfy condition 2 of C (see *Stage 3* of Section 5.1), MP's witness must choose Λ_w to be Λ'.

2. However, Λ' will not pass the ARA test. To see this, note that (a) $\Lambda \subseteq \Lambda'$ implies that $\varphi(e, \Lambda) \leq \varphi(e, \Lambda')$ for all $e \in E$. Furthermore, (b) there exists an edge e' where the inequality is strict, i.e., $\varphi(e', \Lambda) < \varphi(e', \Lambda')$. From this, we see that

$$\phi = \sum_{e \in E} \varphi(e, \Lambda) = \varphi(e', \Lambda) + \sum_{e \in \Lambda, e \neq e'} \varphi(e, \Lambda)$$
$$\overset{(a)}{\leq} \varphi(e', \Lambda) + \sum_{e \in \Lambda, e \neq e'} \varphi(e, \Lambda')$$
$$\overset{(b)}{<} \varphi(e', \Lambda') + \sum_{e \in \Lambda, e \neq e'} \varphi(e, \Lambda') = \sum_{e \in E} \varphi(e, \Lambda'),$$

i.e., if the witness passes condition 2 of C, then it will fail the ARA test.

$\square$

Therefore the value of ϕ can be used to detect fictitious rides. See Figure 5 for a visualization of ARA.

Remark 7 (Privacy-Preserving computation of ϕ). We note that the computation of ϕ must be done without compromising data privacy. The naïve way to compute ϕ is for MA to collect the values $\varphi(e, \Lambda)$ from each road. This, however, can compromise data privacy. Indeed, if there is only 1 request in Λ, then measuring the number of customer carrying vehicles that traverse each link exposes the trip route of that request: Edges that are traversed 1 time are in the route, and edges

that are traversed 0 times are not. More generally, observing $\varphi(e, \Lambda)$ on all roads $e \in E$ exposes trip routes to or from very unpopular locations.

Fortunately, the ARA sensors can use public key encryption to share their data with each other to compute ϕ so that MA cannot learn $\varphi(e, \Lambda)$ for any $e \in E$ even if it tries to eavesdrop on the communication between the sensors. After ϕ has been sent to MA and the protocol has finished, the data on the sensors should be erased.

Remark 8 (Error Tolerance in ARA). Trip trajectories are often recorded via GPS, so GPS errors can lead to inconsistencies between ARA sensor measurements and reported trajectories. To prevent an honest MP from failing the ARA test due to GPS errors, one can use an error tolerant version of the ARA test defined below

$$\left| \phi - \sum_{e \in E} \varphi(e, \Lambda_w) \right| \leq \epsilon \phi$$

where $\epsilon \in [0, 1]$ is a tuneable tolerance parameter to account for GPS errors while still detecting non-negligible overreporting of demand.

Remark 9 (Honesty of Drivers). It is essential that MA acts truthfully when taking measurement and computing ϕ in ARA, otherwise MP will be wrongfully accused of dishonesty. In the next section, we discuss how the ARA sensors can be designed to ensure truthful computation of ϕ. We also note that ARA assumes that drivers are honest when declaring their current period to ARA sensors, e.g., a driver who is in period 3 will not report themselves as period 1 or 2.

5.2.3   Implementation details for ARA

In this section we describe the implementation details of ARA to ensure that MA conducts its audits truthfully to avoid MP being wrongfully accused of dishonesty.

ARA Sensors - To implement ARA, MA designs a sensor to detect MP vehicles. Concretely, the sensor records the current period of all MP vehicles that pass by. For communication, the sensor will generate a random public and private key pair, and share its public key with the other sensors. The sensor should have hardware to enable it to encrypt and decrypt messages it sends and receives, respectively. To ensure honest auditing by MA, these sensors are inspected by MP to ensure that they detect MP vehicles properly, key generation, encryption and decryption are functioning properly, and that there are no other functionalities. Once the sensors have passed the inspection, the following storage and communication restrictions are placed on them:

1. The device can only transmit data if it receives permission from both MP and MA.

2. The device can only transmit to addresses (i.e., public keys) that are on its sender whitelist. The sender whitelist is managed by both MA and MP, i.e., an address can only be added with the permission of MA and MP.

3. The device can only receive data from addresses that are on its receiver whitelist. The receiver whitelist is managed by both MA and MP.

4. The device's storage can be remotely erased with permission from both MP and MA.

Figure 5: An example of ARA. The true demand is Λ, which results in traffic shown on the left. Here $\varphi(e_{ij}, \Lambda)$ is the total number of trips in Λ that use the edge from i to j. Suppose MP submits a commitment to $\Lambda' = \Lambda \cup \{\lambda'\}$, i.e., inserts a fake trip λ' into the commitment. In this example, λ' is a fake trip from 5 to 2 that MP claims was served via the route $\{e_{56}, e_{63}, e_{31}, e_{12}\}$ (shown in red on the right). λ' increases the total traffic on the roads $e_{56}, e_{63}, e_{31}, e_{12}$ and as a result, we have $\sum_{e \in E} \varphi(e, \Lambda') = \phi + 4$.

Figure 6: An ARA sensor records the vehicle ID number, vehicle period and current timestamp of each ridehailing vehicle that traverses the road. The dashed line around the sensor represents a communication restriction: The sensor data can only be accessed with the consent of both parties.

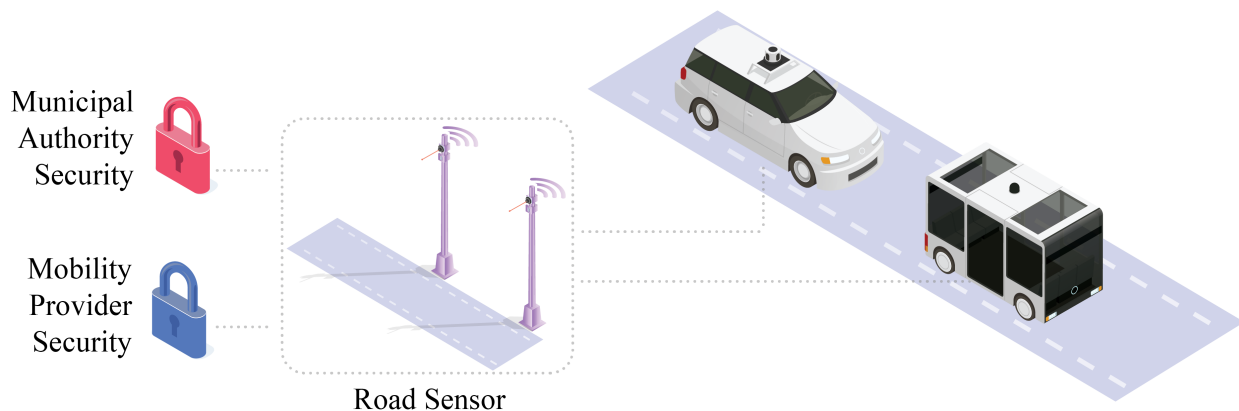*Deployment* - To conduct ARA, a sensor is placed on every road, and will record the timestamp and period information of MP vehicles that pass by during the operation period. During operation, both the sender and receiver whitelists should be empty. As a consequence, MA cannot retrieve the sensor data. After the operation period ends and MP has sent a commitment σ to MA, MP and MA conduct a coin flipping protocol to choose one sensor at random to elect as a leader[10]. A coin flipping protocol is a procedure where several parties can generate unbiased random bits. The leader sensor's public key is added to the whitelist of all other sensors, and all sensors are added to the leader's receiving whitelist. Each sensor then encrypts and sends its data under the leader sensor's public key. Since the MA does not know the leader sensor's secret key, it cannot decrypt the data even if it intercepts the ciphertexts. The addresses of MA and MP are then added to the leader's sender whitelist. The leader sensor decrypts the data, computes ϕ and reports the result to both MA and MP. Once the protocol is over, the sender and receiver whitelists of all sensors are cleared, and MA and MP both give permission for the sensors to delete their data. Figure 6 illustrates the sensor setup for ARA.

6   Discussion

In the case that MP is honest, i.e., σ is guaranteed to be computed via $\mathsf{MCommit}(\Lambda, r)$, PMM can be conducted without revealing any information about Λ beyond the value of $g(\Lambda)$. In such a setting, the integrity checks in stage 1 of the protocol can be skipped. However, if MP cannot be assumed to act honestly, then MA must take some measurements to ensure that MP is reporting σ truthfully. Through Rider Witness, MA will learn the total number of trips and through ARA, MA learns the total volume of MP traffic during the operation period. Due to the level of aggregation, we do not believe such information would lead to a privacy breach for the mobility users.

The protocol requires minimal computational resources from the MA. Indeed, the computation

---

[10]The leader is elected randomly for the sake of robustness. If the leader is the same every time, then the system would be unable to function if this sensor malfunctions or is compromised in any way.

of $g(\Lambda)$, and all data analysis therein, is conducted by the MPs. The MA only needs to construct an evaluation circuit C and zk-SNARK (S, V, P) for each of their queries g. Both of these steps are computationally efficient. In terms of data storage, the MA only needs to store the commitments σ to the demand and the total recorded volume of MP traffic ϕ for each data collecting period. If the Merkle Trees are built using the SHA256 hash function, then σ is only 256 bits, and is thus easy to store. ϕ is a single integer, which is also easy to store.

On the other hand, the hardware requirements for the Aggregated Roadside Audits may be difficult for cities to implement, as placing a sensor on every road in the city will be expensive. To address this concern, we present an alternative mechanism known as Randomized Roadside Audits (RRA) in Appendix E. RRA is able to use fewer sensors by randomly sampling the roads to be audited, however as a tradeoff for using fewer sensors, overreported demand will only be detected probabilistically. See Appendix E for more details.

There is a trade-off between privacy and diagnosis when using zero knowledge proofs. In the event that the zk-SNARK's verification function fails, i.e., $V(\sigma, z, \pi) = \texttt{False}$, we know that z is not a valid message, but we do not know *why* it is invalid. Specifically, $V(\sigma, z, \pi)$ does not specify which step of the evaluation algorithm C failed (See Stage 3 of Section 5.1). Thus in order to determine whether the failure was due to integrity checks, inconsistency between Λ and σ, or a mistake in the computation of g, further investigation would be required. Thus, while the zero knowledge proof enables us to check the correctness of z without directly inspecting the data, it does not provide any diagnosis in the event that z is invalid.

Multi-party computation is a natural way to generalize the proposed protocol to the multiple MP setting. In such a case, the demand $\Lambda = \cup_{i=1}^{k} \Lambda_k$ is the disjoint union of $\Lambda_1, ..., \Lambda_k$, where Λ_i is the demand served by the ith MP, and is hence the private data of the ith MP. Multi-party computation is a procedure by which several players can compute a function over their combined data without any player learning the private data of other players. In the context of PMM with multiple MPs, the MPs are the players and their private data is the Λ_i's. In stage 0, each MP would send to MA a commitment to its demand data, and the computation of z and π in stages 4 and 5 would be done using secure multi-party computation. Verifiability is established using Rider Witness and ARA, as is done in the single MP case. See [LJA$^+$18] and $\texttt{multiparty.org}$ for an open-source implementation of multi-party computation.

7   Incorporating Differential Privacy for the Large Query Regime

One potential concern with the protocol described in Section 5 arises in the large query regime. It was shown in [DN03] that a dataset can be reconstructed from many accurate statistical measurements. One way to address this is to set a limit on the number of times the MA can query the data for a given time period. Such a restriction would not lead to data scarcity since the MP is collecting new data daily. Differential privacy offers a principled way to determine how many times MA should query a dataset (see Remark 10). Differentially private mechanisms address the result of [DN03] by reducing the accuracy of the responses to queries, i.e., responding to a query g with a noisy version of $g(\Lambda)$. In this section we describe how the protocol from section 5 can be generalized to facilitate verifiable and differentially private responses from MP. To this end we first define differential privacy.

Definition 13 (Datasets and Adjacency). A dataset Λ is a set of datapoints. In the context of

transportation demand, a datapoint is the metadata corresponding to a single trip. We say two datasets Λ, Λ' are adjacent if either (a) $\Lambda \subset \Lambda'$ with Λ' containing exactly 1 more datapoint than Λ, or (b) $\Lambda' \subset \Lambda$ with Λ containing exactly 1 more datapoint than Λ'.

Definition 14 (Differential Privacy)**.** Let $\mathcal{F}$ be a σ-algebra on a space Ω. A mechanism $M : \mathcal{D} \to \Omega$ is (ϵ, δ)-differentially private if for any two adjacent datasets $\Lambda, \Lambda' \in \mathcal{D}$ and any $\mathcal{F}$-measurable event S,

$$\mathbb{P}\left(M(\Lambda) \in S\right) \leq e^{\epsilon} \mathbb{P}\left(M(\Lambda') \in S\right) + \delta.$$

In words, the output of a (ϵ, δ)-differentially private mechanism on Λ is statistically indistinguishable from the output of the mechanism on $\Lambda \cup \{\lambda\}$ for any single datapoint $\lambda \notin \Lambda$. Since Λ does not contain λ, $M(\Lambda)$ does not reveal any information about λ. Since $M(\Lambda \cup \{\lambda\})$ is statistically indistinguishable from $M(\Lambda)$, $M(\Lambda \cup \{\lambda\})$ does not reveal much about λ.

Example 9 (Laplace Mechanism for Vote Tallying)**.** Suppose a city is trying to decide whether to expand its railways or expand its roads based on a majority vote from its citizens. The dataset is $\Lambda := \{\lambda_1, ..., \lambda_n\}$ where λ_i is a boolean which is 0 if the ith citizen prefers the railway and 1 if the ith citizen prefers the roads. To implement majority vote, the city needs to compute $g(\Lambda) := \sum_{i=1}^{n} \lambda_i$. The Laplace Mechanism achieves $(\epsilon, 0)$-differential privacy for this computation via

$$M_{\text{laplace}}(\Lambda) := Y + \sum_{i=1}^{n} \lambda_i,$$

where Y has the discrete Laplace distribution: for any $k \in \mathbb{Z}$, $\mathbb{P}[Y = k] \propto e^{-\epsilon|k|}$. To see why this achieves $(\epsilon, 0)$-differential privacy, for any $1 \leq j \leq n$, note that

$$\frac{\mathbb{P}[M(\Lambda) = k]}{\mathbb{P}[M(\Lambda \setminus \{\lambda_j\}) = k]} = \frac{e^{-\epsilon\left|k - \sum_{i=1}^{n} \lambda_i\right|}}{e^{-\epsilon\left|k - \sum_{i \neq j} \lambda_i\right|}} \leq e^{\epsilon \lambda_j} \leq e^{\epsilon}.$$

Note that the noise distribution for Y depends only on ϵ, and is independent of n, the size of the dataset.

Remark 10 (Privacy Budget)**.** By composition rules, the result of k queries to a $(\epsilon, 0)$-differentially private mechanism is $(k\epsilon, 0)$-differentially private. Thus a dataset should only be used to answer k separate $(\epsilon, 0)$-differentially private queries if $e^{k\epsilon}$ is sufficiently close to 1.

7.1   Goal: Differential Privacy without Trust

Given a query function g from MA, let M be an efficiently computable[11] (ϵ, δ)-differentially private mechanism for computing g. For a given dataset Λ we can represent the random variable $M(\Lambda)$ with a function $\widetilde{g}(\Lambda, Z)$ where $Z \in \{0, 1\}^v$ represents the random bits used by M. Here v is an upper bound on the number of random bits needed for the computation of M. By its construction, $\widetilde{g}(\Lambda, Z)$ is (ϵ, δ)-differentially private if Z is drawn uniformly at random over $\{0, 1\}^v$. Therefore differential privacy is achieved if MP draws Z uniformly at random over $\{0, 1\}^v$ and sends $\widetilde{g}(\Lambda, Z)$

---

[11]If $\tau(\Lambda)$ is the time needed to compute $g(\Lambda)$, then the time needed to compute $M(\Lambda)$ should be polynomial in $\tau(\Lambda)$. This also implies that the number of random bits v needed by M is also polynomial in $\tau(\Lambda)$.

to MA. However, as mentioned in Assumption 1, we are studying a model where MP can act strategically. Thus we cannot assume that MP will sample Z uniformly at random if there is some other distribution over Z that leads to a more favorable outcome for MP. We revisit Example 9 to illustrate this concern.

Example 10 (Dishonest Vote Tallying). Consider the setting from Example 9. The Laplace mechanism can be represented as

$$\widetilde{g}(\Lambda, Z) := Y + \sum_{i=1}^{n} \lambda_i, \text{ where } Y = F_{\text{laplace}}^{-1}\left(\frac{\text{int}(Z)}{2^v}\right),$$

where $\text{int}(Z)$ is the integer whose binary representation is the bits of Z.[12] Suppose the MP has a ridehailing service and would thus prefer an upgrade to city roads over an upgrade to the railway system. If this is the case, choosing Z so that $\widetilde{g}(\Lambda, Z) > n/2$ (as opposed to choosing Z randomly) is a weakly dominant strategy for MP, even if $g(\Lambda) < n/2$ and a majority of the citizens prefer railway upgrades.

Thus we need a way to verify that the randomness Z used in MP's evaluation of $g(\Lambda, Z)$ has the correct distribution. We will now show how the protocol can be adjusted to accommodate this, and as a consequence, enable verifiable differentially private data queries for MA.

Remark 11 (MA provided randomness). One natural attempt to ensure that Z is uniformly random is to have MA specify Z. However, this destroys the differential privacy, since for some mechanisms (including the Laplace mechanism) $g(\Lambda)$ can be computed from $\widetilde{g}(\Lambda, Z)$ and Z. Also, it is not clear a priori whether such a setup is strategyproof for MA.

7.2 A Differentially Private version of the protocol

In this section, we present modifications to the protocol from Section 5.1 that enables verifiable differentially private responses from MP. At a high level, the MA and MP jointly determine the random bits Z via a coin flipping protocol [Blu82]. The zk-SNARK can then be modified to ensure that $\widetilde{g}(\Lambda, Z)$ is computed correctly. The protocol has a total of 6 stages, described below.

Stage 0: (Data Collection) MP builds a Merkle Tree T_Λ of the demand Λ that it serves. It computes a commitment $\sigma := \mathsf{MCommit}(\Lambda, r)$ to this demand. Additionally, MP samples Z_{mp} uniformly at random from $\{0,1\}^v$ and computes a Pedersen commitment [Ped91] $z_{\text{mp}} := \mathsf{Commit}(Z_{\text{mp}}, r_{\text{mp}})$. The Pederson commitment scheme is a secure commitment scheme which is perfectly hiding and computationally binding. MP sends both σ, z_{mp} to MA.

Stage 1: (Integrity Checks) Same as in Section 5.1.

Stage 2: (Message Specifications) MA specifies the function g it wants to compute. Additionally, MA samples Z_{ma} uniformly at random from $\{0,1\}^v$ and specifies a differentially private mechanism

---

[12]Here F_{laplace}^{-1} is the inverse cumulative distribution for the discrete Laplace distribution. Thus $F_{\text{laplace}}^{-1}(\text{int}(Z)/2^v)$ is an application of inverse transform sampling that converts a uniform random variable Z into a random variable Y with a discrete Laplace distribution.

$\widetilde{g}$ for the computation of g.

Stage 3: (zk-SNARK Construction) MA constructs an evaluation circuit C for the function $\widetilde{g}$. The public parameters of C are $\sigma, z_{\mathrm{mp}}, Z_{\mathrm{ma}}, z$ and the input to C is a witness of the form $w = (\Lambda_w, r_w, c_w, Z_{\mathrm{mp},w}, r_{\mathrm{mp},w})$. C does the following:

1. Checks whether the Rider Witness and Aggregated Roadside Audit tests are satisfied,

2. Checks whether $\mathsf{MCommit}(\Lambda_w, r_w) = \sigma$,

3. Checks whether $\mathsf{Commit}(Z_{\mathrm{mp},w}, r_{\mathrm{mp},w}) = z_{\mathrm{mp}}$,

4. Checks whether $\widetilde{g}(\Lambda_w, Z_{\mathrm{ma}} \oplus Z_{\mathrm{mp},w}) = z$. (Here $\oplus$ is bit-wise XOR.)

C will return `True` if and only if all of these checks pass. MA constructs a zk-SNARK (S, V, P) for C and sends $g, \widetilde{g}, Z_{\mathrm{ma}}, C, (S, V, P)$ to MP.

Stage 4: (Function Evaluation) If $\widetilde{g}$ is a differentially private mechanism for computing g, then MP computes a message $z = \widetilde{g}(\Lambda, Z_{\mathrm{ma}} \oplus Z_{\mathrm{mp}})$ and a witness $w := (\Lambda, r, c_w, Z_{\mathrm{mp}}, r_{\mathrm{mp}})$ to the correctness of z.

Stage 5: (Creating a Zero Knowledge Proof) Same as in Section 5.1.

Stage 6: (zk-SNARK Verification) Same as in Section 5.1.

We now show that this protocol has two desirable features:

1. Verifiability - If the MA receives a valid proof from MP, then it can be sure that the corresponding message is indeed $\widetilde{g}(\Lambda, Z_{\mathrm{ma}} \oplus Z_{\mathrm{mp}})$.

2. Differential Privacy - The MP's output is differentially private with respect to the dataset Λ if at least one party is honest.

Verifiability is established by steps 1, 3 and 4 of C. Based on the analysis in Section 5.2, a witness satisfies step 1 of C if and only if $\Lambda_w = \Lambda$, i.e., the demand is reported honestly. Since the Pedersen commitment scheme is secure, it is computationally binding, meaning that it is computationally intractable for MP to find $Z'_{\mathrm{mp}}, r'_{\mathrm{mp}}$ with $Z_{\mathrm{mp}} \neq Z'_{\mathrm{mp}}$ and $\mathsf{Commit}(Z'_{\mathrm{mp}}, r'_{\mathrm{mp}}) = z_{\mathrm{mp}}$. So in order for the MP's witness to pass step 3 of C, it must have $Z_{\mathrm{mp},w} = Z_{\mathrm{mp}}$. Given steps 1, 3 have passed, step 4 ensures that the message z is indeed equal to $\widetilde{g}(\Lambda, Z_{\mathrm{ma}} \oplus Z_{\mathrm{mp}})$, which establishes verifiability.

To establish differential privacy, we need to show two things: (a) MA does not know $Z_{\mathrm{ma}} \oplus Z_{\mathrm{mp}}$ (see Remark 11) and (b) $Z_{\mathrm{ma}} \oplus Z_{\mathrm{mp}}$ is uniformly distributed over $\{0,1\}^v$, even if MA and MP are acting strategically. To this end, we consider a game between MA and MP with actions $Z_{\mathrm{ma}}, Z_{\mathrm{mp}} \in \{0,1\}^v$ and outcome $Z_{\mathrm{ma}} \oplus Z_{\mathrm{mp}} \in \{0,1\}^v$. We will show that the strategy profile where both $Z_{\mathrm{ma}}, Z_{\mathrm{mp}}$ are independently sampled uniformly at random is a Nash equilibrium, meaning that differential privacy is achieved as long as at least one party is honest.

To show that independent uniform random sampling of both $Z_{\mathrm{ma}}, Z_{\mathrm{mp}}$ is a Nash equilibrium, we first need to show that $Z_{\mathrm{ma}}, Z_{\mathrm{mp}}$ are independent. In the protocol Z_{mp} is sampled first, and a Pedersen commitment z_{mp} is sent to MA. Since Pedersen commitments are perfectly hiding, the

distribution of z_{mp} does not depend on Z_{mp}. So even if MA samples Z_{ma} based on the value of z_{mp}, the result will be independent of Z_{mp}. Now that we have established independence of Z_{mp}, Z_{ma}, we make use of the following observation.

Observation 3 (One Time Pad). *Suppose Z_{ma}, Z_{mp} are independent random variables. If Z_{ma} is uniformly distributed over $\{0,1\}^v$, then $Z_{ma} \oplus Z_{mp}$ is uniformly distributed over $\{0,1\}^v$, regardless of how Z_{mp} is sampled. If Z_{mp} is uniformly distributed over $\{0,1\}^v$, then $Z_{ma} \oplus Z_{mp}$ is uniformly distributed over $\{0,1\}^v$, regardless of how Z_{ma} is sampled.*

Observation 3 says that if Z_{ma}, Z_{mp} are independent, the distribution of $Z_{ma} \oplus Z_{mp}$ does not depend on Z_{mp} if Z_{ma} is uniformly random, and vice versa. Hence independent uniform sampling of Z_{ma}, Z_{mp} is a Nash equilibrium, establishing condition (b). To establish (a), if at least one party is honest, then we can assume without loss of generality that both parties are acting according to the Nash equilibrium. By observation 3, this means the marginal distribution of $Z_{ma} \oplus Z_{mp}$ and the conditional distribution of $Z_{ma} \oplus Z_{mp}$ given Z_{ma} are both uniform. In particular, MA does not learn anything about $Z_{ma} \oplus Z_{mp}$ from Z_{ma}.

Remark 12 (A note on Local Differential Privacy). Local Differential Privacy [KLN$^+$11] addresses the setting where the data collector is untrusted. Differential privacy is achieved by users adding noise to their data before sending it to the data collector. This is in contrast to the setting we study here where an untrusted data collector has the clean data of many users. We chose to study the latter model due to the way current mobility companies collect high resolution data on the trips they serve. Additionally, local differential privacy requires users to add noise to their data so they become statistically indistinguishable from one another. In the context of transportation, this means the noisy data of users will be statistically indistinguishable from one another, even if they have very different travel preferences. This level of noise significantly reduces the accuracy of any computation done on the data.

8   Conclusion

In this paper we presented an interactive protocol that enables a Municipal Authority to obtain insights from the data of Mobility Providers in a verifiable and privacy-preserving way. During the protocol, a Municipal Authority submits queries and a Mobility Provider computes responses based on its mobility data. The protocol is privacy-preserving in the sense that the Municipal Authority learns nothing about the dataset beyond the answer to its query. The protocol is verifiable in the sense that any deviation from the protocol's instructions by one party can be detected by the other. Verifiability is achieved by using cryptographic commitments and aggregated roadside measurements, and data privacy is achieved using zero knowledge proofs. The protocol is computationally efficient since cryptographic commitments, aggregated roadside measurements, and zero knowledge proofs all have efficient implementations. We showed that the protocol can be generalized to a setting with multiple Mobility Providers using secure multiparty computation. We presented a differentially private version of the protocol to address situations where the Municipal Authority has many queries.

There are several interesting and important directions for future work. First, while this work accounts for strategic behavior of the Municipal Authority and Mobility Providers, it assumes that drivers and customers will act honestly. A more general model which also accounts for potential

strategic behavior of drivers and customers would be of great value and interest. Second, while secure multiparty computation can be used to generalize the protocol to settings with multiple Mobility Providers, generic tools for secure multiparty computation introduce computational and communication overhead. Developing specialized multiparty computation tools for mobility-related queries is thus of significant practical interest. Finally, we suspect there are other applications for this protocol in transportation research beyond city planning and regulation enforcement that could be investigated.

References

[AGMS20]   Vittorio Astarita, Vincenzo Pasquale Giofrè, Giovanni Mirabelli, and Vittorio Solina. A review of blockchain-based systems in transportation. *Information*, 11, 2020.

[Blu82]   Manuel Blum. Coin flipping by telephone - A protocol for solving impossible problems. In *COMPCON'82, Digest of Papers, Twenty-Fourth IEEE Computer Society International Conference, San Francisco, California, USA, February 22-25, 1982*, pages 133–137. IEEE Computer Society, 1982.

[BS20]   Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. 2020.

[But14]   Vitalik Buterin. A next-generation smart contract and decentralized application platform, 2014.

[But16]   Vitalik Buterin. Zk rollup. Available Online at: `https://ethresear.ch/t/on-chain-scaling-to-potentially-500-tx-sec-through-mass-tx-validation/3477`, 2016.

[CGMA85]   Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 383–395. IEEE Computer Society, 1985.

[DMNS06]   Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[DN03]   Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In Frank Neven, Catriel Beeri, and Tova Milo, editors, *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, pages 202–210. ACM, 2003.

[GAC20]   David Gabay, Kemal Akkaya, and Mumin Cebe. Privacy-preserving authentication scheme for connected electric vehicles using blockchain and zero knowledge proofs. *IEEE Transactions on Vehicular Technology*, 69(6):5760–5772, 2020.

[GMR89]   Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GMW87]   Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987.

[GWC19]   Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptol. ePrint Arch.*, 2019:953, 2019.

[HRM10]   Jorge Hortelano, Juan Carlos Ruiz, and Pietro Manzoni. Evaluating the usefulness of watchdogs for intrusion detection in vanets. In *2010 IEEE International Conference on Communications Workshops*, pages 1–5, 2010.

[Isa17]   Mike Isaac. How uber deceives the authorities worldwide. *New York Times*, Mar 2017.

[KLN⁺11]   Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, 2011.

[LJA⁺18]   Andrei Lapets, Frederick Jansen, Kinan Dak Albab, Rawane Issa, Lucy Qin, Mayank Varia, and Azer Bestavros. Accessible privacy-preserving web-based data analysis for assessing and addressing economic inequalities. In Ellen W. Zegura, editor, *Proceedings of the 1st Conference on Computing and Sustainable Societies*, pages 48:1–48:5. ACM, 2018.

[LLW⁺18]   Zhaojun Lu, Wenchao Liu, Qian Wang, Gang Qu, and Zhenglin Liu. A privacy-preserving trust model based on blockchain for vanets. *IEEE Access*, 6:45655–45664, 2018.

[LMGN20]   Wanxin Li, Collin Meese, Hao Guo, and Mark Nejad. Blockchain-enabled identity verification for safe ridesharing leveraging zero-knowledge proof. *arXiv preprint arXiv:2010.14037*, 2020.

[LOL⁺20]   Yuhong Li, Kun Ouyang, Nanxuan Li, Rahim Rahmani, Haojun Yang, and Yiwei Pei. A blockchain-assisted intelligent transportation system promoting data services with privacy protection. *Sensors*, 20(9), 2020.

[MGLB00]   Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, 2000.

[MPJ⁺13]   Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 Internet Measurement Conference, IMC 2013, Barcelona, Spain, October 23-25, 2013*, pages 127–140. ACM, 2013.

[MZN⁺21]   Muhammad Baqer Mollah, Jun Zhao, Dusit Niyato, Yong Liang Guan, Chau Yuen, Sumei Sun, Kwok-Yan Lam, and Leong Hai Koh. Blockchain for the internet of vehicles towards intelligent transportation systems: A survey. *IEEE Internet of Things Journal*, 8(6):4157–4185, 2021.

[Nak08]   Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Available Online at: `http://bitcoin.org/bitcoin.pdf`, 2008.

[NBF⁺16]   Arvind Narayanan, Joseph Bonneau, Edward W. Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies - A Comprehensive Introduction*. Princeton University Press, 2016.

[PCA14]   PCAST. Big data and privacy: A technological perspective, 2014.

[Ped91]   Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[SCG$^+$14] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, 2014.

[Sha79]   Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Szy21]   Sean Szymkowski. Google removes app that calculated if uber drivers were underpaid. *RoadShow*, Feb 2021.

[vdHDLK19]   Rens Wouter van der Heijden, Stefan Dietzel, Tim Leinmüller, and Frank Kargl. Survey on misbehavior detection in cooperative intelligent transportation systems. *IEEE Commun. Surv. Tutorials*, 21(1):779–811, 2019.

A   Mobility Provider Serving Demand

For a given discretization of time $\mathcal{T} := \{0, \Delta t, 2\Delta t, ..., T\Delta t\}$, the demand $\Lambda \in \mathbb{R}^{n \times n \times T}$ can be represented as a 3-dimensional matrix (e.g., a 3-Tensor) where $\Lambda(i, j, t)$ represents the number of riders who request transit from i to j at time t. We use τ_{ij} to represent the time it takes to travel from i to j.

   To serve the demand from i to j, the MP chooses passenger carrying flows $x^{ij} \in \mathbb{R}_+^{mT}$ where $x_t^{ij}(u, v)$ is the number of passenger carrying trips from i to j that enter the road (u, v) at time t. Such vehicles will exit the road at time $t + \tau_{uv}$. There is also a rebalancing flow $r \in \mathbb{R}^{mT}$ which represents the movement of vacant vehicles that are re-positioning themselves to better align with future demand. Concretely, $r_t(u, v)$ is the number of vacant vehicles which enter road (u, v) at time t. The initial condition is $y \in \mathbb{R}_+^n$, where y_i denotes the number of vehicles at location i at time 0.

   The Mobility Provider's routing strategy is thus $x := \left(\left\{ x^{ij} \right\}_{(i,j) \in V \times V}, r \right)$ which satisfies the following multi-commodity network flow constraints:

$$\sum_{v:(v,u) \in E} \left(r_{t-\tau_{vu}}(v, u) + \sum_{(i,j) \in V \times V} x_{t-\tau_{vu}}^{ij}(v, u) \right) = \sum_{v:(u,v) \in E} \left(r_t(u, v) + \sum_{(i,j) \in V \times V} x_t^{ij}(u, v) \right) \quad (4)$$
$$\text{for all } (u, t) \in V \times [T]$$

$$\sum_{v:(u,v) \in E} x_t^{ij}(u, v) = \sum_{v:(v,u) \in E} x_{t-\tau_{vu}}^{ij}(v, u) \text{ for all } (i, j) \in V \times V, t \in [T], u \notin \{i, j\} \quad (5)$$

$$\sum_{\tau=0}^{t} \left(\sum_{v:(i,v)\in E} x_\tau^{ij}(i,v) - \sum_{v:(v,i)\in E} x_{\tau-\tau_{vi}}^{ij}(v,i) \right) \leq \sum_{\tau=0}^{t} \Lambda(i,j,\tau) \text{ for all } (i,j,t) \in V \times V \times [T] \quad (6)$$

$$x_t^{ij}(j,v) = 0 \text{ for all } (i,j) \in V \times V, t \in [T], (j,v) \in E. \quad (7)$$

$$\sum_{j:(i,j)\in E} x_0^{ij} = y_i \text{ for all } i \in V. \quad (8)$$

Here (4) represents conservation of vehicles, (5),(6),(7) enforce pickup and dropoff constraints according to the demand Λ, and (8) enforces initial conditions.

The utility received by the Mobility provider (e.g., total revenue) from implementing flow x for a given demand Λ is $J_{\mathrm{MP}}(x;\Lambda)$. An optimal routing algorithm for demand Λ is a solution to the following optimization problem.

$$\underset{x}{\mathrm{maximize}} \ J_{\mathrm{MP}}(x;\Lambda)$$
$$\text{s.t. } (4),(5),(6),(7),(8).$$

B   Mobility Provider Serving Demand (Steady State)

In a steady state model, the demand can be represented as $\Lambda \in \mathbb{R}_+^{n\times n}$, a matrix where $\Lambda(i,j)$ represents the rate at which riders request transit from node i to node j.

For each origin-destination pair $(i,j) \in V \times V$, the MP serves the demand $\Lambda(i,j)$ by choosing a passenger carrying flow $x_p^{ij} \in \mathbb{R}_+^m$ and a rebalancing flow $x_r \in \mathbb{R}_+^m$ so that $x := \left(\left\{ x_p^{ij} \right\}_{(i,j)\in V\times V}, x_r \right)$ satisfy the multi-commodity network flow constraints with demand Λ:

$$\sum_{v:(u,v)\in E} \left(x_r(u,v) + \sum_{(i,j)\in V\times V} x_p^{ij}(u,v) \right) = \sum_{v:(v,u)\in E} \left(x_r(v,u) + \sum_{(v,u)\in V\times V} x_p^{ij}(v,u) \right) \text{ for all } u \in V$$
$$(9)$$

$$\Lambda(i,j)\mathbb{1}_{[u=j]} + \sum_{v:(u,v)\in E} x_p^{ij}(u,v) = \Lambda(i,j)\mathbb{1}_{[u=i]} + \sum_{v:(v,u)\in E} x_r(v,u) + x_p^{ij}(v,u) \text{ for all } (i,j) \in V \times V, u \in V.$$
$$(10)$$

Here (9) represents conservation of flow and (10) enforces pickup and dropoff constraints according to the demand Λ.

The utility received by the Mobility provider (e.g., total revenue) from implementing flow x is $J_{\mathrm{MP}}(x)$. Therefore the Mobility Provider will choose x according to the following program.

$$\underset{x}{\mathrm{maximize}} \ J_{\mathrm{MP}}(x)$$
$$\text{s.t. } (9),(10)$$

C   Implementation Details and Examples

In this section, we show how driver period information in ridehailing services, and a mobility provider's impact on congestion can be obtained from the protocol. Both cases involve specifying characteristics of the query function g and trip metadata that enable the desired information to be computed by the protocol.

C.1   Obtaining ridehailing period activity

As discussed in Example 3, the pay rate of ridehailing drivers depends on the period they are in. Ridehailing companies use period 2 to tell users that they are matched and a driver is en route, thereby reducing the likelihood that the user leaves the system out of impatience. Due to this utility, period 2 has a higher pay rate than period 1. There is thus a financial incentive for ridehailing companies to report period 2 activity as period 1 activity so that they can have improved user retention while keeping operations costs low. Accurate period information is thus important to protect the wages of ridehailing drivers.

   We achieve accurate period information by including digital signatures in the trip metadata. Recall that the trip metadata includes the request time, match time, pickup time, and dropoff time of the request. The period 2 and period 3 activity associated with a trip can be deduced from these timestamps, as shown in Figure 7. Furthermore, Rider Witness and ARA ensure that reporting the true demand is a dominant strategy for the ridehailing operator.
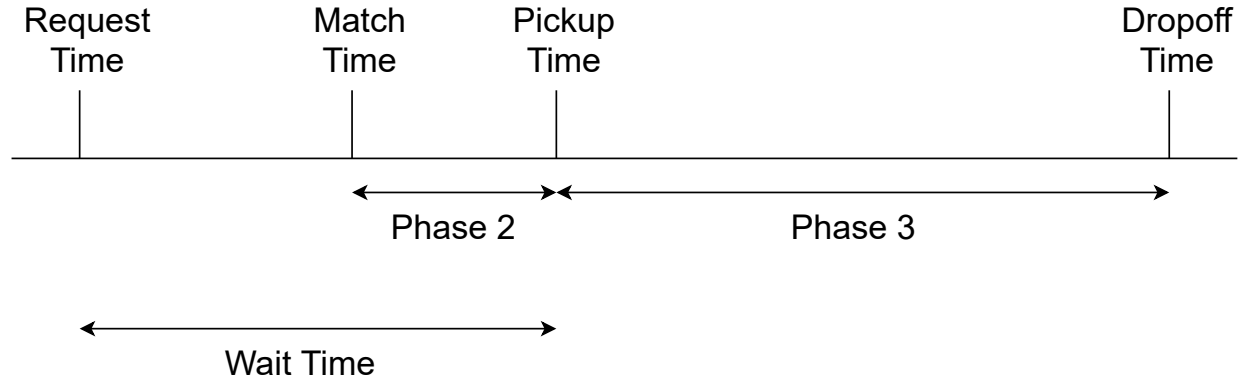

Figure 7: The timesteps within the trip metadata determine the Period 2 and Period 3 activity of the vehicle that serves this trip.

   Therefore to ensure accurate period information, it is sufficient to ensure that the aforementioned timesteps are recorded correctly. For period 2 accuracy, we need to ensure that the match time and pickup time are recorded properly for each trip. To do this, we will use digital signatures. To notify a user that they have been matched, the ridehailing operator will send (m_{pt}, σ_{pt}), where:

$$m_{pt} = \text{You have been matched to vehicle } \texttt{vehID} \text{ at time } \texttt{currtime},$$
$$\sigma_{pt} = \mathsf{sign}(\mathsf{sk}_{\mathrm{mp}}, m_{pt}).$$

The user will only consider the message m_{pt} as genuine if it is accompanied by a valid signature σ_{pt}. Therefore, telling a user they are matched (and thus reducing the likelihood that this user cancels

their trip) requires the ridehailing company to provide an irrefutable and unforgeable declaration of the match time in the form of (m_{pt}, σ_{pt}). The message and signature (m_{pt}, σ_{pt}) is then included in the trip metadata to certify the trip's match time. The same can be done for the pickup time, and as a result, ensure accurate reporting of all period 2 activity.

The accuracy of period 3 activity can be ensured by ensuring that pickup time and dropoff time are recorded correctly.

To implement driver wage inspection through the protocol, the query function g would be

$$g_{\text{wage}}(\Lambda) := \prod_{\lambda \in \Lambda} \mathbb{1}[w(\lambda) = f_{\text{wage}}(\lambda)],$$

where $w(\lambda)$ is the driver wage of ride λ, and f_{wage} is the MP's wage formula which may depend on the period and trajectory information contained in the trip metadata of λ. Note that $g_{\text{wage}}(\Lambda) = 1$ if and only if all drivers were paid properly, and is 0 otherwise.

Remark 13 (Evaluating Waiting Time Equity). Using the idea from Section C.1, one can also evaluate the equity of waiting times throughout the network. It is clear from Figure 7 that the wait time can be determined by the request time and pickup time, both of which can be found in the trip metadata. The trip metadata also includes the pickup location and dropoff location, so the average wait time as a function of pickup location, dropoff location, both pickup and dropoff locations, can all be computed from the trip metadata.

To implement a waiting time evaluation through the protocol, the Municipal Authority would specify a fairness threshold τ. The query function g is then designed to output 1 if and only if the average waiting time across locations does not vary by more than the pre-specified threshold, and outputs 0 otherwise. Concretely, if we want to enforce wait time equity across pickup regions, we could do this with the function

$$g_{\text{wait}}(\Lambda) = \prod_{i,j \in V} \mathbb{1}\left[|\tau_i - \tau_j| \leq \tau\right]$$

where τ_i is the average wait time for requests in region i.

C.2 Evaluating contributions to congestion

The trip metadata contains the trip trajectory which can be used to evaluate a ridehailing fleet's contribution to congestion. The trip trajectory provides the location of the service vehicle as a function of time, which provides two important insights. First, the trip trajectories can be used to determine how many ridehailing vehicles are on a particular road at any given time. Second, from a trajectory one can compute the amount of time the vehicle spends on each road within the trip path. Thus the average travel time for a road can be calculated, which can then be used to estimate the total traffic flow on the road using traffic models. Combining these two pieces of information, the fraction of a road's total traffic that is ridehailing vehicles can be computed from the trip metadata.

D   Necessity of Assumption 1 for Verifiability

In this section we show that Assumption 1 is necessary for verifiable queries on mobility data under the natural assumption that MA does not have surveillance in the interior of MP vehicles. This assumption on limited surveillance ability of MA is formalized in Assumption 2.

Assumption 2. *There does not exist a practical way for MA to determine whether a MP vehicle is carrying a customer or not, without directly tracking all customers. In particular, MA cannot determine the period information of MP vehicles.*

Note that MA can obtain phase information from the drivers or from MP, but in the absence of Assumption 1, drivers and MPs may act strategically, and may not be trustworthy.

The following result shows that under Assumption 2, if the drivers or riders are willing to collude with the MP, then the MP can misreport properties of its mobility demand in a way that is undetectable by the MA.

Observation 4 (Necessity of Assumption 1 for Strategyproofness). *Under Assumption 2, the following events are undetectable by MA, even if MA can track all MP vehicles (i.e., knows the location of each MP vehicle at any time):*

1. *If drivers collude with MP, then MP can overreport demand.*

2. *If riders and drivers collude with MP, then MP can underreport demand, or misreport attributes of the demand.*

Proof of Observation 4. By Assumption 2, MA cannot distinguish between MP vehicles in period 1 and MP vehicles in period 2 or 3. Suppose the drivers are willing to collude with MP. If MP wants to overreport demand from some origin i to some destination j, it can have some drivers drive from i to j without a passenger. This will lead to period 1 traffic from i to j, however the drivers will report themselves in period 3 to MA. This way, even if MA is able to track the MP vehicles, the reported period information from the drivers will be consistent with the demand report from MP.

Now suppose both riders and drivers are willing to collude with MP. If the MP wants to underreport demand from i to j, they can have some drivers who are serving passengers from i to j report themselves in period 1 to MA. □

So in the absence of Assumption 1, the MA has no way of checking whether the messages it receives from MP are computed from the true demand.

Remark 14 (Tracking Users is also insufficient). Even if the MA is able to track users and thus determine whether a MP vehicle has a passenger, this still does not prevent overreporting of demand. In this case, MP can hire people to hail rides from specific trips if it wants to overreport demand.

E   Roadside Audits with fewer sensors

In this section we present the Randomized Roadside Audits (RRA) mechanism. Like ARA, the RRA detects overreporting of demand by conducting road audits. Where ARA places sensors on every road, RRA places sensors on a small subset of randomly selected roads, enabling it to use fewer sensors.

The sensors used in RRA are similar to the sensors used in ARA described in Section 5.2.3, with the following differences:

1. Each sensor has its own pair of public and secret keys $(\mathsf{pk}_s, \mathsf{sk}_s)$ for digital signatures. Everyone knows pk_s, but sk_s is contained in a Hardware Security Module within the sensor so that it is impossible to extract sk_s from the sensor, but it is still possible to sign messages using sk_s.

2. Each sensor now records its own location using GPS.

First, the MA and MP agree on a list of public keys belonging to the sensors. In particular, they must agree on the number of sensors being deployed in the network. Let mp be the number of sensors being deployed (recall that m is the number of roads in the network). We will focus on the case where $p \in (0, 1)$. If $p > 1$, then there are enough sensors to implement ARA.

During the data-collection period, the MA will place the sensors inside vehicles which are driven by its employees. We assume that MP cannot determine which vehicles are carrying sensors. In practice, MA can have much more than mp employees driving around in the network, but only mp of them will have sensors.

The data collection period is divided up into many rounds (e.g., a round could be 1 hour long). In each round, MA will sample a random set of mp roads. Each vehicle with a sensor is assigned to one of these roads, where they will stay (i.e., parked on the side of the street) to measure the MP traffic that pass by them. The sensor will record a measurement u which specifies the time, the period and the location of every MP vehicle that passes by. It will then sign the message with its secret key via $\sigma_u := \mathsf{sign}(\mathsf{sk}_s, u)$. In particular, a sensor assigned to road $e \in E$ in round t will be able to determine $\varphi_t(e, \Lambda)$, which is the total number of period 2 or period 3 MP vehicles that traverse e in round t, formally described below.

$$\varphi_t(e, \Lambda) := \sum_{\lambda \in \Lambda} \mathbb{1}_{[\lambda \text{ traverses } e \text{ in round } t]}.$$

As was the case in ARA, the sensors have a communication constraint that prevents them from transmitting their data unless both MA and MP give permission. Therefore during the data collection period, MP does not know where the sensors are. Once the data collection period is over, both MA and MP give permission to collect the data from the sensors.

Definition 15 (RRA Test). The RRA test checks whether the road usage on sampled roads is consistent with the demand reported by MP. Concretely, a witness $w = (\Lambda_w, r_w, c_w)$ passes the RRA test if $\varphi_t(e, \Lambda) = \varphi_t(e, \Lambda_w)$ for all pairs (e, t) such that e was sampled in round t.

Observation 5 (Efficacy of Random Roadside Audits). *Under Assumption 1, if MP submits a commitment $\sigma' = \mathsf{MCommit}(\Lambda', r)$ to a strict superset of the demand, i.e., $\Lambda \subset \Lambda'$, then with probability at least p, any proof submitted by MP will either be inconsistent with σ' or will fail the RRA test. Hence overreporting of demand will be detected with positive probability.*

Proof of Observation 5. We use a similar analysis to ARA. Suppose MP overreports the demand, i.e., submits a commitment $\sigma' = \mathsf{MCommit}(\Lambda', r)$ where Λ' is a strict superset of Λ. Then there exists $\lambda' \in \Lambda' \setminus \Lambda$. Let e' be any road in the trip trajectory of λ'. and $t(\lambda', e')$ be the round in which trip λ' traverses e'. We then have $\varphi_{t(\lambda', e')}(e', \Lambda) < \varphi_{t(\lambda', e')}(e', \Lambda')$. If e' is audited in round $t(\lambda', e')$, then σ' will be inconsistent with the roadside audit measurements, and will fail the RRA

test. Since MA samples mp roads to audit uniformly randomly in each round, and there are a total of m roads, the probability that e' is chosen in round $t(\lambda', e')$ is p. Since overreporting is detected only probabilistically, in the event that it is detected, MA should fine MP so that MP's expected utility is reduced if it overreports demand.

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\square$

Remark 15 (Comparing RRA to ARA). When compared to ARA, RRA uses fewer sensors. This, however, is not without drawbacks, since RRA detects demand overreporting only probabilistically. Thus in RRA the MA needs to fine the MP in the event that demand overreporting is detected. In particular, the fine should be chosen so that the MP's expected utility is decreased if it decides to overreport demand. Concretely, suppose U_h, U_d are the utilities received by MP when acting honestly and dishonestly respectively. Since dishonesty is detected with probability p, the fine F must satisfy

$$U_h > (1-p)U_d - pF \implies F > \frac{1}{p}(U_d - U_h) - U_d.$$

If MA is using very few sensors or if U_d is much larger than U_h, then F needs to be very large. A large fine, however can be difficult to implement. Recall from Section 5.2.2 that inconsistencies between demand metadata and roadside measurements due to GPS errors can occur even if all parties are honest. If such errors occur, then an MP would incur a large fine even if it behaves honestly. For this reason, even an honest MP may not want to participate in the protocol. One could use an error tolerant version of RRA, but for large F the tolerance parameter ϵ would need to be large, enabling a dishonest MP to overreport demand while remaining within the tolerance parameter.

E.1   Security Discussion

We now make several remarks regarding the two sensor modifications we made for RRA. First, the signatures generated by the sensors' secret keys ensure that MA cannot fabricate or otherwise tamper with the sensor's data. This is important because the sensors are in the possession of the MA and its employees. Even if the MA manages to change the data in the sensor's storage, it cannot produce the corresponding signatures for the altered data since it does not know the secret key, which is protected by a Hardware Security Module.

Second, the sensor's location data is essential to prevent MA from conducting relay attacks. A relay attack is as follows: Suppose Alice and Bob are both MA employees. Alice has a sensor in her car. Bob does not have a sensor in his car, but he wants to collect data as if he had a sensor in his car. The MA can give Bob an unofficial sensor (this sensor does not have a valid public and secret key recognized by the MP), allowing Bob to detect signals from MP vehicles. Since Bob's sensor does not have an official secret key, he cannot obtain a valid signature for his measurements. To get the signatures, Bob sends the detected signal to Alice, and Alice relays the signal to her sensor, which will sign the measurement and record it. In this manner, the MA is able to get official measurements and signatures on Bob's road even though he does not have an official sensor. Fortunately, this attack is thwarted if the sensor knows its own location. If a sensor receives a measurement whose location is very different than its location, then it will reject the message, thus thwarting the relay attack.[13]

---

[13]The measurements can be protected by authenticated encryption so that relayers (e.g., Bob) cannot modify the

F   More Details on Congestion Pricing

When the travel cost is the same as travel time, the prices can be obtained from the following optimization problem

$$
\max \sum_{e \in E} x_e f_e(x_e)
$$
$$
\text{s.t. } x \in S_\Lambda
$$

where S_Λ is the set of flows that satisfy Multi-Commodity Network flow constraints for the transit network G and demand Λ. Here the objective measures the sum of the travel times of all requests in Λ.

Let x^* be a solution to (1). By first order optimality conditions[14] of x^*, for any origin-destination pair (i, j), and any two paths p_1, p_2 from i to j with non-zero flow, we have

$$
\sum_{e \in p_1} \frac{\partial}{\partial x_e} x_e f_e(x_e) \bigg|_{x_e = x_e^*} = \sum_{e' \in p_2} \frac{\partial}{\partial x_{e'}} x_{e'} f_{e'}(x_{e'}) \bigg|_{x_{e'} = x_{e'}^*}
$$
$$
\implies \sum_{e \in p_1} f_e(x_e^*) + x_e^* f_e'(x_e^*) = \sum_{e' \in p_2} f_{e'}(x_{e'}^*) + x_{e'}^* f_{e'}'(x_{e'}^*). \tag{11}
$$

In order to realize x^* as a user equilibrium, the costs of p_1, p_2 should be the same so that no user has an incentive to change their strategy. This can be achieved by setting the toll for each road e as $p_e := x_e^* f_e'(x_e^*)$. By doing so, from (11) we can see that the cost (travel time plus toll) for the two paths will be equal.

In the context of PMM, the function g associated with congestion pricing is

$$
g_{\mathrm{cp}}(\Lambda) := \left\{ x_e^* f_e'(x_e^*) \right\}_{e \in E} \text{ where } x^* \text{ solves (1).}
$$

G   Efficacy of Merkle Proofs

To verify the proof $\pi(m_i) = \{r_i\} \cup \left\{ \text{sibling}(h_{j, x_j}) \right\}_{j=1}^{\ell_q}$ for membership of m_i, the recipient of the proof would compute $v_1, v_2, ..., v_{\ell_q - 1}$ recursively via:

$$
v_1 := H(r_i \| m_i)
$$
$$
v_j := \begin{cases} H(v_{j-1} \| \text{sibling}(h_{j-1, x_{j-1}})) & \text{if } x_{j-1} \text{ is even,} \\ H(\text{sibling}(h_{j-1, x_{j-1}}) \| v_{j-1}) & \text{if } x_{j-1} \text{ is odd,} \end{cases} \text{ for } 1 \le j < \ell_q.
$$

By the construction of the Merkle tree, $v_j = h_{j, x_j}$, and so in particular the Merkle Proof is valid if and only if v_{ℓ_q} is equal to the root, i.e., $v_{\ell_q} = h_{\ell_q, 0}$.

Since there are q leaves in the binary tree, p_i has at most $\log_2 q$ vertices in it, and each hash is d bits, so the length of π is at most $d \log_2 q$.

By collision resistance of H, it is intractable to forge a proof if m_i is not in the tree, and since hiding nonces are used when hashing the items, the proof reveals nothing about the other items in the tree.

---

messages (i.e., changing the vehicle position part of the measurement)

[14]i.e., it should be impossible to decrease the objective function by reallocating flow from p_1 to p_2 or vice versa.