Privacy in Urban Sensing with Instrumented Fleets, Using Air Pollution Monitoring As A Usecase

Ismi Abidi
Indian Institute of Technology Delhi
ismi.abidi@cse.iitd.ac.in

Ishan Nangia
MPI-SWS
ishannangia.123@gmail.com

Paarijaat Aditya
Nokia Bell Labs
paarijaat.aditya@nokia-bell-labs.com

Rijurekha Sen
Indian Institute of Technology Delhi
riju@cse.iitd.ac.in

***Abstract*—Companies providing services like cab sharing, e-commerce logistics and food delivery are willing to instrument their vehicles for scaling up measurements of traffic congestion, travel time, road surface quality, air quality, etc. [1]. Analyzing fine-grained sensors data from such large fleets can be highly beneficial; however, this sensor information reveals the locations and the number of vehicles in the deployed fleet. This sensitive data is of high business value to rival companies in the same business domain, e.g., Uber vs. Ola, Uber vs. Lyft in cab sharing, or Amazon vs. Alibaba in the e-commerce domain. This paper provides privacy guarantees for the scenario mentioned above using Gaussian Process Regression (GPR) based interpolation, Differential Privacy (DP), and Secure two-party computations (2PC). The sensed values from instrumented vehicle fleets are made available preserving fleet and client privacy, along with client utility. Our system has efficient latency and bandwidth overheads, even for resource-constrained mobile clients. To demonstrate our end-to-end system, we build a sample Android application that gives the least polluted route alternatives given a source-destination pair in a privacy preserved manner.**

I. Introduction

Urban sustainability problems like traffic congestion, poor road surface quality, life-threatening levels of air pollution need scalable measurements for appropriate policy design and daily decision making such as which route to take to minimize congestion delays or pollution exposure. Vehicle-mounted sensing holds great promise to scale up urban measurements. Figure 1 shows an example of the spatial coverage by three public bus routes in New Delhi, India, where we instrument the buses with air pollution monitoring sensors. The government deployed static pollution sensors, marked as landmark icons in the same figure, show significantly lower spatial coverage than bus-mounted sensors. Increasing bus count along some routes further increases the temporal granularity of sensed values, as each bus remains active for 16-18 hours per day. While public buses cover the main arterial roads in a city, other vehicle fleets like cab sharing services, food, and essential goods delivery services cover a much wider area. Urban sensing, therefore, can see a dramatic shift leveraging thousands of mobile probes in cities, in the form of cab fleets [2], [3], e-commerce, or food delivery vehicles. Fleet owners are themselves realizing this potential of scalable urban sensing. They are willing to instrument their vehicles for applications like environmental monitoring [1], primarily as part of their corporate social responsibility, to aid more data-driven environmental policy.

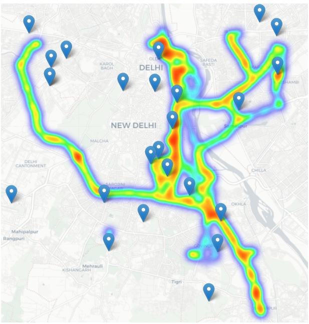


Fig. 1: Coverage with 3 routes where we instrument buses with air pollution sensors in Delhi. Government deployed static pollution sensors are shown with landmark icons.

The recent trend of urban sensing with instrumented fleets has given rise to a novel privacy problem. During discussions with cab-sharing companies to extend our bus-based pollution sensing in Delhi, fleet owners have expressed concerns about revealing their fleet size and locations. It is somewhat different from privacy problems solved in recent years. Existing literature on location privacy focuses on protecting the location of individual mobile users [4], [5], [6], [7] and not on fleet companies' privacy.

Figure 1 shows the exact trajectories of our instrumented buses. Such instrumented vehicle trajectories can become publicly available for any company which participates in urban sensing and publishes collected sensor data. Publishing sensor data tagged with time and location of data collection violates privacy. Rival companies, e.g., Lyft vs. Uber, can use the spatial distribution information of the instrumented fleet for better placement and routing of their competing fleets. Given the high competition among rival companies in a city [8], [9], these privacy concerns about losing business advantage are not overly pessimistic but pragmatic. On the other hand,

Network and Distributed Systems Security (NDSS) Symposium 2022
24-28 April 2022, San Diego, CA, USA
ISBN 1-891562-74-6
https://dx.doi.org/10.14722/ndss.2022.23127
www.ndss-symposium.org

sensor data clients termed as clients henceforth[1] also want to preserve the location privacy for their sensor data query. For example, ordinary citizens query pollution information for their residential, office locations, or their daily trajectories. They might not feel comfortable sharing these details with arbitrary pollution data providers. Environmental agencies, for example, often also have to query pollution values of regions being considered for 'unpopular' policy enforcement, such as shutting down a particular industry or disallowing crop burning. Such decisions are highly contested, often leading to long-drawn court cases. Thus, these agencies have a strong motivation to keep their query locations private, to conduct spatio-temporal monitoring of pollution sources without undue pressure. This motivates us to keep client locations private.

This paper presents an end-to-end system for preserving the privacy of both fleet owners and data clients while maintaining the utility of the sensed data. Unlike [10], [11], we don't release raw trajectory data. We interpolate the sensed data at periodic intervals. Interpolation computes missing values at uniformly sampled grid points in the city. This step hides locations where vehicles are missing. Interpolation is necessary for fleet privacy to hide where vehicles are present vs. absent and for the improved utility to give a reasonable response at locations without vehicles. However, we show in Section V, there can be privacy leaks from the interpolation results. The location and density of vehicles in the sensing fleet can be inferred even from uniformly sampled grid points. We then explore Differential Privacy (DP) [12] to publicly release the interpolated maps with DP guarantee. It allows full privacy to clients who can ask any query, with minimal overhead, on the public maps without revealing query locations. However, the DP scheme gives questionable DP privacy to fleet owners due to inherent correlations in pollution values among nearby areas. Since DP adds noise to interpolated output for privacy, it reduces the utility of the sensed and interpolated values for clients. We show this with real air pollution datasets from Beijing, Zurich, and Delhi. The two metrics of fleet privacy and sensor data utility have an inherent trade-off – more noise added to improve fleet privacy degrades utility and vice versa. We discuss the DP scheme and its limitations in balancing the privacy-utility trade-off in Section VI.

Our final end-to-end system answers client queries on interpolated maps, privately held by a fleet owner, through secure 2-party computation (2PC) [13] mechanisms. The client obfuscates query location within a larger geographical area to preserve its privacy. In order to maintain fleet privacy and prevent fine-grained fleet information leakage per grid cells, the fleet server only answers queries that have been aggregated over more than a threshold number of grid cells. We give ϵ-DP guarantee to the aggregated response by adding noise. This DP further protects fleet privacy against smartly crafted series of 2PC queries intersecting at one grid location, whose privacy can be compromised otherwise. Since information release is changed from per grid fine-grained values to coarse-grained server's aggregation threshold number of grids, the DP noise needed for the aggregate answers is less than the DP noise needed per grid. Hence, data utility is maintained at the server's grid aggregation level. We describe this 2PC based query-response system termed Prac2PC, in Sections VII and IX.

The main contributions of this paper are as follows:

- We define a real-life privacy-vs-utility trade-off problem for air pollution monitoring based on anecdotal discussions. Servers and clients each have critical privacy requirements. The utility of reporting accurate pollution values and associated measurement errors is also necessary, as the whole effort of scalable pollution measurement using instrumented vehicles will be pointless otherwise. Based on our real-world deployment experience for pollution measurement, we use air pollution monitoring as a use case in our paper. However, any crowd-sourced or fleet-based system for scaling measurements (e.g., road surface quality monitoring for potholes, noise pollution, traffic congestion information, and cellular measurements for 4G signal coverage in urban areas) can use our system to preserve fleet and client privacy.
- Through careful empirical analyses, we show that the privacy requirements of server as well as client, and client utility, cannot all be simultaneously maximized in presence of inherent trade-offs. The trade-offs are evaluated using various real datasets and careful experimental designs.
- Besides the extensive micro-benchmarks to quantify privacy-vs-utility trade-off for each mechanism, we also build an end-to-end system. It demonstrates the practical impact Prac2PC can have, by showing how the system can support *pollution-aware route planning* queries in a university campus, in a privacy preserving manner, at acceptable latency, bandwidth and energy overheads.

II. Background

In this section, we briefly describe the key techniques used in our system: namely Gaussian Process Regression, Differential Privacy, and Secure two-party computation.

Gaussian Process Regression(GPR) is a non-parametric Bayesian regression approach that uses sampled data (training) to provide interpolation estimates. A Gaussian Process is a stochastic process which can be described by its mean $\mu(x)$ and covariance function $k(x, x')$ [14]. For a known training input-output pair (X, y), GPR predicts the values of the underlying latent (unobserved) function, f_*. This generates the test output, y_*, on the set of input test points, X_*. Considering Gaussian noise with zero mean and σ_n^2 variance, the posterior distribution for interpolating attribute is given by:

$$f_*|X, y, X_* \sim \mathcal{N}(\mu, \Sigma) \quad (1)$$

where,

$$\mu = k(X_*, X)(k(X, X) + \sigma_n^2 I)^{-1} y \quad (2)$$

$$\Sigma = k(X_*, X_*) - k(X_*, X)(k(X, X) + \sigma_n^2 I)^{-1} k(X, X_*) \quad (3)$$

Σ_{ii} gives the posterior variance values of test points

Differential Privacy (DP) is a standard privacy technique for statistical analysis with provable guarantees. According to [12], an algorithm $\mathcal{M}$ is ϵ-differential private if, for all

[1] not to be confused with regular application clients of the fleet companies, like people ordering food and other commodities or hailing shared cabs

databases, D and D', which differ on a single row and for each query output m,

$$P(\mathcal{M}(D) \in m) \leq e^{\epsilon} P(\mathcal{M}(D') \in m) + \delta$$

where, ϵ is known as privacy budget. The smaller the value of ϵ, the more is the privacy. The above equation implies that the output of the query should not change much if we change a single row of the database. If $\delta = 0$, then the algorithm $\mathcal{M}$ is ϵ-differential private. Laplace mechanism is the most commonly used mechanism that satisfies ϵ-differential privacy in which a random noise drawn from the Laplace distribution is added to the output of the query function. By the definition in [12], the Laplace mechanism satisfies ϵ-DP if:

$$\tilde{f}(D) = f(D) + Lap(\Delta f/\epsilon)$$

where D is the database over which the function f is defined, ϵ is known as privacy budget and Δf is the sensitivity of a function f. The global sensitivity of a function f can be computed as:

$$\Delta f = \max_{D \simeq D'} |f(D') - f(D)|$$

where D and D' differs in atmost one record.

Secure Two-Party Computation (2PC) is a special case of Secure Multiparty Computation (SMC) that allows two parties to compute a function on their private inputs jointly. The function output is revealed to either or both parties. *Yao's Garbled Circuit (GC)* is a 2PC protocol where a server (garbler or creator) generates a Boolean circuit corresponding to any desired function. For each gate, it garbles the truth table using symmetric encryption (mainly, AES). The garbler then sends the circuit, the garbled truth tables and its encrypted input to the client (evaluator). The evaluator is then engaged in oblivious transfer (OT) [15], [16] to get its garbled inputs from the garbler. Once the evaluator receives both its and server's inputs, it will compute the function and get the decrypted output. In recent years, several enhancements have been done in GC to make it faster such as Free-XOR [17], Halfgates [18], pipelining [19] and maliciously secured [20], [21].

Mixed protocol (MP) is a relatively new circuit-based 2PC protocol that combines GC with either Homomorphic Encryption (HE) [22], [23] or Secret Sharing schemes [24], [25]. This combination supposedly allows more efficient computations of particular functions. In this, each party has its own input share of a secret defined over a field. The parties can recover the secret only when they combine their shares. Secret shares can be Arithmetic or Boolean. To hide intermediate results, it utilizes OT or Beaver multiplication triples for Boolean share and Arithmetic share, respectively.

III. Related work

In preserving privacy of raw location data (trajectories), several works add noise to location coordinates using DP [10], [11], [26], [27], [28], or add dummy trajectories to actual trajectories [29], [30], or use chaff vehicles for variable traffic density [31]. Our system also preserves the privacy of fleet trajectories while answering queries on pollution or some other sensor value for a city. Instead of adding noise to trajectories using DP, we add it to our query outputs. It will allow trajectory information to remain clean for any further processing. In [32], the user locations are obfuscated before sending the data to the server using an obfuscation matrix, learning which has a high overhead. It applies compression sensing to solve data sparsity for temperature sensors. In our work, we employ GPR for interpolating sensed raw trajectory data to alleviate data sparsity and provide basic privacy of hiding locations without vehicles.

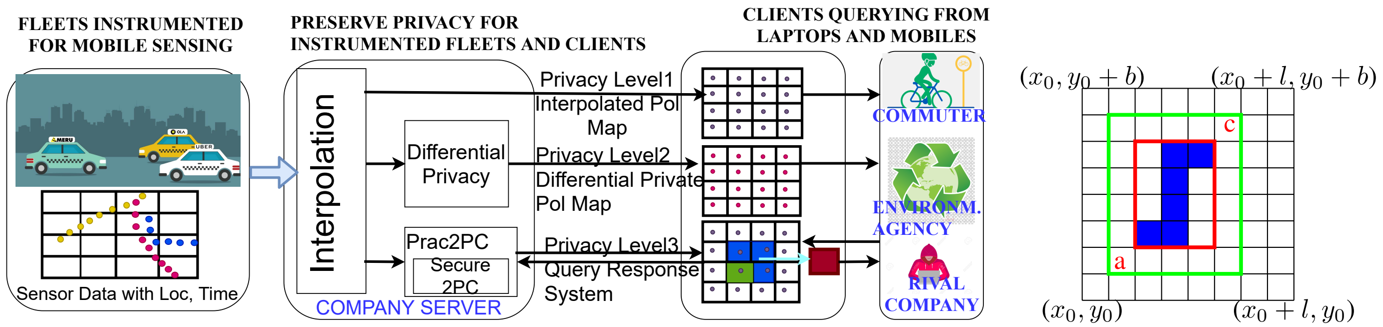
In addition to DP, Secure MultiParty Computation (SMC) has been extensively used to preserve privacy, specifically location privacy, viz. detecting proximate users without revealing individual locations [4], [33], detecting users with overlapping trajectories without revealing individual trajectories in ride sharing [5], [34], [35]. These works have only location sensor values and do not have utility concerns (e.g., pollution sensor values). Hence the problem formulations and solutions for Prac2PC are disjoint. Works like [19], [36], [37], [38], [39] specifically explore SMC for mobile applications with different optimizations like an extra server for outsourcing garbled operations or using a dedicated compiler. We also thoroughly evaluate the latency and bandwidth requirements of Prac2PC with real datasets and laptops and smartphone clients through careful implementations and system optimizations.

Our system is related to the wide literature of Mobile Crowd Sensing (MCS), where typically mobile clients are data providers requiring privacy [6], [7], [40]. Servers, aggregating the data, act as data consumers, requiring little to no privacy. In our system, the roles of data consumers and providers are reversed, and both the server and client have a set of privacy requirements. Kong et al. [41] provides location privacy in a sensing environment to both server and client for an average query using Homomorphic Encryption (HE) under the semi-honest assumption. Their solution requires a trusted third party and also reveals the fleet size. HE will only be able to support a limited subset like sum/average computations and is thus not practical for our queries. In comparison to [41], we support a wide variety of queries in this paper (Table I), and successfully hide both fleet size and location under semi-honest as well as malicious threat models.

We first use GPR to hide locations without vehicles. The GPR output still leaks fleet density. We then use 2PC to answer queries with GPR output at low latency and bandwidth for mobile clients. It preserves client privacy and fleet privacy with reasonable data utility. We additionally use DP on the 2PC results to protect against smartly crafted queries. We empirically motivate 2PC, showing only DP after GPR may not balance the fleet privacy vs. data utility trade-off. GPR, DP, and 2PC are all well-known methods. The novelty of this paper lies in applying them in an appropriate combination for an important privacy and sustainability problem, with empirical utility and performance trade-off analyses on real-world air pollution and trajectory datasets.

IV. System Architecture

Figure 2(a) shows the architecture for the system satisfying the privacy and utility requirements of the server and the client. The vehicle fleet of a company is instrumented for data collection. Collected data is sent to the private server of the company. For the sample use case of air pollution measurement, a server can either release public pollution maps or respond to different client queries on its private data (Prac2PC). When clients use public pollution maps, they


FLEETS INSTRUMENTED FOR MOBILE SENSING
Sensor Data with Loc, Time
PRESERVE PRIVACY FOR INSTRUMENTED FLEETS AND CLIENTS
Interpolation
Differential Privacy
Prac2PC
Secure 2PC
COMPANY SERVER
Privacy Level1 Interpolated Pol Map
Privacy Level2 Differential Private Pol Map
Privacy Level3 Query Response System
CLIENTS QUERYING FROM LAPTOPS AND MOBILES
COMMUTER
ENVIRONM. AGENCY
RIVAL COMPANY
$(x_0, y_0 + b)$
$(x_0 + l, y_0 + b)$
c
a
(x_0, y_0)
$(x_0 + l, y_0)$

(a) (b)

Fig. 2: a) System architecture preserving privacy for server and client with different privacy, utility, and overhead. b) City $\mathcal{A}$ divided into squares with blue squares representing the client's area of interest and red/green rectangles representing the client's preferred obfuscation levels.

Query function	Client Input	Output
Average	cells of interest	average μ_p and propagated error σ_p^2, for average function over those grid cells
Minimum	cells of interest	minimum μ_p and propagated error σ_p^2, for minimum function over those grid cells
Maximum	cells of interest	maximum μ_p and propagated error σ_p^2, for maximum function over those grid cells
Count	threshold θ, cells of interest	count of grid cells and their locations, having $\mu_p > \theta$ and propagated error σ_p^2 for count function
Range	thresholds θ_1, θ_2, cells of interest	count of grid cells and their locations, having $\theta_1 < \mu_p < \theta_2$ and propagated error σ_p^2 for range function

TABLE I: Prac2PC supported queries, with client inputs and outputs. Propagated error discussed in Section VII-C

see minimal overhead in latency and network bandwidth in getting a response. In Prac2PC, they will see some overhead (evaluation in Section X) as they have to participate in secure computation. Our system ensures that rival companies faking as ordinary clients do not get privacy-sensitive information in both schemes. However, the privacy-utility trade-offs vary across schemes as analyzed in subsequent sections.

Query Granularity: The queries and responses for sensor data occur at the granularity of square grids, motivated by the methods in [41], [42], [43]. Consider a city $\mathcal{A}$ be enclosed by a large rectangle of side l and b. The large rectangle is divided into small squares of side z meters (Figure 2(b)). Each square is labeled with a unique id i for referencing. Individual square's property (air pollution, traffic density) is represented by its centroid. A universal mapping of the coordinate system in terms of <LAT,LON> to grid reference id i is known to all the parties involved. Each query's location data is in terms of i and all the parties convert the <LAT,LON> to i beforehand. To map the coordinate (x_i, y_i) to the square i, we compute

$$\lfloor \frac{x_i - x_0}{z} \rfloor + \lfloor \frac{y_i - y_0}{z} \rfloor \cdot \frac{l}{z}$$

where (x_0, y_0) is the leftmost vertex in Figure 2(b). Smaller grid cells mean more granular spatial information, resulting in longer queries comprising more grid cells for a given client trajectory length. Longer queries would, in turn, have more computation and communication in 2PC. We experiment with different query lengths in Section X, based on real cab trajectory data, to analyze these query length-dependent overheads in Prac2PC.

Query types and output: While queries are at the granularity of grid cells, vehicles can be randomly distributed in those cells in the course of their natural movement across the city. Fleet companies need to periodically run spatio-temporal interpolation, through which each grid cell will get a $< \mu_p, \sigma_p^2 >$ tuple, μ_p denoting the interpolated sensor value and σ_p^2, the interpolated variance indicating confidence. When the pollution information is available publicly as pollution maps (interpolated or DP), clients can ask any query. However, Prac2PC's secure 2PC computation supports the queries in Table I, computing not only the μ_p, but also σ_p^2. The client can accept or reject the output if σ_p^2 is low or high, respectively. Computing σ_p^2 for different mathematical functions (average, minimum, maximum, count, range) over a grid, combining different interpolation variance values for each individual grid cell, is non-trivial in secure 2PC (detailed later in Table III). These wide range of queries and error functions, which might not have homomorphic properties, make HE less suitable. Instead, we use circuit-based 2PC methods with the flexibility to handle any function, as described in Section VII.

Threat Model and Privacy Policy: The adversary can be data consumers, i.e., a rival company that wants to learn about the distribution of cabs in the city and their fleet size. The other type of adversary is the data provider, i.e., cab server, which wants to learn about the behavioral preferences of the querier. We assume server and client to be both semi-honest (follows protocol) and malicious (can modify protocol).

For client's location privacy, suppose the client wishes to query a function on the blue squares in Figure 2(b). These blue squares might denote a route from her home to office. In public pollution maps, she can select the values of blue squares without any privacy concerns and then locally compute the required function. However, in Prac2PC, the client intelligently obfuscates the blue squares with a larger rectangle (red

Method	RMSE	σ_p^2
GPR	32.89	yes
GraphSAGE [44]	34.85	no
ANN [45]	41.03	no

TABLE II: Interpolation techniques comparison

rectangle in Figure 2(b)). The client is comfortable in revealing that she is present somewhere inside this rectangular region. A more privacy-seeking client will create a larger obfuscation rectangle to query the server by padding more small squares (green rectangle in Figure 2(b)). Prac2PC does not reveal which squares are blue within the client's obfuscation rectangle to the server. The server applies interpolation to generate uniformly distributed pollution values on the map for the server's fleet privacy. Before releasing the interpolated values, the server can apply either DP to preserve privacy or use Prac2PC to reveal aggregated $< \mu_p, \sigma_p^2 >$ to the client only if the number of blue squares in a query is greater than the threshold. The server decides this threshold based on grid cell size and dataset. Companies can set this threshold larger for better privacy protection.

V. Interpolation with Gaussian Process Regression

To select the best method for interpolation, we evaluate three different techniques i.e. Gaussian Process Regression (GPR) [14], GraphSAGE [44] and Artificial Neural Network (ANN) [45]. For evaluation, we select five random days from the Delhi dataset. *Delhi Dataset* is collected by fitting pollution measuring units by us in 10 DIMTS buses of Delhi. These buses collect spatiotemporally dense PM1, PM2.5, and PM10 values along with other meteorological factors while roaming around the city (routes shown in Figure 1). We separate out the data hour-wise from 9:00 AM to 9:00 PM and then split the data randomly into train and test using 80:20 ratio for each day. We further split the train data into training and validation sets using the same ratio. Table II reports the root mean square error (RMSE) on the test data compared to the ground truth PM2.5 values and also whether the method outputs posterior variance,σ_p^2. Low RMSE means better client utility. *Posterior variance* provides a sense of confidence in the predicted value, which aids in statistical analysis. Furthermore, in real-world deployments, data is expected to arrive in a streaming manner. Hence we trained the models for 100 epochs only to check whether a model can be trained to achieve low error rates within a short period of time. In this paper, we select GPR for interpolation based on its low RMSE (as shown in Table II) as well as the additional *posterior variance* output associated with each interpolated value.

In GPR, we do not consider the full covariance matrix (Eq 3) and only focus on the diagonal entries of the matrix (variance model). The diagonal elements of Eq 3 give the pointwise posterior variances at the testpoints. For a single test point x_* specified as centroids of the grid cells, the equation for posterior mean and the equation for posterior variance are rewritten as:

$$\mu_p = k_*^T A^{-1} y \tag{4}$$

$$\sigma_p^2 = k(x_*, x_*) - k_*^T A^{-1} k_* \tag{5}$$

where $A^{-1} = (K + \sigma_n^2 I)^{-1}$, y is the set of known pollution values at trajectory locations X, μ_p is the predicted pollution value at test point x_*, σ_p^2 is the associated uncertainty with our mean prediction i.e. posterior variance, $k(x_*, x_*)$ is the test point's prior variance, k_* denotes the vector of covariances between the test point and the training points and K denotes covariance matrix between training points.

The covariance or kernel function k captures the relationship between the pollution attribute and how it varies with change in distance between different trajectory points. The most common kernels used in GP are RBF, Matern32, and Matern52 [14]. The kernel function has some hyperparameters that were learned while optimizing the GP model. These hyperparameters are the variance and the lengthscale of the kernel and the variance of the noise. We used Matern32 kernel based on the accuracy comparison of different kernels (Appendix A). The training time for one hour data, i.e. 1400 samples takes around 1.2min (details in Appendix B).

A. GPR Output

Using GPR, the data server computes an interpolated value of the air pollution attribute and variance $< \mu_p, \sigma_p^2 >$, a tuple for each grid cell's centroid. If more than one pollution attribute is measured and interpolated, there will be more than two elements in the tuple per centroid. To analyze performance and privacy mechanisms, we consider one tuple representing one pollution attribute along with its posterior variance, which can be easily generalized to include more attributes if needed. Both elements of the tuple $< \mu_p, \sigma_p^2 >$ are important. The first gives the actual pollution estimate, while the second tells us how much confidence the model has in predicting that particular pollution value. It also gives the estimate of maximum and minimum of $< \mu_p, \sigma_p^2 >$ of a grid. A client interested in pollution information can easily work with this data by selecting the μ_p values for their region of interest. The client can also accept or reject the predicted value based on the magnitude of the corresponding σ_p^2.

B. Privacy Guarantees with GPR

In the absence of interpolation, specific grid cells without vehicles would give a *"no measurement available"* response. Having access to such information could give competitive advantage to a rival cab company, which can route more vacant cabs to that area, increasing revenue. Interpolation preserves basic server privacy, against revealing cells without vehicles, as all cells respond uniformly with interpolated values. The location of specific vehicles also remain hidden with interpolation, as queries and responses are in terms of grid centroids, each of which subsumes all individual vehicles within that cell (contrast Figure 3 on left with Figure 3 on right where the former reveals the exact trajectory, and the latter shows only grid based interpolated values). Thus, interpolation itself gives some privacy guarantees for raw trajectories. However, as we will see next, there can be some location information leakage from interpolation outputs that needs further handling for stronger privacy guarantees.

C. GPR Output Leaks Information

Posterior mean μ_p (Eq 4), depends on both training input variables (i.e. instrumented vehicle location in our application) and the training output variable (i.e. measured pollution values). The magnitude of μ_p is more dependent on the output

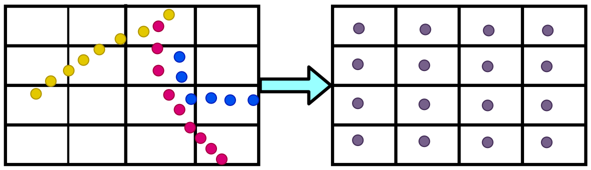


Fig. 3: Trajectory shows every vehicle locations (left side), Interpolation subsumes individual vehicle location in a grid cell (right side)

variable than the input variable as the kernel function, which is applied on the latter, results in terms that are generally smaller in magnitude than the output variable values.

However, the posterior variance, σ_p^2 in Eq 5 does not depend on the training output variable (pollution values) but only on the training input (sensor locations). σ_p^2 will be high for a test point that has less or far-off neighboring sensors when compared to a point having more or nearer neighbor sensors. Thus, unlike μ_p, σ_p^2 is more closely related to training sample locations. For sensors shown in Figure 4(a), we see a contour plot of σ_p^2 in Figure 4(c). We can clearly make out regions with cabs as they have low posterior variances (darker regions).

Thus, given this correlation between σ_p^2 values of grid cells and the location of vehicles in those grid cells (as shown in Figure 4), taxi companies would likely not want to share σ_p^2 information. This is problematic for environmental agencies who will have no clue about the confidence of the model in the interpolated values. We explore how to balance this trade-off between fleet privacy and client utility using DP and 2PC in subsequent sections.

VI. Differentially Private Pollution Maps

After interpolating the grid, $< \mu_p, \sigma_p^2 >$ for individual grid cells are released by the cab fleet owning company as a web map. The σ_p^2 values taken together may leak the trajectories of the vehicles, compromising server privacy (Figure 4(c)). In this section, we use Differential Privacy to add noise to σ_p^2, which we refer to as DP noise henceforth, so that no original σ_p^2 value can be computed from the noisy result and no trajectories may be discerned. Smith et al. [46] apply DP to GPR posterior mean, μ_p. It considers the training output to be private while training input to be public. This simplifies the sensitivity calculation for μ_p. However, in our system, only training input is private, so we cannot apply the same method of sensitivity calculation for σ_p^2. To the best of our knowledge, we are the first to apply DP to GPR posterior variance. We calculate the sensitivity for σ_p^2 as follows.

A. Sensitivity Calculation for σ_p^2

We have to calculate the sensitivity of the σ_p^2 before applying DP. To compute the sensitivity, consider a database D containing the location, μ_p, and σ_p^2 as rows. The equation of σ_p^2 for Database D and D' which differ at row j is:

$$f(D) = k(x_*, x_*) - k_{*,\ D}^T\ A_D^{-1} k_{*,\ D} \tag{6}$$

$$f(D') = k(x_*, x_*) - k_{*,\ D'}^T\ A_{D'}^{-1} k_{*,\ D'} \tag{7}$$

$$f(D') - f(D) = k_{*,\ D}^T\ A_D^{-1} k_{*,\ D} - k_{*,\ D'}^T\ A_{D'}^{-1} k_{*,\ D'} \tag{8}$$

In our sensitivity calculation, A^{-1} is private. Using the positive semidefinite property of A^{-1}, we know that $k_*^T A^{-1} k_* > 0$. Therefore Eq 8 reduces to the following equation:

$$|f(D') - f(D)| < k_{*,\ D}^T\ A_D^{-1} k_{*,\ D} \tag{9}$$

Since $k_*^T A^{-1} k_* < k(x_*, x_*)$, as $\sigma_p^2 \geq 0$, the upper bound of $|f(D') - f(D)|$ comes out as $k(x_*, x_*)$. For selected kernel i.e. Matern32, $k(x, x_*)$ is defined as $\sigma_f^2 \left(1 + \frac{\sqrt{3}d}{l}\right) exp\left(-\frac{\sqrt{3}d}{l}\right)$, where $d = ||x - x_*||$ i.e. the distance between x and x_*, and l is the length-scale of the kernel. When $x = x_*$, $k(x, x_*) = \sigma_f^2$ as $d = 0$. The sensitivity of posterior variance i.e. $k(x_*, x_*)$, has a maximum value of 1 if we take variance of the kernel (σ_f^2) equals to one (works well for simple kernel like RBF or Matern32 and standardized data [46]).

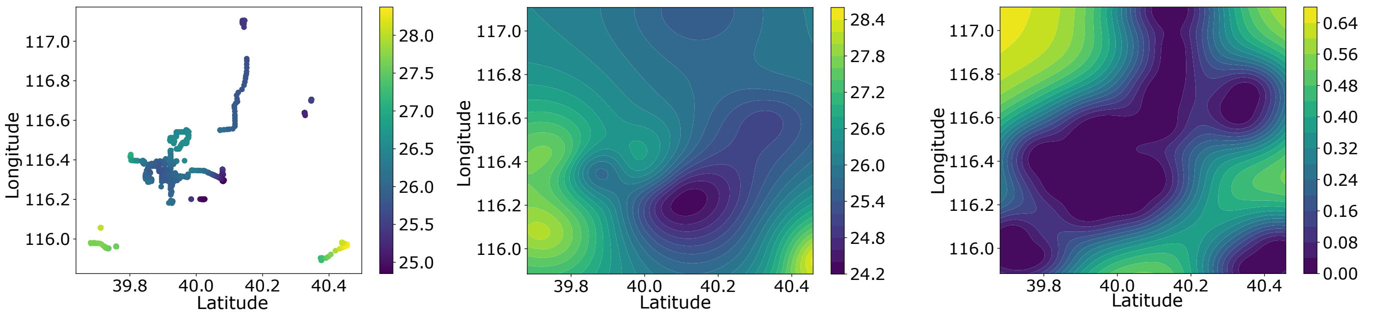
As the support of the Laplace function is unbounded, the random noise added to the output function could be arbitrarily large and negative. To avoid the negative and large variance value, we applied the bounded Laplace mechanism [47]. This works on the principle of rejection sampling and adds Laplace noise such that the noisy posterior variance values fall within a specified range. We take the upper and lower bounds to be 0 and σ_f^2.

B. DP Pollution Maps Output

Figure 4(c) highlights the area where the density of vehicles is more (darker color implies lower σ_p^2 implies more vehicle density). To prevent this leakage, we add bounded noise to σ_p^2. Figure 5(a) shows the contour plots after adding noise with privacy budget $\epsilon = 10$. We can see that, though the contour looks more spread out here than Figure 4(c), the area with no cabs can still easily be discerned. The trace of actual trajectories is completely removed from the plots when we add noise with $\epsilon = 0.5$ (Figure 5(b)). Since the plot for σ_p^2 looks very chaotic for $\epsilon = 0.5$, we need to measure the utility for the dataset. As adding more noise degrades utility, the noisy σ_p^2 values might not be too useful for the environmental analysts. We analyze this privacy vs. utility trade-off with different values of ϵ next.

C. Air Pollution Datasets to Evaluate Client Utility

We consider three different datasets for evaluation. The first dataset is our Delhi Dataset (description in Section V). Our second dataset is the Opensense Zurich dataset, collected by fitting environmental sensor boxes on top of trams. These trams roam in the city of Zurich. We take O_3 as a pollution attribute collected by a tram [48]. We also generate a third dataset using Bejing's open-source datasets for realistic cab trajectories. The Taxi Beijing dataset [49], [50] contains all the trips taken by 24 drivers from 2 February 2008 to 8 February 2008 across Beijing. This dataset only has GPS coordinates vs. time and does not contain pollution values. So, to construct a dataset that has pollution values corresponding to real cab trajectories, we interpolate PM2.5 along each cab's location using a separate air pollution dataset [51], which has PM2.5 values measured at different static pollution sensors. The newly interpolated dataset is the one we use for experimentation (referred to as Taxi-Pol dataset henceforth). The Taxi-Pol dataset consists of latitude, longitude, time, and PM2.5 values. We divide the



(a) Trajectories of different cabs (b) Posterior mean (c) Posterior variance without DP

Fig. 4: Interpolation output for the case where trajectory can be seen in the posterior variance contour plot

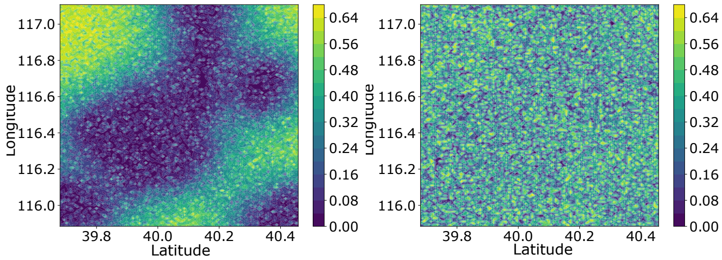
dataset into 150 windows of 1 hour each and apply GPR on each window separately to predict the $< \mu_p, \sigma_p^2 >$ of a grid.

D. Balancing Client Utility vs. Fleet Privacy

In this section, we discuss the effect of applying DP on utility. Our utility metric is the RMSE of σ_p^2 as environmental analysts need accurate values of σ_p^2. RMSE is calculated between the values of differentially private posterior variance ($\sigma_p^2 + BoundedLap(\sigma_f^2, \epsilon, \delta)$, where we fix $\delta = 0$, $\sigma_f^2 = 1$) and the original posterior variance (σ_p^2) for the whole grid. This metric shows how far σ_p^2 values are from the original σ_p^2 values after applying DP. This metric measures variance change due to DP noise. Lower RMSE means higher utility for environmentalists. Since we are not applying DP on μ_p, the accuracy of the posterior mean will remain unaffected.

In Figure 6, we plotted the CDF of the calculated RMSE of different hours for the three different datasets. It also includes the CDF of the maximum value of the original σ_p^2 of the grid for every hour of the day. This is to give an idea of the range of the original σ_p^2. In Figure 6(a)-(b), the RMSE for most of the hours lies below 1 for $\epsilon = 10$. This gives very good utility in comparison to $\epsilon = 1$ and $\epsilon = 0.5$ for both Beijing and Zurich datasets. However, as seen in Figure 5(a), we can still predict the areas where there are no cabs with $\epsilon = 10$, which is not the case for $\epsilon = 0.5$. Once we start reducing the privacy budget,ϵ, the percentage of hours that have low RMSE values start reducing (shown as y-axis in Figure 6). For Beijing and Zurich datasets, the DP-based method gives low RMSE at a low ϵ value, and hence the utility remains high.

In contrast, Figure 6(c), shows the utility-privacy trade-off CDF for PM2.5 data collected by 10 buses on 13 Nov 2020 in New Delhi, India. The trends for the three datasets are very similar. However, there is a huge difference in the range of RMSE of posterior variance for the Delhi Dataset. The RMSE and the actual posterior variances of the grid are dependent on the standard deviation of training data (effect of reversing the standardization of data before publishing pollution maps). If the standard deviation of training data is high, as is in the case of Delhi, the resultant RMSE gets high. Thus data utility for DP based method is low for the Delhi dataset.



(a) Posterior variance for $\epsilon = 10$ (b) Posterior variance for $\epsilon = 0.5$

Fig. 5: DP plots corresponding to different ϵ for the case where trajectory can be seen in the posterior variance contour plot

E. Further Fleet Privacy Concerns with DP

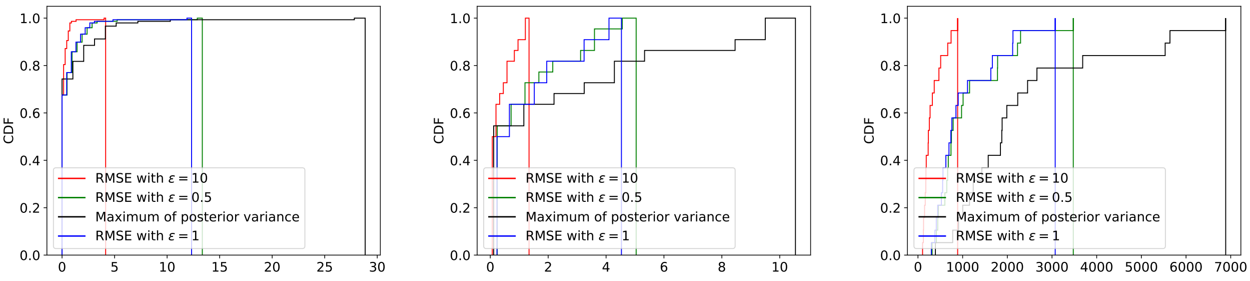
If the adversary has strong computation power, she can launch Machine Learning based attacks on the released μ_p, as it is published without applying DP. In this case, the server may want to apply DP on μ_p along with σ_p^2 to preserve its fleet locations. This, however, degrades the accuracy of air pollution values, and the client might not want to use the system. Air pollution datasets also have correlations among nearby areas. This correlation may reduce the ϵ-DP guarantee. An adversary can try to learn it and may launch a correlation-based attack. To prevent this, a server can define the correlation between different data points as a Correlation Matrix, M. This increases the sensitivity of the dataset by a factor of the correlation matrix ($|M \ f(D') - f(D)|$). Hence, the Laplace noise to be added in the σ_p^2 values will also increase.

In the cases where noise added in σ_p^2 is too high or required noise addition to μ_p as well, achieving the good utility with strong privacy will be difficult. In such cases, DP pollution maps cannot be made public as noise added will be too high for meaningful utility. As an alternative solution, 2PC based query response system, namely Prac2PC, has been designed, implemented, and evaluated next.

Differentially Private pollution maps provide absolute privacy to clients. Clients can compute any function locally with minimal overhead using the released data. However, balancing fleet privacy and client utility at the same time seems impossible for pollution datasets.

VII. Query Response System Using Secure 2PC

Query Response system uses information aggregation for server privacy and location obfuscation for client privacy. It reveals σ_p^2 aggregated over at least α blue squares to the client, to maintain server location privacy. This α is decided by the server based on grid cell size and dataset.

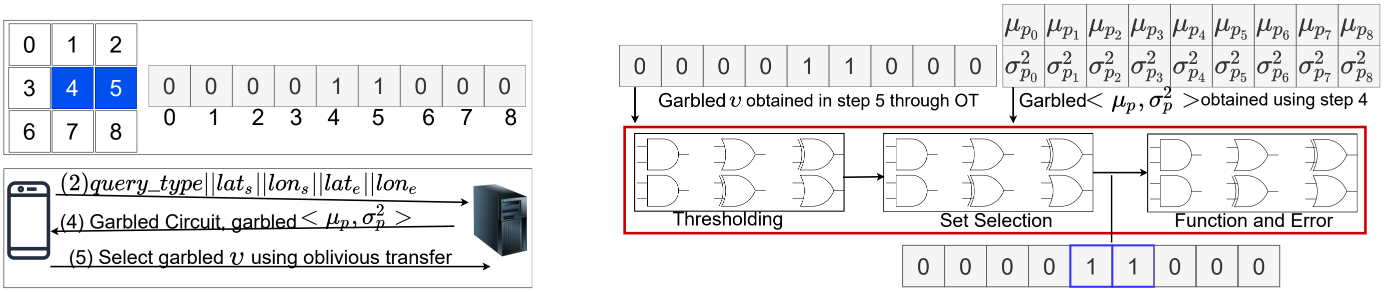


(a) Beijing

(b) Zurich

(c) Delhi

Fig. 6: Utility-vs-Privacy for different cities in terms of CDF of RMSE for different hours.


0 1 2
3 4 5
6 7 8
0 0 0 0 1 1 0 0 0
0 1 2 3 4 5 6 7 8
(2)$query\_type||lat_s||lon_s||lat_e||lon_e$
(4) Garbled Circuit, garbled$< \mu_p, \sigma_p^2 >$
(5) Select garbled υ using oblivious transfer
0 0 0 0 1 1 0 0 0
μ_{p_0} μ_{p_1} μ_{p_2} μ_{p_3} μ_{p_4} μ_{p_5} μ_{p_6} μ_{p_7} μ_{p_8}
$\sigma^2_{p_0}$ $\sigma^2_{p_1}$ $\sigma^2_{p_2}$ $\sigma^2_{p_3}$ $\sigma^2_{p_4}$ $\sigma^2_{p_5}$ $\sigma^2_{p_6}$ $\sigma^2_{p_7}$ $\sigma^2_{p_8}$
Garbled υ obtained in step 5 through OT
Garbled$< \mu_p, \sigma_p^2 >$obtained using step 4
Thresholding
Set Selection
Function and Error
0 0 0 0 1 1 0 0 0

(a) Private input generation (above). Prac2PC steps involving client server communication (below).

(b) Circuit evaluation, final Prac2PC step on Client. Circuit (red rectangle) is created by Server and sent to Client in encrypted form.

Fig. 7: Circuit details of Prac2PC

A. Prac2PC- Secure 2PC Protocol

Prac2PC is a circuit-based secure computation protocol between two parties. It comprises the following six steps between a client and a server (shown in Figure 2). Steps 2, 4 and 5 require server-client communication (shown in Figure 7(a) bottom). Steps 1 and 6 require computations on the client, and step 3 requires computations on the server. We present the generic protocol in this section. Implementation details, and evaluations of communication and computation overheads, will be presented in subsequent sections.

[1] Private input generation: Client maps each grid cell inside her obfuscation rectangle to either one or zero. If client is interested in a particular grid cell (blue color in Figure 2(b)), she sets $\upsilon_i = 1$, otherwise, $\upsilon_i = 0$. She generates a bit vector υ (sample shown in Figure 7(a) top) but does not send this vector to the server. The size of the bit vector (represented by n) is equal to the number of grid cells in the obfuscation rectangle.

[2] Query request: Client sends a query to the server. The query consists of the type of function (one of the functions defined in Table I), represented as *Query_Type*. It also has coordinates of the diagonal vertices a and c of the obfuscation rectangle.

$$\text{Request } \leftarrow \{Query\_Type||lat_s||lon_s||lat_e||lon_e\}$$

where, lat_s and lon_s are the coordinates of vertex a, and lat_e and lon_e are the coordinates of vertex c.

[3] Circuit generation: Both client and server have the mapping of coordinate to grid id. On receiving the query, the server performs coordinate to grid id conversion and gets the obfuscation rectangle. Server then selects $< \mu_p, \sigma_p^2 >$ tuples which are inside the obfuscation rectangle. These tuples form the server's input in the secure computation. The server also prepares the circuit (different circuits for each query type) for the requested function. The circuit components will be detailed in Section VII-B.

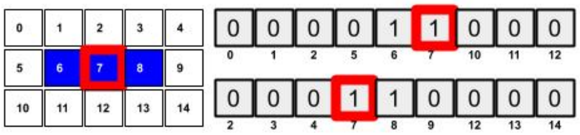


Fig. 8: Smartly crafted queries where subsequent queries overlap by a single grid square (marked in red).

[4] Circuit and garbled values transfer: Server sends this generated circuit in an encrypted format to the client. It also sends the garbled values of its input $< \mu_p, \sigma_p^2 >$.

[5] Oblivious Transfer: Client engages in Oblivious Transfer (OT) [15], [16] with server to get garbled values of the client inputs (υ). Oblivious transfer ensures that the server does not get to know the client's inputs.

[6] Circuit Evaluation: Once the client receives all the garbled inputs, it evaluates the circuit on its own. The client does not know the intermediate results of the circuit. It only gets to know the output of the requested function at the end. The circuit evaluation on the client is shown in Figure 7(b). Details of the circuit (red rectangle) will be discussed next.

Protection against smartly crafted queries By restricting computations over at least α grid cells, Prac2PC makes location or fleet size information retrieval significantly hard. Still, if client queries are smartly crafted, for example, subsequent queries differ by a single square, then the σ_p^2 value of that square might be computed using two queries (shown in Figure 8). The correlation between location information and σ_p^2 is much higher than that between location and μ_p. Thus, to protect against smart queries, the server adds Laplacian noise [52] to the output of the error propagation function by applying ϵ-DP [12], [53]. It ensures that no individual σ_p^2 value can be computed from the noisy result. It is done by extending the circuit in Figure 7(b). Prac2PC database comprises grid cell and their σ_p^2 values, and f is the error propagation function.

B. Circuit Components

Prac2PC has been carefully designed to handle different query functions, as well as the privacy requirements of server and client. Following are the different sub-circuits of Prac2PC: (1) Private thresholding, (2) Private set selection, (3)Private statistical query function computation with error propagation.

(a) Private thresholding: To compute results over at least α grid cells and prevent σ_p^2 information leakage, the server imposes a restriction on the minimum number of $\upsilon_i = 1$ in a query. To count ones in υ and compare it to α, a circuit is created for the following function:

$$\beta = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} \upsilon_j \geq \alpha \\ 0 & \text{Otherwise} \end{cases} \tag{10}$$

The output of the threshold circuit is represented by β, which is fed to the next stage of the circuit. All later stages of the circuit will result in zero if the number of squares in the query vector is less than α.

(b) Private set selection. Once the thresholding is done, the next part of the circuit is to select all the $< \mu_p, \sigma_p^2 >$ pairs corresponding to the blue squares in Figure 2(b). The blue squares correspond to the locations in which the client is interested (υ has ones only in these places). Based on the υ vector, we select locations' property ($< \mu_p, \sigma_p^2 >$) stored at the server's side obliviously such that:

$$y_i = \beta \wedge \upsilon_i \wedge \mu_{p_i},\ i \in [1, n] \tag{11}$$

$$\sigma_i^2 = \beta \wedge \upsilon_i \wedge \sigma_{p_i}^2,\ i \in [1, n] \tag{12}$$

(c) Private statistical query function computation with error propagation: The final sub-circuit changes based on the query function. Prac2PC supports a variety of realistic functions as listed in Table I. These functions are computed on the client over the blue squares, for which $\upsilon = 1$ and the client gets the output. The server has $< \mu_p, \sigma_p^2 >$ tuple for each square in the obfuscation rectangle, which is sent in the garbled form to the client. The circuit should compute the necessary function on the μ_p values for those tuples, where $\upsilon = 1$. In addition, each μ_p estimate in the different tuples is associated with a different interpolation variance σ_p^2. For a function computed with μ_p from different tuples, the circuit needs to compute the cumulative variance of the result, combining the errors of the individual μ_p values. This process is called error propagation in mathematics [54]. Without error propagation, the client would not know how much confidence to have in the query result. We detail these query functions and their error propagation functions next, in Section VII-C.

C. Query function with error propagation

For each query, a separate circuit is invoked based on $Query\_Type$. The functions to compute each query and the corresponding error propagation are listed in Table III. For any function f, error propagation function [54] is derived from the normally distributed errors. It can be represented as:

$$E(x) = \sigma_{f_i} = \sqrt{\sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i}\right)^2 \sigma_i^2} \tag{13}$$

where σ_f represents the standard deviation of function f, σ_i is the standard deviation of x_i.
Error propagation for a mathematical function requires the function to be differentiable. Among the query function supported by Prac2PC, only the average function is differentiable. We estimate the remaining logical functions, i.e., minimum, maximum, count, and range, with a differentiable alternative. The differentiable function has been selected based on the accuracy of the approximation. Furthermore, we also ensure that the function does not result in overflow while implementing the circuit. We describe below how the query functions listed in Table III are mapped to their differential counterpart.

1. **Average**: The server and the client obliviously compute the sum of the pollution values and the number of the locations in which the client is interested. Both the parameters are then used to evaluate average function(Eq 14) as follows:

$$f = \frac{\Sigma_{i=1}^{n} y_i}{\Sigma_{i=1}^{n} \upsilon_i} \tag{14}$$

To calculate the error propagation of average function, we compute the partial derivative of the average (given in Eq 15) and then substitute the partial derivative of f and posterior variance $< \sigma_p^2 >$ of the blue squares (represented by σ_i), in the Eq 13, we get Eq 16:

$$\frac{\partial f}{\partial y_i} = \frac{1}{\Sigma_{i=1}^{n} \upsilon_i} \tag{15}$$

$$E(y) = \frac{\sqrt{\sigma_1^2 + \sigma_2^2 \ldots \sigma_n^2}}{\Sigma_{i=1}^{n} \upsilon_i} \tag{16}$$

2. **Minimum or maximum**: For minimum or maximum, we use the *LogSumExp* function. *LogSumExp* is a popular smooth approximation of maximum function used in ML [55].

$$f = y^* + \frac{1}{\rho} \log\left(\Sigma_{i=1}^{n} \exp(\rho(y_i - y^*))\right), \rho \leq 0 \tag{17}$$

where $y^* = \max\ \{y_1, .., y_n\}$, ρ characterises the behavior of f, if $\rho \leq 0$, then the function calculates minimum else maximum. The following equation shows the partial derivative of Eq 17:

$$\frac{\partial f}{\partial y_i} = \frac{\exp(\rho(y_i - y^*))}{(\Sigma_{i=1}^{n} \exp(\rho(y_i - y^*)))} \tag{18}$$

Similar to average error propagation function, we get the following equation for error propagation of Minimum function:

$$E(y) = \sqrt{\begin{array}{c} \left(\frac{\exp(\rho(y_1 - y^*))}{(\Sigma_{i=1}^{n} \exp(\rho(y_i - y^*)))}\right)^2 \sigma_1^2 + \\ \left(\frac{\exp(\rho(y_2 - y^*))}{(\Sigma_{i=1}^{n} \exp(\rho(y_i - y^*)))}\right)^2 \sigma_2^2 + \ldots \end{array}} \tag{19}$$

Query Function (f)	**Function Formula**	**Propagated Error**
Average	$\frac{\Sigma_{i=1}^{n} y_i}{\Sigma_{i=1}^{n} \upsilon_i}$	$\frac{\sqrt{\sigma_1^2+\sigma_2^2 \ldots \sigma_n^2}}{\Sigma_{i=1}^{n} \upsilon_i}$
Max or Min	$y^* + \frac{1}{\rho} \log\left(\Sigma_{i=1}^{n} \exp(\rho(y_i - y^*))\right)$	$\sqrt{\sum_{j=1}^{n} \left(\frac{\exp(\rho(y_j - y^*))}{\Sigma_{i=1}^{n} \exp(\rho(y_i - y^*))} \sigma_j \right)^2}$ (Max $\rho \to \infty$, Min $\rho \to -\infty$)
Count	$\sum_{i=1}^{n} S(y_i, \theta)$	$\sqrt{\sum_{j=1}^{n} \left(S(y_j, \theta)(1 - S(y_j, \theta))\sigma_j\right)^2}$
Range	$\sum_{i=1}^{n} (S(y_i, \theta_1)$ - $S(y_i, \theta_2))$, $\theta_1 < \theta_2$	$\sqrt{\sum_{j=1}^{n} \left(S(y_j, \theta_1)(1 - S(y_j, \theta_1)) - S(y_j, \theta_2)(1 - S(y_j, \theta_2))\right)^2 \sigma_j^2}$

TABLE III: Query function and error propagation formulae, implemented in the Garbled Circuit where $S(y_i, \theta)$=$\frac{1}{1+\exp(-(y_i - \theta))}$

3. **Count or range**: To calculate the count of grid cells which have the pollution level less than the set threshold θ, we use *Sigmoid* as the approximation function. *Sigmoid* function has a property of limiting its output to 0 and 1 when the input value is less than zero and greater than zero, respectively. By utilizing this property, we use shifted Sigmoid to compute the number of locations that have pollution greater than some threshold. The total count of such locations are calculated using a summation over all the region of interest, represented by the following equation:

$$f = \sum_{i=1}^{n} \frac{1}{1 + \exp(-(y_i - \theta))} = \sum_{i=1}^{n} S(y_i, \theta) \qquad (20)$$

The partial derivative and propagation of error function can be represented mathematically as follows:

$$\frac{\partial f}{\partial y_i} = S(y_i, \theta)(1 - S(y_i, \theta)) \qquad (21)$$

$$E(y) = \sqrt{\sum_{j=1}^{n} \left(S(y_j, \theta)(1 - S(y_j, \theta))\sigma_j\right)^2} \qquad (22)$$

For the range query, where the client requests for the number of locations having pollution in the range of $[\theta_1, \theta_2]$, we compute the count function on both θ_1 and θ_2 independently and then take the difference of the count returned by it. This function has the most complicated circuit as for a single element in υ vector, it requires two exponential computations.

Finally, the circuit also supports the query in which the client asks the location for which air pollution is greater than some threshold within the obfuscation rectangle. Unlike previous queries, which return a pollution value with an error, this query needs a location as an answer. In this case, the circuit returns a bit vector of length *n*, where any 1 at index *j* will represent the pollution value at square *j* to be greater than the threshold. Returning the index of maximum and minimum pollution will be a special case of these queries. To preserve the client's location privacy, the response to all these queries will be available to only the client.

D. Adding Differential Privacy

The server gets an estimate of sensitivity (Δf) from interpolation variance values. We take the average query as an example to describe the sever side computation. The global sensitivity for the corresponding error propagation function is defined as $\Delta f = \sigma_{max}^2 - \sigma_{min}^2$. The error propagation for average query over n squares is represented as $E(y)$ from Eq 16. Based on the algorithm discussed in [12], to satisfy ϵ-DP the noise equivalent to $(Lap(\sigma_{max}^2 - \sigma_{min}^2/\epsilon))/n^2$ is added to $E(y)^2$. We apply the bounding Laplace [12] to restrict the unbounded Laplace. The server assigns the value of σ_{min}^2 to $E(y)^2$, if $E(y)^2 < \sigma_{min}^2$. Similarly, if $E(y)^2$ is greater than σ_{max}^2, the server assigns σ_{max}^2. The σ_{min}^2 and σ_{max}^2 are known after GPR modeling. Prac2PC circuit is modified to accommodate the DP operations. Comparison with σ_{min}^2 or σ_{max}^2 and assignment, if needed to bound the DP output, are all done within the 2PC circuit. Since the server knows the sensitivity of the error function, it maintains a table of random numbers drawn from the Laplace distribution. When the client query arrives, the server selects at random a number from the table and sends it to the client along with the server's private input $< \mu_p, \sigma_p^2 >$. This number is the noise fed to the circuit to be added to $E(y)^2$.

VIII. End-to-end Security Analysis

As discussed in Section IV, we seek to protect three pieces of information (a) server's fleet location, (b) server's fleet size, and (c) client's query location. We use GPR, 2PC, and DP in succession to provide end-to-end privacy guarantees.

The primary privacy property that we offer is for the data providers - hiding fleet size and location information. It is provided by GPR as a first step. GPR is performed on the data obtained from different vehicles. Once the tuple $< \mu_p, \sigma_p^2 >$ for each grid cell is obtained, any query is computed on the interpolated grid cells. GPR subsumes any vehicle information present in the grid cell by a single centroid tuple; thus, one cannot correlate it to the number of vehicles present in a grid. We also do not use fleet size to calculate any query response. This prevents the client from getting any idea of the fleet size or location of individual vehicles in a grid. We have empirically evaluated the privacy of this step (in Section V). It is strictly better than having no GPR. However, as shown in the information leakage through the variance plots in Figure 4, server privacy cannot be maintained by GPR alone, and we need 2PC+DP in addition to GPR to prevent information leakage through per-grid variance.

Next, 2PC checks that the client's query vector υ always contains more than α bits set to 1. Here α is the threshold on the minimum number of grid cells required for answering a 2PC query. This threshold is a privacy knob for the server leading to a trade-off with client utility. A higher aggregation threshold indicates more server privacy and less client utility as the client gets less granular information. Server privacy then boils down to the server using 2PC to securely compute the query's result using the output of GPR and then adding differentially private (DP) Laplacian noise in the aggregated result of propagation error. The 2PC protocol simply computes a deterministic function securely (i.e., without revealing the inputs of one party to the other). It does not negate or undo the privacy preserved by GPR or DP techniques. Based on the threat model (semi-honest vs. malicious), the corresponding 2PC protocol is used.

The Laplacian noise (based on the sensitivity) reduces the possibility of leaking the location of vehicles via the propagated error in the query output. Propagated error combines the GPR variance across the α threshold grids. Adding DP noise is especially important if subsequent 2PC queries overlap in a single grid cell, and the 2PC answers are aggregated across subsequent queries to reveal per grid GPR variance. This noise addition takes place within the garbled circuit, and the client gets to know the output only after noise addition. Even though we use the bounded Laplace mechanism for DP, it still provides ϵ-DP as shown in [47], [12]. Thus, the client can only obtain the aggregated results for its region of interest, but it cannot recover the value of each individual sensed data.

For the client's privacy, the client first identifies an obfuscation rectangle to construct the grid cell mapping of <LAT,LON>. That is, clients' two-dimensional location vector is converted to a single υ vector. This obfuscation rectangle contains the client's region of interest, i.e., wherever $\upsilon_i = 1$. Based on the υ vector, nothing could be learned about the client's region of interest besides the fact that the client is interested in α or more grid cells in the defined rectangle. The client does not send her υ vector in the query requests. Since the output of the query is only revealed to the client, the server cannot reverse map the υ vector at any time.

IX. PRAC2PC IMPLEMENTATION

To use our designed Prac2PC protocol in an end-to-end system, we need to implement the circuits described. As discussed in Section III, GC and mixed protocols are circuit-based 2PC alternatives. We use an empirical system design approach here – implement different Prac2PC versions using both circuit alternatives, and then compare their performance using real datasets.

A. Prac2PC Variants

We list below our different Prac2PC implementations.

[1] Prac2PC-GC: We implement the Prac2PC GC version using a C++ framework, EMP-toolkit [56]. It provides secure data types and operators for integer and floating-point (FP) numbers. Prac2PC uses an exponential function in some queries, which can only be implemented using 32-bit FP numbers in EMP. Hence, the server's input tuple $< \mu_p, \sigma_p^2 >$ is represented by two 32-bit floating-point numbers, whereas the client's input υ is represented by a bit vector.

[2] Prac2PC-MP: We cannot implement Mixed Protocol in EMP. So instead, we use ABY, another C++ MPC framework. It allows the implementation of the Arithmetic (A) circuit, Boolean (B) circuit, and Yao's (Y) GC or a combination of these. As of now, ABY's arithmetic circuits do not support FP operations fully. To perform FP operations, we have used Boolean circuits. This constraint leads us to implement Prac2PC using a B+Y mixed protocol instead of A+B+Y.

[3] Prac2PC-Malicious The above variants are secure against a semi-honest adversary. To handle the case when both parties (malicious) might not follow the correct protocol, we implement Prac2PC in EMP using authenticated garbling [21]. It is based on the combination of GMW Protocol based secret sharing and Yao's garbled circuit.

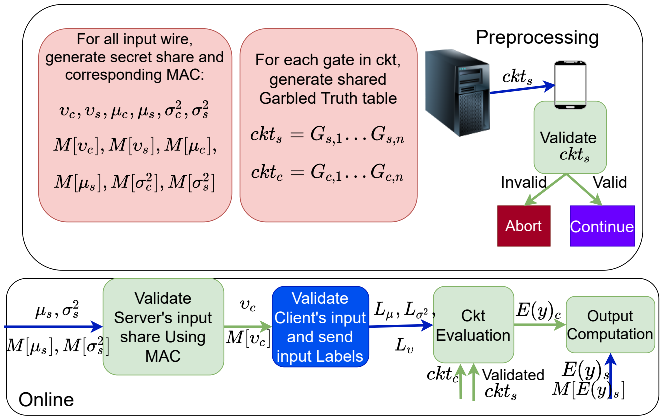

Preprocessing
For all input wire, generate secret share and corresponding MAC:
$\upsilon_c, \upsilon_s, \mu_c, \mu_s, \sigma_c^2, \sigma_s^2$
$M[\upsilon_c], M[\upsilon_s], M[\mu_c],$
$M[\mu_s], M[\sigma_c^2], M[\sigma_s^2]$
For each gate in ckt, generate shared Garbled Truth table
$ckt_s = G_{s,1} \ldots G_{s,n}$
$ckt_c = G_{c,1} \ldots G_{c,n}$
ckt_s
Validate ckt_s
Invalid
Valid
Abort
Continue
Online
μ_s, σ_s^2
$M[\mu_s], M[\sigma_s^2]$
Validate Server's input share Using MAC
υ_c
$M[\upsilon_c]$
Validate Client's input and send input Labels
$L_\mu, L_{\sigma^2},$
L_υ
Ckt Evaluation
ckt_c
Validated ckt_s
$E(y)_c$
Output Computation
$E(y)_s$
$M[E(y)_s]$

Fig. 9: Prac2PC-Malicious workflow. Red boxes represent the steps that are performed by both client and server independently. Blue boxes represent computation at only the server-side, and blue arrows represent data coming from the server. Similarly green boxes and arrows are for Client (subscript s denotes server share and subscript c denotes client share, υ, μ, σ^2 are inputs as discussed in VII).

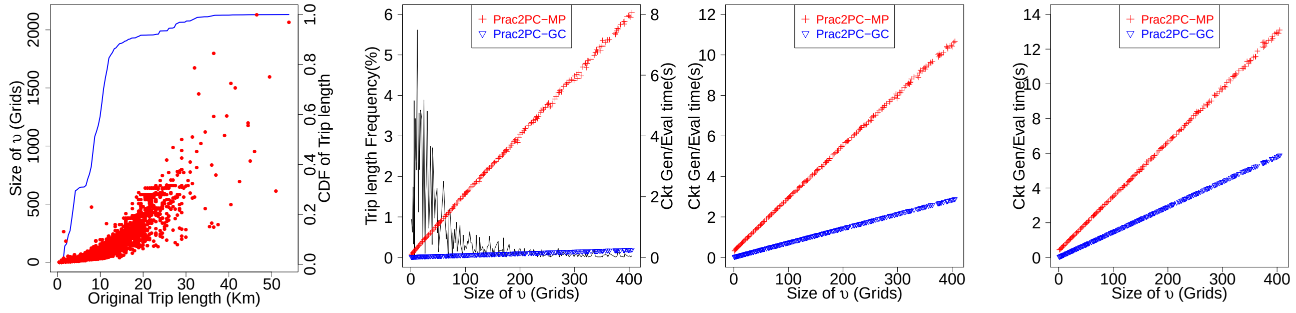
Configuration	Client	Server
Processor	C1: Intel(R) Core(TM) i7-4790 C2: ARM Cortex-A73 & A53	Intel(R) Xeon(R) CPU E5-2698 v4
Memory	C1: 16GB C2: 6GB	256GB
Frequency	C1: 3.60GHz C2: 4@2.3 GHz& 4@1.7 GHz	2.20GHz
No. of Cores	C1:8, C2: 8	40
OS	C1: Ubuntu 18.04 C2: Android 11	Ubuntu 18.04

TABLE IV: Client-Server configuration

Figure 9 shows the server and client interactions, who have their own secret share for the input and output wires. To protect against the malicious server, (1) MAC is generated to authenticate secret shares, and (2) the garbled truth tables are secretly shared between client and server, unlike Prac2PC-GC. A party can learn that the other party is cheating if, at any stage, the MAC sent by the other party does not match with the share sent. The overall protocol is divided into two phases for efficiency, i.e., *preprocessing* and *online*. The preprocessing phase needs the *Query_Type* and the υ size to be decided. It will generate the information needed to construct the authenticated garbled circuit. The inputs to the circuit (which bits of υ are 1, and the $< \mu_p, \sigma_p^2 >$ values) and authenticated garbled circuit are needed only in the online phase to evaluate a particular query.

B. Experimental Setup For Evaluations

Server and client machines: The client will run one circuit for each query, whereas the server has to run as many circuits as there are simultaneous client requests. We use the system configurations given in Table IV for server and client. We also use the Android phone as a client (represented by C2 in table) by porting the Prac2PC C++ library to ARM.



(a) (b) Average (c) Minimum (d) Range

Fig. 10: (a) Trip length vs. υ size for 7500 trips. (b)-(d) Runtimes for different query types and Prac2PC versions

Datasets: Beijing Taxi dataset lack start and endpoints of a trajectory, and hence, we use the Porto Taxi Trajectory dataset [57] to evaluate the practicality of Prac2PC and compare its variants. The Porto dataset contains trip data of 441 taxis running across Porto, Portugal, for around nine months. Each trip is represented by a set of <LAT,LON> pairs with time. Prac2PC efficiency is entirely dependent on υ, and not on $<\mu_p, \sigma_p^2>$. So this *trip* dataset is important to test Prac2PC with realistic υ vectors, based on actual commuter trips. We use random float values as $<\mu_p, \sigma_p^2>$ in these experiments.

To evaluate DP's utility, we need real interpolation variance values as random $<\mu_p, \sigma_p^2>$ values will not be useful to analyze the privacy-utility trade-offs. We use the Opensense Zurich dataset [58] and the Delhi dataset we collected (Figure 1) in our DP evaluations.

X. Prac2PC Evaluation

Prac2PC is secure by design, but it is important to evaluate how practical its different variants are. For extensive evaluations, we divide the different queries described in Table I into three categories: average, minimum, and range. It is because the maximum query function and its error propagation are very similar to a minimum query. The count function and its error propagation are a subset of the range query.

A. Realistic υ sizes

The size of the client's private input (υ vector) depends on the actual trajectory length and the size of the obfuscation rectangle chosen by her, which is proportional to her privacy concern. For our 7500 trajectories, we choose the bounding rectangle of each trajectory as its obfuscation rectangle. Our grid square size is 500m. Figure 10 (a) shows the trip lengths along x-axis, with obfuscation rectangle sizes along left y-axis. From the right y-axis, 80% of the taxi trajectories are less than 15 km (intuitive as cabs are too costly for higher distances). The obfuscation rectangle for 15 km is less than 100 km or 200 grids. We present results till 400 grids in subsequent sections, highlighting the overheads for 100-200 grids, which are more realistic for commuters.

B. Prac2PC Runtimes and CPU Utilization

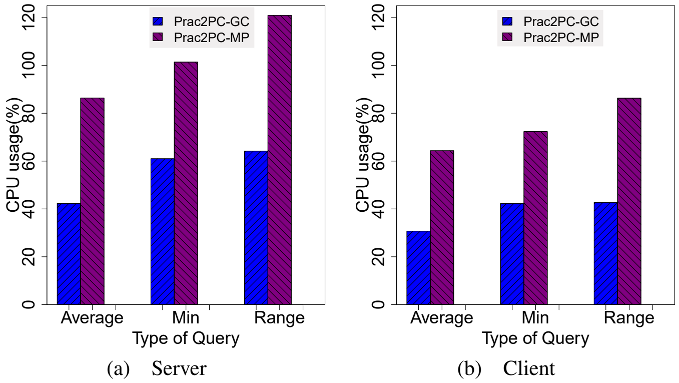
As described earlier, each client encodes her location range in the υ vector, the size of which is equal to the number of squares in the obfuscation rectangle. Squares of interest are set to bit value 1, and extra squares in the obfuscation rectangle are set to bit value 0. An increase in the size of υ implies a larger circuit size. Figure 10 (b)-(d) show the circuit generation and evaluation times in seconds on the right and left y-axis, respectively. The x-axis denotes increasing υ lengths, and the different curves denote different Prac2PC versions. Latency (including computation and communication time) shows the following trends – (a) higher values for higher υ lengths and (b) average, minimum, and range query functions having increasing values (c) Prac2PC-GC performing better than Prac2PC-MP in each case. As network conditions might have changed during the course of these experiments, we also record the server-client average ping latencies, which is around 0.56ms-1.71ms.

Query	**Latency(s)**			**CPU Usage (%)**
	υ			
	10	50	100	
Average	0.57	1.50	2.67	10.27
Min	3.56	16.02	31.18	11
Range	6.03	29.38	57.5	12.3

TABLE V: Latency and CPU utilization for Android as client with ping latency of 20ms $\sim$ 80ms

In addition to latency trends that increase based on query function complexity and υ length, it is also important to look at the absolute latency values. As described above, the x-axis in Figure 10(b) denotes the υ lengths for the 7500 trips selected from the Porto dataset. The left y-axis in Figure 10(b) denotes the percentage frequency of each υ length in the dataset. Based on the x-axis and the left y-axis, we see that most υ vectors are smaller than 100 grid cells. For the vectors less than 100 grid cells predominant in the dataset, latencies for any query (average, max, or range) vary between a few milliseconds to seconds. For longer values of υ, the average query takes at most 0.5 seconds, the max query at most 3 seconds, and the range query at most 6 seconds for Prac2PC-GC.

Table V shows the latency trends for different queries and υ on Samsung Galaxy M21 as the client. The trend is similar to a laptop client. However, the absolute values are higher due to computational limitations and higher ping latencies than the laptop client experiments. Average CPU usage is given



(a) Server

(b) Client

Fig. 11: Average CPU Usage for different query

in Figure 11 (for server and laptop client and Table V (for Android phone client). The low CPU usage value for any version of Prac2PC on the 40 core server guarantees that scaling to many clients will not be an issue for a fleet company. Similarly, the client CPU usage is quite low. To increase CPU utilization on the client and reduce query latency, we have implemented a multi-threaded version of Prac2PC-GC for X86 (details in Appendix D). We will extend this implementation to ARM in the future to further reduce query latencies on mobile devices, which are also quad-core or octa-core now and can take advantage of thread-based parallelism.

Prac2PC query times are very practical, as most commuters with real-time requirements will ask the average query with small to moderate trajectory lengths and get the response within sub-second for laptop users to a few seconds for mobile users. Environmental analysts asking the more varied minimum or range queries using a laptop can wait up to a few seconds, even with very long query lengths covering large city areas.

In addition to Prac2PC's low runtime overhead, another interesting takeaway here is GC outperforming MP implementation. MP is a very recent and exciting cryptographic research domain [25] and we expected to see a huge performance benefit by implementing Prac2PC using the state-of-the-art ABY framework. However, we get far better performance with GC protocol! One reason is recent optimization in GC (pipelining, half gate optimizations). Mixed Protocols have high conversion overheads, e.g., B to Y. This shows the importance of implementation and empirical analysis-based system design, which brings out surprising results.

C. Prac2PC Bandwidth

We capture the data packets transferred over a socket connection from the client to the server using tcpdump. Figure 12 plots the data transferred for different queries and υ lengths for Prac2PC-GC and Prac2PC-MP. In Prac2PC-GC, the number of bytes the server receives is equal, for any query type, for a particular υ length, too negligible to be visible (lower part of left bars in Figure 12). It is due to the same υ vector for all the queries for a given trip. Thus, the client sends the same amount of data in OT to get garbled input in GC. While in Prac2PC-MP, the input is divided into Boolean and Yao's shares, the Boolean shares are shared using XOR-based secret sharing, and Yao share is shared using OT. Range query with two exponential operations per <LAT,LON> pair has around twice the amount of data transferred for Min query (requires only one exponential per <LAT,LON> pair). Average having the simplest and smallest circuit requires the least bandwidth. As the size of the client's input increases, the data transfer increases, which can be seen across all the queries and for all the protocol. It gives us an estimate of the network bandwidth requirement for a particular υ length and query type. One thing to note is that the data transfer between the server and laptop as the client; and between the server and mobile phone as the client is the same.

Commuters would typically query average values using smartphones with network bandwidth constraints. Environmentalists might query all different statistics, possibly using laptops with wired connections, and care less about bandwidth. Prac2PC-GC shows the most practical bandwidth requirement (few KB to MB) irrespective of query size and type.

D. Prac2PC-Malicious

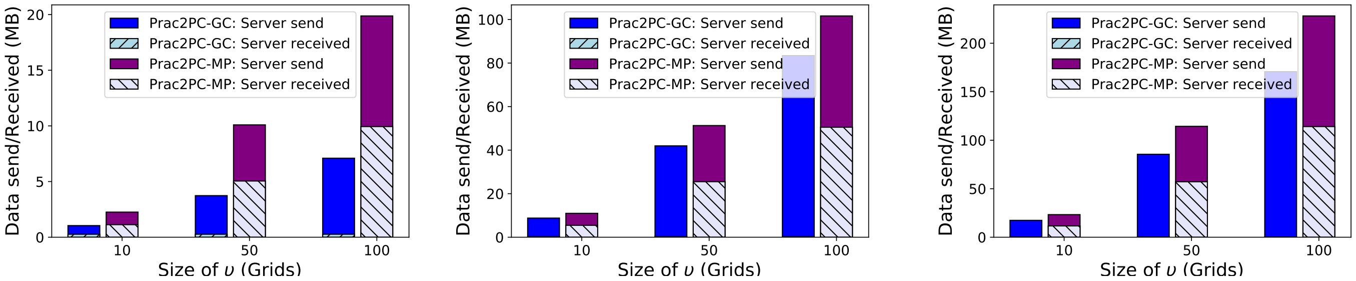
Rival business companies posing as clients will follow the Prac2PC protocol correctly while querying a server (semi-honest threat model) is too optimistic to assume. Hence it is vital to evaluate Prac2PC-Malicious, as described in Figure 9. Figure 13(a) shows the total latency for different queries and different υ. Figure 13(b) shows the online phase latency.

Based on the total latency values, along the right y-axis in Figure 13(a), it is clear that Prac2PC-Malicious indeed needs a two-phase protocol. Preprocessing will be too slow when a client needs an answer from the server in real-time. Preprocessing also has bandwidth requirements in MBs. Thus, it needs to run as a background operation whenever the client device is connected to a wired or WiFi network and keep the client application ready with preprocessed values to be used in the online phase of the next query. The subsequent online phase will then require a few KBs and milliseconds, for most practical υ sizes, as seen from the right y-axis in Figure 13(b).

Preprocessing phase needs only the *Query_Type* and the υ size, which the client app can choose offline based on her interests and obfuscation rectangle. When her trajectory becomes known, she can input the actual 1 and 0 bits in the υ vector in the online phase. This real-time input will not change the Prac2PC circuit computed in the preprocessing phase. The server needs to store the preprocessed values to be used in the online phase, with its most recent $< p, s^2 >$ values for the relevant obfuscation rectangle input during the online phase. So there needs to be a client record at the server for the two-phase Prac2PC-Malicious protocol to maintain preprocessing state. For further optimizations, the same preprocessing can be amortized over many online queries – in case the client repeatedly uses the same *Query_Type* and the υ size. For example, an ordinary citizen checks average travel time or pollution for her work-home-work trajectories every day, or an environmental analyst asks bulk queries for pollution levels in different areas of a city, for many of which *Query_Type* and the υ sizes remain the same. The application can purge the preprocessed data after an appropriate timeout.

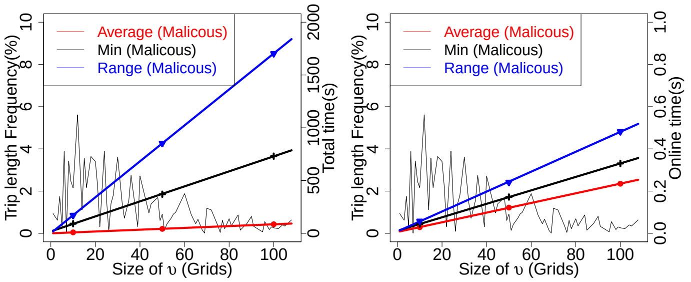
E. Server Privacy vs. Client Utility

Finally, we evaluate the effect of adding Laplace noise to protect against smartly crafted queries. The privacy budget parameter ϵ balances the privacy-utility trade-off. Figure 14



(a) Average (b) Minimum (c) Range

Fig. 12: Data transfer between server and client for different query types and Prac2PC versions. Here, the number of bytes server received in Prac2PC-GC is negligible enough to be visible in plots.



(a) Total Time (b) Online Time

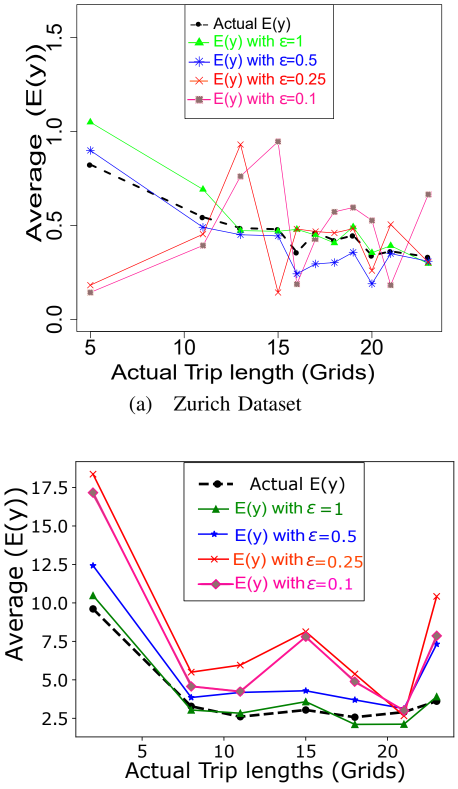
Fig. 13: Query latency for Prac2PC-Malicious

(a) and (b) show the effect of choosing different ϵ on the output of the error propagation function of the average query. As ϵ increases, the utility increases (line becomes closer to the *actual* line), with highest value at $\epsilon = 1$. However, the smaller the ϵ, the better the privacy. As ϵ becomes smaller, more random noise is added to $E(y)$ in Eq 16. $E(y)$ represents the output of the statistical query function given to the client in an end-to-end system. The selection of ϵ to optimally balance this privacy-utility trade-off depends on the dataset. It can be seen from Figure 14 that the best choice for the values of ϵ, which balance privacy and utility, are 0.5 and 1 for Zurich and Delhi, respectively. These provide ϵ-DP guarantees with a small deviation from the actual output.

XI. End-to-End System with GPR and Prac2PC

We implement an end-to-end system to demonstrate a use case that Prac2PC can support. We deploy 40 stationary PMS7003 sensors in a university campus (Figure 15(c)), which measure PM 2.5 and PM 10 values and send the measurements every 30 seconds to a university server over Wi-Fi. The server computes spatio-temporal interpolated values for the grid spanning the whole campus using GPR (Figure 15(b)). Campus residents, if they, input source and destination in our Android application, can see color-coded route choices based on average PM 2.5 values for each route, received through Prac2PC-GC secure computations (Figure 15(a)).

The Android app takes in source and destination from the user and calls Google Map API to get route alternatives. Each route, augmented with obfuscation rectangle, is then represented in-terms of campus grid cells and passed to Prac2PC-GC (C++ library ported to ARM by us) through JNI. The C++ library communicates with the campus PM server over WiFi or 4G and gets the average PM2.5 values for the different routes through the Prac2PC-GC protocol. The averages are passed back from C++ to the Java code, which then displays the color-coded routes based on PM2.5 values.

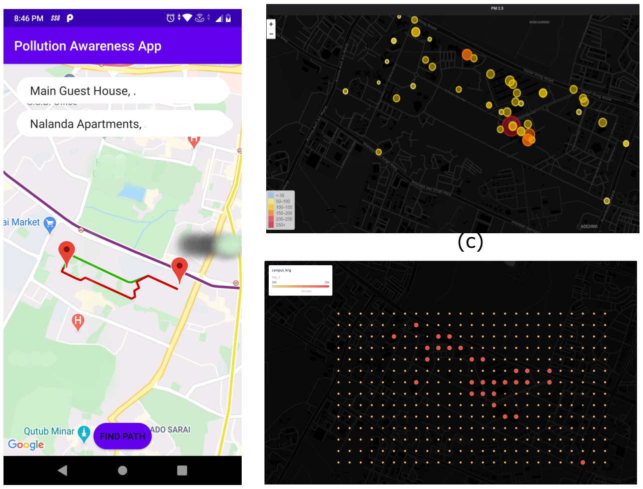


(a) Zurich Dataset

(b) Delhi Dataset

Fig. 14: Privacy-utility trade-off

The median time taken for querying and rendering two paths having 150-190 elements each in the υ vector on 100m x 100 m grid cell are a few seconds on WiFi and 4G. We also monitor the average power and energy consumption for the heaviest range query for $\upsilon = 100$ using Battery Historian and Android Studio, respectively. Figure 16 shows the energy profile monitored using Android Studio; this accounts for about 0.24% of the total device power usage. Based on low CPU



(c)

(a)

(b)

Fig. 15: Privacy-aware Pollution Route App

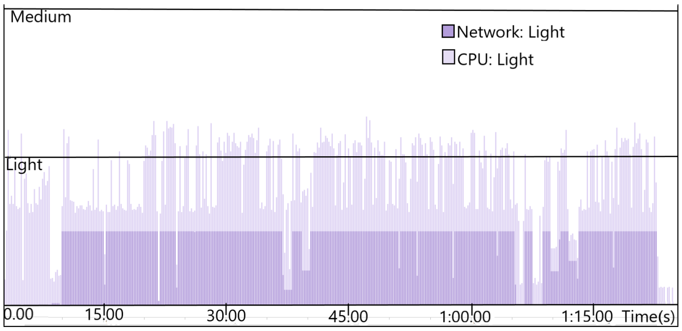


Fig. 16: Energy profile monitored using Android Studio for range query with $v = 100$. Both network and CPU energy usage profiles are low.

and power usage, we can optimize these latencies further by carefully separating the preprocessing and online phases; and extending our ARM library with multi-threading, which are shown to give good results on a laptop (results for multi-threading is shown in Appendix D). This application can be extended to city-scale when pollution measurements scale up with private fleet companies, as the private companies become more confident about location privacy with Prac2PC like systems.

XII. Conclusion and Future Work

Private companies with large fleets are promising partners for vehicle-mounted scalable sensing. This paper presents a privacy-vs-utility trade-off study for fleet companies and sensor data clients. We first use GPR to hide locations without vehicles. The GPR output still leaks fleet location information. We then use 2PC to answer queries with the GPR output at low latency and bandwidth for mobile clients, preserving client privacy, fleet privacy at perfect data utility. We additionally use DP on the 2PC results to protect against smartly crafted 2PC queries. We empirically motivate 2PC, showing only DP after GPR cannot balance the fleet privacy vs. data utility trade-off. Our pollution-aware Android route app shows a glimpse of how scalable data sharing can aid citizens to make more informed choices in reducing personal pollution exposure.

In the future, we will further optimize smartphone query latencies porting our multi-threaded Prac2PC implementation for X86 (Appendix D) to ARM. Secondly, we will add support for multiple private fleet companies [1] to collaborate and publish sensed data. Different fleet companies can have geographical coverage in different parts of a city at different times of the day, and interpolating using sensor data from many such companies will invariably improve the prediction for unseen locations and times. Privacy-Preserving Machine Learning (PPML) methods [59], [60] can be explored in this context, with the machine learning problem being spatio-temporal interpolation (Gaussian Process, Graph Convolutional Networks, etc.). We will also explore extending our DP protocol under the pay-as-you-go scenario for a stronger privacy guarantee [61], [62], [63].

Acknowledgment

We would like to thank all the anonymous reviewers for their valuable comments; it allowed us to immensely improve the paper. The first author would also like to thank MeitY, Govt. of India, to provide support under Visvesvaraya PhD Scheme.

References

[1] E. Times, “Ola partners with microsoft research to study street-level air quality in delhi-ncr,” 2019. [Online]. Available: https://economictimes.indiatimes.com/news/politics-and-nation/ola-partners-with-microsoft-research-to-study-street-level-air-quality-in-delhi-ncr/articleshow/72113952.cms

[2] ——, “Millennial mindset of using ola, uber adversely affecting auto sector, says sitharaman,” 2019. [Online]. Available: https://economictimes.indiatimes.com/industry/auto/auto-news/millennial-mindset-of-using-radio-taxi-adversely-affecting-auto-sector-says-sitharaman/articleshow/71067237.cms

[3] C. Business, “The recession in global car sales shows no sign of ending,” 2020. [Online]. Available: https://edition.cnn.com/2020/01/20/business/global-auto-recession/index.html

[4] W. Dong, V. Dave, L. Qiu, and Y. Zhang, “Secure friend discovery in mobile social networks,” in *2011 Proceedings IEEE INFOCOM*, 2011.

[5] E. Pagnin, G. Gunnarsson, P. Talebi, C. Orlandi, and A. Sabelfeld, “Toppool: Time-aware optimized privacy-preserving ridesharing,” *Proceedings on Privacy Enhancing Technologies*, 2019.

[6] H. Jin and P. Papadimitratos, “Resilient privacy protection for location-based services through decentralization,” *ACM Transactions on Privacy and Security (TOPS)*, vol. 22, no. 4, pp. 1–36, 2019.

[7] S. T. Peddinti and N. Saxena, “On the limitations of query obfuscation techniques for location privacy,” in *Proceedings of the 13th international conference on Ubiquitous computing*, 2011, pp. 187–196.

[8] CNBC, “Indian uber rival ola to launch in london in the coming weeks,” 2019. [Online]. Available: https://www.cnbc.com/2019/11/26/ola-india-uber-rival-backed-by-softbank-prepares-london-launch.html

[9] L. Mint, “Uber vs ola: The battle for dominance in india’s taxi market,” 2016. [Online]. Available: https://www.livemint.com/Companies/okLbTyf5OtqKnO1roYBAeP/Uber-vs-Ola-the-battle-for-dominance-in-Indias-cab-market.html

[10] X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, and D. Srivastava, “Dpt: differentially private trajectory synthesis using hierarchical reference systems,” *Proceedings of the VLDB Endowment*, vol. 8, no. 11, pp. 1154–1165, 2015.

[11] M. Li, L. Zhu, Z. Zhang, and R. Xu, “Achieving differential privacy of trajectory data publishing in participatory sensing,” *Information Sciences*, vol. 400, pp. 1–13, 2017.

[12] N. Li, M. Lyu, D. Su, and W. Yang, “Differential privacy: From theory to practice,” *Synthesis Lectures on Information Security, Privacy, & Trust*, vol. 8, no. 4, pp. 1–138, 2016.

[13] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 1986, pp. 162–167.

[14] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[15] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols." in *SODA*, vol. 1, 2001, pp. 448–457.

[16] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer and extensions for faster secure computation," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 535–548.

[17] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free xor gates and applications," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2008, pp. 486–498.

[18] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 220–250.

[19] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits." in *USENIX Security Symposium*, vol. 201, no. 1, 2011, pp. 331–335.

[20] B. Kreuter, A. Shelat, and C.-H. Shen, "Billion-gate secure computation with malicious adversaries," in *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)*, 2012, pp. 285–300.

[21] X. Wang, S. Ranellucci, and J. Katz, "Authenticated garbling and efficient maliciously secure two-party computation," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 21–37.

[22] M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, "Secure evaluation of private linear branching programs with medical applications," in *European Symposium on Research in Computer Security*. Springer, 2009, pp. 424–439.

[23] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 334–348.

[24] P. Pullonen and S. Siim, "Combining secret sharing and garbled circuits for efficient private ieee 754 floating-point computations," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 172–183.

[25] D. Demmler, T. Schneider, and M. Zohner, "Aby-a framework for efficient mixed-protocol secure two-party computation." in *NDSS*, 2015.

[26] J. Hua, Y. Gao, and S. Zhong, "Differentially private publication of general time-serial trajectory data," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 549–557.

[27] G. Acs and C. Castelluccia, "A case study: Privacy preserving release of spatio-temporal density in paris," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 1679–1688.

[28] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 901–914.

[29] H. Kido, Y. Yanagisawa, and T. Satoh, "An anonymous communication technique using dummies for location-based services," in *ICPS'05. Proceedings. International Conference on Pervasive Services, 2005.* IEEE, 2005, pp. 88–97.

[30] S. Narain and G. Noubir, "Mitigating location privacy attacks on mobile devices using dynamic app sandboxing," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 2, pp. 66–87, 2019.

[31] C. Vaas, M. Khodaei, P. Papadimitratos, and I. Martinovic, "Nowhere to hide? mix-zones for private pseudonym change using chaff vehicles," in *2018 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2018, pp. 1–8.

[32] L. Wang, D. Zhang, D. Yang, B. Y. Lim, and X. Ma, "Differential location privacy for sparse mobile crowdsensing," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016.

[33] S. Stirbys, O. A. Nabah, P. Hallgren, and A. Sabelfeld, "Privacy-preserving location-proximity for mobile apps," in *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 2017.

[34] P. Hallgren, C. Orlandi, and A. Sabelfeld, "Privatepool: Privacy-preserving ridesharing," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, 2017.

[35] Y. Huang, D. Evans, and J. Katz, "Private set intersection: Are garbled circuits better than custom protocols?" in *NDSS*, 2012.

[36] P. Aditya, R. Sen, P. Druschel, S. Joon Oh, R. Benenson, M. Fritz, B. Schiele, B. Bhattacharjee, and T. T. Wu, "I-pic: A platform for privacy-compliant image capture," in *Proceedings of the 14th annual international conference on mobile systems, applications, and services*, 2016, pp. 235–248.

[37] B. Mood, L. Letaw, and K. Butler, "Memory-efficient garbled circuit generation for mobile devices," in *International Conference on Financial Cryptography and Data Security*. Springer, 2012, pp. 254–268.

[38] S. G. Choi, K.-W. Hwang, J. Katz, T. Malkin, and D. Rubenstein, "Secure multi-party computation of boolean circuits with applications to privacy in on-line marketplaces," in *Cryptographers' Track at the RSA Conference*. Springer, 2012, pp. 416–432.

[39] H. Carter, C. Lever, and P. Traynor, "Whitewash: Outsourcing garbled circuit generation for mobile devices," in *Proceedings of the 30th Annual Computer Security Applications Conference*, 2014, pp. 266–275.

[40] S. T. Peddinti, A. Dsouza, and N. Saxena, "Cover locations: Availing location-based services without revealing the location," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, 2011, pp. 143–152.

[41] Q. Kong, R. Lu, M. Ma, and H. Bao, "Achieve location privacy-preserving range query in vehicular sensing," *Sensors*, vol. 17, no. 8, p. 1829, 2017.

[42] R. K. Balan, K. X. Nguyen, and L. Jiang, "Real-time trip information service for a large taxi fleet," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 2011, pp. 99–112.

[43] R. Schlegel, C.-Y. Chow, Q. Huang, and D. S. Wong, "User-defined privacy grid system for continuous location-based services," *IEEE Transactions on Mobile Computing*, vol. 14, no. 10, pp. 2158–2172, 2015.

[44] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *arXiv preprint arXiv:1706.02216*, 2017.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015.

[46] M. Smith, M. Álvarez, M. Zwiessele, and N. D. Lawrence, "Differentially private regression with gaussian processes," in *International Conference on Artificial Intelligence and Statistics*, 2018, pp. 1195–1203.

[47] N. Holohan, S. Antonatos, S. Braghin, and P. Mac Aonghusa, "The bounded laplace mechanism in differential privacy," *Journal of Privacy and Confidentiality*, vol. 10, no. 1, Dec. 2019. [Online]. Available: https://journalprivacyconfidentiality.org/index.php/jpc/article/view/715

[48] OpenSense, "Community-based sensing using wireless sensor network technology to monitor air pollution," https://gitlab.ethz.ch/tec/public/opensense/.

[49] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 316–324.

[50] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, 2010, pp. 99–108.

[51] H. Wang, "Air pollution and meteorological data in Beijing 2016-2017," 2019. [Online]. Available: https://doi.org/10.7910/DVN/RGWV8X

[52] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*. Springer, 2006, pp. 265–284.

[53] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[54] H. H. Ku *et al.*, "Notes on the use of propagation of error formulas," *Journal of Research of the National Bureau of Standards*, vol. 70, no. 4, pp. 263–273, 1966.

[55] G. C. Calafiore, S. Gaubert, and C. Possieri, “A universal approximation result for difference of log-sum-exp neural networks,” *IEEE transactions on neural networks and learning systems*, vol. 31, no. 12, pp. 5603–5612, 2020.

[56] X. Wang, A. J. Malozemoff, and J. Katz, “EMP-toolkit: Efficient MultiParty computation toolkit,” https://github.com/emp-toolkit, 2016.

[57] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, “Predicting taxi–passenger demand using streaming data,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1393–1402, 2013.

[58] J. J. Li, B. Faltings, O. Saukh, D. Hasenfratz, and J. Beutel, “Sensing the air we breathe—the opensense zurich dataset,” in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[59] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, “Crypten: Secure multi-party computation meets machine learning,” 2021.

[60] S. Tan, B. Knott, Y. Tian, and D. J. Wu, “Cryptgpu: Fast privacy-preserving machine learning on the gpu,” 2021.

[61] A. Bkakria, A. Tasidou, N. Cuppens-Boulahia, F. Cuppens, F. Bouattour, and F. B. Fredj, “Optimal distribution of privacy budget in differential privacy,” in *International Conference on Risks and Security of Internet and Systems*. Springer, 2018, pp. 222–236.

[62] D. Durfee and R. M. Rogers, “Practical differentially private top-k selection with pay-what-you-get composition,” in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[63] R. M. Rogers, A. Roth, J. Ullman, and S. Vadhan, “Privacy odometers and filters: Pay-as-you-go composition,” in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[64] GPy, “GPy: A gaussian process framework in python,” http://github.com/SheffieldML/GPy, since 2012.

Appendix

A. Accuracy of Posterior Mean μ_p

For measuring the accuracy of Gaussian Process Regression, we use 5-Fold cross-validation with five random restarts in GPy [64]. In each fold, out of the different random restarts, the objective function with a minimum value is selected. The accuracy of GPR is measured in terms of root mean squared error (RMSE) of the predicted pollution value on the Beijing dataset. Out of the different kernels available in GPy, we tested three kernels' performance, i.e., RBF, Matern32, Matern52. Matern32 kernel gives out the best RMSE among the three for the dataset (Figure 17). This is the kernel we use in subsequent analyses.

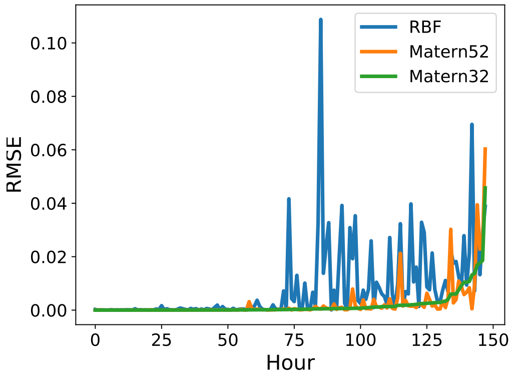
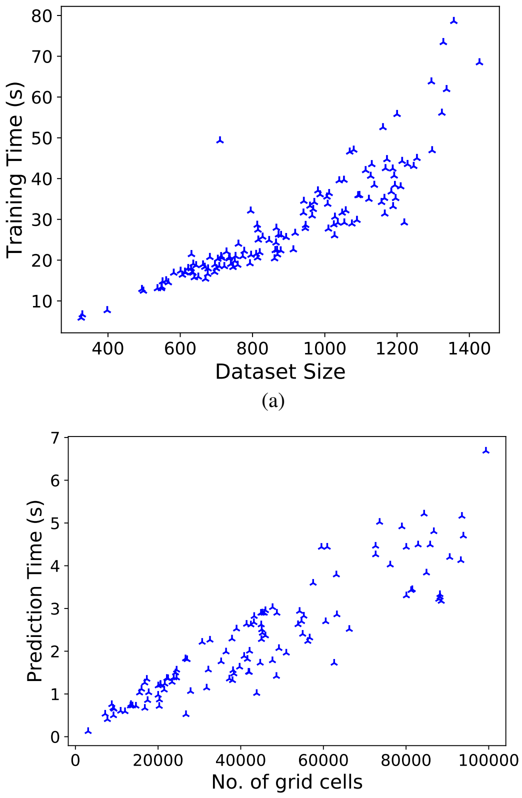


Fig. 17: RMSE of Posterior mean (μ_p) vs. Different Kernels



(a)

(b)

Fig. 18: (a) Time taken for training different number of sample locations sensed by the 24 cabs in 24 hours for 7 days.
(b) Prediction Time vs Number of grid cells

B. Computational Complexity of GPR

We evaluate the cost of training the GP model. When GPS values are sampled by the cabs every minute and a window of one hour is selected for training, the maximum and the minimum number of GPS training samples come out to be around 1400 and 300, respectively. The maximum number of GPS locations are sensed in the evening when the movement of each cab is high. However, at night, we have fewer samples as fewer people are commuting. The training for 1400 samples takes around 1.2min, as can be seen from Figure 18(a). As the number of training samples increases, the training time increases by order of O(N^3) as matrix inversion required in GPR is O(N^3).
After fitting the Gaussian Process on the training set, we measure the time taken to predict each grid cell centroid. We create a bounding box around the trips taken by all the drivers in a given hour and then select all the cells which are inside and intersect the bounding box for prediction.

The number of grid cells varies with grid cell size. As the size of grid cells increases, the number of grid cells decreases. Hence, the time taken to predict the grid also decreases. Figure 18(b) shows the time taken to compute $<\mu_p, \sigma_p^2>$ for different number of grid cells. The plot follows a somewhat linear trend as we predict each centroid sequentially. This time can easily be reduced by parallelizing the prediction process for the centroids using multiple processor cores, as the prediction of these centroids is independent of each other.

C. Computation Time for DP

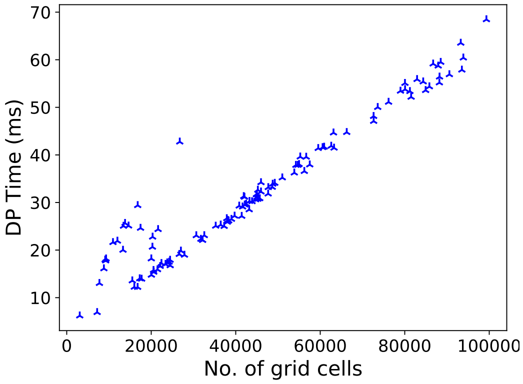


Fig. 19: DP time vs Number of Grid cells

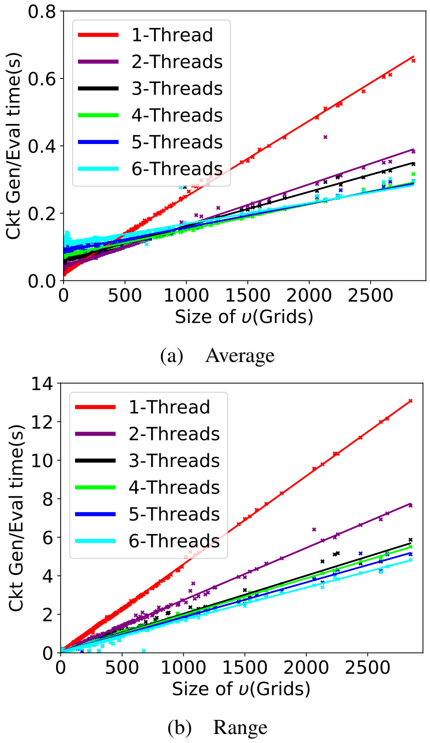
Figure 19 shows that the cost of applying DP (around 65 ms) is negligible in comparison to prediction time (around 7s). Similar to prediction time, the DP time also scales linearly with respect to the number of grid cells. It is also easily parallelizable. Figure 19 has some outliers at the starting of the curve. This could be either due to running the experiments on the shared server (waiting for CPU cycles) or because of using the bounding Laplace mechanism. The bounding mechanism employs rejection sampling when noise is added to σ_p^2. So unless and until the σ_p^2 becomes positive and within a bound, we reject the addition of noise to the sample σ_p^2, thereby increasing runtime.

D. Prac2PC-threaded vs Sequential:

Laptops and mobile platforms come with multi-core CPU processors these days, which our client software should exploit for efficiency. We have implemented Prac2PC-threaded using pthreads. For server-client communications, we open as many ports as there are threads. We divide the input of client and server into equal parts and assign them to separate threads on both client and server, respectively. In the threaded implementation, the set-thresholding circuit will always output 0 as counting of 1's in υ vector is distributed over the multiple threads. To prevent this, we define a local threshold such that each thread must get at least a local threshold number of 1's in its corresponding part of υ. The client must also set at least a global threshold of 1's in the υ vector. We ensure the synchronization of communication between client and server using mutex locks. The number of threads is decided by OS configuration.

We run an experiment omitting network latency to see the actual overhead/ benefit of multi-threaded implementation. We run server and client on the same machine.

Figure 20(a) and (b) show the latency when the client and server are running on the same machine, which has six cores. The improvement of latency is significant with multi-threading when one has a large input vector. This is the direct effect of dividing the workload among the threads. Management of threads in a small υ vector overshadows the benefit of using it in the Average query. The υ vector depends on the size of the individual grid cell (500m X 500m chosen for the Porto dataset) and the obfuscation rectangle selected for a trip. If the client is more conscious of its privacy, it can choose a bigger obfuscation rectangle and hence a large υ vector. In such cases, the client can set a threshold on the size of υ to select between sequential and parallel configurations. This will smartly minimize the latency for any type of query. The server can also use the Prac2PC-threaded in the pollution-aware route-based apps(Section XI), where multiple color-coded paths are returned. In this case, the server can run one thread per path to optimize the latency.



(a) Average

(b) Range

Fig. 20: Query latency for Prac2PC-threaded