An Approach to Privacy Preserving Queries for Spatio-Temporal Data

John Sullivan

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-69
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-69.html

May 17, 2019

Copyright © 2019, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

An Approach to Privacy Preserving Queries for Spatio-Temporal Data

Jack Sullivan

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee

David Culler
Research Advisor

(Date)

★ ★ ★ ★ ★ ★ ★

Randy Katz
Second Reader

(Date)

Abstract

The increasing amount of spatio-temporal data from mobile devices and individual participation can help drive domains such as environmental monitoring, health care, and urban planning. Spatio-temporal data can be individually-identifiable, which has lead to attempts to provide differential privacy for such data. But such data breaks differential privacy guarantees due to its inherent correlation. Previous works have ensured sufficient privacy through synthetic data release and low privacy budgets. Unfortunately the lack of aggregation options for the data release approach weakens analyst utility and protection of users and their data in the interactive query setting. We propose acceptable error (AEs) and range count (RCs) query algorithms. RCs ensure a stronger level of differential privacy than AEs, but typically use a greater privacy budget. We additionally propose two query data access mechanisms: data access policies (DAPs) and data access degrees (DADs) that grant or limit data access and specify the allowed amount of data leakage respectively. We propose a privacy-preserving system that implements these aggregation options to securely store and aggregate sensitive user trajectory data. Our goal is to incentivize users to participate in spatio-temporal aggregations while safely providing valuable data to analysts spanning a multitude of domains.

Contents

1 Introduction

The increasing computation and storage capabilities of mobile devices are enabling the rise of spatio-temporal data collection. This data provides opportunities to make sophisticated inferences about not only people, but also their surroundings, and thus can help improve people's health and lives. While this data can provide immense benefits for users, the release of this user sensitive data to analysts and the public is concerning from a privacy perspective. Spatio-temporal origin-destination pairs and their associated trajectories are known to be sensitive at any granularity. For example, only four spatio-temporal points are enough to uniquely identify a user 95% of the times in a dataset of one and a half million users [6]. Our report setting focuses on how to safely aggregate fine-grained spatio-temporal trajectory data which is the most challenging to preserve individual privacy with, but has potential to provide the most granular queries and overall benefit.

Data privacy mechanisms are used to protect individuals from being identifiable from queries that use their sensitive spatio-temporal data. Differential privacy is a recent theoretical approach to protect individual privacy. Spatio-temporal data can be heavily correlated and breaks the data independence assumption of differential privacy, leading to differential privacy leaking more sensitive information than in the independent data setting. Privacy can be preserved with a low enough privacy budget (i.e., total allowed data leakage), which makes maximizing the information gain from a privacy budget important in this domain.

Spatio-temporal related work has largely focused on the continual publishing of private spatio-temporal data (synthetic data release) as opposed to allowing for interactive, one at a time queries that return noisy results from the raw data. Publishing privacy-injected data to the public safely enables any number of queries and algorithms. But this published private data is at a fixed granularity and typically does not support fine-grained (street intersection level) queries. There is also a lack of accuracy as, in terms of privacy budget, either very little is allocated for each release, or each window of releases is bounded by a privacy budget, which breaks correlation assumptions. The problem we address is insufficient aggregation options to maintain both privacy and utility in the spatio-temporal interactive query setting. The second problem approached is the absence of a privacy-preserving query system that ensures secure user data storage and secure aggregation.

This report proposes aggregation query operators and data access mechanisms to empower analysts and users respectively. The query operators proposed are a relative and an absolute count query that are named acceptable error count queries (AEs) and range count queries (RCs). AEs return a count result within a specified offset relative to the true query answer with a specified accuracy. RCs answers whether the count result is within two specified absolute counts with a specified accuracy. The analytical results in Section 3.1.3 show that RCs ensure a stronger level of differential privacy, but typically use more privacy budget to ensure the same accuracy as AEs. AEs are also shown to use privacy budget less than or equal to the privacy budget used by RCs when wanting a scalar count (AEs are optimal in this case) and vice versa when wanting a range.

The data access mechanisms are policies (DAPs) that specify which queries are allowed to access certain user data. Also we propose data access degrees (DADs) that specify the allowed amount of user data leakage (e.g., privacy budget) a query has. Analysts should have the role to specify the query, including the query operator, while users should be able to control how their personal data is used, given a query.

In addition to proposing aggregation options for analysts and users, this report addresses securely storing and aggregating spatio-temporal user data. A privacy-preserving query system is implemented to deploy queries with their associated query operators and data access mechanisms. This system relies on secure containers (enclaves) to prevent the leakage of sensitive user data through data encryption and remote attestation. The user add protocol securely stores sensitive user data and the aggregation protocol securely computes queries. Scalability and correctness experiments are conducted on our implemented system. Our scalability results show that query aggregate response time increases more with number of users than with data. Correctness experiments are presented that confirms that the implemented aggregation options function as we would expect and demonstrate how these options operate in our system.

2 Background

2.1 Privacy Mechanisms

A privacy mechanism in a query setting is a transformation to the dataset or query result that ensures some privacy guarantees. All privacy mechanisms have trade-off between privacy and utility. Applying an effective privacy mechanism requires a loss in utility, which is known as "no free lunch" [13]. Having maximally accurate private data is impossible. Thus, each mechanism has differing privacy guarantees and utilities.

The two most common privacy models used are syntactic and semantic. Syntactic models focus on syntactic requirements of the query without any further guarantees of any sensitive information adversaries can learn about individuals. Syntactic privacy models, such as k-anonymity which requires that each of the released records be indistinguishable from at least $k - 1$ other records when projected on the public attributes, rely on anonymity through groups. But groups can still leak a person's sensitive information, if the group exhibits the same or highly correlated behaviors. A spatio-temporal example of a k-anonymity limitation is if all k trajectories visited the same granular place (e.g., a street block). This means that the person also traveled to that sensitive place. We instead turn to semantic models that ensure user-level privacy, such as differential privacy.

Differential Privacy

The concept of differential privacy was first introduced by C. Dwork [7] and ensures that an individual is not at increased risk of privacy when they participate in a statistical database that is being queried. Essentially, a differentially private query result should be approximately the same whether any participant is included or excluded from the query. The traditional differential privacy setting involves a trusted aggregator adding statistical noise to the query result before releasing it to the data analyst.

We consider ϵ-differential privacy. The parameter ϵ is defined to be a positive real number to control the privacy level, such that larger values of ϵ will lead to larger privacy and data leakage. Let D_1 and D_2 be neighboring databases that differ by a single database row, i.e., D_1 and D_2 differ

by one user if each row represents a user. Let A denote a randomized algorithm over the databases and $Range(A)$ denote the space of possible outputs for A. The algorithm A is said to provide ϵ-differential privacy if for all datasets D_1 and D_2 and for all subsets S of $Range(A)$:

$$Pr[A(D_1) \in S] \leq e^{\epsilon} Pr[A(D_2) \in S]$$

Differential privacy in our setting ensures that a single user and the collection of their trips and trajectories does not significantly impact any query result.

However, adding statistical noise to each query result is not sufficient in ensuring differential privacy if an analyst repeats a query or if the queries are different but correlated. Averaging these results will lead to a more precise query answer that will eventually reveal individual users. Privacy budgets, the ϵ in the definition above, are introduced to represent the maximum privacy data leakage. We can think of each query as a privacy expense, which incurs an incremental privacy loss. Once the privacy budget is depleted, access to that data is blocked. Privacy budgets and users being able to modify them is discussed in Section 3.2.

Query Noise Procedure

The most simple and well known way to ensure ϵ-differential privacy is by adding statistical noise, such as Laplace noise [2], to the final query result. The Laplace Distribution (centered at 0) with scale b is the distribution with probability density function:

$$Lap(x|b) = \frac{1}{2b} exp(-\frac{|x|}{b})$$

To ensure ϵ-differential privacy, the value for b is $\frac{\triangle f}{\epsilon}$ [2]. $\triangle f$ represents the global sensitivity of the query. Global sensitivity is defined as the maximum query result difference between any two databases that only differ by one element (i.e., an individual being included or not included in a database in our case). The formal definition of global sensitivity is:

$$\triangle f = max||f(D_1) - f(D_2)||_1$$

where the maximum is over all pairs of datasets D_1 and D_2 from the space of all possible databases differing by at most one element, and $||\cdot||_1$ denotes the l_1 norm. For example, the global sensitivity of a counting query is 1, as the overall query result can only change by one whether any person is included or excluded in a query.

Therefore ensuring ϵ-differential privacy involves adding $Lap(\frac{\triangle f}{\epsilon})$ noise to the query result when having a trusted aggregator and independent data. Our setting ensures a trusted aggregator through attestable system components, but it is still necessary to discuss how to handle the spatio-temporal data correlations.

2.2 Spatio-Temporal Data Setting and Problem Definition

Spatio-Temporal Correlations

Most differentially private settings and related works have the implicit assumption about the data is independent (i.e., no correlation between database rows). Recent studies [13] [12] [14] suggest that traditional differential privacy techniques do not ensure sufficient privacy with correlated data.

Roads leading to deterministic location are an instance of temporal correlations that degrades expected differential privacy guarantees. For example, consider a straight road with no exits for 5 miles. Denote location l_1 as the first half of this road and location l_2 as the second half. Assume a person is driving in l_1 in the first time step t_1 and is likely to be in l_2 a few minutes later t_2 (i.e., $\Pr(l_2 \text{ at } t_2 \mid l_1 \text{ at } t_1) = 1$). Thus, given the count at l_1 during t_1, the analyst would know the location of the individual at t_2. An analyst can perform inference due to these spatio-temporal correlations and adding Laplace($\frac{1}{\epsilon}$) to a count guarantees 2ϵ-DP. Extending this to a worse case (for instance, a traffic jam), if individuals stay in the same location for T time steps, then each Laplace($\frac{1}{\epsilon}$) count will guarantee $T\epsilon$-DP. Repeated travel patterns and similar trajectories lead to a long correlation range (e.g., potentially making T very large) that can greatly degrade the privacy of a user's sensitive trip data.

Point and Trajectory Count Queries

With related works covering both spatio-temporal counts and trajectory data, it is important to distinguish between the two by introducing some new terminology and showing how they differ. We introduce two different location definitions: point and trajectory. We denote a *point* as a fixed spatial polygon around a spatial point while a *trajectory* is defined to be a series of successive points. It is important to note that trajectories are a generalization of points (i.e., a point is a trajectory of length 1). A *point count query* is the total unique individual count over a point area over a specified time. A *trajectory count query* is the total unique count of individuals that cover the specified successive spatio-temporal points.

We present an example for each kind of count query. A point clout query example is how many people were on Main Street from 3pm-5pm on November 3rd, 2018? The set of spatio-temporal points is defined by the fixed location point (Main Street) and the specified time (3pm-5pm on November 3rd, 2018). Any user that satisfies any other those spatio-temporal points is counted. A trajectory count query example is how many people on Main Street took a left on First Street from 1pm-2pm on weekends? The set of spatio-temporal points is defined by the two successive points: Main Street and First Street, and the specified time (1pm-2pm on weekends). Any user that satisfies moving from the Main Street point to the State Street point within the time is counted. This intersection trajectory count example requires very fine-grained spatio-temporal data and multiple, successive points to understand how individuals travel through space and time.

Problem Definition

The problem we address is how to provide aggregation options in an interactive spatio-temporal query setting while maintaining user level privacy. Interactive queries allows for optimizing each query to satisfy differentially privacy while providing greater query accuracy. We suggest possible count query algorithms and data access mechanisms that would increase analyst and user utility while maintaining user-defined privacy. To test these options, our goal is to construct a practical, privacy-preserving system that ensures differentially private queries to protect individual user trajectories over (i) sequences of fine-grained user successive spatio-temporal data points (trajectories), (ii) an increasing time range, (iii) interactive, one at a time queries, (iv) multiple query algorithms, and (v) user-defined data access mechanisms.

2.3 Related Work

We split related work into spatio-temporal synthetic data publishing and constructing interactive query systems.

Spatio-temporal Synthetic Data Release

Starting with spatio-temporal interactive differential privacy works, Rastogi and Nath [18] introduce a spectral approach that greatly decreases the noise error for a batch of n correlated queries from $\Theta(n)$ to $\Theta(k)$, where k is the number of Fourier coefficients that can (approximately) reconstruct all the query answers, for limited temporal ranges. Cummings, et al., [5] explore interactive, one at a time queries for growing databases, but do not address data correlations and assumes the data is independent.

Spatio-temporal related work has focused on continual publishing, which injects privacy into the data before publicly releasing it. Acs, et al., [1] publish point count queries for a limited time range and use sampling, clustering nearby locations, and the spectral approach to minimize noise to satisfy sufficient differential privacy [18]. RescueDP [22] focus on publishing point count queries for an infinite time range by ensuring w-event privacy which protects any event sequence occurring within any window of w timestamps. w-event privacy assumes that only ranges of length w are correlated, which can deteriorate privacy over a long, correlated period of time. DP-WHERE [16], DPT [9], and DP-Star [8] focus on publishing coarse synthetic trajectories for infinite time ranges, with somewhat granular location areas ranging from 50m x 50m to 1000m x 1000m and the time sampling rate ranging from 30 seconds to 5 minutes. This report's data contains precise lat-long pairs and frequent sampling rates of 10 seconds.

Distributed Differential Privacy Systems

Our privacy-preserving system aims to mimic the trusted aggregator from traditional differential privacy, but with attestable system components to support any number of differentially private query algorithms. Our approach shares similarities with distributed differentially privacy related work. Rastogi and Nath [18] and Shi, et al., [20] propose distributed differentially privacy aggregation protocols that are dependent on communicating directly with the users. [18] propose the first differentially private aggregation algorithm for distributed time-series data that offers good practical utility without a trusted central server and [20] propose a Diffie-Hellman-based encryption scheme. Both schemes are limited to sum queries, which our system solves with using attestable system components as the trusted aggregator. The schemes also suffer from insufficient user fault tolerance (e.g., user mobility and spotty connectivity) and data storage limitations, as data is stored on mobile devices. Our privacy-preserving system avoids these limitations by deploying the system and data on the cloud.

Practical Privacy-Preserving Systems

There has also been prior work in constructing practical privacy-preserving systems. PrivStats [17] is a system for computing aggregate statistics over location data in a privacy-preserving manner. However, PrivStats relies on the use of an anonymization network and any aggregation functions must be able to run on homomorphically encrypted data. Sampaio, et al., propose a data dissemination platform that supports untrusted infrastructures [19]. However, this data dissemination system does not support persistent user data as aggregate computations are performed on live streaming data from clients. In Prio [4], another privacy-preserving system for the

collection of aggregate statistics, all queries must be performed on Affine-Aggregatable Encodings (AFEs). We avoid this limitation by performing all queries on plaintext values. [15] provides a new communication efficient and privacy-preserving data aggregation protocol but relies on the use of a trusted third party for key setup. Chorus [10] presents the first differentially private practical approach optimized for and only SQL queries and does not apply to flexibly structured data like in our scenario.

2.4 Limited Aggregation Options Insight

For spatio-temporal data publishing related work, there is a lack of query operators and data access options. As seen in this section, low fixed privacy budgets ensure sufficient privacy in this correlated setting. Having multiple query operator options with different differential privacy guarantees would be beneficial for analysts to maximize utility with the limited privacy budget. Also users being able to specify their data access options would enable them and analysts if they choose to increase allowed leakage. Therefore, aggregation options need to be present in the interactive setting to ensure more usable and sufficiently private queries as motivated in this section.

Continual private data publishing has been preferred over interactive queries due to the unlimited query use. Ensuring privacy over an infinite time ranges typically satisfies w-event privacy [11] which protects any event sequence occurring within any window of w timestamps. In other words, the sum of privacy budgets used for each release point for any w successive timestamps cannot exceed a specified privacy budget ϵ. A finer spatio-temporal grained setting would require more total release points, thus leading to even smaller privacy budgets used to release data at these points. Thus, it is difficult for data release to support fine grained queries (e.g., how many cars turned left on State Street) due to unusably high noise. Interactive queries in comparison would have a limited number of queries with potentially higher utility and query flexibility.

The privacy guarantee in our setting is to protect against an analyst from learning individual user trajectories without the user's consent. Recall from Chapter 1 that it is possible to uniquely identify an individual trajectory with as little as four points with high confidence. Protecting the count of any spatio-temporal point, successive sequences of spatio-temporal points, and any combination of spatio-temporal points along a trajectory are necessary in guaranteeing individual privacy. We demonstrate below that we can apply the continual private data publishing setup and parameters to protect individual privacy in our specific interactive setting.

Definitions and Assumptions

The continual private data setup relies on ensuring w-event level privacy that fulfills a privacy budget ϵ. Recall that w-event level privacy ensures that the privacy budget used for any w successive releases is bounded by a privacy budget ϵ. We assume that w is very large (i.e., close to infinity) as spatio-temporal data can be correlated across all of time in the worst case. Privacy budgets in these related works are low ($\epsilon \leq 1$) and fixed. Thus, we assume that users would have a low and fixed non-configurable privacy budget ϵ for the entire spatio-temporal space.

Another important aspect of defining our limited control options scenario is the database layout. We assume a logical database with each row representing an individual and all their trips (and trajectories) stored in their row (user level database). One main assumption made is that users and their trips are independent from other users in the database. This assumption combined with the user database layout with iid rows enables the use of traditional differential privacy to protect users and their individual trajectories. A trip or trajectory level database alternatively could have been used to keep each database row simple and immutable. This is not a natural format for protecting privacy of individuals and the rows would not be iid as individual user trips are highly correlated with each other.

Although our user level database layout helps to evade spatio-temporal correlations on a database level, these correlations will be reflected and present across the results of correlated queries. But standard differential privacy mechanisms can preserve privacy of individual trajectories with sufficiently low privacy budgets and through our user group definitions. Thus, correlated results may be less of an issue. There are two cases in which we need to prove the preservation of privacy for individual trajectories: sufficiently and insufficiently large groups for individual privacy.

Group Definitions

A sufficiently large group for individual privacy (SLGIP) is a group of at least h users that visit the same locations in the same order across the same time range. To produce a formal definition, it is necessary to introduce some preliminary notation. Let U denote the set of all users, L denote the set of all possible disjoint spatial partitioned locations, T denote the set of all discrete times between the starting time (t_s) and the ending time (t_{present}). Additionally, let S denote all the spatio-temporal points where a spatio-temporal point is represented as s_{t_i,l_j} where t_i represents a time in the time range ($t_i \in T$), l_j represents a location in the set of possible locations ($l_j \in L$).

A SLGIP G is defined by set of users $U_G \subseteq U$, a minimum number of required users h, and by a set of spatio-temporal points $S_G \subseteq S$ where all $u_k \in U_G$ visit all spatio-temporal points

$s_{t_i,l_j}, \forall s_{t_i,l_j} \in S_G$. The minimum number of user condition is expressed as $|U_G| \geq h$. An example of a SLGIP with a minimum count threshold of $h = 30$ is a group of 100 individuals on a train.

The definition of an insufficiently large group for individual privacy (ILGIP) is the number of users that are bounded and traveling together is less than the count threshold h. An ILGIP is the compliment of a SLGIP. An example of a ILGIP is one individual driving to work. We assume that both ILGIPs and SLGIPs are independent of all other groups, meaning that each group's trajectory can not be used to learn more about the trajectory of another group.

Protecting ILGIP Individual Trajectory Privacy

Focusing first on ILGIPs, traditional differential privacy noise mechanisms guarantee individual privacy for $h = 1$. Group sizes of 1 combined with the assumption of groups traveling independently and the user-level database layout ensure that standard differential privacy can adequately provide individual privacy. In the case where $h > 1$ and $h \ll n$, where n denotes the number of database rows, these mechanisms should intuitively still provide individual privacy as a very small subset of the rows are correlated. But a formal proof of the extension of traditional differential privacy to $h > 1$ is outside our scope. Additionally, it is important to note that the noise produced by traditional differential privacy will be larger relative to the small true counts because this noise is independent of number of people.

For individual trajectories to be revealed, an analyst must query multiple points along a trajectory with high accuracy and high privacy budget. The most correlated spatio-temporal points are similar to repeated queries as all spatio-temporal points rely on the same privacy budget. Protecting against repeated queries will protect against any series of spatio-temporal points as repeated queries inherently have maximum correlation. Thus, the safeguard against averaging repeated queries is the privacy budget.

As an example, using $\epsilon = 1$ for user privacy budgets allows the analyst to learn an exact count at one spatio-temporal point with 0.63 probability (using $1 - \exp(-\epsilon)$ per query to represent the probability of returning the actual count plus or minus one). When attempting to reconstruct a trajectory, the probability of learning an exact count at all four spatio-temporal points when using $\epsilon = 0.25$ for each is 0.002. Thus, the risk of leaking individual trajectories in ILGIPs can be protected with the proper privacy budget.

Protecting SLGIP Individual Trajectory Privacy

Querying the SLGIP repeatedly and across successive spatio-temporal points using traditional differential privacy noise adding will generate correlated query count results. These correlated

results come from the fact that at least h rows in the user level database are correlated. In theory, although we are able to obtain more precise query answers across correlated queries, no sensitive information can be learned about any individual as all users in the SLGIP travel predictably together. The only hope of learning anything about individuals in this large group setting is when considering when users enter or leave the SLGIP, formally known as a divergence point.

A divergence point is a spatio-temporal point that breaks the bounding of users in a SLGIP and allows any number of users to leave or enter the group. Formally, at spatio-temporal point s_{divpoint}, U_{subset} is randomly repopulated and its cardinality can change: $|U_{\text{subset}}| = c$ where $c \geq 0$. Using our running example, a train station would be a divergence point for individuals in a group as users in a group have the chance to leave the group.

In the worst case, individual privacy loss occurs when only one user diverges from the SLGIP. An individual would need to transition into another SLGIP or an ILGIP which is proved to be safe. To only know that an individual diverged with high confidence, the analyst would have had to conduct two queries with relatively high privacy budgets (one before divergence, one after).

For example, with overall $\epsilon = 1$ as in the ILGIP section, an analyst conducts a query before a divergence point and one after a divergence point each with high $\epsilon = 0.4$. The analyst would have low confidence in obtaining the accurate counts as the probability of getting the exact count for both queries is 0.108. The analyst has also exhausted the majority of the user's overall privacy budget as there is now 0.2 remaining out of 1. To learn that an individual diverged from a group, the analyst has very little user privacy budget left and power to follow the possible branched paths.

Privacy budget ϵ was seen as a double edged sword throughout this section. Privacy is properly enforced in this domain by setting low enough privacy budgets for each correlated range. But privacy budgets also greatly limit the number of queries and their associated accuracies. Our goal is to empower analysts by providing multiple ways to conduct count queries to obtain their desired results, while simultaneously empowering users to control how their personal data is used given a query.

3 Proposed Aggregation Options for Spatio-Temporal Queries

Although individual trajectories are properly protected with the Laplace count query algorithm and small, fixed privacy budgets, an interactive query system allows for flexible aggregation options in order to simultaneously maximize analyst and user utility. Figure 3.1 illustrates the logical query process which begins with an analyst specifying a query and its parameters, the users the data the query accesses, and the query result being calculated and sent to the analyst. We first propose a way to parametrize an interactive spatio-temporal query and two different types of count algorithms: (i) a relative perspective that specifies the acceptable amount of error for a query (acceptable error) or (ii) an absolute perspective that determines whether a count is within a specified range (range count). The next section discusses possible data control methods given the query. Lastly, analyst and user roles are explained; analysts construct the queries and choose the query operator, while users control the data each query accesses.

Query Parameterization

Queries can be parametrized by space, time, algorithm, and query origin. Spatio-temporal related works parametrize queries typically by space, time and algorithm. But the query data can be the bottleneck for the kind of algorithms run. For example, [1] only releases coarse count data, which can limit query granularity. Related works that publish synthetic private trajectories, such as [16], [9], and [8], can allow for more fine query granularities. Interactive queries can support more fine-grained space and time ranges. The trade-off is that continual private data publishing of counts or trajectories enable algorithms that the data can support, while preserving privacy. Providing interactive differential privacy mechanisms for spatio-temporal data has been notoriously difficult and is algorithm dependent.

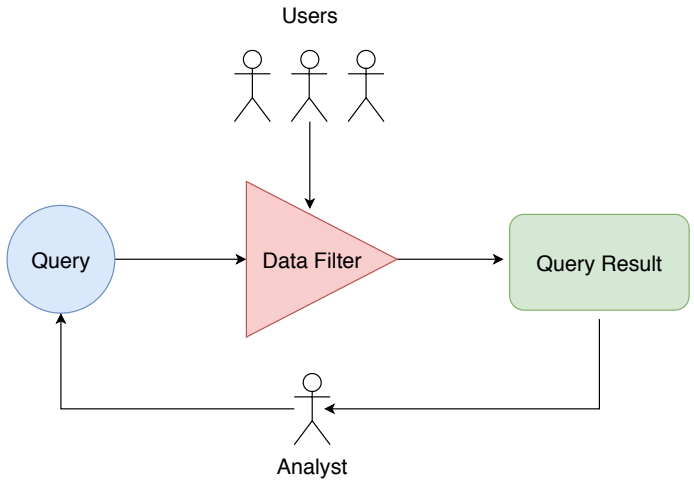


Figure 3.1: Logical Query Pipeline Diagram

3.1 Query Operators

Count queries are among the simplest query types to apply differential privacy, as they require less noise due to their low global sensitivity ($\triangle f = 1$). Although spatio-temporal data is correlated and breaks differential privacy assumptions of data being iid, our trip group framing allows for count queries to still provide adequate privacy as discussed in Section 2.4. The proposed algorithms in this section offer two different count algorithm perspectives: AEs and RCs. These operators are formally defined in the following two sections. The analytical comparison between AEs and RCs show that RCs ensure a stronger level of differential privacy, but typically use more privacy budget to ensure the same accuracy as AEs. AEs are also shown to use privacy budget less than or equal to the privacy budget used by RCs when wanting a scalar count (AEs are optimal in this case) and vice versa when wanting a range.

3.1.1 Acceptable Error Count Queries

Acceptable error count queries (AEs) represent specifying an acceptable error amount relative to the true count. AEs are simply an alternative framing of the traditional differential privacy Laplace mechanism. This is because AEs are parametrized by an accuracy, as opposed to a relatively unintuitive privacy budget.

Acceptable Error Count Queries (AEs) Definition: An AE is parametrized by an acceptable offset o ($o > 0$) and a confidence level α ($0 \leq \alpha \leq 1$). AEs return a noisy count $\tilde{c} \in \mathbb{N}$, where

$\tilde{c} = c + Y$ such that $\tilde{c} \in (c - o, c + o)$ with probability $1 - \alpha$, where c denotes the true count and $Y \sim Lap(\frac{1}{\epsilon_{AE}})$, where ϵ_{AE} denotes the privacy budget used to ensure the $1 - \alpha$ accuracy.

AEs are similar to the subsequent range count queries (RCs) except that the AE range is relative to the true count. An example of using an AE is when asking for the number of people who turned left on State Street between 3pm and 6pm with an acceptable error of 30 people.

Privacy Budget to Ensure Accuracy

Although a query is parametrized by accuracy $(1 - \alpha)$, privacy budget is still the mechanism that restricts access to sensitive data. It is necessary to understand the relationship between accuracy and privacy budget which is defined in the below definition:

Definition 3.1.1. AE Privacy Budget

$$\epsilon_{AE} = \frac{-\ln(\alpha)}{o}$$

Definition 3.1.1 is proved in the Appendix. Changing the privacy budget ϵ is how to achieve the specified accuracy as ϵ adjusts the distribution scale of the Laplace noise. Given offset o, privacy budget is the log of accuracy and increases rapidly as α decreases (accuracy increases). Thus, AEs ensure a unique and natural mapping from accuracy to privacy budget.

3.1.2 Range Count Queries

Range Count Queries (RCs) are an absolute perspective of count queries. The rationale is that count queries do not have to always return exact counts. Alternatively, a broad range of counts may be sufficient for analysis purposes. Similar to AEs, RCs utilize the accuracy interface by asking whether a noisy count lies within the given absolute count range with some probability. The consumption of privacy budget is then derived from the accuracy perspective. Before providing the formal RC definition, it is necessary to define a range in this context.

Range Definition: A range r is an exclusive interval defined by a start count point $s \in \mathbb{N}$ and an ending count point $e \in \mathbb{N}$ which contains all the real numbers between s and e ($\forall y \in \mathbb{R}$ s.t. $s < y < e$). An example range r^1 contains 30 to 80 (exclusive range) people would have $r^1_s = 30$ and $r^1_e = 80$. The real numbers are considered although true counts are natural numbers, they become real numbers once the Laplace noise is added to the true count.

Range Count Queries (RCs) Definition: A RC is parametrized by a range r and an accuracy level α ($0 \leq \alpha \leq 1$). RCs return a boolean indicating whether a noisy count $\tilde{c} \in \mathbb{R}$ is in range r

$\tilde{c} \in (r_s, r_e)$ where $\tilde{c} = c + Y$ with probability $1 - \alpha$, where c denotes the true count and $Y \sim L(\epsilon_{RE})$ where the noise $L()$ ensures differential privacy with the appropriate use of and value of ϵ_{RE} that denotes the privacy budget used to ensure the $1 - \alpha$ accuracy.

RCs are similar to AEs but the count range is absolute and defined at the start of the query as opposed to AEs where the count range moves with the true count. One key observation is that RCs and AEs are similar and would use the same privacy budget when the ranges are the same and when the true count is in the middle of this range ($(c - o, c + o) == (r_s, r_e)$). For example, assuming the same accuracy level α, if the true count $c = 55$, the AE offset $o = 25$, and RC range is $r = (30, 80)$, then the privacy budget used for both queries should be the same. This observation will be important for Section 3.1.3. The next example demonstrates a scenario where an AE and a RC are not equivalent with the same range lengths. Consider an AE with offset $o = 5$ and RC with $r_s = 10$ and $r_e = 20$. If the true count is at 12 for instance, the AE will return a noisy count within the range of $(c - o = 7, c + o = 17)$ while the RC will return that the count lies within the specified range (also known as "in range") with both queries ensuring the a result within these ranges with probability $1 - \alpha$.

Improved utility and a reduced privacy budget can result from utilizing RCs. For instance, if an analyst wants to know if there are 100-500 people that cross a street intersection on Mondays, and the true count is 498, AEs would return noisy counts that are outside the specified range almost half of the time, while range count queries return a boolean of whether the count is within the query with $1 - \alpha$ probability.

RCs are intuitively more difficult to ensure the specified accuracy if the true count lies near the starting or ending interval points. In this case, more privacy budget should be used. Increasing privacy budget in a standard Laplace scenario will decrease the scale and thus make the distribution more skinny to ensure that more of the area lies within the specified range. Thus, the distribution of the Laplace is determined by the data and the true count. To help verify these hypothesises, the next sections are dedicated to deriving probability expressions in terms of privacy budget ϵ.

Privacy Budget to Ensure Accuracy (In Range)

When the true count lands in the specified range, the probability that the noisy count will still be in the range is:

Definition 3.1.2. RC Symmetric Probability of In Range

$$1 - \alpha = 1 - \frac{1}{2}[\exp(-\epsilon(c - r_s)) + \exp(-\epsilon(r_e - c))]$$

Definition 3.1.2 is proved in the Appendix. The above expression does not have a closed-form solution for ϵ, thus this value can be solved via an optimizer such as Newton's method. Privacy budget is therefore dependent not only on accuracy but also where the true count lies inside the range. More details about privacy guarantees can be seen in Section 3.1.3.

Privacy Budget to Ensure Accuracy (Not In Range)

When the true count falls outside of the specified range, the true count is either less than the start of the interval or greater than the end of the interval. First considering the true count being less than the range starting point, the probability of the noisy count remaining outside of the range is:

Definition 3.1.3. RC Symmetric Probability of Not In Range

$$\text{if } c < r_s\text{: } 1 - \alpha = 1 - \frac{1}{2}(\exp(-\epsilon(r_s - c)) - \exp(-\epsilon(r_e - c)))$$

$$\text{if } c > r_e\text{: } 1 - \alpha = 1 - \frac{1}{2}(\exp(-\epsilon(c - r_e)) - \exp(-\epsilon(c - r_s)))$$

Definition 3.1.3 is proved in the Appendix.

Minimizing Privacy Budget Usage

For range count queries, it is necessary to modify the privacy budget used on a query-by-query basis to achieve the specified accuracy. Privacy budgets are extremely valuable and are set low to ensure sufficient privacy (e.g., often $\epsilon < 1$). Thus, there is incentive to see if this range count process can be optimized to fulfill the same privacy guarantees while decreasing the privacy budget used.

An example is to consider when a real count lies inside the range and is very close to one of the range endpoints. The noise distribution must decrease its scale in order to still ensure the specified accuracy that the count is in range. This is because the scale of the Laplace noise is $\frac{1}{\epsilon}$, an increase in ϵ makes the distribution more condensed and the noisy count can still be in the range for high accuracies. An optimization would be if the distribution could be skewed away from the closer range endpoint. An asymmetric distribution could theoretically require less change in scaling to achieve the same accuracy. This idea can be realized with the asymmetric Laplace distribution.

Asymmetric Laplace Distribution

The asymmetric Laplace is a generalization of the standard (symmetric) Laplace distribution. The pdf is:

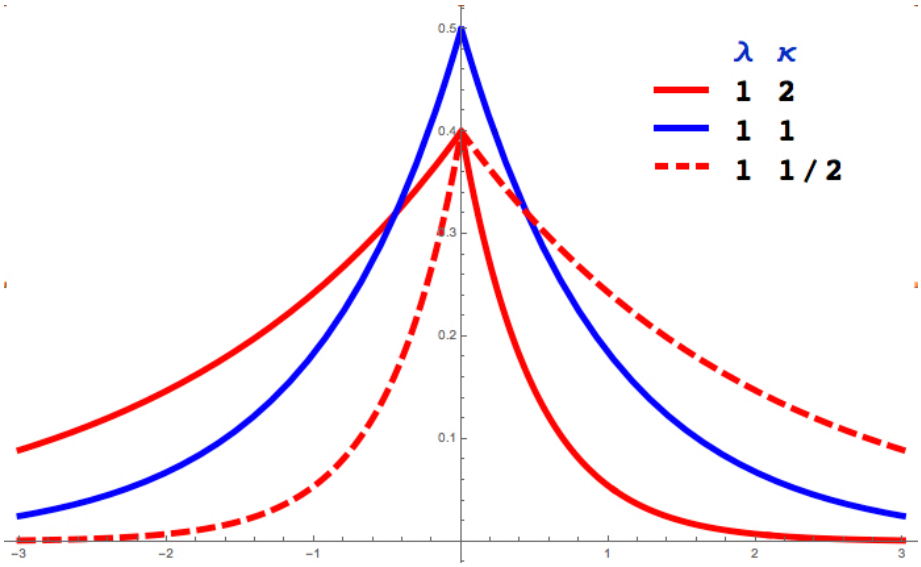


Figure 3.2: An asymmetric laplace centered at 0 with different values of κ (image from the asymmetric laplace wikipedia page).

$$f(x; m, b, \kappa) = \frac{1}{(\kappa + 1/\kappa)b} \begin{cases} \exp\left(\frac{1}{\kappa b}(x - c)\right) & \text{if } x < c \\ \exp(-(\frac{\kappa}{b}(x - c)) & \text{if } x \geq c \end{cases}$$

where κ is the asymmetry parameter and represents the skew. When $\kappa = 1$, the asymmetric distribution simplifies to the symmetric Laplace distribution. A visualization of different values of κ can be seen in Figure 3.2.

The probability of the noisy count $\tilde{c}$ being in the range given that the actual count is in the range can be seen as a generalization of the symmetric Laplace expression:

Definition 3.1.4. RC Asymmetric Probability of In Range

$$Pr[\tilde{c} \in r | r, c \in r, \epsilon] = 1 - \alpha = \frac{1}{1 + \kappa^2}[\kappa^2(1 - exp(-\frac{\epsilon}{\kappa}(c - r_s))) + (1 - exp(-\epsilon\kappa(r_e - c)))]$$

This expression is solved using the same approach as Definition 3.1.2 and is additionally dependent on the value of κ. To minimize the privacy budget used, it is important to first find the optimal value of κ that maximizes the above probability expression. Similar to ϵ, κ does not have a closed-form solution and must be solved using an optimizer. Once the distribution is appropriately

skewed, an optimizer can also be used to solve for the privacy budget ϵ to shape the noise to achieve the specified $1 - \alpha$ accuracy.

Similarly, the probabilities that the noisy count $\tilde{c}$ being outside the range given that the actual count is outside the range can also be seen as generalizations of the symmetric Laplace expressions.

Definition 3.1.5. RC Asymmetric Probability of Not In Range

$$\text{if } c < r_s\text{: } Pr[\tilde{c} \notin r | r, c \notin r, \epsilon] = 1 - \alpha = 1 - \frac{1}{1+\kappa^2}(\exp(-\epsilon\kappa(r_s - c)) - \exp(-\epsilon\kappa(r_e - c)))$$

$$\text{if } c > r_e\text{: } Pr[\tilde{c} \notin r | r, c \notin r, \epsilon] = 1 - \alpha = 1 - \frac{\kappa^2}{1+\kappa^2}(\exp(-\frac{\epsilon}{\kappa}(c - r_e)) - \exp(-\frac{\epsilon}{\kappa}(c - r_s)))$$

This expression is solved using the same approach as Definition 3.1.3.

3.1.3 Analytical Query Evaluation

Query Operators Satisfying Differential Privacy

Acceptable error count queries (AEs) that are parameterized by offset o and accuracy level α use privacy budget ϵ_{AE} (as defined in Definition 3.1.1) and satisfy ϵ_{AE}-DP as they are the exact same as the traditional Laplace mechanism with the release of scalar counts.

The differential privacy levels and findings are more interesting for RCs as symmetric and asymmetric RCs are shown to ensure a stronger level of differential privacy than AEs, especially for small range lengths and small alphas. To measure a level of differential privacy an algorithm ensures, the ratio of the probabilities of neighboring databases are bounded by privacy level ϵ. Formally, for a discrete random variable R_{A,D_i} parameterized by algorithm A and database D_i, where D_1 and D_2 are neighboring databases and differ by only one person:

$$\frac{Pr[R_{A,D_1} = x]}{Pr[R_{A,D_2} = x]} \leq \exp(\epsilon)$$

Range count queries have only two possible random variable values denoted by a Boolean indicating whether the noisy count is within the specified absolute count range. Even with the more simple symmetric Laplace probability expression, it is difficult to find a closed-form bound for differential privacy. We empirically solve for this bound.

The first step is to identify that the probabilities of being in range change the most near the range endpoints. This is because the change in privacy budget (scale) is the largest as shown by Figure 3.3. Figures 3.4a and 3.4b visualize the ratio of the probabilities of neighboring databases as

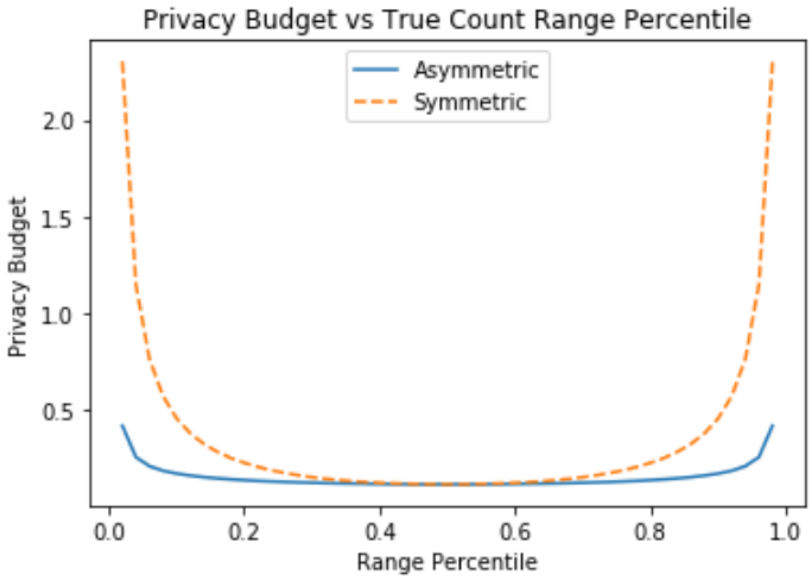
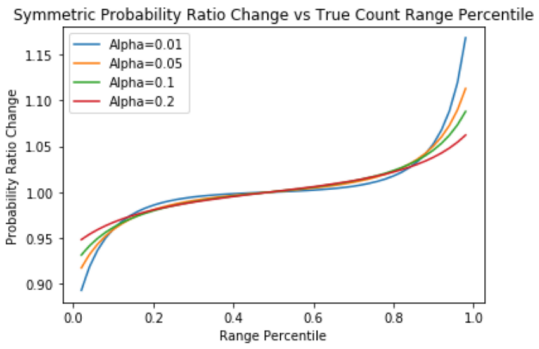
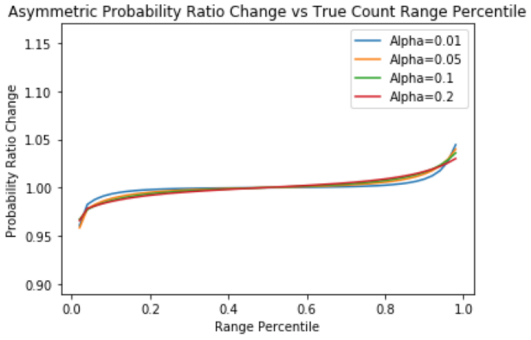


Figure 3.3: Privacy budget increases the most near the range endpoints for an example of any range with length 50 and $\alpha = 0.05$.



(a) $\frac{Pr(c \text{ in range})}{Pr(c+1 \text{ in range})}$ where c denotes the true count when using the symmetric Laplace for a range length of 50 when using the optimal RC privacy budget ϵ_{AE}.



(b) $\frac{Pr(c \text{ in range})}{Pr(c+1 \text{ in range})}$ where c denotes the true count when using the asymmetric Laplace (skewed away from r_e) for a range length of 50 when using the optimal RC privacy budget ϵ_{AE}.

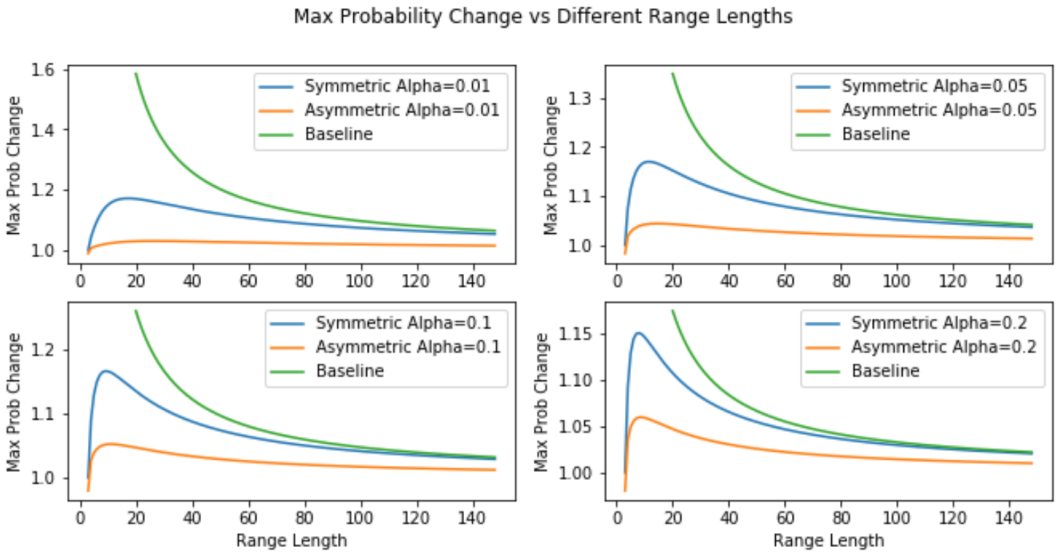


Figure 3.5: Maximum probability change across different range lengths up to 150 compared to the ϵ_{AE} baseline.

the true count increases over range percentile. These figures verify that the ratio of probabilities are the largest near the range endpoints and that the asymmetric Laplace ensures a much lower maximum probability ratio.

Knowing that the maximum probability change is near the range endpoints, Figure 3.5 shows the symmetric and asymmetric maximum probability ratio relationship as the range length T increases. Both the symmetric and asymmetric largest probability ratios are upper bounded by the baseline ϵ_{AE}. The symmetric curve converge to the baseline ϵ_{AE} differential privacy level and converges more quickly the larger alpha is. The level of differential privacy of symmetric and asymmetric becomes stronger (ensuring a tighter bound than ϵ_{AE}) as alpha decreases. The asymmetric Laplace always produces a lower (stronger) level of differential privacy ($\exp(\epsilon_{asym}) \leq \exp(\epsilon_{sym}) \leq \exp(\epsilon_{AE})$) which means that for the same privacy budget ϵ_{AE}, the asymmetric mechanism for RCs preserve stronger individual privacy than both the symmetric RC and AE operators. Additionally the asymmetric and symmetric have relatively large differential privacy improvements over the baseline for small range lengths T which show that with the same privacy budget, the asymmetric and symmetric RCs ensure a significantly stronger level of differential privacy than ϵ_{AE}.

As seen in Section 3.1.3, RCs often use a privacy budget ϵ_{RC} that is greater than ϵ_{AE} to ensure the specified $1 - \alpha$ accuracy. Even if $\epsilon_{\text{RC}} >= \epsilon_{\text{AE}}$ with $\alpha_{\text{RC}} = \alpha_{\text{AE}}$ and the same range lengths, RCs leak less per privacy budget unit spent than AEs. RCs ensuring stronger differential privacy

intuitively makes sense as RCs only have two possible values ("in range" and "not in range") while AEs output scalars.

Privacy Budget Guarantees

AEs have a privacy budget independent of the true count c and is calculated given a query's specified offset o and accuracy level α:

$$\epsilon_{\text{AE}} = -\frac{log(\alpha)}{o}$$

RCs and their privacy budgets are, in contrast, dependent on the true count c and its position on the specified range. Because RC privacy budgets can vary, our goal is to identify lower and upper bounds for the privacy budget.

The lower bound for both the symmetric and asymmetric case is $\epsilon_{lb} = \frac{-2log(\alpha)}{r_e - r_s}$ which is equivalent to ϵ_{AE} when $o = \frac{r_s - r_e}{2}$. The lower bound is achieved when the true count lands on the center of the range the scale ($\frac{1}{\epsilon}$) to achieve the $1 - \alpha$ accuracy is the lowest. When the true count c is not in the middle of the range, The noise distribution scale has to decrease when the true count c is not in the middle of the range. Thus, privacy budget increases in order to ensure the specified accuracy.

Solving for the upper bound is more difficult. One very important thing to note is that the upper bound for the "not in range" privacy budgets is the same for the "in range". We limit our analysis to the "in range" case. Recall Figure 3.3 which shows that the maximum privacy budget used is close to the range start and end points. Given a range r, the max privacy budget is spent when the true counts are at either $r_s + 1$ or $r_e - 1$. The probability expression when setting true count c as either $r_s + 1$ or $r_e - 1$ simplifies to form without a closed-form ϵ expression:

Definition 3.1.6. RC Symmetric Upper Bound Privacy Budget Expression

$$\ln(\epsilon + \ln(2\alpha)) \leq -(r_e - r_s - 2)\epsilon$$

Although it is not possible to get a closed-form expression for ϵ in terms of α and r, we can analyze this expression to still yield a tight upper bound as the two expressions are typically very close in equality as seen through the derivation of Definition 3.1.6 in the Appendix. Figure 3.6 shows that the intersection occurs during the steep incline of the log function. Note that the range length in the Figure 3.6 example is 7, which is relatively small. The intersection will have an even lower function value with larger ranges as the slope of the linear right expression will decrease proportionally to range length and will intersect the Left (ln()) function at a lower function value.

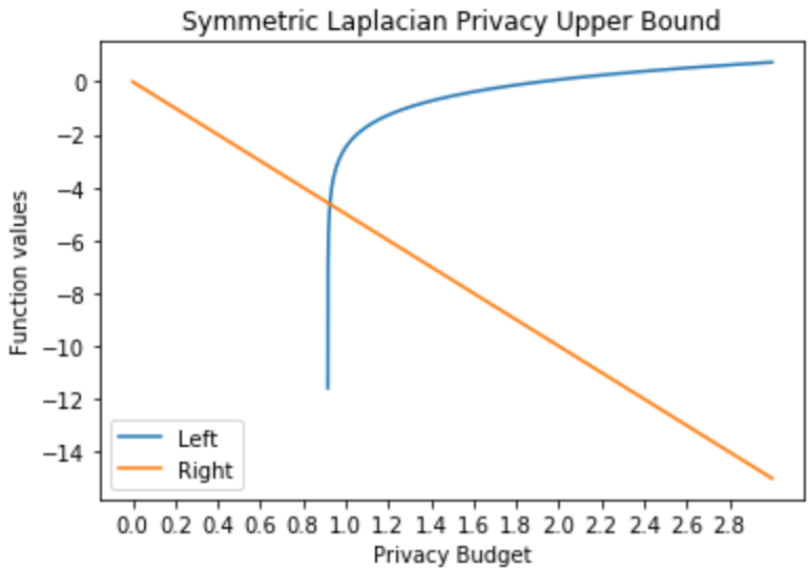


Figure 3.6: Graphical intersection showing the privacy upper bound of $\epsilon = 0.926$ for the symmetric laplace with $\alpha = 0.2$ and $r_e - r_s = 7$. Left represents $\ln(\epsilon + \ln(2\alpha))$ and Right represents $-(r_e - r_s - 2)\epsilon$. These parameters were chosen as they displayed the intersection well.

For large range length, since we know from the domain of ln() that $\epsilon > -\ln(2\alpha)$, the intersection function value being a somewhat large negative value means that ϵ will still approximately be $-\ln(2\alpha)$. Precisely for sufficiently large range lengths T in counts ($T > 5$) where $T = r_e - r_s$ the upper bound is $\epsilon = -\ln(2\alpha) + \delta$ where δ is typically very small ($\delta < 0.001$). Otherwise when $2 \leq T \leq 5$, $\epsilon \leq 1 - \ln(2\alpha)$ is the worst case upper bound and is achieved when $T = 2$ as the natural log on the left has to equal to 0.

The probability "in range" expression using the asymmetric laplace does not simplify to an analyzable expression. Another way to obtain an upper bound is use empirical max privacy budgets e and the $\frac{1}{T}$ relationship shown in Figure 3.7 to produce a linear regression approximation using $x = \frac{1}{T}$ and $y = e$. The issue with unconstrained linear regression is that the line's function values are not guaranteed to be greater than or equal to the points. Thus a constrained linear regression problem is solved with the additional constraint of $\theta_1 x + \theta_0 \geq e_x$ where θ_1 denotes the slope and θ_0 denotes the bias. The upper bounds for both symmetric and asymmetric laplaces are showed in Table 3.1.

Table 3.1 shows that the asymmetric laplace always has a lower upper bound for privacy budget than the symmetric laplace when $T \geq 2$ as when T increases, the bias is weighted much more and

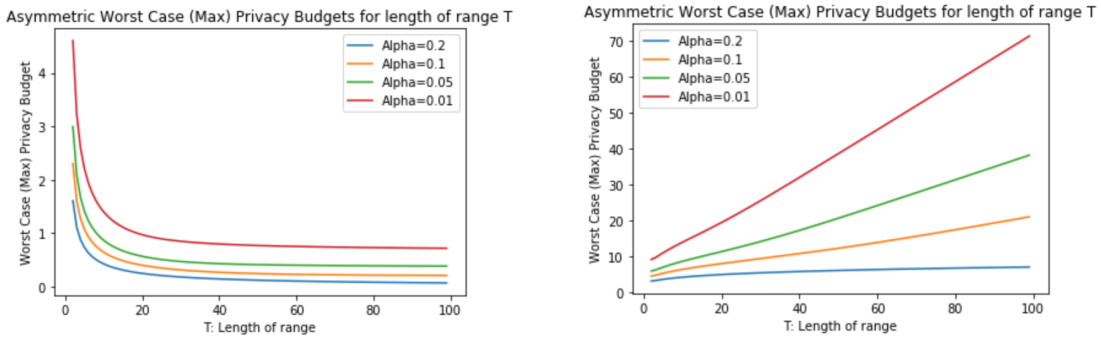


Figure 3.7: Left plot shows worst case asymmetric privacy budgets for increasing interval lengths. Right plot shows the same plots when the privacy values are multiplied by T which verifies that the max privacy budget is proportional to $\frac{1}{T}$.

Alpha	Symmetric Upper Bound	Asymmetric Upper Bound
0.01	1.415/T + 3.898	7.93/T + 0.64
0.05	1.415/T + 2.289	5.45/T + 0.331
0.1	1.415/T + 1.595	4.635/T + 0.182
0.2	1.415/T + 0.902	3.481/T + 0.081

Table 3.1: Privacy budget upper bounds dependent on range length T.

the symmetric case has much larger biases than the asymmetric ones. RCs should only use the asymmetric as it is an optimization of the standard symmetric laplace. Any time an RC is used in the remainder of this document, assume it uses the asymmetric laplace mechanism.

Preferring AEs over RCs for Scalar Counts

When wanting to obtain a scalar count, it is optimal (uses less privacy budget) to use AEs over RCs. Recall that the privacy budget AEs ensure are the lower bounds for RCs and that these queries provide the same information when the true count lies in the middle of the specified range. For the same AE and RC queries with an accuracy level α, offset o for the AE, and range r for the RC such that $\frac{r_e - r_s}{2} = o$ and true count c, if $c = \frac{r_s + r_e}{2}$, then both queries give the same information and use a privacy budget $\epsilon = -\frac{\ln(\alpha)}{o}$.

Running an AE and RC query with $\alpha = 0.05, c = 25, o = 10, r_s = 15, r_e = 35$ as an example will both give information that the true count is somewhere within the 15-35 range with probability $1 - \alpha = 0.95$ and uses $\epsilon = 0.3$.

But a given RC with an alternative range of the same size will either not overlap with the true count or overlapping with the true count not in the middle, which uses more privacy budget than if the count were in the middle. Continuing with the same example as above, resetting the range values to $r_s = 30$ and $r_e = 50$ indicates that the true count is not in that range with 0.95 probability and uses $\epsilon = 0.25$. Additional RCs will have to pinpoint the true count, which will use more privacy budget. Also if we reset the range values to $r_s = 20$ and $r_e = 40$, the privacy budget used is $\epsilon = 0.33$, which is greater than privacy budget used when the true count was in the middle of the range.

AEs still use lower privacy budgets to obtain counts as opposed to using multiple ranges to estimate counts. A realistic approach to using ranges to estimate a query count value is to start with a large range and use binary search with ranges to hone in on the true count. For example, resetting the range values to $r_s = 0$ and $r_e = 100$ would use $\epsilon = 0.067$ and two proceeding ranges with $r_s = 10$ and $r_e = 40$ will use $\epsilon = 0.2$ and range $r_s = 60$ and $r_e = 90$ will use $\epsilon = 0.1$ which has already exceeded the 0.3 optimal privacy budget.

Preferring RCs over AEs for Ranges

When obtaining a range of values, less privacy budget is consumed compared with using RCs over AEs. We will argue that using AEs to learn the same information with less privacy budget than using one RC is impossible.

Similar to preferring counts over ranges section, equivalent AEs and RCs use the same privacy budget. If the true count is not in the center of the wanted range, the probability that the AE noisy query result being in the range is less than if it were in the middle of the range. AEs use a symmetric laplace mechanism and the probability of being in range given the position on the range can be seen in Figure 3.8. Because RCs use the asymmetric mechanism, they will always ensure a higher probability of being in the range given the same privacy budget.

3.2 Data Access Mechanisms

Once a query and its operator have been defined, the next stage in the logical query pipeline is accessing user data to compute the query result. Continual private data publishing related works release data from all users to all analysts at fixed privacy levels. An interactive query setting, in contrast, provides room for customizable data access mechanisms that controls the data that a query has access to. The goal is to provide data access according to the given query and its

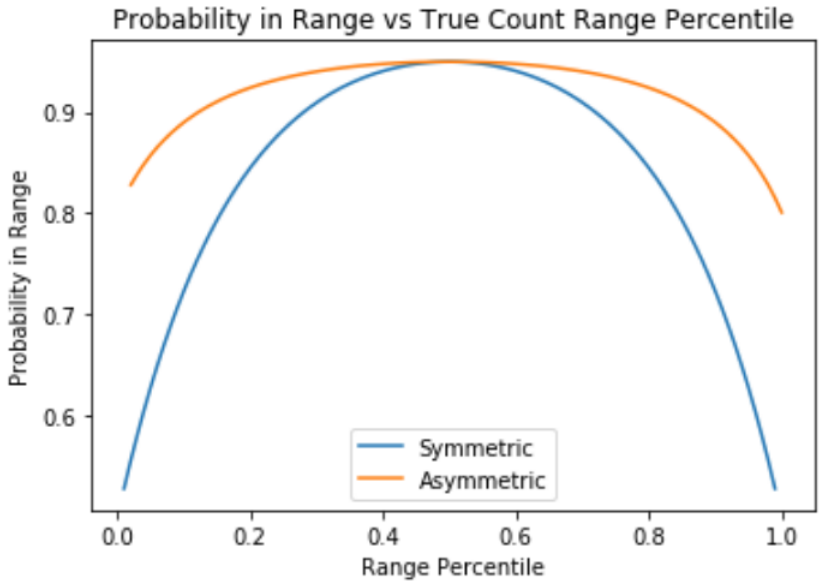


Figure 3.8: Probability of being in the range using the symmetric and asymmetric laplace mechanism with $\alpha = 0.05$, $r_s = 0$, and $r_e = 100$.

parameters. Similarly to query operators, we propose two data access mechanisms perspectives: data access policies and data access degrees.

Data Access Policies

Data access policies specify which queries are allowed to access to certain data.

Data Access Policies (DAPs) Definition: If the logical database is split into n disjoint partitions where each partition is denoted as P, a DAP of a partition P is the subset $D_P \subseteq Q_{Pc}$ of the space of all possible query parameter combinations Q_{Pc} such that any query parameter combination $q_{Pc} \in Q_{Pc}$ has access to the data if $q_{Pc} \in D_P$ and does not otherwise. The aggregated DAP across all n partitions is defined as $D = \bigcup_{P=1}^{n} D_P$.

Thus, DAPs are binary policies that indicate whether a query has data access for each database partition. For a single partition ($n = 1$), the query would have access to either all or none of the total data.

DAPs provide the ability to define access for any combination of spatio-temporal range, query origins and analysts, and algorithms. An example includes only being able to conduct count queries over a certain spatio-temporal range to a select few analysts.

Data Access Degrees

The main limitation of DAPs is not being able to specify how much access to the data query has. One way to address this limitation is to provide a mechanism to set how much access queries should have on the data. Data access degrees specify the allowed amount of data leakage (e.g., privacy budget) for a given query.

Data Access Degrees (DADs) Definition: If the logical database is split into n disjoint partitions where each partition is denoted as P, a DAD of a partition P is a function $DAD_P()$ that maps a query parameter combination $q_{Pc} \in Q_{Pc}$ to a degree access value $v_{q_{Pc}} \in \mathbb{R}$. The aggregated DAD across all n partitions is $DAD(q_c) \rightarrow \sum_{P=1}^{n} v_{q_{Pc}}$.

Privacy budgets from differential privacy are a possible application of the DAD() output degree values. An example is to set high data access values for times that are more than three years old and restricting access for all analysts except for the government. One limitation of DADs are that it is difficult to set and keep track of values for the combination of possible query parameters. Our implementation assigns one customizable privacy budget value for all query parameters as seen in Section 4.1.

3.3 Analyst and User Roles

Aggregation options we proposed have introduced two query operators and two data access mechanisms. Analysts should have the role of specifying the query, which includes the query operator, while users should be able to control how their personal data is used given a query. Analyst and user roles in the logical query lifecycle can be seen in Figure 3.1.

Analyst Specifying Queries and Operators

The analyst role is to specify a query that should include space, time, algorithm (and its associated parameters), and query origin. Because the spatio-temporal data is fine-grained in our setting, an analyst should be able to specify any level of spatio-temporal granularity. Additionally, analysts should also be able to choose their query result precision with the specified accuracies and offsets for AEs or ranges for RCs.

Users Specifying Query Data Access

The user role is to specify how their personal data can be used by the given query. Users should have the ability to customize their data access settings through policies (DAPs) and/or privacy

budgets (DADs) for each analyst. Each user would control their personal partition of the overall dataset.

Users may want to protect themselves against malicious analysts and have the ability to consent their data through the use of DAPs. DAPS would enable users to provide full or no access to certain queries. For instance, controlling whether their data can be accessed by their city and not by Russian hackers (granting access by analyst). Continuing with the previous example, a user may only want to provide their city with only count query algorithms (granting access by algorithm). Lastly, a user can utilize DADs to set their own privacy budget for any combination of query parameters. For instance, a user moves to another state and chooses to increase their privacy budget for a trusted analyst for their travel data from their previous state.

4 Privacy Preserving Query System Implementation

4.1 Model Overview

Our implementation includes a privacy-preserving query system to deploy the aforementioned spatio-temporal queries and their associated query operators and data access mechanisms. The high level design is that users upload their spatio-temporal trip data to a secure location on the cloud. Analysts want to conduct queries on user data, and the goal is to allow queries in a privacy-preserving manner that will not leak this sensitive user data.

The system primarily relies on enclaves to ensure that the code interacting with the raw user data is unhampered, while also being encrypted RAM without OS support. It is importantly noted that enclaves are the security backbone of this system. The current implementation does not use secure enclaves, as there are difficulties with combining existing research with remote attestation. A downstream concern when wanting to deploy this system on a large scale is the lack of secure enclave option support from cloud providers as enclave technology is relatively premature.

System Components

The system consists of five high level components: user clouds, user file store, an aggregator, query microservices, and the controller, and two entities interacting with the system: users and an analyst. We discuss each component and how they contribute to the high level design of securely storing and aggregation sensitive user data. All system components are visualized in the aggregation protocol in Figure 4.2.

User Clouds: A user cloud is a partitioned space that manages the user's data and policies. The goal of a partitioned space is to ensure privacy guarantees through isolation over CPU, memory, and runtime guarantees. These clouds can be viewed as the partitioning of machines (or cloud instances), but at the container level rather than server consumer level (i.e., everyone having their own servers). Specifically, a user cloud is implemented as a container that hosts both a server

enclave and a database enclave. The server is in charge of uploading data and communicating with the database component to stream data when queries are conducted. The purpose of streaming data is to not require user clouds to have large memory. Using an enclave for the server is necessary to prevent leakage of raw user data. The database is an application level interface that accesses the user file store and also needs to be a secure enclave because of the handling of sensitive user data.

The user cloud stores the unique user key that is generated when a user joins to encrypt their data, stores the one privacy budget that covers all of their data, and implements user policies.

User File Store: A user file store is the system component that is in charge of securely storing user trip data. A user file store formally is a user partitioned database that stores user trip data encrypted at rest with the unique user key. Our implementation opted to use ECryptFS because it is installed by default on Ubuntu and was easy to deploy with simple to use instructions. An encrypted database was chosen as it protects user data from being leaked if the database is compromised while also being simple to access by the user cloud.

Aggregator: The aggregator receives and sends a query while also performing the final aggregation step with the intermediate query results. A one time use enclave is used for the aggregator as it needs to be attestable to ensure the correct computations are run on the intermediate queries results. The aggregator functions as the trusted aggregator in the traditional differential privacy setting as its code is attestable.

Query Microservices: When computing a query, each user has a designated query microservice that receives raw user data from the user cloud, computes the user specific intermediate query result, and sends it to the aggregator for the final computation. These microservices are also enclaves as they need to be encrypted and attestable as they are handling sensitive user data.

Controller: The controller communicates with system components to provide necessary information to other components. A standard web server is used to implement the controller as it does not observe user data and has no possibility of leaking this data.

Analyst: An analyst constructs a query by defining the time range, geo-polygon location area, accuracy or privacy budget, and an algorithm and its associated parameters. The query is sent to the created aggregator (i.e., the analyst proxy) that computes the query result and sends back to the analyst.

Users: The entities that generate the trip data and specify the policy and data access settings that their user cloud enforces such as determining the value of their privacy budget ϵ and specifying which analysts and algorithms are allowed to access their data. Users do not directly participate in the aggregation protocol.

Threat Model

The system can be split into three main parts for the threat model: users (mobile devices uploading data), analysts (clients who would like to perform aggregate queries on user data), and the online service provider (i.e., all of the system components). We assume that a majority of distributed users are honest and that analysts are untrusted. The online service provider we assume is also untrusted as most system components are attestable. Overall, this is a strong threat model that assumes trust only in a majority of users.

Aggregation Options

Aggregation options we implemented include supporting any granularity of time and location, query operators, and data access mechanisms. The query operators are the acceptable error queries (AEs) and range count queries (RCs) that are described in Chapter 3 with their associated parameters. The implementation supports defining any query algorithm that is differentially private or privacy-preserving through implementing simple interfaces in the query microservices and aggregator (in terms of lines of code, AEs are 36 and RCs are 110). The data access mechanisms are policies (DAPs) that are limited to filtering by analyst and algorithm and through one fixed privacy budget (DADs) which allows the user to specify how much of their data can be accessed.

Implementation Details

We provide a Python implementation that appends to the e-mission server code located at https://github.com/e-mission/e-mission-server. The Bottle Python Web Framework is used to facilitate communication and requests between system components. The aggregator, query microservices, and user clouds are Docker containers that include libraries and other system dependencies such that the system can run on any machine despite the machine's configuration. More in depth information about this system implementation will be covered in a technical report written by Nicholas Riasanovsky that has yet to be released. The system code is open source and can be viewed at https://github.com/njriasan/e-mission-server/tree/newArch.

4.2 User Add Data Protocol

The first part of the high level design goal of ensuring the secure storage of sensitive user trip data is handled by the user add data protocol. Although this part was not part of this implementation (generated trip data was used for our evaluations in Chapter 5), this is the protocol that users

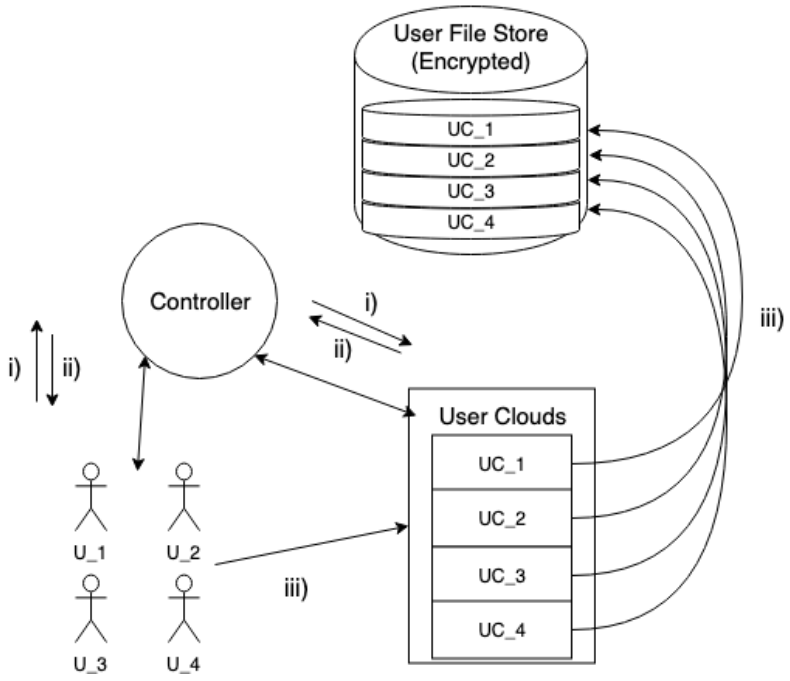


Figure 4.1: The user add data protocol and its various steps.

would use securely upload their data to their encrypted file store. The protocol is visualized in Figure 4.1 and the steps are outlined below:

- Step i) Each user sends a request to the controller to obtain their user cloud address. The controller sends a request to the user cloud to unpause their container (if necessary).
- Step ii) If the user cloud is running and responds to the address request, the controller sends the user the user cloud address.
- Step iii) The final step involves the users sending their new trip data to their user clouds, where the data is encrypted with the unique user key and forwarded for storage in the user file store.

The user add data protocol ensures the secure storage of user data. Our threat model illuminates one potential malicious consideration: the controller sending the users fake user cloud addresses. The users confirm that they are communicating to the correct system components via remote attestation of the user cloud server enclave. Hence the user add data protocol is secure and fulfills the vision of secure storage used for privacy-preserving queries.

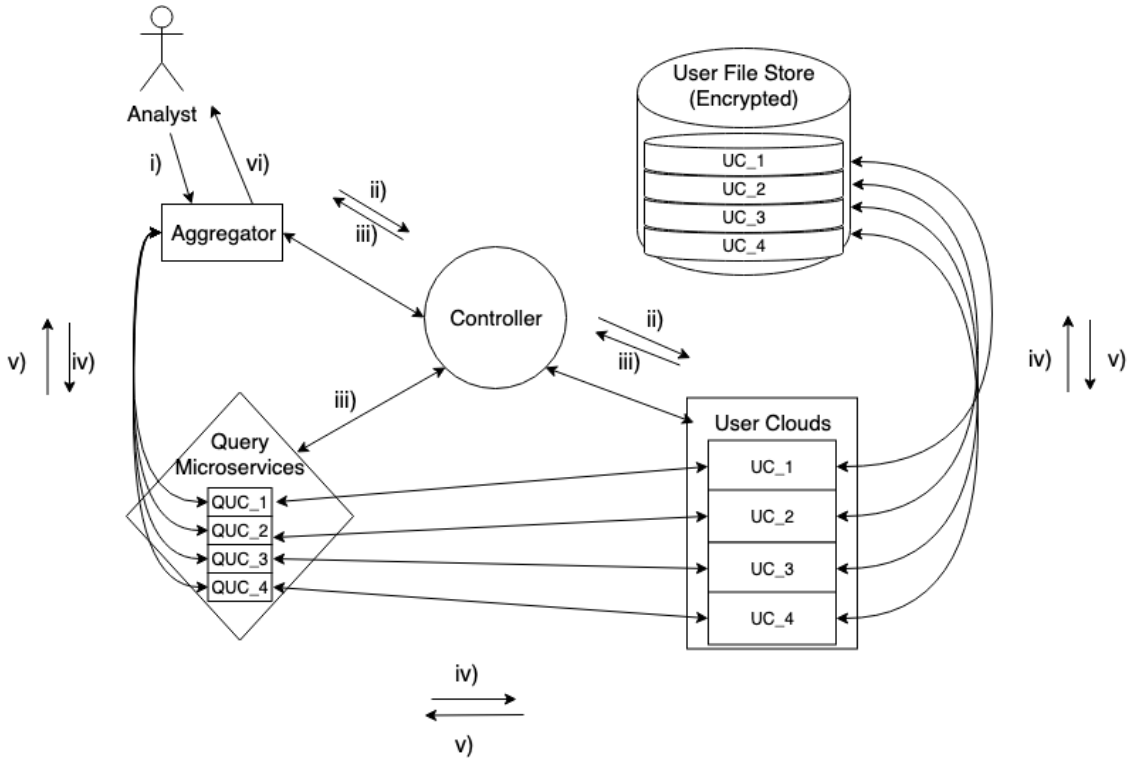


Figure 4.2: The aggregation protocol and its various steps.

4.3 Aggregation Protocol

After user data is securely stored, analysts are able to safely conduct privacy-preserving queries on user data through our system's aggregation protocol. This protocol is the crux of our implementation and is visualized by Figure 4.2 with the steps are outlined below:

- Step i) The analyst specifies a query q which is defined by a time range, geo-polygon location area, accuracy or privacy budget, and an algorithm and its associated parameters. The aggregator is launched for this single protocol.
- Step ii) The aggregator sends a query signal to the controller in order to get the addresses of the user clouds and query microservices. The controller sends a signal to the user clouds to unpause their containers if necessary.
- Step iii) The user cloud addresses are obtained through the running user clouds sending their addresses to the controller and query microservices are created for each user by the controller for this single protocol.

- Step iv) This step passes the query to every other system component all the way through to the user file store. The aggregator first sends each query microservice the query q and the associated user address, which the query microservice passes onto the associated user cloud. If the analyst and algorithm user policy checks pass, then the user data is accessed.
- Step v) Data is streamed from the user file store to the user clouds where the data is decrypted, then streamed into the query microservices where intermediate query results for each user are generated. For example, if the query algorithm is a count, then each query microservice will return either 0 or 1. These intermediate results are then sent to the aggregator for the final round of aggregation.
- Step vi) With the intermediate user results, the aggregator combines them in order to obtain the true query answer. An additional algorithm-dependent postprocessing step is required to generate a noisy query answer.

It is important to review why this protocol is indeed secure with the consideration of the threat model. The aggregator and query microservices being attestable enclaves ensures that the aggregations are consistent and can not be tampered with. Remote attestation before communicating with enclave system components helps guard against the untrusted controller attempting to provide malicious query microservice or user cloud addresses and the user cloud sending decrypted data to malicious destinations. Lastly, all connections between components are secure as required by enclaves. Thus the aggregation protocol satisfies the goal of computing query results while not leaking user data.

5 Empirical Evaluation

We have shown that the proposed interactive system is secure by not leaking sensitive user data. In this section, we assess the implemented system based on scalability and providing correctness of the aforementioned query operators and data access mechanisms. We see that the overall query aggregate response time increases more as we scale up the number of users. Unpausing the user clouds and query microservices both depend on the number of users while only the streaming data process relies on the amount of data.

Experimental Setup

The experiments utilize three machines from the UC Berkeley millennium cluster and Docker Swarm to orchestrate and manage the containers across the different machines. Docker Swarm was chosen for these experiments over Kubernetes, as Swarm can deploy containers relatively quickly and this allows faster reaction times to scale on demand. As mentioned in Section 4.1, enclaves were not used as each container would have required its own enclave and the BETS lab ante Intel SGX machine could only support around 20 enclaves with our setup using SCONE [3]. Utilizing SGX enclaves on this machine as a result would have been infeasible as the number of enclaves is directly proportional to the number of users (which would only support around 10 users as each user has a user cloud and a query microservice at aggregation time).

This experiment creates fake users and generates user experiment data with the e-mission synthetic trip generator that takes deterministic trips (routes via OpenStreetMap) between specified locations over a time range. This allows for the deterministic configuration of users and their trip data to ensure expected true count query results. Each trip is roughly around 700 database entries.

Many of the experimental steps are independent, which allows for parallelizing across multiple machines. The generations of fake users and their trips, the controller preparing the query microservices, and the aggregator sending query requests to the user query microservices are all independent and thus multiprocessing greatly optimized the experiment runtimes.

Query microservices for these experiments are alternatively always available and not killed after every aggregation as these experiment queries are stateless and an optimization in the setting of

frequent aggregation protocols. Therefore pausing and unpausing the query microservices makes more sense as our experiments can be seen as submitting batches of queries.

Further details of the system implementation and experiments can be found in Nicholas Riasanovsky's future technical report.

5.1 System Scaling and Performance

It is important to consider our implemented system's scalability as the number of users and data increases. Response times for multiple steps of the aggregation protocol are recorded for different combinations of number of users and number of trips (user data). The aggregation protocol steps that contribute the significant response time to the query are:

1. The controller unpausing the user clouds and sending their addresses back to the aggregator (Step 2 and 3 in Section 4.3).
2. The controller unpausing query microservices for each user and sending their addresses to the aggregator (Step 3 in Section 4.3).
3. The aggregator passing on the query to the query microservices that streams user data from the user cloud and returns the intermediate query results to the aggregator (Step 4 and 5 in Section 4.3).

Query times are recorded for different combinations of number of users and number of trips and are averaged over 40 queries as the time standard deviation is low across queries. The number of users considered are 10, 50, and 100 and the number of trips considered are 10, 50, and 100. The average times for each of the three significant aggregation steps can be seen in Tables 5.1, 5.2, and 5.3.

	10 Trips	50 Trips	100 Trips
10 Users	4.61	4.45	4.30
50 Users	23.52	23.73	23.72
100 Users	50.39	49.74	50.23

Table 5.1: Average time to unpause user cloud containers and get user addresses in seconds.

	10 Trips	50 Trips	100 Trips
10 Users	3.19	3.11	3.22
50 Users	17.80	19.09	18.70
100 Users	37.66	39.16	39.26

Table 5.2: Average time to unpause the query microservices in seconds.

The average time to prepare the user containers and for the aggregator to receive the user cloud addresses linearly increases with the number of users as seen in Table 5.1. Only depending on

	10 Trips	50 Trips	100 Trips
10 Users	4.14	21.79	43.74
50 Users	4.88	25.97	48.01
100 Users	6.76	31.43	57.29

Table 5.3: Average time to get the intermediate query results in seconds.

number of users make sense as there are proportionally more user clouds with the increase of number of users. The average time to prepare the user query microservices also only depends on the number of users and increases linearly with the number of users. This is because each user receives their own query microservice. Receiving the intermediate query results from the microservices increases with data as there is more data to stream and process. The time slightly increases with the increase of users as there are more containers to stream and partition resources to.

If enclaves were used, every significant aggregation section would presumably have increased time due to remote attestation and an overhead associated with running inside an enclave [21]. The next steps are to test scalability with many more users (users $>$ 1000) and with enclaves which is future work deferred to the upcoming technical report.

5.2 Verifying Aggregation Options Correctness

In addition to analyzing scalability and performance, correctness experiments test whether the aggregation options proposed in Chapter 3 function as we expect in our implemented privacy-preserving system. AE and RC query algorithms are first verified to provide approximately the given query accuracy levels. Verifying the correctness of the user data access mechanisms consists of two experiments: testing the enforcement of user policies and privacy budget. The correctness tests match their expected behavior and provide tangible examples of how the aggregation options can be used in this system.

Query Correctness

The query correctness experiment tests if AEs and RCs satisfy approximately the specified accuracy level. AEs and RCs are run across four different accuracy levels ($\alpha = 0.01, 0.05, 0.1, 0.2$) and are run across 100 queries to ensure that the queries are providing approximately the expected accuracy. This experiment neglects user policies and privacy budgets which allows for the running of unlimited queries. 20 identical users were generated with the users satisfying the spatio-temporal

query conditions which leads to a true query count result of 20. The first query q_1 is an AE with an offset of 10 and the second query q_2 is a RC that has range starting and ending points at $r_s = 10$ and $r_e = 30$. The results in Table 5.4 demonstrate that the percentages are close to the theoretical $(1 - \alpha)\%$ for the 100 queries and converge will converge closer to $(1 - \alpha)\%$ with more queries. Figure 5.1 verifies that lower values of α lead to more precise answers while larger values tend to be more noisy on average.

$\alpha =$	0.01	0.05	0.1	0.2
q_1	100%	96%	87%	83%
q_2	100%	100%	89%	79%

Table 5.4: Percentage of queries within the specified ranges.

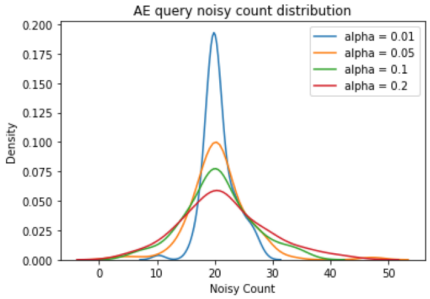


Figure 5.1: q_1 distribution of noisy counts.

User Policy Enforcement

The first data access mechanism test verifies that user policies effectively limits user data and subsequently lowers the true counts and noisy counts on average. Two experiments are designed with one testing that user data is only available to their specified analysts and one testing that data should be accessible only to specified query algorithms. For an accuracy level ($\alpha = 0.05$), q_3 is an AE that has an offset of 5 and q_4 is a RC with starting and ending points at 15 and 25 with the number of fake users and true query count results being 20 similar to q_1 and q_2. There are 100 total queries for each experiment: (i) all analysts and algorithms allowed, (ii) 50 users do not allow any analyst, and (iii) 50 users do not allow any algorithm. As seen in Table 5.5, when half of the users do not allow any analyst or algorithm, we get on average half of the total user count and "not in range" responses from the RC.

Privacy Budget Enforcement

The final correctness experiment is to test whether privacy budgets are properly being depleted and that user data is restricted once privacy budgets are exhausted. This experiment assigns two different privacy budgets to two groups of 20 users (40 users overall): group 1 (g_1) receives $\epsilon_{g_1} = 2$

	All	Half Analyst	Half Algorithm
q_3	19.50	10.17	10.21
q_4	92%	3%	5%

Table 5.5: Mean values of the AE q_3 and RC q_4 (α = 0.05).

Queries Elapsed:	1-10	11-20	21-30
q_5	37.58	20.62	0.94
q_6	100%	10%	0%

Table 5.6: Mean values of the AE q_5 and RC q_6 (α = 0.05) as privacy budgets deplete every 10 queries.

and group 2 (g_2) receives $\epsilon_{g_2} = 4$. Privacy budgets are not advised in practice to be set this high, but we wanted to provide somewhat accurate results and run an adequate number of queries.

The privacy budget correctness queries are q_5 which is an AE with an offset value of 15 and q_6 which is a RC with starting and ending range points at 25 and 55. Using $\alpha = 0.05$ and Definition 3.1.1 show that g_1 has enough privacy budget for 10 queries and g_2 has enough for 20 queries and both will be depleted during the last 10 queries (30 queries overall). Table 5.6 demonstrates that less data is becoming accessible across elapsed queries (by 20 people after every 10 queries).

6 Conclusion and Future Work

Conclusion

In this report, we offer an approach to constructing a privacy-preserving interactive system that supports differential privacy of spatio-temporal trajectories. Namely, maintaining low privacy budgets while proposing count query algorithms and data access mechanisms can enable analysts to maximize their query utility while users have the ability to control how their data is used. These options will hopefully incentivize both analyst and user participation. Relative and absolute count queries (AEs and RCs) are proposed with RCs ensuring a stronger level of differential privacy than AEs, as range length T and α decrease, but RCs typically use more privacy budget than AEs. AEs use privacy budget less than or equal to the privacy budget used by RCs when wanting a scalar count (AEs are optimal in this case) and vice versa when wanting a range.

The interactive query system approach we propose relies on user data being securely stored on the cloud via an encrypted, partitioned file system and on system components that are encrypted and attestable (enclaves) to prevent the leakage of user data and other malicious behavior. Our scalability results demonstrate that the overall query aggregate response time is more sensitive to an increase of users than an data increase. Because two of the significant aggregator steps depend on interacting with system components that are directly proportional to the number of users. Correctness experiments are presented that confirms that the implemented aggregation options function as we would expect and demonstrate how these options operate in our system.

Practically providing differentially privacy for interactive, one at a time queries is inherently a difficult problem due to highly correlated data and one that has not yet been adequately explored. A privacy-preserving architecture must be thought of first and differentially private query algorithms can then be devised such that they are compatible with the architecture. Overall we hope that users will be more incentivized to participate in spatio-temporal aggregations and that their data can enable analysts to make data-driven decisions and impacts.

Future Work

There are many potential interactive differentially private spatio-temporal data future work directions. Query algorithms are limited to counts in this paper, which leaves optimizing and exploring other query algorithms, such as sums, for future work. Trajectories can also be used to not only learn counts, but also locations. A trajectory geo-polygon query would return a geo-polygon that would correspond to people that follow a certain trajectory. For example, where did people start their trips when using this road? Additionally, it would be worth exploring an incentive to prevent analysts from wanting to exhaust user privacy budgets such as an analyst cost that is proportional to the privacy budget or differential privacy level.

The approach of the privacy-preserving implementation in this domain has room for optimization. Implementing this system for a production environment would likely benefit more from using Kubernetes over Docker Swarm. This is because Swarm has limited functionality as per the availability in the Docker API and limited fault tolerance, making it difficult to deploy in production environments. Our implementation query aggregation response times were high, even without using enclaves. A promising avenue for future work is to tweak or redesign the system architecture to increase scalability.

Bibliography

1. G. Acs, G. Biczók, and C. Castelluccia. “Privacy-Preserving Release of Spatio-Temporal Density”. In: *Handbook of Mobile Data Privacy*. Springer, 2018, pp. 307–335.
2. Y. et al. “The algorithmic foundations of differential privacy.” In: *"Foundations and Trends in Theoretical Computer Science"*, 2014.
3. S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O’keeffe, M. L. Stillwell, et al. “{SCONE}: Secure Linux Containers with Intel {SGX}”. In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 689–703.
4. H. Corrigan-Gibbs and D. Boneh. “Prio: Private, Robust, and Scalable Computation of Aggregate Statistics.” In: *NSDI*. 2017, pp. 259–282.
5. R. Cummings, S. Krehbiel, K. A. Lai, and U. Tantipongpipat. “Differential privacy for growing databases”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 8864–8873.
6. Y.-A. De Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. “Unique in the crowd: The privacy bounds of human mobility”. In: *Scientific reports* 3, 2013, p. 1376.
7. C. Dwork. “"Differential privacy"”. In: *"Invited talk at ICALP"*, 2006.
8. M. E. Gursoy, L. Liu, S. Truex, and L. Yu. “Differentially private and utility preserving publication of trajectory data”. In: *IEEE Transactions on Mobile Computing*, 2018.
9. X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, and D. Srivastava. “DPT: differentially private trajectory synthesis using hierarchical reference systems”. In: *Proceedings of the VLDB Endowment* 8:11, 2015, pp. 1154–1165.
10. N. Johnson, J. P. Near, J. M. Hellerstein, and D. Song. “Chorus: Differential Privacy via Query Rewriting”. In: *arXiv preprint arXiv:1809.07750*, 2018.
11. G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. “Differentially private event sequences over infinite streams”. In: *Proceedings of the VLDB Endowment* 7:12, 2014, pp. 1155–1166.
12. D. Kifer and A. Machanavajjhala. “A rigorous and customizable framework for privacy”. In: *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*. ACM. 2012, pp. 77–88.
13. D. Kifer and A. Machanavajjhala. “No free lunch in data privacy”. In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM. 2011, pp. 193–204.

14. D. Kifer and A. Machanavajjhala. “Pufferfish: A framework for mathematical privacy definitions”. In: *ACM Transactions on Database Systems (TODS)* 39:1, 2014, p. 3.

15. Liu, Chuyi. “An Application of Secure Data Aggregation for Privacy-Preserving Machine Learning on Mobile Devices”. MA thesis. 2018. URL: http://hdl.handle.net/10012/13870.

16. D. J. Mir, S. Isaacman, R. Cáceres, M. Martonosi, and R. N. Wright. “Dp-where: Differentially private modeling of human mobility”. In: *2013 IEEE international conference on big data*. IEEE. 2013, pp. 580–588.

17. R. A. Popa, A. J. Blumberg, H. Balakrishnan, and F. H. Li. “Privacy and Accountability for Location-based Aggregate Statistics”. In: *Proceedings of the 18th ACM Conference on Computer and Communications Security*. CCS ’11. ACM, Chicago, Illinois, USA, 2011, pp. 653–666. ISBN: 978-1-4503-0948-6. DOI: 10.1145/2046707.2046781. URL: http://doi.acm.org/10.1145/2046707.2046781.

18. Rastogi and S. Nath. “"Differentially Private Aggregation of Distributed Time-Series with Transformation and Encryption"”. In: *Proc. ACM SIGMOD Int’l Conf. Management of Data*, 2010.

19. L. Sampaio, F. Silva, A. Souza, A. Brito, and P. Felber. “Secure and Privacy-Aware Data Dissemination for Cloud-Based Applications”. In: *Proceedings of the10th International Conference on Utility and Cloud Computing*. UCC ’17. ACM, Austin, Texas, USA, 2017, pp. 47–56. ISBN: 978-1-4503-5149-2. DOI: 10.1145/3147213.3147230. URL: http://doi.acm.org/10.1145/3147213.3147230.

20. E. Shi, T. Chan, E. Rieffel, R. Chow, and D. Song. “"Privacy-Preserving Aggregation of Time-Series Data"”. In: *Proc. Network and Distributed System Security Symp. (NDSS ’11)*, 2011.

21. C.-C. Tsai, D. E. Porter, and M. Vij. “Graphene-SGX: A Practical Library {OS} for Unmodified Applications on {SGX}”. In: *2017 {USENIX} Annual Technical Conference ({USENIX}{ATC} 17)*. 2017, pp. 645–658.

22. Q. Wang, Y. Zhang, X. Lu, Z. Wang, Z. Qin, and K. Ren. “RescueDP: Real-time spatio-temporal crowdsourced data publishing with differential privacy”. In: *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE. 2016, pp. 1–9.

A Appendix

Definition 3.1.1 Proof

Let $f(x|\mu, b)$ denote the pdf of a Laplacian random variable centered at mean $\mu = c$ and with scale $b = \frac{1}{\epsilon}$. Calculating the probability of the noisy count being within the $(c - o, c + o)$ range follows a straight forward pdf integration.

$$1 - \alpha = \int_{c-o}^{c+o} f(x|c, \frac{1}{\epsilon})\, dx$$

$$= \frac{\epsilon}{2}[\int_{c-o}^{c} \exp(-\epsilon(c - x)) + \int_{c}^{c+o} \exp(-\epsilon(x - c))]\, dx$$

$$= \epsilon \int_{c}^{c+o} \exp(-\epsilon(x - c))\, dx \text{ (by symmetry)}$$

$$= -\exp(-\epsilon(x - c))\Big|_{c}^{c+o}$$

$$= 1 - \exp(-\epsilon o)$$

which simplifies to $\epsilon = \frac{-\ln(\alpha)}{o}$ when solving the above equality for the privacy budget.

Definition 3.1.2 Proof

When the true count lands in the specified range, the probability that the noisy count will still be in the range is calculated by integrating the probability density function (pdf) from r_s to r_e of a $Lap(c, \frac{1}{\epsilon})$ distribution centered at the true query count $r_s < c < r_e$ with scale $\frac{1}{\epsilon}$:

$$1 - \alpha = \int_{r_s}^{r_e} \frac{\epsilon}{2} \exp(-\epsilon|x - c|)dx$$

$$= \int_{r_s}^{c} \frac{\epsilon}{2} \exp(-\epsilon(c - x))dx + \int_{c}^{r_e} \frac{\epsilon}{2} \exp(-\epsilon(x - c))dx$$

$$= \frac{1}{2}[\exp(-\epsilon(c-x))\Big|_{r_s}^{c} - \exp(-\epsilon(x-c))\Big|_{c}^{r_e}]$$

$$= \frac{1}{2}[\exp(-\epsilon(c-x))\Big|_{r_s}^{c} - \exp(-\epsilon(x-c))\Big|_{c}^{r_e}]$$

$$= 1 - \frac{1}{2}[\exp(-\epsilon(c-r_s)) + \exp(-\epsilon(r_e-c))]$$

Definition 3.1.3 Proof

The probability of being outside the range when the true count is less than the starting range point can be seen as the probability complement of the noisy count being inside of the range:

$$1 - \alpha = 1 - \int_{r_s}^{r_e} \frac{\epsilon}{2} \exp(-\epsilon(x-c))dx$$

$$= 1 - \frac{1}{2}(\exp(-\epsilon(r_s-c)) - \exp(-\epsilon(r_e-c)))$$

Similarly, the probability of being outside the range when the true count is greater than the ending interval point is:

$$1 - \alpha = 1 - \int_{r_s}^{r_e} \frac{\epsilon}{2} \exp(-\epsilon(c-x))dx$$

$$= 1 - \frac{1}{2}(\exp(-\epsilon(c-r_e)) - \exp(-\epsilon(c-r_s)))$$

Definition 3.1.6 Proof

Substituting either $r_s + 1$ or $r_e - 1$ as c into the symmetric Laplace in range probability (Definition 3.1.2) yields:

$$1 - \alpha = 1 - \frac{1}{2}[\exp(-\epsilon(1)) + \exp(-\epsilon(r_e - r_s - 1))]$$

$$2\alpha = \exp(-\epsilon)[1 + \exp(-\epsilon(r_e - r_s - 2))]$$

$$\ln(2\alpha) = -\epsilon + \ln([1 + \exp(-\epsilon(r_e - r_s - 2))])$$

$$\epsilon + \ln(2\alpha) = \ln(1 + \exp(-\epsilon(r_e - r_s - 2)))$$

$$\epsilon + \ln(2\alpha) \leq \exp(-\epsilon(r_e - r_s - 2)); \text{ using the fact that } \ln(1 + x) \leq x \text{ for } x \geq 0$$

$$\ln(\epsilon + \ln(2\alpha)) \leq -(r_e - r_s - 2)\epsilon$$

This inequality is typically very close to equality and converges to equality when $\exp(-\epsilon(r_e - r_s - 2))$ is small from the property $\ln(1 + x) \approx x$ when $x \approx 0$.