PRIVACY-PRESERVING DATA COLLECTION AND SHARING IN MODERN MOBILE INTERNET SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Mehmet Emre Gursoy

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology

August 2020

PRIVACY-PRESERVING DATA COLLECTION AND SHARING IN MODERN MOBILE INTERNET SYSTEMS

Approved by:

Dr. Ling Liu, Advisor
School of Computer Science
Georgia Institute of Technology

Dr. Joy Arulraj
School of Computer Science
Georgia Institute of Technology

Dr. Margaret Loper
Georgia Tech Research Institute
Georgia Institute of Technology

Dr. Calton Pu
School of Computer Science
Georgia Institute of Technology

Dr. Christin Seifert
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente

Date Approved: May 7, 2020

To my mother Gulen Gursoy, and in memory of my father Sahir Gursoy

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Prof. Ling Liu, for her guidance and continuous support throughout my Ph.D. study. I am indebted for the immense amount of advice she has given me and the many hours she has spent on my work. Her enthusiasm and energy are inspiring. I appreciate her for giving me the flexibility in pursuing the research problems I found intriguing while guiding me in the right directions. She has undoubtedly broadened my perspective and taught me many lessons not just in research but also in life.

Next, I would like to thank my Ph.D. committee members Prof. Calton Pu, Dr. Margaret Loper, Prof. Joy Arulraj and Prof. Christin Seifert. Their advice and constructive feedback have helped me improve the quality of my research and my dissertation. I feel privileged to have such a distinguished group of researchers on my committee.

I am grateful for the support and friendship provided by the members of the DiSL research group. In particular, I would like to extend my special thanks to Emre Yigitoglu, Stacey Truex and Semih Sahin. I am forever indebted to Emre Yigitoglu for his friendship, his fun and outgoing personality, and his help during difficult times. I am extremely grateful to have Stacey in my life, who is not only a great friend but also an amazing researcher. I was lucky to have her by my side throughout my Ph.D. (she was literally by my side in the lab, too!) and I thank her for being my partner in crime. I would also like to thank Semih for his friendship and support, starting from the early days of my study.

I wholeheartedly thank the remaining members of the DiSL research group, whom I spent time and collaborated with, including Wenqi Wei, Lei Yu, Yanzhao Wu, Ka-Ho Chow, Wenqi Cao, Qi Zhang, and James Bae. I also had the pleasure of working with several undergraduate researchers. I especially thank Vale Tolpegin and Vivek Rajasekar for being great collaborators and researchers.

I am grateful to my Turkish friends and the members of the Turkish Student Organiza-

tion at Georgia Tech, who made Atlanta feel like home away from home. I wholeheartedly thank Abdurrahman, Berkay Yilmaz, Berkay Yucel, Bige, Duygu Deniz, Goktug, Gozde, Guliz, Hakki, Murat, Sezen, Su, and many others whom I had the pleasure of spending time with. Our experiences made my years in Atlanta truly joyful and memorable. My Ph.D. journey would not have been the same without them.

The one person who has brought most joy and happiness to my life has been my fiancee, Beren. She is one of the best things that ever happened to me, and words cannot describe how lucky and grateful I feel to have her in my life. Beren is my source of endless love and happiness, and she is also my best friend. I am thankful to have had her in every step of my Ph.D. journey by my side. My success is also hers. I look forward to the next chapters of our life together.

Finally, I would like to thank my mother Ayse Gulen Gursoy and my father Birol Sahir Gursoy for their continuous support of my education and encouragement for my academic career. This dissertation would not have been possible without their unconditional love and support. I dedicate this dissertation to my late father, who passed away in the second year of my Ph.D. study, and whose untimely passing has changed my life forever. I am sure he would have been proud of my achievements. His memory will always live with me, and may he rest in peace.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

With the ubiquity and widespread use of mobile devices such as laptops, smartphones, smartwatches, and IoT devices, large volumes of user data are generated and recorded. While there is great value in collecting, analyzing and sharing this data for improving products and services, data privacy poses a major concern.

This dissertation research addresses the problem of privacy-preserving data collection and sharing in the context of both mobile trajectory data and mobile Internet access data. The first contribution of this dissertation research is the design and development of a system for utility-aware synthesis of differentially private and attack-resilient location traces, called AdaTrace. Given a set of real location traces, AdaTrace executes a four-phase process consisting of feature extraction, synopsis construction, noise injection, and generation of synthetic location traces. Compared to representative prior approaches, the location traces generated by AdaTrace offer up to 3-fold improvement in utility, measured using a variety of utility metrics and datasets, while preserving both differential privacy and attack resilience.

The second contribution of this dissertation research is the design and development of locally private protocols for privacy-sensitive collection of mobile and Web user data. Motivated by the excessive utility loss of existing Local Differential Privacy (LDP) protocols under small user populations, this dissertation introduces the notion of Condensed Local Differential Privacy (CLDP) and a suite of protocols satisfying CLDP to enable the collection of various types of user data, ranging from ordinal data types in finite metric spaces (malware infection statistics), to non-ordinal items (OS versions and transaction categories), and to sequences of ordinal or non-ordinal items. Using cybersecurity data and case studies from Symantec, a major cybersecurity vendor, we show that proposed CLDP protocols are practical for key tasks including malware outbreak detection, OS vulnerability analysis, and inspecting suspicious activities on infected machines.

The third contribution of this dissertation research is the development of a framework and a prototype system for evaluating privacy-utility tradeoffs of different LDP protocols, called LDPLens. LDPLens introduces metrics to evaluate protocol tradeoffs based on factors such as the utility metric, the data collection scenario, and the user-specified adversary metric. We develop a common Bayesian adversary model to analyze LDP protocols, and we formally and experimentally analyze Adversarial Success Rate (ASR) under each protocol. Motivated by the findings that numerous factors impact the ASR and utility behaviors of LDP protocols, we develop LDPLens to provide effective recommendations for finding the most suitable protocol in a given setting. Our three case studies with real-world datasets demonstrate that using the protocol recommended by LDPLens can offer substantial reduction in utility loss or in ASR, compared to using a randomly chosen protocol.

CHAPTER 1
INTRODUCTION

In recent years, Big Data has become ubiquitous thanks to advancements and widespread use of electronic and mobile devices such as laptops, smartphones, smartwatches, and Internet-of-Things (IoT) devices. Users interact with dozens of smart electronic devices and software apps every day, which leads to massive amounts of data to be generated. Companies and stakeholders want to make use of such data by promoting big data-driven analytics, machine learning, and business intelligence. One example is Netflix, a popular media-services provider and production company, which estimates that 75% of what people watch on Netflix is determined by recommendation algorithms that are powered by past viewers' and customers' data [1]. Another example is the banking sector, which uses customer data to determine credit and loan approvals, reduce customer churn, design personalized offerings and individualized services, and so forth [2]. Furthermore, with the increasing amounts of online shopping and e-commerce on the Internet, web users' browsing, search and shopping histories have become lucrative sources for trend prediction, personalized ads, and even product price optimization [3].

While there is great value in collecting, analyzing and sharing user data for improving products and services, data privacy poses a major concern. According to a survey reported by the Mobile Ecosystem Forum, privacy (or the lack of it) is the number one concern of consumers when envisioning a world of connected devices [4]. Privacy concerns are further aggravated due to privacy scandals and data mishandlings, including the recent Facebook - Cambridge Analytica scandal [5], fitness app Polar revealing where U.S. military personnel worked and lived, and the Exactis data exposure scandal [6]. Data privacy and data protection concerns have grown so rapidly over the last years that new laws and regulations are now being implemented, a popular example being Europe's General Data Protection

Regulation (GDPR).

This dissertation contributes formal and technical solutions to the problem of data privacy in modern computerized systems, e.g., mobile, IoT, and Internet-based systems. In particular, we consider two research settings within the umbrella of data privacy: privacy-preserving *data collection* and privacy-preserving *data sharing*.

In the data collection setting, each user has some privacy-sensitive data recorded and stored on their personal device. Examples include web browsing histories in browsers such as Chrome, application telemetry and usage statistics on smartphones, and so forth. The data collector (aggregator) would like to collect relevant information from users' devices; however, doing so directly and naively would violate users' privacy. Hence, the research challenge is to design and develop protocols that enable privacy-conscious collection of user data that offer formal privacy guarantees to users while simultaneously allowing the data collector to infer accurate population-level statistics.

In the data sharing setting, the data collector already has potentially sensitive data pertaining to individuals. This is the case with U.S. Census database, NYC Taxi and Limousine Commission's database of taxi trips in New York City [7], and Uber Movement [8], to name a few. In order to facilitate analytics and learning, the data collector wants to share the data with third parties such as Machine Learning as a Service providers (MLaaS), city officials and policymakers, university researchers, or even the general public. However, straightforward sharing of privacy-sensitive data with untrusted third parties may lead to attacks such as re-identification, record linkage, and membership inference attacks [9, 10, 11, 12]. Consequently, the research challenge in the data sharing setting is to design and develop methods to ensure that the data shared with third parties will ensure formal privacy and attack-resilience guarantees.

In order to provide formal and provable privacy guarantees in the data collection and data sharing settings, we build on top of the notion of *differential privacy*, among other privacy and security metrics. Differential privacy is a well-known notion that is being used

increasingly in different academic communities as well as the industry [13, 14, 15, 16]. It has several desirable properties, including the ability to: (i) provide a mathematically provable privacy guarantee, (ii) compose multiple differentially private building blocks and still satisfy differential privacy with the same privacy budget or a relaxed budget, and (iii) post-process differentially private outputs without hurting the differential privacy guarantee. Yet, differential privacy is not a magic bullet – it does not necessarily protect against all types of adversaries and it does not necessarily provide sufficient data utility in all applications. To address protections that fall outside the scope of differential privacy, we propose complementary security metrics appropriate for the setting and data type under consideration, e.g., in case of location traces, we additionally promote resilience to partial trace sniffing attacks against adversaries who may track users' movements in areas under surveillance. In data collection scenarios, we observe that existing local differential privacy protocols provide sufficient utility when data is collected from a large population, but their utility suffers when the population size is small. We therefore propose an alternative privacy notion and protocols targeting data collection in smaller population sizes.

1.1 Dissertation Statement and Contributions

This dissertation research makes the following contributions to address privacy-preserving data collection and sharing in the context of both mobile trajectory data and mobile Internet access data.

1.1.1 Differentially Private and Attack-Resilient Location Trace Synthesis

The first contribution of this dissertation research is the design and development of a system for utility-aware synthesis of differentially private and attack-resilient GPS location traces, called AdaTrace. Individuals' location traces are highly sensitive, and sharing location trace datasets with unauthorized third parties may disclose individuals' travel records, home and work locations, or visits to sensitive locations such as hospitals or religious

events. It is therefore imperative to take appropriate measures to protect individuals' privacy when sharing location traces. To this end, we design and develop AdaTrace, a scalable and quantitative framework that synthesizes location traces while providing three desirable features: (i) differential privacy, (ii) resilience against relevant location trace attacks, (iii) strong preservation of trajectory utility and authenticity. AdaTrace builds a generative model from a given set of real traces through a four-phase synthesis process consisting of feature extraction, synopsis learning, privacy and utility preserving noise injection, and generation of synthetic location traces. The location traces synthesized by AdaTrace bear a statistical resemblance to real traces, but they do not leak sensitive information about any individual due to the enforcement of differential privacy and attack-resilience guarantees. We validate the effectiveness of AdaTrace by comparing it with three state of the art approaches using three datasets (Geolife, Taxi, Brinkhoff) and seven trajectory utility metrics. Our results show that AdaTrace provides up to 3-fold improvement in trajectory utility, and is significantly faster than prior art.

1.1.2 Secure and Utility-Driven Data Collection

The second contribution of this dissertation research is the design and development of locally private protocols for privacy-sensitive collection of mobile and Web user data. While Local Differential Privacy (LDP) is popularly used in practice for privacy-preserving data collection, we find that existing LDP protocols offer high utility only when data is collected from large enough user populations, but they perform poorly in terms of utility when the population size is small. Motivated by the excessive utility loss of existing LDP protocols under small populations, this dissertation introduces the notion of Condensed Local Differential Privacy (CLDP) and a suite of protocols satisfying CLDP to enable the collection of various types of user data, ranging from ordinal data types in finite metric spaces (malware infection statistics), to non-ordinal items (OS versions and transaction categories), and to sequences of ordinal or non-ordinal items. Extensive experiments are conducted

on multiple datasets, including datasets that are an order of magnitude smaller than those used in existing approaches, which show that proposed CLDP protocols yield high utility. In addition, case studies with Symantec datasets demonstrate that our protocols accurately support key cybersecurity-focused tasks of detecting ransomware outbreaks, identifying targeted and vulnerable OSs, and inspecting suspicious activities on infected machines.

1.1.3 LDP Protocol Analysis Through a Bayesian Lens

The third contribution of this dissertation research is the design of a principled framework for privacy and utility analysis of LDP protocols, and a prototype implementation of this framework called LDPLens. While the popularity of Local Differential Privacy (LDP) has led to the development of several LDP protocols, few have engaged in analyzing the privacy relationships of these protocols across different factors. In this dissertation, we propose a Bayesian adversary model and perform mathematical and empirical analysis of six representative LDP protocols under this adversary model (GRR, BLH, OLH, RAPPOR, OUE, SS). Our analysis shows that the Adversarial Success Rate (ASR) changes from protocol to protocol, and that there are multiple factors affecting ASR, including privacy budget ε, data domain, encoding parameters, adversarial background knowledge, and statistical data distribution. Consequently, we find that no single protocol is consistently better than others across all possible settings. We therefore develop a prototype system called LDPLens, which considers the aforementioned factors through customizable modules, and enables the selection of the most suitable LDP protocol for the given set of application-specific settings. We perform three real-world case studies with LDPLens, showing that the best protocol in each case is different, and using the protocol recommended by LDPLens rather than a random protocol may yield substantial reduction in utility loss or ASR.

1.2 Dissertation Organization

The rest of this dissertation is organized into several chapters as follows. Chapter 2 presents AdaTrace, our proposed location trace synthesizer with three features: differential privacy guarantee, resilience against relevant location trace attacks, and strong utility preservation. Chapter 3 presents the notion of Condensed Local Differential Privacy (CLDP) and its application to secure and utility-driven data collection through proposed CLDP protocols. Chapter 4 presents LDPLens: our system for Bayesian analysis of LDP protocols and optimized protocol and budget selection. Chapter 5 presents our concluding remarks and discusses the ongoing and future research directions.

1.3 Bibliographic Notes

Chapter 2 of this dissertation contains material from our conference and journal publications [12] and [17]. Chapter 3 of this dissertation contains material from our publication [18]. The implementations of the systems and protocols presented in each chapter are available at `https://github.com/git-disl`.

CHAPTER 2

UTILITY-AWARE SYNTHESIS OF DIFFERENTIALLY PRIVATE AND ATTACK-RESILIENT LOCATION TRACES

As mobile devices and location-based services become increasingly ubiquitous, the privacy of mobile users' location traces continues to be a major concern. Traditional privacy solutions rely on perturbing each position in a user's trace and replacing it with a fake location. However, recent studies have shown that such point-based perturbation of locations is susceptible to inference attacks and suffers from serious utility losses, because it disregards the moving trajectory and continuity in full location traces.

In this chapter, we argue that privacy-preserving synthesis of *complete* location traces can be an effective solution to this problem. We present AdaTrace, a scalable location trace synthesizer with three novel features: provable statistical privacy, deterministic attack resilience, and strong utility preservation. AdaTrace builds a generative model from a given set of real traces through a four-phase synthesis process consisting of feature extraction, synopsis learning, privacy and utility preserving noise injection, and generation of differentially private synthetic location traces. The output traces crafted by AdaTrace preserve utility-critical information existing in real traces, and are robust against known location trace attacks. We validate the effectiveness of AdaTrace by comparing it with three state of the art approaches (ngram, DPT, and SGLT) using real location trace datasets (Geolife and Taxi) as well as a simulated dataset of 50,000 vehicles in Oldenburg, Germany. AdaTrace offers up to 3-fold improvement in trajectory utility, and is orders of magnitude faster than previous work, while preserving differential privacy and attack resilience.

2.1 Introduction

A growing number of location-based services and applications, such as those offered by Google, Uber, Lyft, and Waze, continue to collect users' location traces in order to learn about their spatio-temporal movement patterns and offer life enriching, real-time experiences through location-based convenience and entertainment [8, 7, 19]. On the other hand, the sensitive nature of location trace data raises legitimate privacy concerns. Unauthorized exposure of users' private location traces may disclose their travel records, home and work locations, frequent meeting points, or visits to sensitive locations such as hospitals, health clinics, and religious events.

Traditional location privacy protection techniques have mostly focused on point-based location privacy, which is often achieved by perturbing or obfuscating each location point in a user's trace using a cloaked region or a fake location, with the goal of ensuring location k-anonymity or geo-indistinguishability [20, 21, 22, 23, 24, 25, 26, 27]. However, these point-based privacy mechanisms are insufficient for protecting the privacy of users' *trajectories*, i.e., spatially correlated, temporal sequences of locations. Several studies show that independent perturbation of each point-based location in a trajectory suffers from fatal shortcomings, including susceptibility to reverse engineering and inference attacks [28, 29]. In these attacks, adversaries observe a sequence of perturbed locations to infer a movement pattern and then link specific movement patterns with specific users. Such perturbations also suffer from acute spatial utility losses [30], and are vulnerable to known location trace attacks.

The aforementioned shortcomings motivated recent interest in synthesizing privacy-preserving, complete location traces [29, 31, 32]. These recent trace synthesis mechanisms focus on either differential privacy [31, 32] or plausible deniability [29], and while they offer desirable theoretical guarantees, they are limited by the extent of their chosen privacy definitions. For example, Dwork showed the impossibility to achieve absolute disclosure

prevention [33]. A resulting limitation is that differential privacy cannot bound *all* prior and posterior knowledge distributions of an adversary, which sparked interest in augmenting a Bayesian form of privacy with prior-to-posterior comparison [34, 35, 36]. This Bayesian property is clearly useful for location privacy, e.g., even if we cannot bound all possible knowledge that is disclosed to an adversary, a Bayesian formulation can be used to limit disclosure specifically in sensitive zones such as hospitals or schools. Another example is regarding outliers – it is well-known that differential privacy requires large noise amounts to mask the existence of outliers. Location traces are typically high-dimensional and contain diverse behavior. Hence, it is worthy to seek alternative attack resilience notions to combat privacy leakages concerning numerous kinds of outlier traces.

We argue that a practical solution to synthesizing privacy-preserving location traces should achieve three goals simultaneously: (i) robust, well-defined statistical privacy, (ii) deterministic resilience against location privacy attacks, and (iii) strong preservation of trajectory utility and authenticity. We present AdaTrace, a scalable and quantitative framework that provides utility-aware synthesis of differentially private and attack resilient location traces. AdaTrace builds a *generative* model over a dataset of real location traces by performing feature extraction, noise injection, and feature synthesis throughout four phases. In each phase, it first extracts utility-critical features and then allocates an adequate differential privacy budget to inject sufficient perturbation (noise), while maintaining attack resilience and preserving desired statistical and spatial utility. We guarantee that the synthetic trajectory generation process is differentially private, i.e., the generation of synthetic traces is not strongly dependent on any specific trace in the real trace dataset. This ensures unlinkability between an output trace and any real trace (input), thus thwarting identity and record linkage attacks. As such, although the location traces crafted by AdaTrace bear a statistical resemblance to real traces, they do not leak information about any individual that participates in the synthesis process.

We also ensure that the traces generated by AdaTrace are robust against three attacks:

Bayesian inference, partial sniffing, and outlier leakage. While the generation process is probabilistic to satisfy differential privacy, we enforce attack resilience by imposing deterministic constraints on the generated traces. Finally, by maintaining resemblance to real traces, the crafted traces preserve many useful statistical features that are in common with real traces, and therefore they can be effectively used in geo-spatial queries and geo-spatial data mining tasks.

2.1.1 Contributions

The design of AdaTrace is novel in three aspects. First, we identify three privacy threats that are relevant and common in trajectory data, and are neither addressed by differential privacy nor by other existing trajectory generators. We formalize these threats as Bayesian inference, partial sniffing, and outlier leakage attacks, and propose defenses for mitigation. We show that in many cases, our defenses can be realized with little or no aggregate utility loss, demonstrating their effectiveness. Second, we combine differential privacy with attack resilience to offer a two-layer privacy approach. This allows us to integrate statistical, algorithmic privacy with output privacy, enabling deterministic attack resilience in a probabilistic trace synthesis setting. Third, we design a utility-aware trace synthesizer by analyzing and categorizing various types of location trace utility and by devising noise-resilient utility extraction and preservation techniques.

We validate the effectiveness of AdaTrace by comparing it with three representative existing approaches (ngram [31], DPT [32], and SGLT [29]) using both real location trace datasets (Geolife [37] and Taxi [19]), as well as a dataset simulated by Brinkhoff simulator [38] of 50,000 vehicles in Oldenburg, Germany. We empirically measure how well each utility type is preserved, and our results show that AdaTrace offers up to 3-fold improvement in trajectory utility while preserving both differential privacy and attack resilience. AdaTrace is also computationally more efficient, as it is on average 1.5x faster than DPT, 8x faster than ngram, and 1000x faster than SGLT.

2.2 Related Work

We study the related work in three main directions: individual location perturbation, location trace anonymization, and synthetic location and trace generation.

2.2.1 Location Perturbation

Research in location privacy started a decade ago with the notion of location k-anonymity and two landmark results: uniform location k-anonymity [39] and user-defined, personalized location k-anonymity [20]. After the proliferation of differential privacy (DP), two stronger notions of location privacy were proposed based on statistical quantification of attack resilience. The first one is geo-indistinguishability [21], which employs DP by measuring the posterior information gain of an adversary based on the release of pseudo-locations of mobile users. The second one is the expected inference error [22, 23], which proposes a privacy notion powered by attack resilience to the prior knowledge of an adversary. A number of location perturbation mechanisms have been developed to satisfy these privacy notions [21, 24, 40, 23, 26]. However, these location perturbation mechanisms whose main functionality is to perturb an *individual* location point of the user, were shown to be ineffective when the goal is to protect the privacy of users' *complete* location traces, as they tend to suffer from inference attacks [29] as well as temporal and sequential correlation attacks [28, 41, 42].

2.2.2 Location Trace Anonymization

One approach to protecting users' location traces is through location trace anonymization and de-identification. Common approaches under this direction employ extensions of k-anonymity and ℓ-diversity. The (k, δ)-anonymity notion introduced in [43] ensures that at least k different trajectories exist in cylinders of radius δ. (k, δ)-anonymity is enforced via clustering and space translation. [44] enforces trajectory k-anonymity using location

generalization, and reduces the anonymization problem to time and space shifted alignment of location sequences. [45] introduces attack graphs to break traditional k-anonymity on trajectories using timestamps as quasi-identifying information, and proposes a Hilbert indexing method to circumvent such attacks. [46] applies $(K, C)_L$-privacy to thwart identity linkage and sensitive attribute disclosure attacks when trajectories contain sensitive information. However, recent studies on human mobility patterns [47, 48] show that individuals' location traces are highly unique and predictable, and location trace uniqueness decreases only slightly despite spatial coarsening (e.g., generalization). Recent reports on re-identification and de-anonymization attacks [49, 50, 51] further render the traditional methods such as anonymization, generalization and spatial cloaking insufficient to provide strong privacy protection.

2.2.3 Synthetic Location and Trace Generation

This line of work is inspired by the security research on generating fake or synthetic records for privacy protection in web search, anonymous communication, and authentication systems [52, 53, 54]. The main challenge is to generate synthetic data in a manner that resembles genuine user-produced data while offering practical privacy protection at the same time. In the context of location-based systems, some work has been on generating and injecting a synthetic location point within a user's trajectory [22, 55, 56]. A few recent projects studied the problem of generating synthetic trajectories [31, 32, 29]. ngram [31] and DPT [32] proposed to generate fake trajectories with differential privacy. Although offering strong statistical privacy guarantees, we argue that relying solely on differential privacy has two shortcomings. First, its probabilistic nature does not allow deterministic protection against targeted attacks such as those considered in this chapter. Second, it places restrictions on the generation algorithm but not its output, and therefore sensitive inferences remain possible [57, 58, 59]. In contrast to ngram and DPT, SGLT [29] enforces *plausible deniability* to generate privacy-preserving fake location traces. It first introduces

trace similarity and intersection functions that map a fake trace to a real trace under similarity and intersection constraints. Then, it generates one fake trace using one real trace as its seed. If the fake trace satisfies plausible deniability, i.e., there exist k other real traces that can map to the fake trace, then it is said to preserve privacy of the seed trace. We compare our AdaTrace system against ngram, DPT and SGLT in Section 2.6 and show that AdaTrace provides superior trajectory utility across a variety of utility metrics of interest.

2.3 AdaTrace Overview

2.3.1 Problem Statement

Consider a database of real location traces (trajectories) collected from mobile travelers on the road, denoted by D_{real}. We want to build a generative model over D_{real} using feature extraction, noise injection, and synopsis formation while upholding statistical utility and attack resilience. We then employ the generative model to output a set of synthesized traces, denoted by D_{syn}. This probabilistic generative model should be differentially private such that removing any real trace from D_{real} has no significant impact on the outcome D_{syn}. In addition, the synthetic traces D_{syn} should collectively retain a high resemblance to the real traces D_{real}, so that D_{syn} has many useful statistical and spatial features in common with D_{real}. Finally, both the generative model and the synthetic traces D_{syn} should be robust against inference attacks, in order to strengthen privacy by maximizing attackers' probability of error in estimating the true behavior of users over time.

We design and develop AdaTrace, a scalable and utility-aware trajectory synthesizer with differential privacy and attack resilience. To demonstrate its feasibility and effectiveness, we compare AdaTrace with existing state of the art mechanisms on both real and simulated location trace datasets [19, 37, 38]. AdaTrace methodically unites statistical privacy (e.g., differential privacy) and syntactic privacy (e.g., attack resilience). It maximizes privacy by mitigating location inference attacks, and minimizes utility loss by extracting and upholding many useful types of statistical and spatial features of real location traces

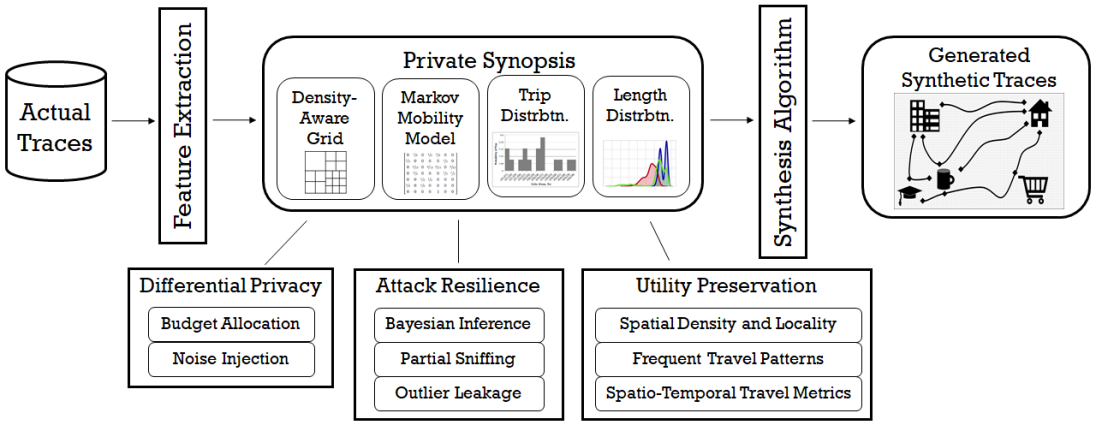


Figure 2.1: AdaTrace system architecture

throughout each phase of synthesis.

Figure 2.1 illustrates the system architecture of AdaTrace. First, AdaTrace analyzes D_{real}, the database of real location traces, to extract four spatial and statistical features and establish them in its private synopsis. The features include a density-aware grid to capture the complex spatial density and locality distributions; a Markov mobility model to capture intra-trace mobility patterns; a trip distribution to learn the correlations between trips' pickup and destination locations; and a length distribution to capture travel distances and durations. Each of the four features are perturbed by controlled noise under differential privacy and attack resilience constraints. Although several mechanisms can be used to add noise during each feature extraction and perturbation phase, in AdaTrace we carefully choose the mechanisms such that: (i) we find a balance in the trade-off between privacy, attack resilience and utility, and (ii) we maximize utility under a certain desired level of privacy and robustness.

2.3.2 Differential Privacy

We adapt the well-known notion of differential privacy to achieve statistical privacy in AdaTrace. We enforce differential privacy with respect to the complete location trace of an individual. It was recently argued that such use of the differential privacy notion can combat membership inference attacks relevant in machine learning as well as aggregate

location statistics [60, 61].

For an actual database D_{real}, let $nbrs(D_{real})$ denote the set of databases neighboring D_{real} such that for all $D'_{real} \in nbrs(D_{real})$, it holds that $(D_{real} - D'_{real}) \cup (D'_{real} - D_{real}) = \{T\}$, where T denotes one user's trajectory. Further, let $\mathcal{A}$ be a probabilistic algorithm, ε be the privacy parameter, and *Range*($\mathcal{A}$) denote the set of $\mathcal{A}$'s possible outcomes. $\mathcal{A}$ is said to be ε-differentially private [33], if for all D_{real} and $D'_{real} \in nbrs(D_{real})$, and for all possible outcomes $S \in$ *Range*($\mathcal{A}$):

$$Pr(\mathcal{A}(D_{real}) = S) \leq e^{\varepsilon} \times Pr(\mathcal{A}(D'_{real}) = S)$$

A special case of this definition is when $\mathcal{A}$ is a synthetic trajectory generation process and *Range*($\mathcal{A}$) $= \{D^1_{syn}, ..., D^n_{syn}\}$ is the set of all possible synthetic databases, yielding AdaTrace's differential privacy requirement. This requirement guarantees that the output of a private algorithm does not enable an adversary to distinguish, beyond a probability controlled by parameter ε, between two input databases D_{real} and D'_{real} that differ in one user's GPS trace. Hence, an adversary observing D_{syn} or an intermediate product of AdaTrace, including the features of the private synopsis, will not be able to infer the existence or content of a user's location data in D_{real} with strong confidence. Smaller ε yields tighter privacy [13].

We use two private building blocks in AdaTrace, namely the Laplace and Exponential mechanisms, for numeric and categorical queries respectively. Let f be a real-valued function $f : D_{real} \rightarrow \mathbb{R}^m$ that maps a database D_{real} into a fixed-size vector of m real numbers. The sensitivity of f, denoted Δf, is defined as the magnitude of the maximum L_1-norm change in the result of the function on all possible neighboring databases:

$$\Delta f := \max_{D_{real}, D'_{real}} ||f(D_{real}) - f(D'_{real})||$$

When answering a set of numeric (i.e., real-valued) queries, the Laplace mechanism re-

trieves the true answers of these queries, and then perturbs each answer by adding random noise scaled according to their sensitivity. That is, let $Lap(\alpha)$ denote a random variable sampled from the Laplace distribution with mean 0 and scale parameter α. The differentially private algorithm $\mathcal{A}$ answers f by [62]:

$$\mathcal{A}(f, D_{real}) = f(D_{real}) + (Y_1, ..., Y_m)$$

where Y_i are i.i.d. random variables drawn from $Lap(\Delta f/\varepsilon)$.

When the query is categorical instead of real-valued, the Exponential mechanism is used. It selects a discrete output r from a domain of outputs R in a private manner [63]. It employs a scoring function (i.e., quality criterion) q that associates each output $r \in R$ with a non-zero probability of being selected. Let $q(D_{real}, r)$ denote the quality of returning output r given database D_{real}, and let Δq be the sensitivity of q, defined similarly to Δf above. Then, the Exponential mechanism returns output $x \in R$ with probability equal to:

$$Pr(\mathcal{A}(q, D_{real}) = x) = \frac{e^{\frac{\varepsilon q(D_{real}, x)}{2\Delta q}}}{\sum_{r \in R} e^{\frac{\varepsilon q(D_{real}, r)}{2\Delta q}}}$$

The composition properties enable us to combine the building blocks and generate more sophisticated algorithms [64]. Because of the composition properties, ε is also called the *privacy budget*. Given a total budget ε, our goal is to create a sophisticated algorithm by cleverly combining the building blocks according to the composition properties. We employ three composition properties. First, for n algorithms $\mathcal{A}_1 \ldots \mathcal{A}_n$, each satisfying differential privacy with parameter $\varepsilon_1 \ldots \varepsilon_n$, the sequential execution of these algorithms on D_{real} consumes $\sum_{i=1}^{n} \varepsilon_i$ from the budget. Second, if the algorithms are executed on disjoint subsets of the database, the resulting execution consumes $\max(\varepsilon_i)$. Third, post-processing the output of a differentially private algorithm does not consume any budget or deteriorate privacy. For example, modifying the output of $\mathcal{A}_1$ without accessing D_{real}, and releasing the modified version still consumes ε_1.

2.3.3 Threat Model and Attacks

Differential privacy is regarded as the de facto standard for privacy-preserving data sharing due to its provable privacy guarantees. However, like any other security mechanism, the resilience of generic differential privacy against targeted, syntactic attacks has recently been criticized [57, 58, 59, 34]. We describe three attacks which are highly relevant in sharing sanitized location traces, but are beyond the scope of protection offered by differential privacy. We describe AdaTrace's threat model with respect to each attack and discuss why differential privacy is not sufficient to solve the threats.

Bayesian Inference Threat

Certain geographic zones (regions) are sensitive by nature, e.g., hospitals, schools, health clinics. Let Z denote a privacy sensitive zone. An adversary may have a prior belief $\mathcal{B}(Z)$ regarding the users who visit these zones, e.g., where they live, work, or which other locations they frequently visit. We allow informed and uninformed priors, but by default address stronger adversaries with *informed priors*, e.g., knowledge obtained from publicly available information such as census records and population averages. For example, if 10% of the general population lives in a certain neighborhood, then $\mathcal{B}(Z)$ may assume that 10% of the users who visit the hospital also live in that neighborhood.

The adversary formulates a posterior belief after observing the synthesized trajectories, denoted by $\mathcal{B}(Z|D_{syn})$. If the difference between $\mathcal{B}(Z)$ and $\mathcal{B}(Z|D_{syn})$ is significant, we assert that there is a privacy leakage. For example, the adversary observes from D_{syn} that 50% of the hospital's visitors live in a certain neighborhood rather than the assumed prior of 10%, yielding the inference that there is a flu outbreak in that neighborhood. Differential privacy is not sufficient to prevent this threat, because despite its noise injection, priors and posteriors will be related. For example, if indeed 50% of the hospital's visitors come from a certain neighborhood in D_{real}, after Laplace noise we may have a 48% visiting rate from the same neighborhood in D_{syn}. This is still significantly higher than the 10%

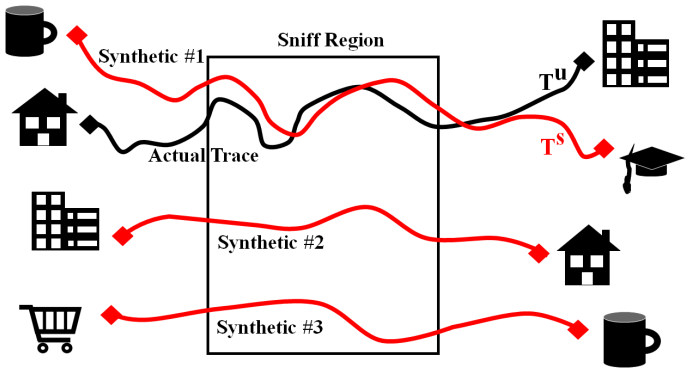


Figure 2.2: User's actual trace T_u drawn in black, synthetic traces passing through the sniff region drawn in red. Synthetic #1 is closest to T_u within the sniff region, hence it is labeled T_s.

prior. A proper defense should ensure that the difference between $\mathcal{B}(Z)$ and $\mathcal{B}(Z|D_{syn})$ is sufficiently small, so that an adversary's inference power is crippled.

Partial (Subtrajectory) Sniffing Threat

Assume that an adversary can track users when they are in certain regions, such as grocery stores or airports. We call these regions *sniff regions*. Tracking in sniff regions can be enabled via surveillance cameras, biometric scanners, or simply by following users' location check-ins on a social network. The sniffed locations constitute a subtrajectory (i.e., portion) of the user's full trajectory. Armed with subtrajectory knowledge, if the adversary can infer which trajectory belongs to the user with high probability, the adversary can learn the remaining locations visited by the user outside the sniff region, such as home and work locations. Let u denote the sniffed user and $T_u \in D_{real}$ denote her trajectory. Even when D_{syn} is shared in place of D_{real}, an adversary can launch this attack by linking the subtrajectory $T_s \in D_{syn}$ which is geographically most similar to T_u in the sniff region. This inference process is illustrated in Figure 2.2. The threat is especially acute if T_s and T_u share points in one or more sensitive zone.

Outlier Leakage Threat

A trajectory database may contain outliers that are unique in one or more ways such as the locations they visit, their start/destination points, and their time of travel. It is these outlier trajectories that are often hunted by adversaries. For example, while a $15 taxi ride in downtown Manhattan may not reveal much, a $125 trip from an isolated location to the airport leaks the identity and travel plans of a particular individual – note that this attack has actually been successfully exercised using NYC Taxi data [65].

We argue that relying solely on differential privacy cannot combat outliers for two reasons. First, since trajectories are often high-dimensional and unique [47, 48], there may exist skewed instances that are easily singled out and mapped to particular users with high confidence. Differential privacy needs to inject large noise amounts to hide such skewed instances. Second, even when large noise amounts are added, an outlier may still result from the probabilistic nature and algorithmic randomness of differential privacy. We should therefore place deterministic constraints to combat outliers. We combat an outlier trajectory T_{out}'s leakage threat by ensuring that it plausibly blends into a crowd of κ users' trajectories. This guarantees that T_{out} can at best be associated with a group of users (of size κ), and does not disclose the identity or secrets of any one user.

2.3.4 Utility Reference Model

The utility reference model used in AdaTrace is primarily designed based on the common utilities of location traces used in many geo-spatial data analysis and mining tasks, such as: users' frequency of visiting popular semantic locations, location entropy calculation, spatial decomposition, aggregate human mobility modeling (e.g., spatial, temporal and semantic mobility mining), and training predictive models for next location and destination inference. We identify the following three core utility notions as the prominent categories of trajectory utility in our reference model.

(1) Spatial density and locality distribution, e.g., for analyzing where users visit and

their popular points of interest. This can have various commercial benefits, including choosing ideal geographic placement for advertisements or retail stores, spatial hotspot discovery, and recommendation in location-based social networks.

(2) Spatio-temporal travel metrics, e.g., correlations between users' trip start and destination locations, how long their trips take, etc. Since real datasets often consist of trips such as taxi and Uber rides, trajectories have well-defined pickup and destinations. The commercial benefits of preserving the correlations and related statistics include taxi service prediction and passenger demand analysis, as well as governance benefits such as emergency response planning.

(3) Frequent and representative travel patterns, e.g., commonly preferred routes of travel. Preserving this information helps discover associations or sequentiality between road segments, ultimately benefiting road network performance analysis, traffic flow control, and path recommendation in navigation services.

Although these utility categories are listed separately, they can be used in conjunction to enable more sophisticated geo-spatial data analysis and machine learning tasks, some of which are mentioned above. A natural question here is *what if instead of synthesizing privacy-preserving traces, we kept the original traces without modification but governed data analysts' access with a differentially private querying interface?* This interactive approach suffers from several drawbacks. First, it places an additional burden on the data owner to create, maintain, and enforce a privacy-preserving querying interface. Second, it jeopardizes the analysts' ability to apply off-the-shelf ML tools or libraries, since these tools assume availability of raw records and are not compatible with noisy query interfaces. Third, our generated traces can be used in arbitrary tasks, including those that were unforeseen at time of generation. In contrast, the querying interface is an ad hoc solution whose functionality must be enhanced each time a new type of query or analysis is desired. Due to these factors, our non-interactive trace generation strategy has much higher benefit and convenience for all parties.

2.4 Synthetic Trajectory Generation

In this section, we will describe the features in AdaTrace's private synopsis, as well as its trajectory synthesis algorithm. For each of the features, we will discuss how noise and perturbation is injected so that privacy is preserved. A central privacy parameter here is ε, the differential privacy budget. Since AdaTrace's synopsis consists of four features as shown in Figure 2.1, ε is divided into four sub-budgets ε_1 to ε_4, one for each feature, such that $\sum_{i=1}^{4} \varepsilon_i = \varepsilon$.

2.4.1 Density-Aware Grid

Effective discretization of $\Omega(D_{real})$, the geographic location space of D_{real}, constitutes the first step towards preserving spatial densities. Without discretization, $\Omega(D_{real})$ is continuous and we have an unbounded number of possibilities to simulate a move from one location to another when generating a trajectory. To develop an efficient and accurate synthesis algorithm, we need to bound such transition possibilities by high utility choices. This motivates our design of a *grid* structure for space discretization. Although grids have been previously used in the location privacy literature [66, 67, 68], choosing an appropriate grid size and structure is not trivial under our privacy and utility constraints. If the grid is too coarse (e.g., 2x2), then each grid cell covers a large spatial area, and knowing that T visited a certain cell is uninformative. If the grid is too fine-grained (e.g., 50x50), we risk having many empty cells in which there are very few or no visits, and more additions of Laplace noise to each cell leads to higher inaccuracy as well as inefficiency.

We therefore implement a grid with density-adaptive cell granularity [68]. That is, for low density regions in $\Omega(D_{real})$ we place large, coarse cells; whereas for high density regions we divide the region into smaller cells with finer granularity. The grid is built in two levels. In the top-level, we lay an $N \times N$ grid with uniform cells. Depending on the density of these cells, in the bottom-level, we subdivide each cell into varying sizes

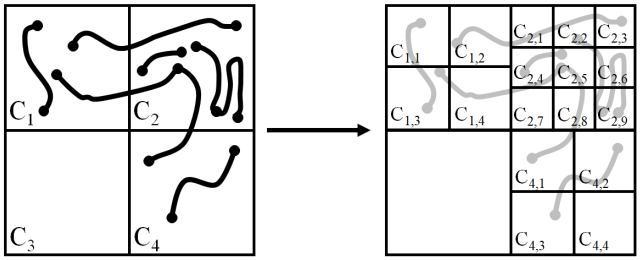


Figure 2.3: Two-step grid construction, top-level with $N = 2$ on the left and density-aware bottom-level on the right.

to reflect the density distribution of D_{real}. This allows us to *selectively* zoom into dense regions of $\Omega(D_{real})$ only when the need arises, which saves us from redundant computation and noise addition to sparse regions.

First, the grid is initialized with $N \times N$ identical cells, according to an input parameter N. This produces N^2 cells in the top-level, which we denote by $C_1, C_2, ..., C_{N^2}$. Then, we encode each trajectory in D_{real} as a list of cells, e.g., observe the trajectory $T : C_1 \rightarrow C_2 \rightarrow C_4$ in Figure 2.3. We denote by $|T|$ the number of cells T contains. Next, for each cell C_i, where $1 \leq i \leq N^2$, we count the *normalized* number of visits in that cell as follows:

$$g(D_{real}, C_i) = \sum_{T \in D_{real}} \frac{\text{\# of occurrences of } C_i \text{ in } T}{|T|}$$

We say that g is normalized since occurrences are divided by trajectory lengths. We need to find how much noise should be added to the query answers to satisfy differential privacy. In Theorem 1 we show that the sensitivity of the set of queries:

$$\boldsymbol{W} = \{g(D_{real}, C_1), g(D_{real}, C_2), ..., g(D_{real}, C_{N^2})\}$$

is $\Delta \boldsymbol{W} = 1$, therefore it suffices to add $Lap(1/\varepsilon_1)$ to each query answer to obtain the noisy answers. That is, the noisy ε_1-differentially private answer $\hat{g}(D_{real}, C_i)$ is computed as:

$$\hat{g}(D_{real}, C_i) = g(D_{real}, C_i) + Lap(1/\varepsilon_1)$$

Theorem 1. *The sensitivity of the query set* $\boldsymbol{W} = \{g(D, C_1), g(D, C_2), ..., g(D, C_{N^2})\}$ *is* $\Delta\boldsymbol{W} = 1$.

Proof. For the real trajectory dataset D, recall that a neighboring dataset D' implies either $D = D' \cup \{T\}$ or $D' = D \cup \{T\}$, where T denotes a single user's location trajectory. Without loss of generality we assume $D' = D \cup \{\tau\}$. We need to show $\Delta\boldsymbol{W} = \max_{D,D'} ||\boldsymbol{W}(D') - \boldsymbol{W}(D)|| = 1$. The derivation follows:

$$
\begin{aligned}
||\boldsymbol{W}(D') - \boldsymbol{W}(D)|| &= \sum_{C_i} (g(D', C_i) - g(D, C_i)) \\
&= \sum_{C_i} ((\sum_{T \in D} \cdot + \sum_{T \in \{\tau\}} \cdot) - \sum_{T \in D} \cdot) \\
&= \sum_{C_i} \sum_{T \in \{\tau\}} \frac{\text{\# of occurrences of } C_i \text{ in } T}{|T|} \\
&= \sum_{T \in \{\tau\}} \sum_{C_i} \frac{\text{\# of occurrences of } C_i \text{ in } T}{|T|} \\
&= \sum_{T \in \{\tau\}} \frac{1}{|T|} \sum_{C_i} (\text{\# of occurrences of } C_i \text{ in } T) \\
&= \sum_{T \in \{\tau\}} \frac{|T|}{|T|} = 1
\end{aligned}
$$

□

In the definition of g, we had normalized cell visits via dividing them by $|T|$. The intuition behind this can be observed from the above derivation. Without normalization, the last step reduces to $\sum_{T \in \{\tau\}} |T|$. In the general case this quantity cannot be bounded, leading to unbounded sensitivity $\Delta\boldsymbol{W}$. Previous works facing this problem resort to: (i) truncation, i.e., keep only the first T_{max} points of trajectories and remove the remaining points (where T_{max} is an input parameter), or (ii) random sampling, i.e., randomly choose T_{max} points from each trajectory and remove the rest [69, 70]. Both approaches yield $\Delta\boldsymbol{W} = T_{max}$. However, we argue that our normalization approach is more appropriate for our density measurement goal. The problem with truncating is that by removing all

points after T_{max}, we lose significant portions of trajectories that are much longer than T_{max}. This can be circumvented by having a high T_{max}, but doing so increases sensitivity, beating the purpose of bounding $\Delta \boldsymbol{W}$ in the first place. The problem with sampling is that a sample will randomly throw away many locations that could significantly contribute to spatial densities. Our approach ensures that all entries are properly considered in density calculation.

The final step in grid construction is to subdivide C_i using the noisy visit counts $\hat{g}(D_{real}, C_i)$. Each C_i is independently divided further into $M_i \times M_i$ cells, where M_i is proportional to $\hat{g}(D_{real}, C_i)$ and a grid constant. This achieves our goal of zooming into dense regions with large visit counts. For example, in Figure 2.3, we have densities $C_2 > C_1 \approx C_4 > C_3$, hence $M_2 = 3$, $M_1 = M_4 = 2$ and $M_3 = 1$. The grid constant acts as a balancing factor to determine the highest and lowest possible value of M, so that both M remains sensitive to changes in spatial density and the total number of cells resulting from this procedure is not excessively large.

2.4.2 Markov Chain Mobility Model

To generate high utility and realistic synthetic trajectories, we need to mimic actual intra-trajectory mobility. AdaTrace employs Markov chains for mobility modeling. A Markov chain of order r asserts that the next location in a trajectory depends on the previous r locations instead of all previous locations. Our grid-based discretization allows us to build a discrete state space Markov chain as follows. We map each cell in the adaptive grid to a state in our Markov chain. We assume each trajectory is represented as a time-ordered sequence of cells, and denote by T[j] the jth entry in T. Following the order r Markov property, we find the transition probability of T to a next cell C_{next} having observed its

previous n locations $T[1]T[2] \ldots T[n]$:

$$
\begin{aligned}
&Pr\Big(T[n+1] = C_{next} \mid T[1] \ldots T[n]\Big) \\
&\quad = Pr\Big(T[n+1] = C_{next} \mid T[n-r+1]T[n-r+2] \ldots T[n]\Big) \\
&\quad = \frac{\text{Number of sequences in } T \text{ containing } T[n-r+1]T[n-r+2] \ldots T[n]C_{next}}{\text{Number of sequences in } T \text{ containing } T[n-r+1]T[n-r+2] \ldots T[n]}
\end{aligned}
$$

We say that the trajectory-specific mobility model, $\Pi(T)$, is the collection of such probabilities $Pr\big(T[n+1] \mid T[1] \ldots T[n]\big)$. The trajectory-specific model captures the mobility of a single user in D_{real}. An aggregate mobility model for the whole D_{real} can be found by averaging the individual mobility models of each user. We slightly abuse notation and write $\Pi(D_{real})$ to denote the aggregate mobility model.

Similar to noise addition during grid construction, the aggregate model $\Pi(D_{real})$ is also perturbed with Laplace noise to satisfy differential privacy. We add noise to the Markov probabilities. Since Markov probabilities are computed using a ratio of sequence counts, and since these counts have sensitivity equal to 1, the required noise amount is limited. Hence, $\Pi(D_{real})$ can remain robust to noise.

2.4.3 Trip Distribution

Real life trace databases often consist of trips, such as taxi trips, Uber trips, home-work commutes, etc. The *trip distribution* aims to preserve the association between the start-end points of these trips when generating D_{syn}. This trip distribution is also useful from a technical sense to guide a random walk on the Markov model, because although we have a mobility model Π for intra-trajectory movement, we need a *start state* and an *end state* for specifying the two endpoints of our walk. Without the trip distribution we can assume a uniform distribution of start and end states, however, as Figure 2.4 shows, the trip distributions of real datasets are often heavily skewed. Thus, assuming a uniform distribution in place of the actual distribution is far from ideal, and will result in bogus

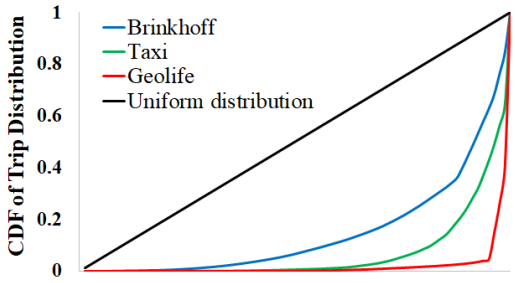


Figure 2.4: Cumulative Distribution Function (CDF) of the trip distribution $\mathcal{R}$ of 3 datasets, compared with the CDF of uniform distribution. For consistency we enforced a 3x3 uniform grid on all datasets and ordered the cell pairs on x-axis in non-decreasing trip frequency.

synthetic trips that jeopardize utility and authenticity.

Assume that we discretize $\Omega(D_{real})$ using grid $\mathbb{A}$. We denote a trip using its start cell C_{start} and destination cell C_{end}, as: $C_{start} \rightsquigarrow C_{end}$. Let $h(D_{real}, C_{start} \rightsquigarrow C_{end})$ be a function that computes the number of trips $C_{start} \rightsquigarrow C_{end}$ in database D_{real}, and $\hat{h}$ denote its private noisy version. For brevity we drop D_{real} from the notation and simply write $h(C_{start} \rightsquigarrow C_{end})$. Then, treating X as a random variable over the domain of the trip distribution $\mathbb{A} \times \mathbb{A}$, the entries in the trip distribution denoted $\mathcal{R}$ are calculated as:

$$Pr\Big(X = (C_{start}, C_{end})\Big) = \begin{cases} \frac{\hat{h}(C_{start} \rightsquigarrow C_{end})}{\sum_{C_i} \sum_{C_j} \hat{h}(C_i \rightsquigarrow C_j)} & \text{for } C_{start}, C_{end} \in \mathbb{A} \\ 0 & \text{otherwise} \end{cases}$$

Note that our definition ensures $\mathcal{R}$ is a probability mass function (pmf) since its entries sum to 1.

In the case of a two-layer grid where one GPS location is indexed by both a top-level and bottom-level cell, we can use constrained inference for improved accuracy and consistency [71, 68, 72]. In particular, we employ the following linear Ordinary Least Squares (OLS) approach. We denote by C_i a cell in the top-level of the grid, and by $C_{i,j}$ a cell in the bottom-level of the grid where $1 \leq j \leq M_i^2$ (see the notation in Figure 2.3). We use budget $\theta\varepsilon_3$ when obtaining top-level trip counts $\hat{h}(C_i \rightsquigarrow C_j)$, and budget $(1-\theta)\varepsilon_3$ when

obtaining bottom-level counts $\hat{h}(C_{i,k} \rightsquigarrow C_{j,l})$. Note that if we had no privacy or perturbation requirement, and we simply used the noise-free counts h instead of $\hat{h}$, the following would hold: $h(C_i \rightsquigarrow C_j) = \sum_k \sum_l h(C_{i,k} \rightsquigarrow C_{j,l})$. However, this may not hold after each h is perturbed with random noise. To re-establish consistency and minimize noise impact, OLS asserts that given the noisy values of $\hat{h}$, we can obtain optimized trip counts, denoted $\hat{h}'(C_i \rightsquigarrow C_j)$, as:

$$\hat{h}'(C_i \rightsquigarrow C_j) = \frac{\theta^2 M_i^2 M_j^2}{(1-\theta)^2 + \theta^2 M_i^2 M_j^2} \cdot \hat{h}(C_i \rightsquigarrow C_j) + \frac{(1-\theta)^2}{(1-\theta)^2 + \theta^2 M_i^2 M_j^2} \cdot \sum_k \sum_l \hat{h}(C_{i,k} \rightsquigarrow C_{j,l})$$

When optimizing the counts for the bottom-level, differences in optimized top-level counts calculated above are distributed equally among the bottom-level cells:

$$\hat{h}'(C_{i,k} \rightsquigarrow C_{j,l}) = \hat{h}(C_{i,k} \rightsquigarrow C_{j,l}) + \frac{\hat{h}'(C_i \rightsquigarrow C_j) - \sum_s \sum_t \hat{h}(C_{i,s} \rightsquigarrow C_{j,t})}{M_i^2 M_j^2}$$

Finally, optimized trip counts $\hat{h}'$ are used in place of $\hat{h}$ in the definition of $\mathcal{R}$.

Example: We now demonstrate the intuition behind our OLS approach and illustrate it with a worked example. Let C_1 and C_2 be two grid cells in the top-level of AdaTrace's adaptive grid. Let $M_1 = 2$ and $M_2 = 3$, i.e., in the bottom-level, C_1 is divided into 4 cells (denoted $C_{1,1}$, $C_{1,2}$, $C_{1,3}$, $C_{1,4}$) and C_2 is divided into 9 cells (denoted $C_{2,1} \ldots C_{2,9}$). Without noise, we trivially have: $h(C_1 \rightsquigarrow C_2) = \sum_{k=1}^{4} \sum_{l=1}^{9} h(C_{1,k} \rightsquigarrow C_{2,l})$. That is, the total is consistent, and equals the sum of its parts. For example, observe the trip counts in Figure 2.3: $h(C_1 \rightsquigarrow C_1) = 1$ at the top-level (LHS), and $h(C_{1,1} \rightsquigarrow C_{1,3}) = 1$ at the bottom-level (RHS). However, with random Laplace noise, consistency is no longer guaranteed, and for noisy counts we may have: $\hat{h}(C_1 \rightsquigarrow C_2) \neq \sum_{k=1}^{4} \sum_{l=1}^{9} \hat{h}(C_{1,k} \rightsquigarrow C_{2,l})$.

Our goal in using OLS is to re-establish consistency by post-processing $\hat{h}$ and obtaining $\hat{h}'$, which denotes a consistent and optimized trip count. $\hat{h}'$ values are then used in the

definition of the trip distribution $\mathcal{R}$, in place of $\hat{h}$. We establish consistency by defining $\hat{h}'(C_1 \rightsquigarrow C_2)$ as a linear combination of $\hat{h}(C_1 \rightsquigarrow C_2)$ and $\sum_{k=1}^{4}\sum_{l=1}^{9}\hat{h}(C_{1,k} \rightsquigarrow C_{2,l})$. A natural first attempt towards linear combination is averaging:

$$\hat{h}'(C_1 \rightsquigarrow C_2) = \frac{1}{2}\hat{h}(C_1 \rightsquigarrow C_2) + \frac{1}{2}\sum_{k=1}^{4}\sum_{l=1}^{9}\hat{h}(C_{1,k} \rightsquigarrow C_{2,l})$$

However, observe that the noise variance in this case is $Var(\hat{h}') = {}^{1}/_{4}Var(\hat{h}) + 36({}^{1}/_{4})$ $Var(\hat{h}) = {}^{37}/_{4}Var(\hat{h})$, significantly higher than $Var(\hat{h})$, which means using the original value $\hat{h}$ is better than computing $\hat{h}'$. This is caused by two factors. First, averaging does not take into account the possibly uneven privacy budget assigned when retrieving top-level counts versus bottom-level counts. (Recall that top-level counts are retrieved with budget $\theta\varepsilon_3$, whereas bottom-level counts are retrieved using budget $(1-\theta)\varepsilon_3$.) Second, averaging does not take into account the difference in the number of times noise is added to each of the two parts – according to the properties of variance, the sum of 36 i.i.d. Laplace random variables has higher variance than 1.

The following OLS approach addresses these shortcomings and calculates $\hat{h}'$ as follows:

$$\begin{aligned}\hat{h}'(C_1 \rightsquigarrow C_2) = {} & \frac{36\theta^2}{(1-\theta)^2 + 36\theta^2} \cdot \hat{h}(C_1 \rightsquigarrow C_2) \\ & + \frac{(1-\theta)^2}{(1-\theta)^2 + 36\theta^2} \cdot \sum_{k=1}^{4}\sum_{l=1}^{9}\hat{h}(C_{1,k} \rightsquigarrow C_{2,l})\end{aligned}$$

It can be verified that $Var(\hat{h}'(C_1 \rightsquigarrow C_2)) < Var(\hat{h}(C_1 \rightsquigarrow C_2))$, which improves accuracy. This computation is bottom-up in nature, since top-level count is optimized using bottom-level counts. As a final step, consistency requires that the sum of the bottom-level counts equals the top-level count. To achieve this, we employ a top-down approach by distributing the acquired difference $\delta_{OLS} = \hat{h}'(C_1 \rightsquigarrow C_2) - \sum_{m=1}^{4}\sum_{n=1}^{9}\hat{h}(C_{1,m} \rightsquigarrow C_{2,n})$ equally among all bottom-level counts. The resulting post-processed bottom level counts

in our example become:

$$\hat{h}'(C_{1,k} \rightsquigarrow C_{2,l}) = \hat{h}(C_{1,k} \rightsquigarrow C_{2,l}) + \frac{\delta_{OLS}}{36}$$

2.4.4 Route Length Distribution

The final component in AdaTrace's private synopsis consists of fitted distributions of route lengths. We fit a different theoretical distribution for each different trip $C_{start} \rightsquigarrow C_{end}$. The distribution is learned based on observed route lengths in D_{real} that make this trip. Our rationale in enforcing trip-specific length distributions is that trajectories travelling between certain regions may take more indirect routes than others, for reasons such as unavailability of roads, traffic avoidance, and so on. Hence, trip-specific length distributions will be more accurate than global distributions (learned using whole D_{real}) used in previous works [73, 74].

We treat observed route lengths as a histogram. We consider multiple well-known distributions with different shapes, such as uniform, exponential and Poisson distributions, as candidates to capture the shape of our histogram. An example is shown in Figure 2.5. The candidate distributions have one characteristic in common: Their parameters are directly related to aggregate summary statistics that can be privately and accurately obtained from D_{real}. For example, the Poisson distribution has a single parameter that is equal to the mean length, whereas the λ parameter of the exponential distribution is related to the median length med as: $\lambda = {}^{ln2}/_{med}$. We then use the Laplace and Exponential mechanisms to privately fetch statistics such as means and medians. A private mean can be fetched by decomposing it into a noisy sum divided by a noisy count, where the Laplace mechanism is used to inject noise. A private median can be fetched using Cormode et al.'s adaptation of the Exponential mechanism with scoring function $q := -|rank(x) - rank(med)|$ [72]. The mechanism returns noisy median x instead of the actual median, and the intuition behind this particular q is that if x is close to the actual median, then its rank will be similar to

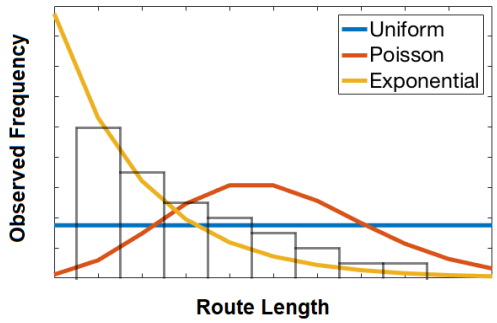


Figure 2.5: Comparing frequencies of observed route lengths (histogram) with candidate distributions.

the median's rank. Thus, the score of each candidate is negatively impacted by how much its rank deviates from $rank(med)$.

After building multiple candidate distributions as above, we select one distribution as the best fit for trip $C_{start} \rightsquigarrow C_{end}$, store it in AdaTrace's memory, and drop the remaining. We use a *goodness of fit* test to determine which distribution is the best fit. While there are several tests to measure goodness of fit, we use the value of the χ^2 test statistic since its differentially private implementation is well known [75]. For example, in Figure 2.5 the goodness of fit test would select the exponential distribution as the best fit, since its shape is closest to the shape of the histogram.

2.4.5 Trajectory Synthesis Algorithm

AdaTrace's synthesis algorithm combines the four features in its private synopsis: the density-aware grid $\mathbb{A}$, the mobility model $\Pi(D_{real})$, the trip distribution $\mathcal{R}$ and the collection $\mathcal{L}$ of length distributions for each trip. It outputs synthetic trajectories based on the skeleton presented in Algorithm 1.

The algorithm can be studied in four steps. First, it determines the start and end points of the synthetic trajectory T_{syn} by sampling from the trip distribution. Second, it determines the length of T_{syn} by sampling from the appropriate route length distribution in $\mathcal{L}$. Third, it initializes T_{syn} with first location as the starting cell of the trip, and last location

Algorithm 1: Trajectory synthesis algorithm

Input : Grid $\mathbb{A}$, trip distribution $\mathcal{R}$, mobility model Π, length distributions $\mathcal{L}$
Output: A candidate synthetic trajectory T_{syn}

Pick a random sample (C_{start}, C_{end}) from pmf of $\mathcal{R}$
Retrieve from $\mathcal{L}$ the fitted probability distribution PD for trip $C_{start} \rightsquigarrow C_{end}$
Pick a random sample ℓ from PD
Initialize T_{syn} with $T_{syn}[1] = C_{start}$ and $T_{syn}[\ell] = C_{end}$
for $i = 2$ *to* $\ell - 1$ **do**
    for $C_{cand} \in \mathbb{A}$ **do**
        Retrieve from Π: $w_1 = Pr\Big(T[\text{i}] = C_{cand} \mid T[1] \dots T[\text{i-1}]\Big)$ and
        $w_2 = Pr\Big(T[\ell] = C_{end} \mid T[1] \dots T[\text{i-1}] C_{cand}\Big)$
        Set the weight of C_{cand} equal to $w_1 \cdot w_2$
    end
    Sample C_{chosen} from $\mathbb{A}$ with probability proportional to its weight calculated above
    Set $T_{syn}[\text{i}] = C_{chosen}$
end
return T^{syn}

as the destination cell of the trip. Fourth, given the two endpoints of T_{syn}, intermediate locations are found using a random walk on the mobility model Π. When determining the i'th position of T_{syn} considering the cells of grid $\mathbb{A}$ as candidates, each candidate is given a *weight* that consists of two sub-weights denoted w_1 and w_2. w_1 performs a look-back and finds the probability that the next location is C_{cand} given the previous locations, as in the straightforward application of the Markov assumption. This is essentially a one-step transition probability. On the other hand, w_2 performs a look-ahead and finds the probability that the final location is C_{end} given the previous locations and assuming the current location is set to C_{cand}. This is an $(\ell - i)$-step transition probability, which is computed using a combination of 1-step transition probabilities. To improve efficiency, we pre-compute multiple-step transition probabilities after learning Π so that the same computation is not repeated for different T_{syn}.

The above generates a trajectory for a *single* trip between well-defined start and end locations. This is sufficient when each user's GPS record in D_{real} corresponds to a short-

term trip, such as an Uber or taxi ride. However, if D_{real} is collected over long periods of time (e.g., several days), then a user's record may contain multiple trips. In this case, the synthesis algorithm can be run multiple times per user, such that in each iteration, the starting location of the next trajectory is equal to the last known location of the previous trajectory. Then, these trajectories can be stitched together to form the final GPS record of the user with a desired number of trips.

2.5 Privacy Analysis

AdaTrace has differential privacy and attack resilience as its two privacy goals. In this section we give concrete definitions and privacy parameters with which AdaTrace fulfills these goals. Our definitions are parameterized to enable the data owner to explicitly control the leakage potential of synthetic trajectories by specifying the desired privacy parameters.

2.5.1 Differential Privacy Preservation

AdaTrace satisfies ε-differential privacy as a whole. Recall that we treat ε as the total privacy budget and distribute it to four sub-budgets (one for each feature in the synopsis) such that $\sum_{i=1}^{4} \varepsilon_i = \varepsilon$. Learning and perturbing a feature consumes the ε_i allocated to it, thus depleting the total ε after the perturbation phase is complete, according to the sequential composition property. Then, any additional data-independent perturbation performed to satisfy attack resilience counts as post-processing. Note that sequential composition holds even when subsequent computations incorporate the outcomes of preceding computations [64], hence the fact that AdaTrace uses the perturbed grid in subsequent steps (Markov chain, trip distribution) does not violate ε-differential privacy. Also note that AdaTrace's synthesis algorithm performs sampling and calculation on perturbed features without accessing the actual database D_{real}. As a result, AdaTrace remains ε-differentially private.

Distribution of ε into ε_i can be done either by AdaTrace automatically or according to the specifications of the data owner. The current implementation of AdaTrace comes

with a default budget distribution, which we empirically determined to yield high average utility: $\varepsilon_1 = \varepsilon/9$ for the grid, $\varepsilon_2 = 4\varepsilon/9$ for the Markov mobility model, $\varepsilon_3 = 3\varepsilon/9$ for the trip distribution and $\varepsilon_4 = \varepsilon/9$ for the length distribution. We can observe from this that not all features require high budgets to be accurate, and allocating an unnecessarily high budget to such features will negatively impact utility since it would steal from those features that do require high budgets to remain accurate. For example, the grid and length distribution components need lower budgets in comparison to others. One way to further optimize budget distribution is to leverage regression-based learning on D_{real} and ε, a plan for future work.

Although the above strategy is beneficial from a utility maximization perspective, we should also discuss its privacy implications. Since each individual feature will satisfy ε_i-differential privacy, a higher ε_i will cause the feature to be accurately preserved, whereas a lower ε_i will cause it to be more perturbed. A variable budget distribution allows the data owner (e.g., service provider) to distribute ε in a way that reflects which artifact he perceives is more sensitive and should thus be more perturbed to protect its privacy. For example, if the data owner feels that spatial densities are more sensitive, then a lower ε_1 can be assigned to grid construction. If trip distributions are sensitive, a lower ε_3 can be assigned, causing a more perturbed trip distribution. This provides flexibility with respect to different perceptions regarding what needs to be protected.

2.5.2 Enforcement of Attack Resilience

We now discuss how AdaTrace ensures attack resilience against the 3 privacy threats listed in Section 2.3.3.

Bayesian Inference Threat

Recall that we denote by Z a sensitive zone such as a hospital, health clinic or religious place, by $\mathcal{B}(Z)$ the prior belief of an adversary regarding the users who visit Z, and by

$\mathcal{B}(Z|D_{syn})$ the posterior belief having observed D_{syn}. The threat stems from $\mathcal{B}(Z|D_{syn})$ being significantly different than $\mathcal{B}(Z)$. To combat the threat, AdaTrace enforces the following defense.

Defense 1 (ϑ)**.** *Denoting by EMD the Earth Mover's Distance, we say that D_{syn} is attack-resilient if the following holds, and susceptible otherwise.*

$$EMD\big(\mathcal{B}(Z), \mathcal{B}(Z|D_{syn})\big) \leq \vartheta$$

Intuitively, the privacy guarantee is that upon observing D_{syn}, an adversary's belief regarding users visiting Z does not change significantly, where the level of significance is controlled by parameter ϑ. This prohibits both positive and negative disclosures, i.e., the adversary is prohibited not only from learning that significantly higher than 10% of Z's visitors live in a certain neighborhood, but also that significantly lower than 10% of Z's visitors live in a certain neighborhood. The latter is useful when, for instance, Z is a university and the adversary may infer the lack of education.

AdaTrace follows an iterative perturbation strategy to implement Defense 1. Let D_Z denote the subset of trajectories that visit zone Z. We perturb the features concerning D_Z in the private synopsis, e.g., $\Pi(D_Z)$, $\mathcal{R}(D_Z)$, $\mathcal{L}(D_Z)$. In the most relaxed setting of $\vartheta = +\infty$ no perturbation is necessary; whereas in the strictest setting $\vartheta = 0$, the features must be maximally perturbed so that they exactly equal population averages. For any ϑ in between, we perturb the features iteratively, converging to population averages in each iteration, until Defense 1 is satisfied. After the defense is satisfied, we plug the perturbed features back to the private synopsis.

Partial (Subtrajectory) Sniffing Threat

Armed with user u's subtrajectory from the sniff region, the adversary aims to link u with a synthetic trajectory $T_s \in D_{syn}$. Since the user's actual trajectory T_u is not in D_{syn} the link-

age is not genuine, however the threat stems from two cases: (i) T_s sufficiently resembles T_u, thus the adversary can still make correct inferences using T_s. (ii) T_s contains visits to a sensitive zone Z – although u may or may not have visited Z in reality, such an inference can have discriminating or harmful impact on u, and therefore should be prohibited.

Defense 2 (φ, ϱ)**.** *Let SR denote the sniff region, $T_{sniffed}$ denote the sniffed subtrajectory, Z denote a sensitive zone, and visit(T, Z) denote the number of times trajectory T visits zone Z.*

1. *Find the matching synthetic trajectory:*

$$T_s = \underset{T \in D_{syn}}{\operatorname{argmin}} \; \textit{DTW}((T \cap SR), T_{sniffed})$$

2. *If $|$intersection$(T_s, T_u)| > \varphi$, mark as susceptible.*

3. *If visit$(T_s, Z) > \varrho$, mark as susceptible.*

4. *If no susceptibility is found in the above steps, mark as attack-resilient.*

AdaTrace enforces the above defense to thwart the partial sniffing threat. Its first step is to identify T_s using Dynamic Time Warping (DTW) as the geographical distance metric, which is a prominent method for measuring trajectory distance [76, 77, 78]. Then, in step 2, we prohibit correct location disclosures. Since exactly matching GPS coordinates and trajectories are rare, we allow for inexact intersections within a small error margin. φ is a threshold controlling the maximum amount of intersection allowed between T_s and T_u. It can be defined using absolute length (e.g., 100 meters, 1 kilometer) or using percentage of relative trajectory length (e.g., 10% of T_s). In step 3, we prohibit sensitive location disclosures. To prohibit *any* sensitive disclosure, we set $\varrho = 0$, which is suitable when trajectories are short, e.g., corresponding to taxi trips or home-work commutes. If trajectories are collected over long periods of time such as one month, it is advisable to set higher ϱ.

Outlier Leakage Threat

In the outlier leakage attack, the threat stems from an adversary's ability to make sensitive inferences from atypical, outlier trajectories. To combat this threat, we must first identify and detect outliers. We adapt the most popular distance-based outlier definition [79, 80]: Outliers are those trajectories that have highest distance to their respective k'th nearest neighbor. It remains to define an appropriate distance function d so that distances between trajectories can be measured, and the nearest neighbors of each trajectory (according to d) can be identified. AdaTrace considers three distance functions, each of which corresponds to a different *type* of outlier:

(1) Trip outlier: Trajectories with non-traditional trip start and end locations. In this case, d is set as the geographical distance between trajectories' trip start and end locations.

(2) Length/duration outlier: Trajectories with unusually long or short lengths. In this case, d is set to measure the difference between two trajectories' route lengths.

(3) Mobility outlier: Trajectories with unusual mobility patterns, in which case d is set to measure the Jensen-Shannon Divergence (JSD) between the mobility models of two trajectories: $d(T_1, T_2) = \text{JSD}(\Pi(T_1), \Pi(T_2))$.

The distance functions above allow us to compute distances between pairs of trajectories and detect whether a trajectory is a certain type of outlier based on distance to its k'th nearest neighbor. After detecting outliers, AdaTrace enforces Defense 3.

Defense 3 (β, κ)**.** *Let $T_{out} \in D_{syn}$ be a distance-based outlier, and d be the distance function used in detecting T_{out}.*

1. *Find the trajectory in D_{real} that is most similar to T_{out} with respect to d:*

$$T_{akin} = \underset{T \in D_{real}}{\operatorname{argmin}}\ d(T_{out}, T)$$

2. *Let y be the number of trajectories $T_{sim} \in D_{real}$ that satisfy:*

$$d(T_{out}, T_{sim}) - d(T_{out}, T_{akin}) \leq \beta$$

3. *If $y \geq \kappa$, mark as attack-resilient, otherwise susceptible.*

The intuition behind this defense is as follows. We first identify T_{akin}, the real trajectory most similar to T_{out}. We then check how many real trajectories exist which have distance similar to T_{akin}'s distance to T_{out}, and denote this quantity by y. This effectively searches for a crowd of y trajectories with maximum distance β to the candidate outlier. Note that we always have $y \geq 1$ due to $T_{sim} = T_{akin}$ in step 2. Finally, if $y \geq \kappa$, where κ is the privacy parameter enforcing a minimum crowd size, we conclude that T_{out} plausibly blends in a crowd of actual trajectories. Hence, the trajectory is similar to a *group* of actual users' trajectories, it cannot be conclusively linked to a particular user, and outlier leakage is not a concern. However, if $y < \kappa$, i.e., T_{out} does not blend into a group of trajectories, then T_{out} must be discarded and replaced with a non-outlier trajectory.

2.6 Experimental Evaluation

2.6.1 Experiment Setup

Implementation Details and Competitors: We implemented AdaTrace in Java. We compare it with 3 state of the art location trace generators: ngram, DPT and SGLT. We obtained their implementations from the respective authors [29, 31, 32]. In our comparison we use the parameters recommended by the authors when applicable, otherwise we experiment with different parameter values and report only the best results. Since all generators are probabilistic, each experiment is repeated 5 times and results are averaged.

We note the configuration used for ngram and SGLT: For ngram, although it is originally designed for sequential data generation, it has been used in [32] for comparison against DPT and in our experiments we adopt their setting, i.e., we discretize trajectories

Table 2.1: Datasets used in our experiments (μ:mean, σ:stdev)

	$\lvert D_{real}\rvert$	$\Omega(D_{real})$	$\lvert T\rvert$ ($\mu \pm \sigma$)	GPS sampling rate
Geolife	14,650	Beijing	931.4 ± 1602.5	$\sim$ 2.5 sec
Taxi	30,000	Porto	43.1 ± 33.5	$\sim$ 15 sec
Brinkhoff	50,000	Oldenburg	64.6 ± 35.9	$\sim$ 15.6 sec

at different granularities, run ngram on discretized sequences and convert output synthetic sequences back to trajectories. We repeat this process at different granularities of discretization, and use the results of the granularity that yields highest accuracy. For SGLT, it is required that the input set of trajectories must be of same duration. In real life and in our datasets, trajectories vary in duration and sampling rate. To be able to run experiments with SGLT, we repeated the last known locations of shorter trajectories several times so that all trajectories would have equal duration.

Datasets: We used three datasets in our experiments: Geolife, Taxi and Brinkhoff. Geolife and Taxi are real datasets, whereas Brinkhoff is simulated. Information regarding the datasets is given below and in Table 2.1. Geolife was collected and released by Microsoft as part of the Geolife project [37]. It contains GPS traces of 182 users over 5 years. Majority of the data is from Beijing, China, with few outliers in other cities in China, Europe, etc. We pre-processed the data to remove these outliers and obtain a more even data distribution and well-defined $\Omega(D)$. The resulting dataset contained 14,650 trajectories. Taxi contains GPS traces of taxis operating in the city of Porto, Portugal. We extracted 30,000 trips from the denser areas in the dataset made available as part of the Taxi Service Prediction Challenge at ECML-PKDD 2015 [19]. Brinkhoff contains trajectories of vehicles simulated using Brinkhoff's network generator for moving objects [38]. The map of Oldenburg, Germany was used to simulate movements of 50,000 vehicles. Locations were sampled at equal time intervals.

2.6.2 Evaluation Metrics

We generated synthetic databases D_{syn} with cardinality $|D_{syn}| = |D_{real}|$, and used the following metrics to quantify the difference (or resemblance) between D_{real} and D_{syn}. When keeping the privacy level constant, lower difference (resp. higher resemblance) is desired for improved utility. According to Section 2.3.4, the three trajectory utility categories we aimed to preserve in AdaTrace were spatial densities and localities, frequent travel patterns, and spatio-temporal travel metrics. The evaluation metrics we present in this section fall under these categories, as described below.

Spatial Density and Locality Metrics

A large number of geodata analysis tasks, including PoI and popular location extraction, hotspot discovery, heatmaps and recommendation engines rely on spatial densities. We employ the following two metrics to measure the quality of spatial density and locality resemblance.

Our first metric is *query error*, which is a popular measure for evaluating data synthesis algorithms ranging from tabular data to location and graph data [31, 81, 82]. We consider spatial counting queries of the form: "Retrieve the number of trajectories passing through a certain region R". Let Q denote a query of this form, and $Q(D)$ denote its answer when issued on database D. The relative error (RE) of Q is defined as:

$$RE = \frac{|Q(D_{real}) - Q(D_{syn})|}{max\{Q(D_{real}), b\}}$$

where b is a sanity bound to mitigate the effect of extremely selective queries. We set $b = 1\% \times |D|$. We generate 500 random queries by choosing their regions R uniformly at random and compute average relative error (AvRE) by averaging the RE of all queries.

Our second metric is for measuring the discrepancies in locations' popularity ranking. Let $L_1, ..., L_n$ be a list of locations and let $pop(D, L_i)$ measure the popularity of L_i,

i.e., number of times L_i is visited by trajectories in D. We compute $pop(D_{real}, L_i)$ and $pop(D_{syn}, L_i)$ for all L_i. Then, we say that a pair of locations (L_i, L_j) are *concordant* if either of the following hold:

$$(pop(D_{real}, L_i) > pop(D_{real}, L_j)) \wedge (pop(D_{syn}, L_i) > pop(D_{syn}, L_j))$$
$$(pop(D_{real}, L_i) < pop(D_{real}, L_j)) \wedge (pop(D_{syn}, L_i) < pop(D_{syn}, L_j))$$

That is, their popularity ranks (in sorted order) agree. They are said to be *discordant* if their ranks disagree. The Kendall-tau coefficient can then be applied as:

$$KT = \frac{\text{(\# of concordant pairs)} - \text{(\# of discordant pairs)}}{n(n-1)/2}$$

In our experiment, we determine the locations $L_1, ..., L_n$ using a fine-grained grid and use $n = 400$ locations.

Frequent Travel Pattern Metrics

Mining popular travel patterns has also received significant attention from the geodata analysis community, for understanding traffic flows and road network performance, and providing better navigation services. The following two metrics measure resemblance with respect to frequent patterns (FPs). For both metrics, we project trajectories on a uniform grid $\mathcal{U}$, thus obtaining the sequence of cells they pass through. We define a pattern P as an ordered sequence of cells, e.g., $P : C_2 \rightarrow C_4 \rightarrow C_3$. We define the support of a pattern, $supp(D, P)$, as the number of occurrences of P in database D. We mine the top-k patterns in D, i.e., the k patterns with highest support, denoted by $\mathcal{F}^k_{\mathcal{U}}(D)$.

Our first metric measures difference in patterns' support. We calculate the relative difference between $supp(D_{real}, P)$ and $supp(D_{syn}, P)$ for each frequent pattern $P \in \mathcal{F}$.

Formally, the FP average relative error is:

$$FP\ AvRE = \frac{\sum_{P \in \mathcal{F}_{\mathcal{U}}^k(D_{real})} \frac{|supp(D_{real},P) - supp(D_{syn},P)|}{supp(D_{real},P)}}{k}$$

Our second metric measures the set similarity between the top-k patterns in D_{real} and the top-k patterns in D_{syn}. Following the F1-measure, i.e., harmonic mean of precision and recall, we define the FP similarity metric as:

$$FP\ F1\ Similarity = \text{F1}(\mathcal{F}_{\mathcal{U}}^k(D_{real}), \mathcal{F}_{\mathcal{U}}^k(D_{syn}))$$

FP similarity score is between 0 and 1, where higher score implies better set preservation, and is therefore more desired. For the AvRE metric, lower error values are more desired. In our experiments we use $k = 100$, as higher k returned patterns with insignificant support. Also, we assume a 6x6 uniform grid for $\mathcal{U}$ and only consider patterns that are at least 3 cells long.

Spatio-Temporal Travel Metrics

Since many trajectory databases consist of taxi/Uber rides or daily commutes, analysis of spatio-temporal features of these trips becomes critical. We employ 3 evaluation metrics for trip and travel analysis.

Our first metric, called *trip error*, measures how well the correlations between trips' start and end regions are preserved. We make use of the empirical trip distribution $\mathcal{R}$ defined in Section 2.4.3. Given the grid $\mathcal{U}$, we compute the trip distributions of the real and synthetic databases, denoted $\mathcal{R}(D_{real})$ and $\mathcal{R}(D_{syn})$ respectively. The trip error is defined as: JSD$(\mathcal{R}(D_{real}), \mathcal{R}(D_{syn}))$ where JSD is the Jensen-Shannon divergence.

Our second metric, called *length error*, measures the error in trip lengths (i.e., distances travelled in each trip). We calculate the total distance travelled in a trip by adding up the distance between each consecutive GPS reading. Upon learning the maximum length from

D_{real}, we quantize trip lengths into 20 equi-width buckets: $\{[0, x), [x, 2x), ..., [19x, 20x]\}$, where $20x$ is the longest length present in D_{real}. For each bucket we determine how many trips' length fall into that bucket, thereby obtaining a histogram of lengths. Let $\mathcal{N}(D_{real})$ and $\mathcal{N}(D_{syn})$ denote this empirical, bucketized histogram of real and synthetic databases respectively. The length error is calculated as: JSD($\mathcal{N}(D_{real}), \mathcal{N}(D_{syn})$).

Our final metric, called *diameter error*, is adopted from [32]. The diameter of a trajectory T is defined as the maximum distance between any pair of its GPS readings (not necessarily consecutive). Similar to length error, we learn the maximum diameter from D_{real}, perform equi-width bucketing and histogram extraction. Let $\mathcal{E}(D_{real})$ and $\mathcal{E}(D_{syn})$ denote the diameter distributions of the real and synthetic database respectively. The diameter error is calculated as: JSD($\mathcal{E}(D_{real}), \mathcal{E}(D_{syn})$).

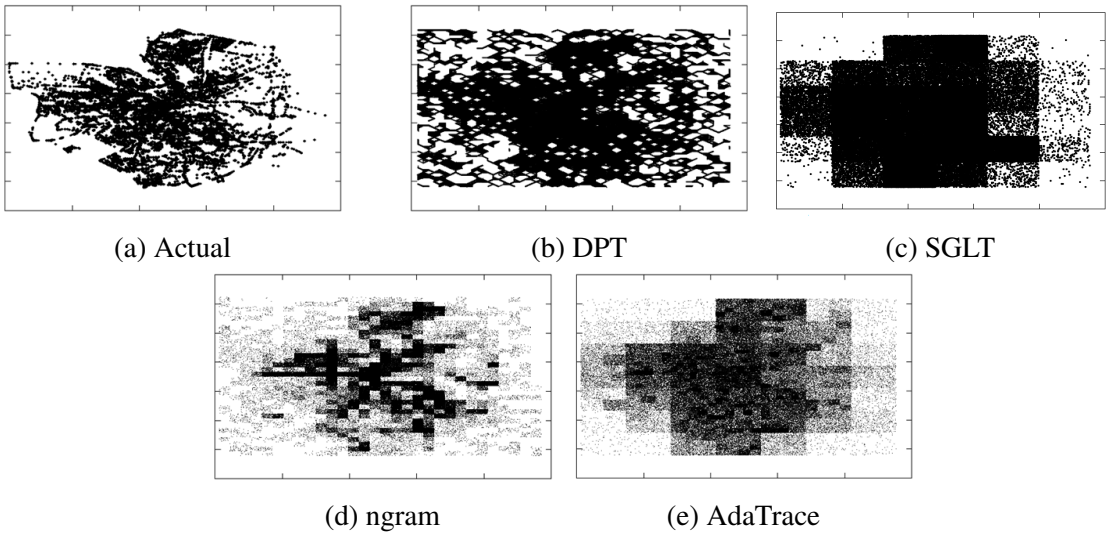
2.6.3 Comparison with Existing Generators

In this section we compare AdaTrace with ngram, DPT and SGLT. Since all generators except SGLT have ε (differential privacy budget) as a parameter, we perform our comparison by varying ε. Results are summarized in Table 2.2. We make two general observations. First, AdaTrace provides superior utility when subjected to the same level of privacy: Out of 63 total comparison settings, AdaTrace outperforms its competitors in 53 cases. In certain settings, AdaTrace's utility improvement is significant (2-3 fold or more). In addition to a utility advantage, AdaTrace also has a privacy advantage over its competitors due to its attack resilience property. Second, we observe that errors decrease and utility increases when higher ε is used. As such, we can conclude that data quality is responsive to changes in the privacy parameters, making AdaTrace flexible.

Next, we perform an in-depth analysis by studying the results under each utility category one by one, starting with *spatial density and locality*. We expect that Query AvRE is heavily related to the spatial distribution of trajectories. Hence, upon observing the numerical results in Table 2.2 we performed the visual comparison illustrated in Figure 2.6, where

Table 2.2: Comparing AdaTrace with its competitors. Best result in each category is shown in bold. For FP F1 Similarity and location Kendall-tau, higher values are better. For remaining metrics, lower values are better.

		Geolife				**Taxi**				**Brinkhoff**			
		ngram	DPT	SGLT	AdaTrace	ngram	DPT	SGLT	AdaTrace	ngram	DPT	SGLT	AdaTrace
Query AvRE	ε=0.5	0.391	0.158	**0.123**	0.168	0.232	0.381	0.158	**0.091**	0.201	0.386	0.216	**0.151**
	ε=1.0	0.383	0.161	**0.123**	0.162	0.233	0.380	0.158	**0.089**	0.158	0.342	0.216	**0.142**
	ε=2.0	0.355	0.154	**0.123**	0.155	0.222	0.369	0.158	**0.088**	0.142	0.291	0.216	**0.138**
FP AvRE	ε=0.5	0.60	0.58	0.75	**0.47**	0.43	0.65	0.53	**0.41**	0.83	0.53	0.52	**0.39**
	ε=1.0	0.61	0.57	0.75	**0.41**	0.42	0.69	0.53	**0.37**	0.64	0.48	0.52	**0.38**
	ε=2.0	0.59	0.56	0.75	**0.41**	0.43	0.68	0.53	**0.36**	0.54	**0.36**	0.52	0.38
Trip Error	ε=0.5	0.418	0.339	0.143	**0.048**	0.225	0.492	0.279	**0.042**	0.156	0.170	0.298	**0.052**
	ε=1.0	0.412	0.341	0.143	**0.025**	0.220	0.561	0.279	**0.033**	0.150	0.120	0.298	**0.045**
	ε=2.0	0.317	0.296	0.143	**0.013**	0.236	0.514	0.279	**0.027**	0.139	0.106	0.298	**0.043**
Length Error	ε=0.5	0.020	0.131	0.173	**0.011**	0.085	0.020	0.106	**0.017**	0.094	0.057	0.126	**0.042**
	ε=1.0	0.021	0.124	0.173	**0.010**	0.060	0.017	0.106	**0.016**	0.057	0.051	0.126	**0.041**
	ε=2.0	0.019	0.125	0.173	**0.008**	0.025	0.019	0.106	**0.014**	**0.032**	0.038	0.126	0.039
Diameter Error	ε=0.5	0.125	0.255	0.170	**0.085**	0.239	0.372	0.129	**0.026**	0.151	0.143	0.172	**0.022**
	ε=1.0	0.124	0.251	0.170	**0.067**	0.229	0.374	0.129	**0.025**	0.103	0.084	0.172	**0.023**
	ε=2.0	0.125	0.249	0.170	**0.065**	0.234	0.373	0.129	**0.024**	0.092	0.061	0.172	**0.022**
FP F1 Similarity	ε=0.5	0.38	0.46	0.38	**0.57**	0.42	0.27	0.49	**0.66**	0.41	0.56	0.47	**0.61**
	ε=1.0	0.42	0.55	0.38	**0.61**	0.54	0.31	0.49	**0.68**	0.39	**0.64**	0.47	0.62
	ε=2.0	0.41	**0.60**	0.38	**0.60**	0.61	0.32	0.49	**0.69**	0.42	**0.71**	0.47	0.62
Loc. Kendall-tau	ε=0.5	0.53	0.39	0.62	**0.71**	0.68	0.45	0.69	**0.82**	0.71	0.68	0.61	**0.73**
	ε=1.0	0.55	0.44	0.62	**0.68**	0.72	0.46	0.69	**0.83**	**0.76**	0.65	0.61	0.74
	ε=2.0	0.52	0.54	0.62	**0.69**	0.76	0.51	0.69	**0.83**	0.75	**0.86**	0.61	0.74



(a) Actual (b) DPT (c) SGLT

(d) ngram (e) AdaTrace

Figure 2.6: Visual density distributions of synthetic databases generated by various generators, using Brinkhoff as their D_{real}.

we compare the density distributions of synthetic databases produced by various trajectory generators using Brinkhoff as input. Figure 2.6 validates the finding in Table 2.2 that AdaTrace does better in terms of spatial density. DPT and SGLT's errors stem from low-density or medium-density regions becoming exceedingly dense, and ngram's errors stem from medium-density regions becoming exceedingly sparse. AdaTrace's density-adaptive grid plays an important role in density preservation, which we show in more detail in the next section. Another interesting note is that although Query AvRE and location Kendall-tau metrics have a positive correlation, they do not necessarily agree in all settings. For example, observe that SGLT's AvRE is lowest on Geolife, however, AdaTrace performs better than SGLT in terms of Kendall-tau. This demonstrates that there can be benefit in using multiple evaluation metrics for measuring one type of trajectory utility.

In terms of *frequent travel patterns*, there is correlation between FP AvRE and F1 similarity metrics, but also some discrepancies. AvRE indicates that there is 30-40% error in FP support when using AdaTrace which might seem high, but we observed that this is caused mostly by FPs with highest support not having as high support in synthetic data. That is, even though an actually frequent FP is correctly identified as frequent in synthetic data,

Table 2.3: Execution time measurements

	ngram	**DPT**	**SGLT**	**AdaTrace**
Geolife	7 mins	1.9 mins	11 hrs	0.7 mins
Taxi	12 mins	2.3 mins	2.4 days	1.7 mins
Brinkhoff	18 mins	3.5 mins	6.9 days	2.2 mins

since its frequency is not as high, considerable support error is incurred. This explains the competitive F1 scores despite high error in AvRE.

In terms of *spatio-temporal travel metrics*, we first note that AdaTrace is significantly superior in terms of trip error. Second, we notice that although AdaTrace's length errors are similar to its competitors, its diameter errors are much smaller. This is surprising, given that the two metrics are clearly related. To better understand this behavior, we individually studied some trajectories from each generator. We observed that DPT and ngram are biased towards generating trajectories with low displacement. For example, their trajectories contain loops and U-turns, which means that although they travel long distances, their total displacement is not large. In contrast, AdaTrace's synthesis algorithm prevents unrealistic loops and U-turns by guiding a trajectory's random walk with its final destination. That is, at each step of the random walk AdaTrace considers the final destination that the trajectory will eventually reach and how many steps are left to reach there. Hence, the resulting trajectory becomes goal-driven, which realistically captures the mobility of human beings since humans often travel with a destination in mind.

Finally, we comment on the efficiency of AdaTrace versus ngram, DPT, and SGLT. We performed our experiments on a high-end laptop with Intel i7 CPU, but since all generators are single-threaded, they utilized a single 2.8 GHz CPU. Also, 16 GB main memory was sufficient for the computation performed by all generators. In terms of execution time, we obtained the results given in Table 2.3. Overall, AdaTrace is very efficient – it is able to process and generate 50,000 trajectories in slightly longer than 2 minutes. ngram and DPT are also reasonably efficient since they finish in several minutes, but not as efficient as AdaTrace. On the other hand, SGLT is significantly slower, as it takes several days to

produce the same number of trajectories AdaTrace can produce in 2 minutes. This greatly hinders its use on commodity hardware or mobile devices. Note that this observation is consistent with the authors' claims in their paper [29], as they also report up to 2 minutes per generated trajectory. This has not caused problems in their experiments since their datasets consisted of < 100 users, but our datasets were several orders of magnitude larger (15-50k trajectories), surfacing an inefficiency problem.

2.6.4 Usefulness of Density-Aware Grid

To illustrate the usefulness of our density-aware adaptive grid, we compare it with a uniform grid with and without privacy constraints. We run an experiment similar to the spatial query experiment using $D_{real} =$ Brinkhoff. We first find how many times each cell in Brinkhoff's grid is visited. Then, we issue random counting queries of the form: *How many times was region R visited?*, where R typically spans portions of multiple cells. Queries are answered assuming locations are uniformly distributed within each cell, e.g., if query region R contains $1/2$ of C_1 and $1/3$ of C_2, and C_1 is visited 10 times and C_2 is visited 12 times, the query answer would be computed as $5 + 4 = 9$. We find the relative error in the query answer by comparing this computed answer with the actual answer (i.e., if the query was answered using D_{real}). While the above procedure is followed for non-private grids, for private grids, an additional step is taken before the query answers are computed: Appropriate Laplace noise is added to the visit count of each cell, and queries are answered using noisy counts instead of actual counts.

We give the results of the above experiment in Figure 2.7. We make three observations. First, by analyzing the private and non-private versions of the uniform grid, we observe the two sources of error in $\Omega(D)$ discretization. When grid granularity is too low, the dominating error is the coarse granularity. This is confirmed by the observation that increasing the grid granularity will almost always decrease error in the non-private case. However, in the private case, after the grid granularity exceeds 14x14, query error starts increasing.

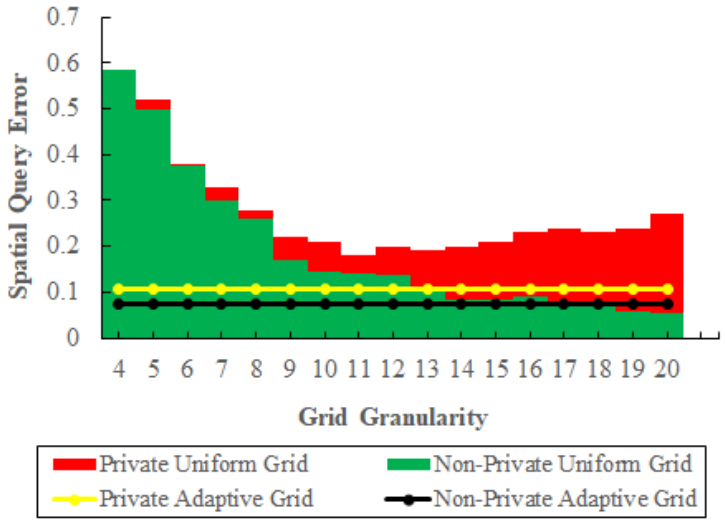
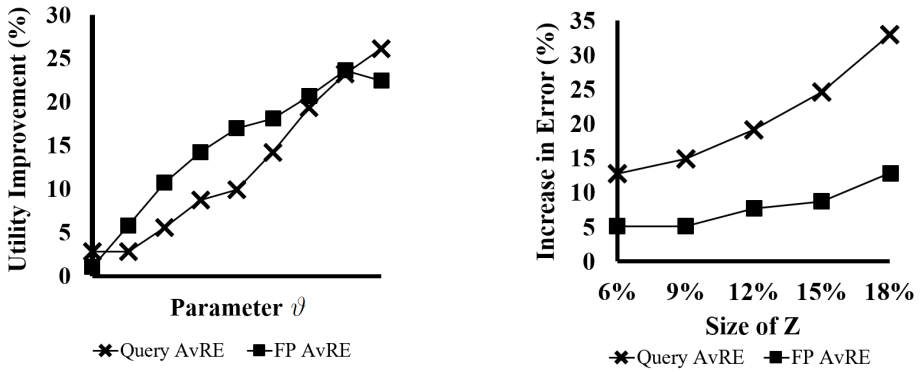


Figure 2.7: Comparing private and non-private versions of uniform grids versus our density-aware adaptive grid. Results are obtained using the Brinkhoff dataset and $\varepsilon = 1.0$. Grid granularity (x axis) controls the granularity of the uniform grid, e.g., a value of 10 means the uniform grid is 10x10.

This is caused by the Laplace noise, which explains the inverse bell-shaped readings for the private uniform grid. Second, we observe that the non-private adaptive grid is as good as a high-granularity uniform grid (e.g., 16x16). The uniform grid performs better only after its granularity exceeds 19x19. Thus, the adaptive grid is useful and efficient even when no noise is present. Third, upon observing that non-private and private versions of the adaptive grid perform similarly, we can conclude that our process of building the grid is not too negatively impacted by Laplace noise, as opposed to the uniform grid where Laplace noise destroys utility when grid granularity is high.

2.6.5 Impact of Attack Resilience on Utility

We now analyze the impacts of the attack resilience parameters used in Defenses 1-3 on various forms of utility.



(a) (b)

Figure 2.8: (a) Impact of ϑ on aggregate trajectory utility. (b) The size of sensitive zone Z is positively correlated with error amounts. Both experiments are performed on the Taxi dataset.

Defense 1: Bayesian Inference

Our defense asserts that the difference between priors and posteriors must be $\leq \vartheta$. When $\vartheta = 0$ the defense is strictest, and it is relaxed as ϑ increases. In Figure 2.8a, we initially set $\vartheta = 0$, increase it gradually, and report the percentage improvement (decrease) in Query AvRE and FP AvRE. We only report these metrics since the remaining metrics are non-linear, involving the likes of F1 or JSD calculation, hence percentages are not meaningful. The results show the expected trait that both spatial densities (implied by Query AvRE) and frequent patterns (implied by FP AvRE) are positively impacted when ϑ is relaxed. We observed that this is mostly caused by queries and FPs involving the sensitive zone Z, which is central to Defense 1. When the privacy setting is stricter, trajectories visiting Z are more random, hence FPs including Z will not be preserved.

An implicit parameter here is the choice (size and density) of Z. In the extreme case where $Z = \Omega(D_{real})$, Defense 1 dictates that the adversary should gain no knowledge that adds to his prior belief. In this case the trajectories in D_{syn} will be most random, and their features will converge to population averages rather than capturing the utility in D_{real}. To experimentally confirm this, in Figure 2.8b we perform an experiment in which we vary the size of Z. (In the previous experiment, size of Z was fied to be roughly 3% of D_{real}.)

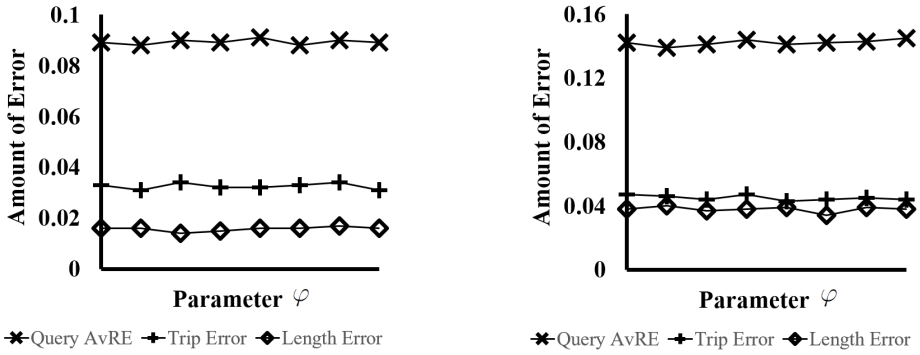
Results indicate that there is indeed a positive correlation between error amounts and the size of Z.

Defense 2: Partial Sniffing

The defense is based on two parameters: φ that bounds the intersection between a sniffed actual trajectory and its counterpart synthetic trajectory, and ϱ that limits the possibility that the counterpart visits a sensitive zone. For both parameters, lower values imply stricter privacy. We set a fixed sniffing region that is visited by 4% of the actual trajectories, and assume all trajectories in this region have been sniffed by the adversary. We set $\varrho = 0$ which is the strictest privacy setting, and vary φ to observe its utility impact. In Figure 2.9a and 2.9b we illustrate the utility impact with respect to 3 metrics. Although one could expect utility to decrease as privacy becomes stricter, this does not appear to be the case – we observe that aggregate utility stays constant despite changes in φ. Upon further analysis with more data points and remaining utility metrics, we confirm that the impact of φ on overall utility is negligible. This is because our utility metrics are aggregate metrics, taking whole D_{real} and D_{syn} into consideration rather than comparing individual trajectories one by one. This is also the case in many machine learning tasks and real-life uses of trajectory data. Thus, although an actual trajectory and its synthetic counterpart do not closely match, utility can be re-injected by implicitly adding the mismatched part into a different trajectory that is not sniffed.

Defense 3: Outlier Leakage

The two parameters in this defense are β and κ. κ dictates that a synthetic outlier must blend into a crowd of κ actual trajectories. β dictates the uniformity of the crowd: When $\beta = 0$ all trajectories in the crowd must appear exactly the same (in terms of the trajectory feature considered), whereas a large β allows the crowd to have higher non-uniformity. The privacy requirement becomes stricter when β decreases and κ increases.

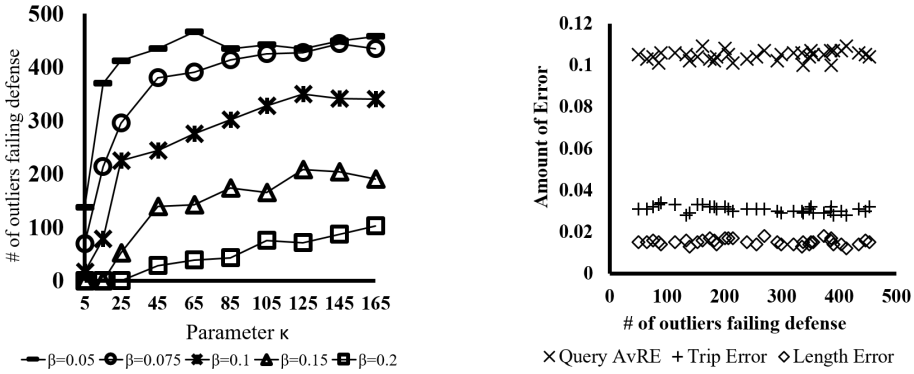


(a) Taxi dataset (b) Brinkhoff dataset

Figure 2.9: Impact of φ on aggregate trajectory utility.

We first analyze the behavior of Defense 3 with respect to parameters β and κ in Figure 2.10a. Here, we set β normalized after observing the maximum possible distance in the database, e.g., if the max distance is 100, $\beta = 0.1$ implies that the allowed distance in forming the crowd is 10. When the privacy requirement is stricter, we indeed observe that there are more outlier trajectories failing the defense, with a notable increase when κ becomes ≥ 25. We attribute this to the curse of dimensionality: Crowd-blending notions of privacy, such as k-anonymity and our defense, rely on the ability to find a crowd comprised of similar records. While this is easier for low-dimensional data, trajectory data is inherently high-dimensional, and typically no two trajectories are alike. Hence, forming a similar crowd is difficult, and while higher κ values can be suitable for low-dimensional databases, the same κ may yield an excess number of outliers on a high-dimensional or trajectory database.

Nevertheless, we are interested in measuring the utility impact of arbitrary (β,κ) pairs, and wish that our system preserves utility even in the strictest cases. We therefore design the experiment in Figure 2.10b. Instead of trying to isolate β and κ, we take a holistic approach and base our utility analysis on the number of outliers failing the defense. Note that outliers failing the defense cannot be released, and must be perturbed or regenerated. We observe from the experiment results that this perturbation or regeneration process has little to no impact on aggregate trajectory utility. Despite different privacy settings, Query



(a) (b)

Figure 2.10: (a) Analyzing the outlier behavior with respect to parameters β, κ. (b) Utility impact of executing Defense 3. Both experiments are performed on the Taxi dataset.

AvRE, Trip Error and Length Error do not seem to be impacted. Hence, we can conclude that we are able to protect against outlier leakage with no additional utility cost. In addition to the visual results in Figure 2.10b, we computed the correlation between the privacy settings and the error amounts. Results indicate weak correlations at most (+0.2), but are often closer to 0 and sometimes even negative.

Summary of Findings

Under reasonable privacy parameters, Defenses 2 and 3 can be implemented with little or no observable impact on aggregate trajectory utility, because they are concerned with individual trajectories that often constitute a small portion of the whole database. The impact of small perturbations on individual trajectories are either negligible on aggregate utility, or they can be counterbalanced by injecting the lost utility in the remaining trajectories. On the other hand, Defense 1 is directly connected to the aggregate information that is obtained from D_{syn}, hence a stricter privacy parameter ϑ or an increased geographic size of sensitive zone Z negatively impacts aggregate trajectory utility.

2.7 Conclusion

In this chapter we presented AdaTrace, a utility-aware location trace synthesizer with differential privacy guarantee and attack resilience. AdaTrace performs feature extraction, learning, and noise injection using a database of real location traces. It then generates synthetic traces while preserving differential privacy, enforcing resilience to inference attacks, and upholding statistical and spatial utility. Our experiments on real and simulated datasets show that AdaTrace is highly effective. Compared to ngram, DPT, and SGLT [29, 31, 32], AdaTrace provides up to 3-fold improvement in trace utility. At the same time, AdaTrace is computationally more efficient, and on average 1.5x faster than DPT, 8x faster than ngram, and 1000x faster than SGLT. An open research direction is to investigate the feasibility of extending AdaTrace to handle incremental updates, in order to make AdaTrace compatible with growing and streaming location trace datasets.

CHAPTER 3

SECURE AND UTILITY-AWARE DATA COLLECTION WITH CONDENSED LOCAL DIFFERENTIAL PRIVACY

Local Differential Privacy (LDP) is popularly used in practice for privacy-preserving data collection. Although existing LDP protocols offer high utility for large user populations (100,000 or more users), they perform poorly in scenarios with small user populations (such as those in the cybersecurity domain) and lack perturbation mechanisms that are effective for both ordinal and non-ordinal item sequences while protecting sequence length and content simultaneously. In this chapter, we address the small user population problem by introducing the concept of Condensed Local Differential Privacy (CLDP) as a specialization of LDP, and develop a suite of CLDP protocols that offer desirable statistical utility while preserving privacy. Our protocols support different types of client data, ranging from ordinal data types in finite metric spaces (numeric malware infection statistics), to non-ordinal items (OS versions, transaction categories), and to sequences of ordinal and non-ordinal items. Extensive experiments are conducted on multiple datasets, including datasets that are an order of magnitude smaller than those used in existing approaches, which show that proposed CLDP protocols yield high utility. Furthermore, case studies with Symantec datasets demonstrate that our protocols accurately support key cybersecurity-focused tasks of detecting ransomware outbreaks, identifying targeted and vulnerable OSs, and inspecting suspicious activities on infected machines.

3.1 Introduction

Organizations and companies are becoming increasingly interested in collecting user data and telemetry to make data-driven decisions. While collecting and analyzing user data is beneficial to improve services and products, users' privacy poses a major concern. Re-

cently, the concept of *Local Differential Privacy (LDP)* has emerged as the accepted standard for privacy-preserving data collection [83, 14, 84]. In LDP, each user locally perturbs their sensitive data on their device before sharing the perturbed version with the data collector. The perturbation is performed systematically such that the data collector cannot infer with strong confidence the true value of any user given their perturbed value, yet it can still make accurate inferences pertaining to the general population. Due to its desirable properties, LDP has been adopted by major companies to perform certain tasks, including Google to analyze browser homepages and default search engines in Chrome [14, 85], Apple for determining emoji frequencies and spelling prediction in iOS [86, 16], and Microsoft to collect application telemetry in Windows 10 [15].

While LDP is popularly used for the aforementioned purposes, one domain that is yet to embrace it is cybersecurity. It can be argued that this nuanced domain has the potential to greatly benefit from an LDP-like protection mechanism. This is because many security products rely on information collected from their clients, with the required telemetry ranging from file occurrence information in file reputation systems [87, 88, 89] to heterogeneous security event information such as system calls and memory dumps in the context of Endpoint Detection and Response systems (see [90] for a survey on core behavioral detection techniques used by such systems). Nevertheless, clients are often reluctant to share such data fearing that it may reveal the applications they are running, the files they store, or the overall cyber hygiene of their devices. Offering formal privacy guarantees would help convince clients that sharing their data will not cause privacy leakages.

On the other hand, existing LDP protocols suffer from problems that hinder their deployment in the cybersecurity domain. First, although they are accurate for large populations (e.g., hundreds of thousands of clients), their accuracy suffers when client populations are smaller. Population size is not necessarily a problem for the likes of Google Chrome and Apple iOS with millions of active users, but it does cause problems in a domain such as cybersecurity where small population sizes are common. For example, if a security analyst

is analyzing the behavior of a particular malware that targets a certain system or vulnerability, only those clients who are infected by the malware will have meaningful observations to report, but the number of infections could be limited to less than a couple of thousand users globally due to the targeted nature of the malware. In such cases, we need a new scheme that allows the security analyst to make accurate inferences while simultaneously giving adequate privacy to end users. Second, existing protocols consider a limited set of primitive data types. To the best of our knowledge, currently no protocol supports perturbation of item sequences (with either ordinal or non-ordinal item domains) to offer privacy with respect to sequence length and content simultaneously. Sequences are more difficult to handle compared to the data types of singleton items or itemsets since they are not only high-dimensional, but also contain an ordering that must be preserved for tasks such as pattern mining. Yet, sequential data is ubiquitous in cybersecurity, e.g., security logs, network traffic data, file downloads, and so forth are all examples of sequential data.

In this chapter, we propose the notion of *Condensed LDP (CLDP)* and a suite of protocols satisfying CLDP to tackle these issues. In practice, CLDP is similar to LDP with the addition of a *condensation* aspect, i.e., during the process of perturbation, similar outputs are systematically favored compared to distant outputs using condensed probability. We design protocols satisfying CLDP for various types of data, including singletons and sequences of ordinal and non-ordinal items. We show that CLDP can be satisfied by a variant of the Exponential Mechanism [63], and employ this mechanism as a building block in our *Ordinal-CLDP*, *Item-CLDP*, and *Sequence-CLDP* protocols. Our methods are generic and can be easily applied for privacy-preserving data collection in multiple domains, including but not limited to cybersecurity.

We consider a Bayesian adversary model for privacy-preserving data collection, which enables us to establish a formal connection between LDP and CLDP and ensures that they provide the same protection against this common adversary. The Bayesian model measures the adversary's maximum posterior confidence (MPC) in predicting the user's true value

over all possible inputs and outputs of a perturbation mechanism, where higher adversarial confidence implies lower privacy protection for users. We derive how the parameters of CLDP protocols can be selected to give as strong protection as LDP protocols *under this Bayesian model*. Experiments conducted using this parameter selection show that our CLDP protocols provide high utility by yielding accurate insights for population sizes that are an order of magnitude smaller than those currently assumed by state-of-the-art LDP approaches.

We also perform extensive experiments to evaluate the effectiveness of our protocols on real-world case studies and public datasets. Experiments show that proposed CLDP protocols enable accurate frequency estimation, heavy hitter identification, and pattern mining while achieving strong protection against the adversary model considered. Using data from Symantec, a major cybersecurity vendor, we show that CLDP can be used in practice for use cases involving ransomware outbreak detection, OS vulnerability analysis, and inspecting suspicious activities on infected machines. In contrast, existing LDP protocols can either not be applied to these problems or their application yields unacceptable accuracy loss. To the best of our knowledge, our work is the first to apply the concept of local differential privacy to the nuanced domain of cybersecurity.

3.2 Background and Problem Setting

We consider the data collection setting, where there are many clients (*users*) and an untrusted data collector (*server*). Each client possesses a secret value. The client's secret value can be an ordinal item (e.g., numeric value or integer), a categorical item, a non-ordinal item, or a sequence of items. The server wants to collect data from clients to derive useful insights; however, since the clients do not trust the server, they perturb their secrets locally on their device before sharing the perturbed version with the server. Randomized perturbation ensures that the server, having observed the perturbed data, cannot infer the true value of any one client with strong probability. At the same time, the scheme allows the

server to derive useful insights from *aggregates* of perturbed data by recovering statistics pertaining to the general population.

In Section 3.2.1, we start by introducing the threat and Bayesian adversary models we consider for measuring privacy protection in the above data collection setting. In Section 3.2.2, we define LDP as the current popular solution to privacy-preserving data collection. In Section 3.2.3, we describe the utility model and provide preliminary analysis that shows LDP's utility loss under small user populations. Section 3.2.4 contains our problem statement.

3.2.1 Adversary Model

We use a Bayesian approach to measure privacy in the local data collection setting. Our Bayesian adversary model is similar to the Bayesian adversary formulations for membership privacy and location privacy in [26, 34, 91, 92]; but differs in the sense that the main goal of local privacy schemes is to offer confidentiality and plausible deniability for each individual user's secret. Higher plausible deniability implies lower adversarial prediction confidence and consequently higher privacy protection.

The threat stems from an untrusted third party (e.g., the data collector) inferring the true value of the user with high confidence from the perturbed value (s)he observes. Then, the goal of randomized perturbation is to stop an adversary $\mathcal{A}$, even if they can fully observe perturbed output y, from inferring the user's true secret v. Given y, the optimal attack strategy for $\mathcal{A}$ is:

$$\mathcal{A}(y) = \underset{\hat{v} \in \mathcal{U}}{\operatorname{argmax}} \Pr[\hat{v}|y] \tag{3.1}$$

$$= \underset{\hat{v} \in \mathcal{U}}{\operatorname{argmax}} \frac{\pi(\hat{v}) \cdot \Pr[f(\hat{v}) = y]}{\sum_{z \in \mathcal{U}} \pi(z) \cdot \Pr[f(z) = y]} \tag{3.2}$$

where $\pi(\hat{v})$ denotes the prior probability of $\hat{v}$, f denotes a perturbation function, and $\mathcal{U}$ denotes the universe of possible items. Then, worst-case privacy disclosure can be measured

using the maximum posterior confidence (MPC) the adversary can achieve over all possible inputs and outputs:

$$\text{MPC} = \max_{v \in \mathcal{U}, y} \Pr[v|y] = \max_{v \in \mathcal{U}, y} \frac{\pi(v) \cdot \Pr[f(v) = y]}{\sum_{z \in \mathcal{U}} \pi(z) \cdot \Pr[f(z) = y]} \tag{3.3}$$

This establishes a mathematical framework under which we can quantify worst-case adversarial disclosure of perturbation protocols, for both informed adversaries (with prior knowledge π) and uninformed adversaries (e.g., by canceling out the π term, or equivalently, setting $\pi(x) = 1/|\mathcal{U}|$ for all $x \in \mathcal{U}$). When MPC is high, the adversary can more easily predict the true input of the user, hence f yields lower privacy. Thus, lower MPC is desired.

3.2.2 Local Differential Privacy (LDP)

The state-of-the-art scheme currently used and deployed by major companies such as Google, Apple, and Microsoft in the data collection setting is LDP [14, 85, 16, 86, 15]. In LDP, each user perturbs their true value v using an algorithm Ψ and sends $\Psi(v)$ to the server. LDP can be formalized as follows.

Definition 1 (ε-LDP)**.** *A randomized algorithm Ψ satisfies ε-local differential privacy (ε-LDP), where $\varepsilon > 0$, if and only if for any inputs v_1, v_2 in universe $\mathcal{U}$, we have:*

$$\forall y \in Range(\Psi): \quad \frac{Pr[\Psi(v_1) = y]}{Pr[\Psi(v_2) = y]} \leq e^{\varepsilon}$$

where $Range(\Psi)$ denotes the set of all possible outputs of algorithm Ψ.

Here, ε is the privacy parameter controlling the level of indistinguishability. Lower ε yields higher privacy. Several works were devoted to building accurate protocols that satisfy ε-LDP. These works were analyzed and compared in [84], and it was found that the two currently optimal protocols are based on: (i) OLH, the hashing extension of the GRR

primitive; and (ii) RAPPOR, the Bloom filter-based bitvector encoding and bit flipping strategy. Next, we briefly present these protocols.

Generalized Randomized Response (GRR): This protocol is a generalization of the *Randomized Response* survey technique introduced in [93]. Given the user's true value v, the perturbation function Ψ_{GRR} outputs y with probability:

$$\Pr[\Psi_{GRR}(v) = y] = \begin{cases} p = \frac{e^{\varepsilon}}{e^{\varepsilon}+|\mathcal{U}|-1} & \text{if } y = v \\ q = \frac{1}{e^{\varepsilon}+|\mathcal{U}|-1} & \text{if } y \neq v \end{cases}$$

where $|\mathcal{U}|$ denotes the size of the universe. This satisfies ε-LDP since $\frac{p}{q} = e^{\varepsilon}$. In words, Ψ_{GRR} takes as input v and assigns a higher probability p to returning the same output $y = v$. With remaining $1 - p$ probability, Ψ_{GRR} samples a fake item from the universe $\mathcal{U} \setminus \{v\}$ uniformly at random, and outputs this fake item. Note that an equal probability q is assigned to all fake items, i.e., there is no preference towards sampling a fake item that is similar to or distant from v.

Optimized Local Hashing (OLH): When the universe size $|\mathcal{U}|$ is large, it dominates the denominator of p and q, thus the accuracy of GRR deteriorates quickly. The OLH protocol proposed in [84] handles the large universe problem by first using a hash function to map v into a smaller domain of hash values and then applying GRR on the hashed value. Formally, the client reports:

$$\Psi_{OLH}(v) = \langle H, \Psi_{GRR}(H(v)) \rangle$$

where H is a randomly chosen hash function from a family of hash functions. Each hash function in the family maps $v \in \mathcal{U}$ to a domain $\{1...g\}$, where g denotes the size of the hashed domain, typically $g \ll |\mathcal{U}|$. We use $g = \lceil e^{\varepsilon} + 1 \rceil$ as the optimal value of g found in [84].

Randomized Aggregatable Privacy-Preserving Ordinal Response (RAPPOR): RAP-

POR was developed by Google and is used in Chrome [14]. In RAPPOR, the user's true value v is encoded in a bitvector B. The straightforward method is to use one-hot encoding such that B is a length-$|\mathcal{U}|$ binary vector where the v'th position is 1 and the remaining positions are 0. When $|\mathcal{U}|$ is large, both communication cost and inaccuracy cause problems, hence RAPPOR uses Bloom filter encoding. Specifically, B is treated as a Bloom filter and a set of hash functions $\mathcal{H}$ is used to map v into a set of integer positions that must be set to 1. That is, $\forall H \in \mathcal{H}$, $B[H(v)] = 1$, and the remaining positions are 0.

After the encoding, RAPPOR uses perturbation function Ψ_{RAPPOR} on B to obtain perturbed bitvector B' as follows:

$$\Pr\Big[B'[i] = \Psi_{RAPPOR}(B[i]) = 1\Big] = \begin{cases} \frac{e^{\varepsilon/2\Delta}}{e^{\varepsilon/2\Delta}+1} & \text{if } B[i] = 1 \\ \frac{1}{e^{\varepsilon/2\Delta}+1} & \text{if } B[i] = 0 \end{cases}$$

where 2Δ is analogous to the notion of *sensitivity* in differential privacy [94], i.e., how many positions can change in neighboring bitvectors at most? In one-hot encoding, $\Delta = 1$; in Bloom filter encoding, $\Delta = |\mathcal{H}|$. Then, the perturbation process considers each position in B independently, and the existing bit $B[i]$ is either kept or flipped when creating $B'[i]$.

3.2.3 Utility Model and Analysis

The most common use of LDP is to enable the data collector learn *aggregate* population statistics from large collections of perturbed data. Much research has been invested in tasks such as frequency estimation (identify proportion of users who have a certain item) and heavy hitter discovery (identify popular items that are held by largest number of users) [84, 95, 96, 97]. Utility is measured by how closely the privately collected statistics resemble the actual statistics that would have been obtained if privacy was not applied.

Frequency estimation and heavy hitter discovery are important tasks in the cybersecurity domain as well. For example, by monitoring the observed frequencies of different malware using aggregates of privatized malware reports, a cybersecurity vendor such as

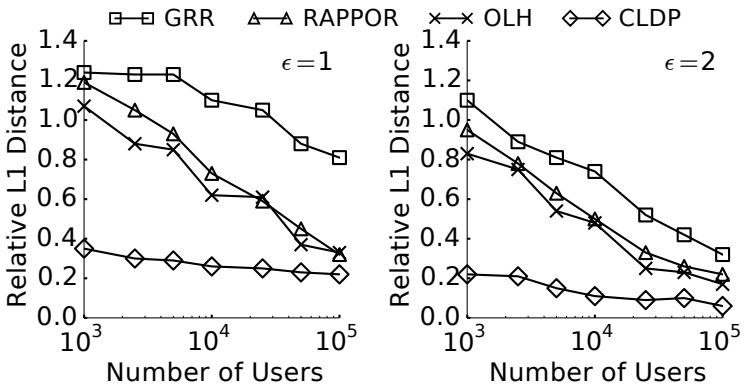


Figure 3.1: Measuring relative error in item frequency estimation by calculating L1 distance between actual and estimated frequencies. L1 distance of d means $100 \cdot d$ % estimation error.

Symantec can identify large-scale malware outbreaks and create response teams to address them. Furthermore, analyzing the heavy hitter operating systems that are most commonly infected by the malware will enable Symantec understand OS vulnerabilities as well as the fraction of clients in its user base that are impacted. Relevant findings may also be used by Symantec when developing its next-generation anti-malware defenses.

A novel challenge posed by the cybersecurity domain, however, is accurately supporting small user populations. Typically, existing LDP literature assumes the availability of "large enough" user populations in the order of hundreds of thousands or millions of users [14, 84, 86, 97]. Yet, population sizes in the cybersecurity domain are typically much smaller. For example, consider a security analyst analyzing the behavior of a specific malware by studying infected user machines. It is often the case that malware targets a specific computing platform or software product, limiting the total number of infections to less than a couple of thousand users globally. We designed the frequency estimation experiment in Figure 3.1 to illustrate how the utility of existing LDP protocols suffer under such small populations. We sampled each user's secret value from a Gaussian distribution with mean $\mu = 50$ and standard deviation $\sigma = 12$, and rounded it to the nearest integer. The goal of the data collector is to estimate the true frequency of each integer. We run this experiment for varying number of users between 1,000 and 100,000 and graph the error in the frequency

estimations made by the data collector.

We observe from Figure 3.1 that although more recent and optimized protocols improve previous ones by decreasing estimation error (e.g., OLH outperforms RAPPOR, which outperforms GRR); in cases with small user populations (e.g., 1000, 2500, or 5000 users) the improvements offered by more recent LDP protocols over previous ones are only 10-20%, whereas our proposed CLDP approach provides a remarkable 60-70% improvement. Furthermore, with 2500 users, estimation error is larger than 80% even for the state-of-the-art OLH algorithm, which is optimized for frequency estimation [84]. In contrast, our proposed CLDP solution is able to handle small user populations gracefully, with estimation errors lower than half of OLH's errors.

3.2.4 Problem Statement

As our analysis shows, low utility levels of existing LDP protocols under small user population sizes constitute an important obstacle towards their deployment in the cybersecurity domain. This motivates us to seek alternative approaches and protocols for privacy-preserving data collection in this challenging scenario. The problem we study in this chapter is to design a data collection scheme such that: (i) it gives at least as strong privacy protection as existing LDP protocols under the MPC adversary model, (ii) while doing so, it provides higher accuracy and data utility compared to existing protocols especially for small user populations, and (iii) it offers extensibility and generalizability to support complex data types such as different types of singleton items, itemsets, and sequences that are present in the cybersecurity domain.

3.3 Proposed Solution

In this section, we introduce our proposed solution aproach to the problem stated above. We start with the notion of *Condensed* Local Differential Privacy (CLDP).

3.3.1 Condensed Local Differential Privacy

Let $\mathcal{U}$ denote the finite universe of possible values (items) and let $d : \mathcal{U} \times \mathcal{U} \rightarrow [0, \infty)$ be a distance function that takes as input two items $v_1, v_2 \in \mathcal{U}$ and measures their distance. We require d to satisfy the conditions for being a metric, i.e., non-negativity, symmetry, triangle inequality, and identity of discernibles. Then, CLDP can be formalized as follows.

Definition 2 (α-CLDP)**.** *A randomized algorithm Φ satisfies α-condensed local differential privacy (α-CLDP), where $\alpha > 0$, if and only if for any inputs $v_1, v_2 \in \mathcal{U}$:*

$$\forall y \in Range(\Phi) : \quad \frac{\boldsymbol{Pr}[\Phi(v_1) = y]}{\boldsymbol{Pr}[\Phi(v_2) = y]} \leq e^{\alpha \cdot d(v_1, v_2)}$$

where $Range(\Phi)$ denotes the set of all possible outputs of algorithm Φ.

LDP and CLDP follow a similar formalism, but differ in how their privacy parameters and indistinguishability properties work. Similar to ε-DP, α-CLDP satisfies the property that an adversary observing y will not be able to distinguish whether the original value was v_1 or v_2. However, in α-CLDP, indistinguishability is controlled also by items' distance $d(\cdot, \cdot)$ in addition to α. Consequently, as d increases, α must decrease to compensate, i.e., we have $\alpha \ll \varepsilon$. By definition, CLDP constitutes a metric-based extension of LDP. Metric-based extensions of differential privacy have been studied in the past under certain settings such as aggregate query answering in centralized statistical databases [98], geo-indistinguishability in location-based systems [21, 24], and protecting sensitive relationships between entities in graphs through k-edge differential privacy [99]. In contrast, we propose the metric-based CLDP extension in the *data collection* setting. Our data collection setting poses novel challenges due to: (i) the local privacy scenario, unlike centralized DP assumed in aggregate query answering and graph mining in which user data is collected in the clear first and privacy is applied after the data has been stored in a centralized database, (ii) data types that are different than tabular datasets, locations, and graphs, and

(iii) establishing relationships and comparison between CLDP and LDP under the assumed adversary and utility models.

Since existing LDP protocols do not satisfy CLDP, we need new mechanisms and protocols supporting CLDP. We show below that a variant of the Exponential Mechanism (EM) [63] satisfies α-CLDP. EM is used in the remainder of the chapter as a building block for more advanced CLDP protocols.

Exponential Mechanism (EM). Let $v \in \mathcal{U}$ be the user's true value, and let the Exponential Mechanism, denoted by Φ_{EM}, take as input v and output a perturbed value in $\mathcal{U}$, i.e., $\Phi_{EM} : \mathcal{U} \to \mathcal{U}$. Then, Φ_{EM} that produces output y with the following probability satisfies α-CLDP:

$$\forall y \in \mathcal{U}: \ \Pr[\Phi_{EM}(v) = y] = \frac{e^{\frac{-\alpha \cdot d(v,y)}{2}}}{\sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v,z)}{2}}}$$

Theorem 2. *Exponential Mechanism satisfies α-CLDP.*

Proof. We start by applying the definition of EM and breaking the odds ratio into two terms:

$$\frac{\Pr[\Phi_{EM}(v_1) = y]}{\Pr[\Phi_{EM}(v_2) = y]} = \frac{\frac{e^{\frac{-\alpha \cdot d(v_1,y)}{2}}}{\sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v_1,z)}{2}}}}{\frac{e^{\frac{-\alpha \cdot d(v_2,y)}{2}}}{\sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v_2,z)}{2}}}} \tag{3.4}$$

$$= \underbrace{\frac{e^{\frac{-\alpha \cdot d(v_1,y)}{2}}}{e^{\frac{-\alpha \cdot d(v_2,y)}{2}}}}_{*} \cdot \underbrace{\frac{\sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v_2,z)}{2}}}{\sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v_1,z)}{2}}}}_{**} \tag{3.5}$$

For *, we observe that:

$$\frac{e^{\frac{-\alpha \cdot d(v_1,y)}{2}}}{e^{\frac{-\alpha \cdot d(v_2,y)}{2}}} = e^{\frac{\alpha \cdot \left(d(v_2,y) - d(v_1,y)\right)}{2}} \tag{3.6}$$

Since d is a metric, it satisfies the triangle inequality. Therefore, it holds that: $d(v_2, y) -$

$d(v_1, y) \leq d(v_1, v_2)$. Combining this with the above, we conclude for *:

$$\frac{e^{\frac{-\alpha \cdot d(v_1,y)}{2}}}{e^{\frac{-\alpha \cdot d(v_2,y)}{2}}} \leq e^{\frac{\alpha \cdot d(v_1,v_2)}{2}} \tag{3.7}$$

Next, we study the second term **:

$$\frac{\sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v_2,z)}{2}}}{\sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v_1,z)}{2}}} = \frac{\sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v_2,z)+\alpha \cdot d(v_1,z)-\alpha \cdot d(v_1,z)}{2}}}{\sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v_1,z)}{2}}} \tag{3.8}$$

Again by triangle inequality: $d(v_1, z) - d(v_2, z) \leq d(v_1, v_2)$. Applying this to the numerator we get:

$$\frac{\sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v_2,z)}{2}}}{\sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v_1,z)}{2}}} \leq \frac{\sum_{z \in \mathcal{U}} e^{\frac{\alpha \cdot d(v_1,v_2)-\alpha \cdot d(v_1,z)}{2}}}{\sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v_1,z)}{2}}} \tag{3.9}$$

$$\leq \frac{e^{\frac{\alpha \cdot d(v_1,v_2)}{2}} \cdot \sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v_1,z)}{2}}}{\sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v_1,z)}{2}}} \tag{3.10}$$

$$\leq e^{\frac{\alpha \cdot d(v_1,v_2)}{2}} \tag{3.11}$$

We established that $* \leq e^{\frac{\alpha \cdot d(v_1,v_2)}{2}}$ and $** \leq e^{\frac{\alpha \cdot d(v_1,v_2)}{2}}$. Plugging them into Equation 3.5 concludes our proof:

$$\frac{\Pr[\Phi_{EM}(v_1) = y]}{\Pr[\Phi_{EM}(v_2) = y]} \leq e^{\frac{\alpha \cdot d(v_1,v_2)}{2}} \cdot e^{\frac{\alpha \cdot d(v_1,v_2)}{2}} = e^{\alpha \cdot d(v_1,v_2)}$$

□

3.3.2 Privacy Protection of LDP and CLDP

We wish to find the appropriate α value to be used in CLDP so that the MPC under α-CLDP will be equal to or lower than the MPC under ε-LDP, which ensures that CLDP gives equal or better protection than LDP against the adversary we consider.

Question: Let $\mathcal{U}$, d and π be given. If there is an LDP protocol currently in place with

privacy budget ε and we are interested in switching to α-CLDP, how should the value of α be selected to achieve equal or better protection than LDP according to the MPC adversary model? **Answer:** Based on the quantification of the adversary's maximum posterior confidence in Equation 3.3, the requirement to have the MPC of CLDP less than or equal to that of LDP can be written as:

$$\max_{v,y} \frac{\pi(v) \cdot \Pr[\Phi(v) = y]}{\sum_{z \in \mathcal{U}} \pi(z) \cdot \Pr[\Phi(z) = y]} \leq \max_{v,y} \frac{\pi(v) \cdot \Pr[\Psi(v) = y]}{\sum_{z \in \mathcal{U}} \pi(z) \cdot \Pr[\Psi(z) = y]} \tag{3.12}$$

where Ψ denotes LDP perturbation and Φ denotes CLDP perturbation. Using $\mathcal{U}$, d, π and ε, we can compute the right hand side, and then search for the largest α such that the left hand side remains smaller than the right hand side, iteratively by incrementing α in each iteration and re-computing the left hand side in each iteration. The complexity of computing the right hand side (left hand side is analogous) is $\mathcal{O}(|\mathcal{U}|^3) \cdot \mathcal{O}(\Psi)$, where $\mathcal{O}(\Psi)$ is the complexity of the perturbation mechanism, often linear or sublinear in $\mathcal{U}$ or constant time. This is assuming access and querying of $\pi(v)$ and $d(v_i, v_j)$ are constant time operations, as they can be pre-computed in $\mathcal{O}(|\mathcal{U}| + |\mathcal{U}|^2)$ time and stored in a form that supports efficient access, which does not affect overall complexity. In deployment, the computation to convert ε to α is a one-time cost performed at protocol setup time, therefore it does not have a negative impact on real-time user experience.

An alternative to solving Equation 3.12 could be to obtain a closed-form relationship between α and ε given $\mathcal{U}$, d, and π. We found that such a closed-form relationship can be established under particular cases such as when π is uniform or when d satisfies certain properties. However, the closed-form relationships require constructing and solving a high degree polynomial, which has higher execution time in practice than the iterative solution to Equation 3.12 which we presented above, due to the inherent hardness and time complexity of solving high degree polynomials. Hence, we give the current version of Equation 3.12 for wider practical applicability and more efficient implementation than the closed-form

relationships we could derive. One of our ongoing future work directions is considering the derivation of efficient closed-form relationships.

Roles of Relevant Factors: Among the relevant factors, $\mathcal{U}$ impacts not only the outcome of ε to α conversion, but also the computational complexity. This is because $\mathcal{U}$ is a relevant factor in the Pr[$\cdot$] calculations in both LDP and CLDP. For example, the size of the bitvector in RAPPOR is affected by $|\mathcal{U}|$. EM of CLDP is also affected by $|\mathcal{U}|$, as each element in $\mathcal{U}$ must be assigned a score and the scores are then normalized. d has no impact on the behavior of LDP, but it affects CLDP. Since our conversion aims to achieve equivalent protection in CLDP compared to LDP against the Bayesian adversary from Section 3.2.1, it is expected that larger the $\max_{v_i,v_j} d(v_i, v_j)$, smaller the α value under the same ε. This can be explained via the fact that under fixed ε, according to the exponent in Definition 2, when d is larger, then α must be lowered to counter-balance the impact of increased d to still achieve equivalent protection. π is also a relevant factor in Equation 3.12. As we will exemplify in the upcoming practical analysis, when π is skewed (rather than uniform) it becomes the dominating factor in both the right and left hand sides of Equation 3.12; therefore under the same ε, typically larger α is obtained when π is skewed compared to uniform. Finally, the number of users does not appear in Equation 3.12 or play a role in the conversion, since the same ε is used across all users in LDP and similarly the same α is used across all users in CLDP.

Practical Analysis: To demonstrate the practicality of the relationship we establish above and derive insights, we solve Equation 3.12 under three example π settings. In all settings, we assume $\mathcal{U}$ is the set of integers between $[0, 99]$ and d measures absolute value distance between two integers. We use π = Uniform, Gaussian and Exponential distributions with the corresponding distribution parameters given in Figure 3.2. Our rationale is that each π represents a different type of skewness: Uniform has no skewness, Gaussian is symmetrically skewed around the mean, and Exponential has positive skew. Users' secrets are samples from these distributions rounded to the nearest integer. Distribution parameters

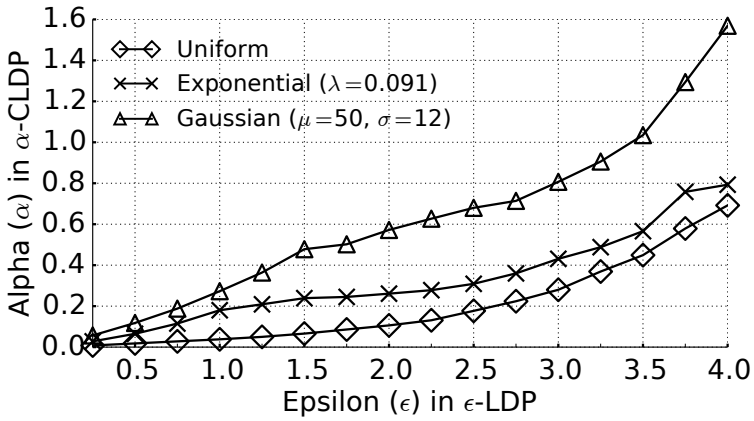


Figure 3.2: Exemplifying the relationship between ε in ε-LDP and α in α-CLDP according to Equation 3.12, with $|\mathcal{U}| = 100$ and $\pi =$ Uniform, Exponential, Gaussian.

are chosen so that users' secrets fall within the universe of $[0, 99]$ with non-negligible tail probabilities. In Figure 3.2, we provide the results of our study for a wide range of ε values used in the literature: $0.25 \leq \varepsilon \leq 4$. Note that this figure is obtained purely by solving Equation 3.12, and does not require a real simulation or execution of the protocol involving end clients, i.e., the solution can be used in setup time before any data collection occurs.

Figure 3.2 allows us to derive several interesting insights. The first important conclusion is that ε and α are positively correlated – as the privacy requirement of LDP is relaxed, we can also relax that of CLDP. Second, it is often the case that LDP and CLDP give equivalent protection when $\alpha \ll \varepsilon$ holds. This is because in order to compensate for the added term of $d(\cdot, \cdot)$ decreasing indistinguishability of distant items under CLDP, we have to use much smaller α in CLDP compared to ε of LDP. Third, we observe that the relationship between ε and α depends on π, e.g., if the data follows a Uniform distribution, CLDP must use a stricter α than the other two distributions. The reason is because there are two determining factors in calculating adversarial confidence: π and $\Pr[f(v) = y]$. For skewed distributions, π becomes the dominating factor and since the same π is shared by LDP and CLDP, the behavior of perturbation functions have relatively less impact on adversarial confidence. In contrast, for the Uniform distribution, since $\Pr[f(v) = y]$ becomes the dominating factor and its value is high for the tail-ends of the domain in the case of CLDP, we must use lower

(stricter) α in CLDP to match the adversarial confidence in LDP. Based on our findings and observation that the Uniform π setting causes lowest α under the same ε compared to other π settings, in the remainder of the chapter (including our experiments and case studies) we assume the Uniform π setting. This helps demonstrate CLDP's utility benefit in the strictest and most challenging setting.

Note on Alternate Threat and Adversary Models: The analysis perform in this section and the rest of this chapter is under the Bayesian adversary and measurement of MPC described in Section 3.2.1. Since Bayesian methods were previously used in the literature for evaluating DP variants and relaxations in different domains, we argue that the Bayesian adversary model is relevant and representative [26, 34, 91, 92]. Yet, we acknowledge that under different adversaries, the protection offered by LDP and CLDP could be different, and thus, a different conversion between ε and α could be needed. For example, one alternative method to compare LDP and CLDP is hypothesis testing [100, 101, 102]. We leave comparison between LDP and CLDP under such different protection models to future work.

3.4 CLDP Mechanisms and Protocols

In this section, we present protocols that can be used in practice to collect data while achieving CLDP. We present three protocols: Ordinal-CLDP, Item-CLDP, and Sequence-CLDP, to address different types of client data.

3.4.1 Ordinal-CLDP for Ordinal Items

Our first protocol is Ordinal-CLDP, which addresses data types that stem from finite metric spaces, i.e., $\mathcal{U}$ is discrete and finite, and there exists a built-in distance metric $d : \mathcal{U} \times \mathcal{U} \rightarrow [0, \infty)$. This setting covers a variety of useful data types: (i) discrete numeric or integer domains where d can be the absolute value distance between two items, (ii) ordinal item domains with total order, e.g., letters and strings ordered by dictionary order A $<$ B $<$ C

Algorithm 2: Ordinal-CLDP using EM

Input : α: CLDP privacy parameter, $\mathcal{U}$: item universe, d: distance metric, v: user's true value

Output: $v' \in \mathcal{U}$: perturbed value

for *each* $y \in \mathcal{U}$ **do**

  Assign $\text{score}(y) = e^{\frac{-\alpha \cdot d(v,y)}{2}}$

end

Pick a random sample v' from $\mathcal{U}$, where $\Pr[v' \text{ is sampled}] = \frac{\text{score}(v')}{\sum_{z \in \mathcal{U}} \text{score}(z)}$

return v'

$<$ ..., and (iii) categorical domains with tree-structured domain taxonomy where distance between two items can be measured using the depth of their most recent common ancestor in the taxonomy tree [103, 104]. In these scenarios, item order and d are naturally defined and enforced.

In Ordinal-CLDP, each client locally applies the Exponential Mechanism (EM) implementation shown in Algorithm 2, and uploads the perturbed output to the server. Notice that $\alpha, \mathcal{U}, d$, and user's true value v are all inputs to the algorithm. The algorithm's output v' is sent to the collector, thereby concluding the protocol in a single round without blocking.

Algorithm 2 has some desirable utility properties. First, since d is a metric, it holds that $d(v, v) = 0$ for all $v \in \mathcal{U}$. As a result, each invocation of Algorithm 2 is unbiased since $\Pr[\Phi_{ORD}(v) = v] > \Pr[\Phi_{ORD}(v) = y]$ where $y \neq v$ and Φ_{ORD} denotes Algorithm 2. Furthermore, when d is selected with the property that $d(v_i, v_j) = c$ for all $v_i, v_j \in \mathcal{U}$ where $v_i \neq v_j$ and c is a constant such as $c = 1$, then the aggregation of the outputs of Algorithm 2 on the server side preserves the relative frequency relationship between v_i and v_j in expectation. In other words, if the true frequency of v_i is larger than v_j [resp. smaller than], after each client uploads perturbed items resulting from Algorithm 2 and the counts of the perturbed items are computed on the server side, the observed frequency of v_i is expected to be larger than the observed frequency of v_j [resp. smaller]. Given that frequency relationships between individual pairs of items are preserved, general item frequency rankings are also preserved.

In order to prove this claim, we first introduce some notation. Let $v_i \in \mathcal{U}$ be an item, let true(v_i) be the true count of v_i in the population, and let obs(v_i) be the observed count of v_i from the aggregation of perturbed outputs of Algorithm 1. Then, the above claim is equivalent to the statement that iff true$(v_i) >$ true(v_j), then $\mathbb{E}[\text{obs}(v_i)] > \mathbb{E}[\text{obs}(v_j)]$. Given this holds for all $v_i, v_j \in \mathcal{U}$, the converse follows trivially that iff true$(v_i) <$ true(v_j), then $\mathbb{E}[\text{obs}(v_i)] < \mathbb{E}[\text{obs}(v_j)]$. Also, if the relative frequency rank relation between all pairs of individual items are preserved in expectation, it implies that that the complete ranking is also preserved. Thus, it suffices to prove the initial statement. We begin the proof by expressing $\mathbb{E}[\text{obs}(v_i)]$:

$$\begin{aligned}\mathbb{E}[\text{obs}(v_i)] = \ & \text{true}(v_i) \cdot \Pr[\Phi_{ORD}(v_i) = v_i] + \text{true}(v_j) \cdot \Pr[\Phi_{ORD}(v_j) = v_i] \\ & + \sum_{v_k \in \mathcal{U} \setminus \{v_i, v_j\}} \text{true}(v_k) \cdot \Pr[\Phi_{ORD}(v_k) = v_i]\end{aligned}$$

Similarly, $\mathbb{E}[\text{obs}(v_j)]$ can be expressed as:

$$\begin{aligned}\mathbb{E}[\text{obs}(v_j)] = \ & \text{true}(v_j) \cdot \Pr[\Phi_{ORD}(v_j) = v_j] + \text{true}(v_i) \cdot \Pr[\Phi_{ORD}(v_i) = v_j] \\ & + \sum_{v_k \in \mathcal{U} \setminus \{v_i, v_j\}} \text{true}(v_k) \cdot \Pr[\Phi_{ORD}(v_k) = v_j]\end{aligned}$$

For the special case choice of d that $d(v_i, v_j) = c$ for all $v_i, v_j \in \mathcal{U}$ where $v_i \neq v_j$ and c is a constant such as $c = 1$; we observe by construction of Φ_{ORD} that $\Pr[\Phi_{ORD}(v_i) = v_i] = \Pr[\Phi_{ORD}(v_j) = v_j]$, $\Pr[\Phi_{ORD}(v_j) = v_i] = \Pr[\Phi_{ORD}(v_i) = v_j]$, and $\Pr[\Phi_{ORD}(v_k) = v_i] = \Pr[\Phi_{ORD}(v_k) = v_j]$. Let constants c_1, c_2, and c_3 denote these three quantities respectively. Then, $\mathbb{E}[\text{obs}(v_i)] - \mathbb{E}[\text{obs}(v_j)]$ can be written as:

$$\begin{aligned}\mathbb{E}[\text{obs}(v_i)] - \mathbb{E}[\text{obs}(v_j)] = \ & c_1 \cdot (\text{true}(v_i) - \text{true}(v_j)) - c_2 \cdot (\text{true}(v_i) - \text{true}(v_j)) \\ & + c_3 \cdot \sum_{v_k \in \mathcal{U} \setminus \{v_i, v_j\}} (\text{true}(v_k) - \text{true}(v_k))\end{aligned}$$

which can be simplified to:

$$\mathbb{E}[\text{obs}(v_i)] - \mathbb{E}[\text{obs}(v_j)] = (c_1 - c_2) \cdot (\text{true}(v_i) - \text{true}(v_j)) \tag{3.13}$$

By construction of Φ_{ORD}, we know that $c_1 > c_2$. Then, Equation 3.13 shows that $\mathbb{E}[\text{obs}(v_i)] - \mathbb{E}[\text{obs}(v_j)]$ and $\text{true}(v_i) - \text{true}(v_j)$ are directly positively correlated. In other words, iff $\text{true}(v_i) > \text{true}(v_j)$, then $\mathbb{E}[\text{obs}(v_i)] > \mathbb{E}[\text{obs}(v_j)]$ and vice versa. This proves the intended statement.

3.4.2 Item-CLDP for Non-Ordinal Items

Our second protocol is Item-CLDP in which each user still holds a singleton true item, but the items come from an arbitrary $\mathcal{U}$ with no pre-defined d or total order. For example, if $\mathcal{U}$ consists of OS names, an order of MacOS $<$ Ubuntu $<$ Windows is neither available nor initially justifiable. This non-ordinal item setting has been assumed in recent LDP research for finding popular emojis, emerging slang terms, popular and anomalous browser homepages, and merchant transactions [14, 84, 86, 16, 96]. We propose Item-CLDP in a generic way to maximize its scope and cover such existing cases. Parallel to previous works, our goal is to uphold relative item frequencies to learn popularity histograms and discover heavy hitters. To this end, we propose that a desirable perturbation strategy should replace a *popular* item with another popular item, and an *uncommon* item with another uncommon item. This achieves our goal of upholding relative item frequencies, as the expected behavior (conceptually) will be that popular items and uncommon items will be shuffled among themselves, and relative frequencies will be preserved.

The proposed Item-CLDP protocol is given in Figure 3.3. In Item-CLDP the server communicates with each client twice, hence the protocol consists of two rounds. The first round contains steps 1-3 and the second round contains steps 3-5. The server executes the first round with each client in parallel (without blocking). At the end of the first round, the

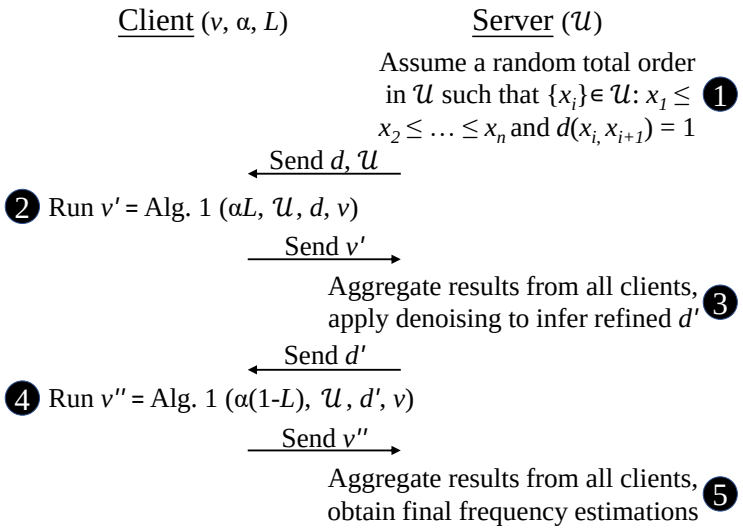


Figure 3.3: Item-CLDP protocol to report non-ordinal items.

server performs the aggregation and de-noising step (Step 3). Then, the server executes the second round of communication. Next, we explain each step in detail.

Step 1: When the protocol starts, the server knows universe $\mathcal{U}$ and each client has a true value v. The value of the privacy budget α and budget allocation parameter $0 < L < 1$ can be publicly known. (The role of L will be explained later.) A random total order is constructed among all items in $\mathcal{U}$ such that for $\{x_i\} \in \mathcal{U}$, $d(x_i, x_{i+1}) = 1$. The server advertises $\mathcal{U}$ and d to all clients.

Step 2: Each client runs Algorithm 2 with budget $\alpha \cdot L$ to obtain a perturbed value v' locally on their device. Then, the clients send their v' to the server.

Step 3: Due to the utility-unaware choice of d in Step 1, the absolute item frequencies discovered at this step contain significant error. A second round is desirable to reduce error. We found that although the server does not accurately learn absolute item frequencies by the end of Step 2, it can learn frequency *ranking* of items after applying a de-noising strategy. A key aspect is how de-noising is performed. Let $y \in \mathcal{U}$ be an item, let true(y) be the true count of y in the population, which we are trying to find, and let obs(y) denote the observed count of y following the first round of client-server communication (i.e., by the beginning

of Step 3). The following holds in expectation:

$$\text{obs}(y) = \text{true}(y) \cdot \Pr[\Phi_{EM}(y) = y] + \sum_{x \in \mathcal{U} \setminus \{y\}} \text{true}(x) \cdot \Pr[\Phi_{EM}(x) = y]$$

Our goal is to solve for true(y), but we cannot do so since true(x) is also unknown. Hence, in our de-noising strategy we make the heuristic decision of plugging obs(x) in place of true(x), thereby obtaining true$'(y)$ as follows:

$$\text{true}'(y) = \frac{\text{obs}(y) - \sum_{x \in \mathcal{U} \setminus \{y\}} \text{obs}(x) \cdot \Pr[\Phi_{EM}(x) = y]}{\Pr[\Phi_{EM}(y) = y]}$$

We apply this to all y's in $\mathcal{U}$ and rank them according to their true$'(y)$. The distance function d' is set to reflect this new ranking instead of the original d from Step 1.

Step 4: After the clients receive d', each client runs Algorithm 2 with d' to obtain v'' and sends v'' to the server. This invocation of the algorithm is with budget $(1 - L) \cdot \alpha$.

Step 5: Upon receiving the v'' values from all clients, the server aggregates all results and obtains the final frequency estimates.

Since Item-CLDP is a two-round protocol, each client sends perturbed information twice. Hence, we need to quantify the total disclosure by the end of two rounds. We introduce the parameter L, which takes values between 0 and 1 and determines how the CLDP privacy budget will be allocated to the two rounds of Item-CLDP. Denoting Item-CLDP by Φ_{ITEM}, we can show that for any two possible inputs v_1,v_2 of a user:

$$\begin{aligned} \frac{\Pr[\Phi_{ITEM}(v_1) = \langle v', v'' \rangle]}{\Pr[\Phi_{ITEM}(v_2) = \langle v', v'' \rangle]} &\leq e^{\alpha \cdot L \cdot d(v_1, v_2)} \cdot e^{\alpha \cdot (1-L) \cdot d'(v_1, v_2)} \\ &\leq e^{\alpha \cdot \max\{d(v_1, v_2), d'(v_1, v_2)\}} \end{aligned}$$

This follows from the fact that the first round satisfies $(\alpha \cdot L)$-CLDP with d, and the second round satisfies $\alpha \cdot (1 - L)$-CLDP with d'. We choose the value of L by finding which L yields minimum frequency estimation error by the end of the second round of Item-CLDP.

According to our experiments with different L, we recommend $L \cong 0.8$ as it often gives best results, which indicates that finding an accurate preliminary ranking in the first round of Item-CLDP is indeed important to obtain a good final result. An important property enabling the composition of Item-CLDP in above fashion is the fact that by construction of d and d' as metrics, $\max\{d(v_1, v_2), d'(v_1, v_2)\}$ is also a metric. This enables composition of the two individual CLDP rounds, and ensures their composed outcome achieves CLDP. We provide the proof of this factor enabling the composition of Item-CLDP below.

Lemma 1. *Given that $d(v_i, v_j)$ and $d'(v_i, v_j)$ are metrics, $d^*(v_i, v_j) = \max\{d(v_i, v_j), d'(v_i, v_j)\}$ is also a metric.*

Proof. First recall the properties of being a metric, which are satisfied by d and d'. We write these properties for d, as d' is analogous:

1. $d(v_i, v_j) \geq 0, \forall v_i, v_j \in \mathcal{U}$
2. $d(v_i, v_j) = 0$ if and only if $v_i = v_j$
3. $d(v_i, v_j) = d(v_j, v_i), \forall v_i, v_j \in \mathcal{U}$
4. $d(v_i, v_k) \leq d(v_i, v_j) + d(v_j, v_k), \forall v_i, v_j, v_k \in \mathcal{U}$

To prove that d^* is also a metric, we must show that d^* satisfies the four properties above. We prove each property one by one.

Property 1: We know already that $d(v_i, v_j) \geq 0$ and $d'(v_i, v_j) \geq 0$ for all v_i, v_j. $d^*(v_i, v_j)$ takes the maximum of these two non-negative values, which must also be non-negative. Thus, $d^*(v_i, v_j) \geq 0$.

Property 2: There are two directions to prove. First, when $v_i = v_j$ then $d(v_i, v_j) = 0$ and $d'(v_i, v_j) = 0$. As a result, $d^*(v_i, v_j) = \max\{0, 0\} = 0$, which concludes the first direction.

For the second direction, we prove by contradiction. Assume that $d^*(v_i, v_j) = 0$, but contrary to expectation $v_i \neq v_j$. If $v_i \neq v_j$, by the first and second properties satisfied by

d and d', either $d(v_i, v_j) > 0$ or $d'(v_i, v_j) > 0$ must hold. However, if either holds, the maximum of $d(v_i, v_j)$ and $d'(v_i, v_j)$ cannot be zero, since at least one of them is positive. Thus, the initial assumption of $d^*(v_i, v_j) = 0$ is violated, which yields a contradiction. It must be that $v_i = v_j$.

Property 3: The symmetry property of d^* follows from the individual symmetry properties satisfied by d and d'.

$$\begin{aligned} d^*(v_i, v_j) &= \max\{d(v_i, v_j), d'(v_i, v_j)\} \\ &= \max\{d(v_j, v_i), d'(v_j, v_i)\} \\ &= d^*(v_j, v_i) \end{aligned}$$

Property 4: By definition of d^*, $d^*(v_i, v_k)$ is either equal to $d(v_i, v_k)$ or $d'(v_i, v_k)$. Without loss of generality assume $d^*(v_i, v_k) = d(v_i, v_k)$; the case with $d^*(v_i, v_k) = d'(v_i, v_k)$ is identical. Using the fourth property of d, we can derive:

$$\begin{aligned} d^*(v_i, v_k) &= d(v_i, v_k) \\ &\leq d(v_i, v_j) + d(v_j, v_k) \\ &\leq d^*(v_i, v_j) + d^*(v_j, v_k) \end{aligned}$$

since for each individual term on the right hand side, it is guaranteed by definition of d^* that $d^*(v_i, v_j) \geq d(v_i, v_j)$ and $d^*(v_j, v_k) \geq d(v_j, v_k)$. □

3.4.3 Sequence-CLDP for Item Sequences

In Ordinal-CLDP and Item-CLDP, each user reports a single item. We now study the case where each user reports a collection of items. We give our Sequence-CLDP protocol assuming this collection forms a sequence and later show the applicability of Sequence-CLDP to set-valued data. Sequential data arises naturally in many domains, including

Algorithm 3: Sequence-CLDP randomization

Parameter: Probabilities *halt*, *gen*, maximum length allowed *max_len*
Input : Client's private sequence X, privacy budget α, item universe $\mathcal{U}$, distance metric d
Output : Randomized sequence S

while X *is shorter than max_len* **do**
  Pad X with stop sign $\perp$ `// Dummy symbol`
end
Initialize empty sequence S
for $i = 1$ *to max_len* **do**
  if $X[i] \neq \perp$ **then**
    With probability *halt*, stop here and return S
    With probability $(1 - \textit{halt})$, run Alg. 2 with inputs (α, $\mathcal{U}$, d, $X[i]$) to obtain v', and append v' to S
  end
  else
    With probability *gen*, randomly pick an item from $\mathcal{U}$ and append it to S
    With probability $(1 - \textit{gen})$, stop here and return S
  end
end
return S

cybersecurity (log files), genomics (DNA sequences), web browsing histories, and mobility traces; thus, a protocol for privacy-preserving collection of item sequences holds great practical value. We denote by X a user's true sequence, and by $X[i]$ the i'th element in X. Each element $X[i]$ is an item from universe $\mathcal{U}$. We assume the distance metric d between individual items is known apriori, e.g., for ordinal $\mathcal{U}$ we can use built-in d as in Ordinal-CLDP; otherwise, we can infer d using a process similar to the first round of Item-CLDP. We measure distance between two sequences $d_{\text{seq}}(X, Y)$ as:

$$d_{\text{seq}}(X, Y) = \sum_{i=1}^{|X|} d(X[i], Y[i])$$

In Sequence-CLDP, each client runs the sequence randomization procedure given in Algorithm 3 to locally perturb their X. The procedure has two probability parameters: $0 < \textit{halt, gen} < 1$, and a length parameter *max_len* denoting the maximum sequence length

allowed. Given true sequence X, the algorithm returns a perturbed sequence S. Our goal in Sequence-CLDP is to hide two complementary types of information: the *length* of X and the *contents* of X. For example, let X consist of a sequence of security events observed on a machine. Hiding the length of X is useful because it disables the adversary from learning that many security events were observed on this machine, hence that the machine is probably infected. Hiding the contents of X is useful because it disables the adversary from learning *which* security events were observed, hence the adversary cannot infer which types of problems exist on the machine, which attacks are successful, and so forth. Denoting Sequence-CLDP by Φ_{SEQ}, we formalize these privacy properties as follows.

Definition 3. *Let $Pr[\Phi_{SEQ}(X) \rightsquigarrow \ell]$ denote the probability that $\Phi_{SEQ}(X)$ produces a perturbed sequence of length ℓ given input sequence X. We say that Φ_{SEQ} satisfies α-length-indistinguishability if for any pair of true sequences X, Y:*

$$\frac{Pr[\Phi_{SEQ}(X) \rightsquigarrow \ell]}{Pr[\Phi_{SEQ}(Y) \rightsquigarrow \ell]} \leq e^{\alpha \cdot abs(|X|-|Y|)}$$

Definition 4. *Let $Pr[\Phi_{SEQ}(X) = S]$ denote the probability that $\Phi_{SEQ}(X)$ produces perturbed sequence S given input sequence X. We say that Φ_{SEQ} satisfies α-content-indistinguishability if, for any pair of true sequences X, Y of same length, it holds that:*

$$\frac{Pr[\Phi_{SEQ}(X) = S]}{Pr[\Phi_{SEQ}(Y) = S]} \leq e^{\alpha \cdot d_{seq}(X,Y)}$$

It can be observed that both properties are adaptations of CLDP for hiding the two types of sensitive information relating to sequences: length and content. Definition 3 is analogous to CLDP with metric d being the absolute value difference between sequence lengths. It ensures that an adversary observing the length of the perturbed sequence ℓ cannot infer the length of the user's true sequence with high confidence. Definition 4 is analogous to CLDP with metric d being d_{seq}, i.e., item-wise difference in sequences' contents. It ensures that an adversary observing the contents of the perturbed sequence S cannot infer the contents of

the user's true sequence with high confidence. The combined privacy protections offered by Definition 3 and Definition 4 can be best explained from the following perspective: Given that the adversary observes a perturbed sequence S with length $\ell = |S|$ that a user sends to the data collector, can the adversary reverse-engineer the length or the contents of the user's true sequence? If the perturbation satisfies Definitions 3 and 4 simultaneously, then the answer is negative, since both sequence length and content are protected analogously to the protection offered by CLDP.

Theorem 3. *Algorithm 3 satisfies α-length-indistinguishability and α-content-indistinguishability simultaneously if halt, gen are chosen either symmetrically as:*

$$\textit{halt} = \textit{gen} = \frac{1}{e^{\alpha}+1}$$

or asymmetrically within the ranges:

$$0 < \textit{halt} < \frac{1}{e^{\alpha}+1} \;\; \textit{and} \;\; 1 - e^{\alpha} \cdot \textit{halt} \leq \textit{gen} \leq 1 - \frac{\textit{halt}}{e^{\alpha}}$$

Proof. We first prove that Sequence-CLDP, denoted here onwards by Φ_{SEQ}, satisfies α-length-indistinguishability for the given parameter choices. Observe from Algorithm 3 that given a true sequence X of length $|X|$, Φ_{SEQ} produces an output sequence of length ℓ with probability:

$$\Pr[\Phi_{SEQ}(X) \rightsquigarrow \ell] = \begin{cases} \textit{halt} \cdot (1-\textit{halt})^{\ell} & \text{when } \ell < |X| \\ (1-\textit{halt})^{|X|} \cdot (1-\textit{gen}) \cdot \textit{gen}^{\ell-|X|} & \text{when } \ell \geq |X| \end{cases}$$

We consider several disjoint cases depending on the length of true sequences X, Y. The parameters must be chosen so that all cases are simultaneously satisfied.

Case 0: $|X| = |Y|$. This case is trivial since Φ_{SEQ} treats their length equally, resulting

in:

$$\frac{\Pr[\Phi_{SEQ}(X) \rightsquigarrow \ell]}{\Pr[\Phi_{SEQ}(Y) \rightsquigarrow \ell]} = 1 = e^{\alpha \cdot 0} \qquad \text{if } |X| = |Y|$$

Remaining cases fall under $|X| \neq |Y|$, and are analyzed case-by-case below.

Case 1: $\ell < |X|$ and $\ell < |Y|$

$$\frac{\Pr[\Phi_{SEQ}(X) \rightsquigarrow \ell]}{\Pr[\Phi_{SEQ}(Y) \rightsquigarrow \ell]} = \frac{\textit{halt} \cdot (1 - \textit{halt})^{\ell}}{\textit{halt} \cdot (1 - \textit{halt})^{\ell}} = 1 \leq e^{\alpha \cdot abs(|X| - |Y|)}$$

Since $abs(|X| - |Y|) \geq 0$, we trivially have $e^{\alpha \cdot abs(|X| - |Y|)} \geq 1$, hence this case is satisfied without placing constraints on the values of *halt*, *gen*.

Case 2: $\ell \geq |X|$ and $\ell \geq |Y|$

$$\begin{aligned}\frac{\Pr[\Phi_{SEQ}(X) \rightsquigarrow \ell]}{\Pr[\Phi_{SEQ}(Y) \rightsquigarrow \ell]} &= \frac{(1 - \textit{halt})^{|X|} \cdot (1 - \textit{gen}) \cdot \textit{gen}^{\ell - |X|}}{(1 - \textit{halt})^{|Y|} \cdot (1 - \textit{gen}) \cdot \textit{gen}^{\ell - |Y|}} \\ &= (1 - \textit{halt})^{|X| - |Y|} \cdot \textit{gen}^{|Y| - |X|}\end{aligned}$$

Divide this into two subcases:

2a: $|X| < |Y|$: When this holds, the requirement that

$$(1 - \textit{halt})^{|X| - |Y|} \cdot \textit{gen}^{|Y| - |X|} \leq e^{\alpha \cdot abs(|X| - |Y|)}$$

can be written as:

$$\begin{aligned}\frac{\textit{gen}^{|Y| - |X|}}{(1 - \textit{halt})^{|Y| - |X|}} &\leq e^{\alpha \cdot (|Y| - |X|)} \\ (\frac{\textit{gen}}{1 - \textit{halt}})^{|Y| - |X|} &\leq e^{\alpha \cdot (|Y| - |X|)}\end{aligned}$$

Thus, we have the constraints for parameters *halt*, *gen* as:

$$\frac{\textit{gen}}{1 - \textit{halt}} \leq e^{\alpha} \tag{3.14}$$

2b: $|X| > |Y|$: When this holds, the same requirement can be written as:

$$\frac{(1-\mathit{halt})^{|X|-|Y|}}{\mathit{gen}^{|X|-|Y|}} \leq e^{\alpha\cdot(|X|-|Y|)}$$
$$(\frac{1-\mathit{halt}}{\mathit{gen}})^{|X|-|Y|} \leq e^{\alpha\cdot(|X|-|Y|)}$$

resulting in the parameter constraints:

$$\frac{1-\mathit{halt}}{\mathit{gen}} \leq e^{\alpha} \tag{3.15}$$

Case 3: $\ell < |X|$ and $\ell \geq |Y|$

$$\frac{\Pr[\Phi_{SEQ}(X) \rightsquigarrow \ell]}{\Pr[\Phi_{SEQ}(Y) \rightsquigarrow \ell]} \leq e^{\alpha\cdot abs(|X|-|Y|)}$$
$$\frac{\mathit{halt}\cdot(1-\mathit{halt})^{\ell}}{(1-\mathit{halt})^{|Y|}\cdot(1-\mathit{gen})\cdot\mathit{gen}^{\ell-|Y|}} \leq e^{\alpha\cdot abs(|X|-|Y|)}$$
$$\frac{\mathit{halt}}{1-\mathit{gen}}\cdot(\frac{1-\mathit{halt}}{\mathit{gen}})^{\ell-|Y|} \leq e^{\alpha\cdot abs(|X|-|Y|)}$$

Since the assumption in this case is $|Y| \leq \ell < |X|$, we can rewrite the RHS as:

$$\frac{\mathit{halt}}{1-\mathit{gen}}\cdot(\frac{1-\mathit{halt}}{\mathit{gen}})^{\ell-|Y|} \leq e^{\alpha\cdot(\ell-|Y|)}\cdot e^{\alpha\cdot(|X|-\ell)}$$

Given the constraint we established in Equation 3.15 holds, the following is a sufficient condition to satisfy the above:

$$\frac{\mathit{halt}}{1-\mathit{gen}} \leq e^{\alpha\cdot(|X|-\ell)}$$

Notice that, by assumption, $|X| > \ell$ and both $|X|$ and ℓ are integers representing sequence length. Then, $|X| - \ell \geq 1$, making the following parameter constraint sufficient:

$$\frac{\mathit{halt}}{1-\mathit{gen}} \leq e^{\alpha} \tag{3.16}$$

Case 4: $\ell \geq |X|$ and $\ell < |Y|$

$$\frac{\Pr[\Phi_{SEQ}(X) \rightsquigarrow \ell]}{\Pr[\Phi_{SEQ}(Y) \rightsquigarrow \ell]} \leq e^{\alpha \cdot abs(|X|-|Y|)}$$
$$\frac{(1-\mathit{halt})^{|X|} \cdot (1-\mathit{gen}) \cdot \mathit{gen}^{\ell-|X|}}{\mathit{halt} \cdot (1-\mathit{halt})^{\ell}} \leq e^{\alpha \cdot abs(|X|-|Y|)}$$
$$\frac{1-\mathit{gen}}{\mathit{halt}} \cdot (\frac{\mathit{gen}}{1-\mathit{halt}})^{\ell-|X|} \leq e^{\alpha \cdot abs(|X|-|Y|)}$$

Since the assumption in this case is $|X| \leq \ell < |Y|$, we can rewrite the RHS as:

$$\frac{1-\mathit{gen}}{\mathit{halt}} \cdot (\frac{\mathit{gen}}{1-\mathit{halt}})^{\ell-|X|} \leq e^{\alpha \cdot (|Y|-\ell)} \cdot e^{\alpha \cdot (\ell-|X|)}$$

Given the constraint we established in Equation 3.14 holds, the following is a sufficient condition to satisfy the above:

$$\frac{1-\mathit{gen}}{\mathit{halt}} \leq e^{\alpha \cdot (|Y|-\ell)}$$

Since $|Y| > \ell$ and both $|Y|$ and ℓ are integers, we have $|Y| - \ell \geq 1$, making the following parameter constraint sufficient:

$$\frac{1-\mathit{gen}}{\mathit{halt}} \leq e^{\alpha} \tag{3.17}$$

Combine all cases: Finally, we combine the parameter constraints we identified at the end of each case (Equations 3.14, 3.15, 3.16, and 3.17) to obtain a system of equations:

$$\frac{\mathit{gen}}{1-\mathit{halt}} \leq e^{\alpha} \qquad \frac{1-\mathit{halt}}{\mathit{gen}} \leq e^{\alpha} \qquad \frac{\mathit{halt}}{1-\mathit{gen}} \leq e^{\alpha} \qquad \frac{1-\mathit{gen}}{\mathit{halt}} \leq e^{\alpha}$$

Given the privacy parameter α, the values of *halt*, *gen* satisfying all four equations simultaneously satisfy α-length-indistinguishability. If we set *halt*=*gen* and solve this system of equations, we observe that the following is a solution (which we call the *symmetric*

parameter choice):

$$\textit{halt} = \textit{gen} = \frac{1}{e^{\alpha}+1}$$

Another solution to the same system of equations, which we call the *asymmetric parameter choice*, is desirable when input sequences X, Y are longer and therefore a smaller *halt*ing probability is preferable:

$$0 < \textit{halt} < \frac{1}{e^{\alpha}+1} \text{ and } 1 - e^{\alpha} \cdot \textit{halt} \leq \textit{gen} \leq 1 - \frac{\textit{halt}}{e^{\alpha}}$$

Next, we prove that for the above choices of parameters, Φ_{SEQ} satisfies α-content-indistinguishability. We divide into 3 possible cases depending on how $|S|$ relates to $|X| = |Y|$.

Case 1: $|S| = |X| = |Y|$. In this case, Algorithm 2 must have behaved as follows. Upon reaching its main loop (lines 5-14), it must have run lines 6-9 for $i = 1$ to *max_len* and, in each iteration, the event with probability $(1 - \textit{halt})$ must have occurred such that $X[i]$ was perturbed and the perturbed item was appended to S. The perturbation is performed by the function call to Algorithm 2, which we denote by Φ_{EM}. Then, in iteration $i = \textit{max\_len} + 1$, the event with probability $(1 - \textit{gen})$ must have occurred so that a sequence S with length exactly equal to $|X| = |Y| = n$ was returned. The odds-ratio probabilities of the overall run can be computed as:

$$\begin{aligned}
\frac{\Pr[\Phi_{SEQ}(X) = S]}{\Pr[\Phi_{SEQ}(Y) = S]} &\overset{?}{\leq} e^{\alpha \cdot d_{\text{seq}}(X,Y)} \\
\frac{(1-\textit{gen}) \cdot \prod_{i=1}^{n}(1-\textit{halt}) \cdot \Pr[\Phi_{EM}(X[i]) = S[i]]}{(1-\textit{gen}) \cdot \prod_{i=1}^{n}(1-\textit{halt}) \cdot \Pr[\Phi_{EM}(Y[i]) = S[i]]} &\overset{?}{\leq} e^{\alpha \cdot d_{\text{seq}}(X,Y)} \\
\frac{\prod_{i=1}^{n} \Pr[\Phi_{EM}(X[i]) = S[i]]}{\prod_{i=1}^{n} \Pr[\Phi_{EM}(Y[i]) = S[i]]} &\overset{?}{\leq} e^{\alpha \cdot d_{\text{seq}}(X,Y)}
\end{aligned}$$

By Theorem 2, for any index i:

$$\frac{\Pr[\Phi_{EM}(X[i]) = S[i]]}{\Pr[\Phi_{EM}(Y[i]) = S[i]]} \leq e^{\alpha \cdot d(X[i], Y[i])} \tag{3.18}$$

Applying Equation 3.18, we obtain:

$$\prod_{i=1}^{n} e^{\alpha \cdot d(X[i], Y[i])} \stackrel{?}{\leq} e^{\alpha \cdot d_{\text{seq}}(X,Y)}$$
$$e^{\alpha \cdot \sum_{i=1}^{n} d(X[i], Y[i])} \leq e^{\alpha \cdot d_{\text{seq}}(X,Y)}$$

which holds by definition of d_{seq}, completing the proof for Case 1.

Case 2: $|S| < |X| = |Y|$. In this case, Algorithm 3 must have run lines 6-9 for $|S| + 1$ iterations; in the first $|S|$ iterations, the event with probability $(1 - \textit{halt})$ must have occurred to build the perturbed sequence S, and in the last iteration the *halt*ing event must have occurred so that the sequence of length $|S| = m$ was returned without adding more elements. The odds-ratio becomes:

$$\frac{\Pr[\Phi_{SEQ}(X) = S]}{\Pr[\Phi_{SEQ}(Y) = S]} \stackrel{?}{\leq} e^{\alpha \cdot d_{\text{seq}}(X,Y)}$$
$$\frac{\textit{halt} \cdot \prod_{i=1}^{m}(1 - \textit{halt}) \cdot \Pr[\Phi_{EM}(X[i]) = S[i]]}{\textit{halt} \cdot \prod_{i=1}^{m}(1 - \textit{halt}) \cdot \Pr[\Phi_{EM}(Y[i]) = S[i]]} \stackrel{?}{\leq} e^{\alpha \cdot d_{\text{seq}}(X,Y)}$$
$$\frac{\prod_{i=1}^{m} \Pr[\Phi_{EM}(X[i]) = S[i]]}{\prod_{i=1}^{m} \Pr[\Phi_{EM}(Y[i]) = S[i]]} \stackrel{?}{\leq} e^{\alpha \cdot d_{\text{seq}}(X,Y)}$$

Notice that this case is different than Case 1 in the bounds of the product—the product goes from $i = 1$ to length of $|S|$, which is shorter than $|X|$. Applying Equation 3.18:

$$e^{\alpha \cdot \sum_{i=1}^{m} d(X[i], Y[i])} \stackrel{?}{\leq} e^{\alpha \cdot d_{\text{seq}}(X,Y)}$$

Distance metric d satisfies the non-negativity property by definition. Therefore, for $m <$

$n = |X|$, we have:

$$\sum_{i=1}^{m} d(X[i], Y[i]) \leq \sum_{i=1}^{n} d(X[i], Y[i]) = d_{\text{seq}}(X, Y)$$

completing the proof for Case 2.

Case 3: $|S| > |X| = |Y|$. In this case, Algorithm 3 must have run lines 6-9 for $|X|$ iterations, and in each iteration the event with probability $(1 - \textit{halt})$ must have occurred. Then, for $|S| - |X|$ iterations, lines 10-13 must have run, and the item generation event with probability *gen* must have occurred in these iterations. Finally, to return S, a final iteration must have occurred with the stopping event having $(1 - \textit{gen})$ probability. Let $m = |S|$ and $n = |X|$. The odds-ratio is:

$$\begin{aligned}
&\frac{\Pr[\Phi_{SEQ}(X) = S]}{\Pr[\Phi_{SEQ}(Y) = S]} \overset{?}{\leq} e^{\alpha \cdot d_{\text{seq}}(X,Y)} \\
&\frac{\begin{array}{l}(1 - \textit{gen}) \cdot \prod_{i=1}^{m-n} \textit{gen} \cdot Pr[\text{random} = S[i]] \\ \quad \cdot \prod_{i=1}^{n} (1 - \textit{halt}) \cdot \Pr[\Phi_{EM}(X[i]) = S[i]]\end{array}}{\begin{array}{l}(1 - \textit{gen}) \cdot \prod_{i=1}^{m-n} \textit{gen} \cdot Pr[\text{random} = S[i]] \\ \quad \cdot \prod_{i=1}^{n} (1 - \textit{halt}) \cdot \Pr[\Phi_{EM}(Y[i]) = S[i]]\end{array}} \overset{?}{\leq} e^{\alpha \cdot d_{\text{seq}}(X,Y)}
\end{aligned}$$

where $Pr[\text{random} = S[i]]$ denotes the probability that the random sampling on line 11 of Algorithm 3 returns item $S[i]$. Note that the random sampling does not depend on the properties of inputs X, Y, therefore we can safely cancel most terms. We end up with:

$$\frac{\prod_{i=1}^{n} \Pr[\Phi_{EM}(X[i]) = S[i]]}{\prod_{i=1}^{n} \Pr[\Phi_{EM}(Y[i]) = S[i]]} \overset{?}{\leq} e^{\alpha \cdot d_{\text{seq}}(X,Y)}$$

Applying Equation 3.18 and continuing in the same fashion as Case 1, it is straightforward to complete the proof. □

In Algorithm 3, high *halt* causes the algorithm to terminate early for a long sequence, causing the perturbed sequence S to be much shorter than X. High *gen* adds random items

to S, thus it causes S to be much longer than X; in addition, since the added items are sampled uniformly at random, S will contain bogus elements. If we consider only the utility perspective, simultaneously decreasing the values of *halt* and *gen* yields higher sequence utility. However, Theorem 3 places bounds on the values of *halt* and *gen*; we cannot arbitrarily decrease them, otherwise we will not satisfy the indistinguishability properties. Among the given choices, asymmetric parameter choice is preferable when we expect users' true sequences to be long, since it assigns a lower halting probability compared to the symmetric case, thereby decreasing the probability that the algorithm is terminated early. This is done at the cost of increased *gen*, which implies that the asymmetric case will more likely add synthetic elements to S. Since this is detrimental to utility especially when users' true sequences are short, for short sequences, we recommend using the symmetric parameter choice.

Application to set-valued data: Although Sequence-CLDP is designed for sequences, it can be applied to set-valued data without information loss as follows. First, each user enforces a random ordering among the items in their itemset, to convert the itemset to a sequence. Second, the user runs Sequence-CLDP on this converted sequence to obtain a perturbed sequence. Third, the user removes the ordering from the perturbed sequence to obtain a perturbed itemset. Finally, the perturbed itemset is sent to the server. On the other hand, we cannot use existing set-valued LDP protocols [97, 105] on sequences without losing their sequentiality (ordering) aspect. Hence, we believe Sequence-CLDP has wider applicability than existing set-valued protocols.

3.5 Experimental Evaluation

We compare our proposed CLDP protocols against the existing LDP protocols on real-world cybersecurity datasets provided by Symantec as well as on public datasets. In singleton item comparison, we use RAPPOR (proposed and deployed by Google [14]) and OLH (recent protocol with improved utility over prior works [84]). In set-valued setting,

we use SVIM as the current state-of-the-art LDP protocol [105]. In each experimental dataset and setting, given $\mathcal{U}$, d and ε, we freshly execute Equation 3.12 and the process in Section 3.3.2 to obtain the appropriate α parameter for CLDP to ensure a fair comparison under each individual setting. Uniform π is used in each execution of Equation 3.12 since it is the most challenging π setting for CLDP to demonstrate its utility.

To ensure our experimental comparison between LDP and CLDP is fair with respect to the level of privacy protection under the Bayesian adversary model, we perform the following steps. First, we simulate data collection with LDP protocols and chosen ε, and measure the utility loss and privacy protection in terms of MPC. Second, we use the technique in Section 3.3.2 to determine the α that should be used for CLDP to match the protection achieved by LDP. Third, we simulate data collection with CLDP protocols and α from the previous step. Finally, we compare the utility loss caused by CLDP versus the utility loss of LDP.

In Section 3.5.1, we consider cybersecurity use cases that reflect the limitations of existing LDP protocols, e.g., user populations are small and sequential datasets cannot be handled. In these experiments, our results show that LDP protocols do not yield sufficient utility while our CLDP protocols offer satisfactory utility in most cases, hence their use in corresponding security products is practical and preferable. In Section 3.5.2, we experiment on public datasets which differ from the above since they do not reflect the limitations of LDP protocols, e.g., user populations are sufficiently large (over half a million). Even so, we show that our CLDP protocols perform at least comparable to, or in many cases better than, existing LDP protocols. An interesting result is that while LDP protocols are capable of finding the frequencies and ranking of heaviest hitter items with good accuracy (e.g., top 5-10% of the universe), CLDP protocols' accuracy is similar for these few heavy hitters, but they significantly outperform LDP protocols for remaining items (medium frequency and infrequent items).

3.5.1 Case Studies with Cybersecurity Datasets

Cybersecurity datasets provided by Symantec allowed us to test the accuracy of our protocols on pertinent real-world use cases and assess their practical applicability. We report three case studies: ransomware outbreak detection, ransomware vulnerability analysis, and inspecting suspicious activity.

Case Study 1: Ransomware Outbreak Detection

Setup: We consider the case where Symantec collects malware reports from machines running its anti-malware protection software. Each machine sends a locally private malware report to Symantec daily, containing the count of malware-related events observed on that machine during that day. Privacy is injected into malware reports by modifying the actual counts with LDP/CLDP.

For our experiments, we obtained the daily infection counts of two ransomware variants within those time periods in which we already know there were global outbreaks. Specifically, we considered the infections reported for *Cerber* between March 22 and April 21 in 2017 with the outbreak happening on April 6, and those reported for *Locky* between February 11 and March 13 in 2018 with the outbreak happening on February 26. We evaluated how accurately the total number of daily infections for these two ransomware variants can be estimated using our Ordinal-CLDP approach versus LDP approaches RAPPOR and OLH. The goal of our experiment is to retroactively test whether LDP/CLDP could identify if and when a ransomware outbreak happened.

Results: We illustrate the results in Figures 3.4 and 3.5 for Cerber and Locky respectively. If we use RAPPOR or OLH to perform detection, we obtain many false positives (days on which RAPPOR/OLH claim there was an outbreak, but in fact there was not) and false negatives (days on which RAPPOR/OLH claim there was no outbreak, but in fact there was). Some important examples are marked on the graphs, e.g., in Figure 3.4, RAPPOR raises false positives on March 23-24 as well as April 12-13. In addition, OLH

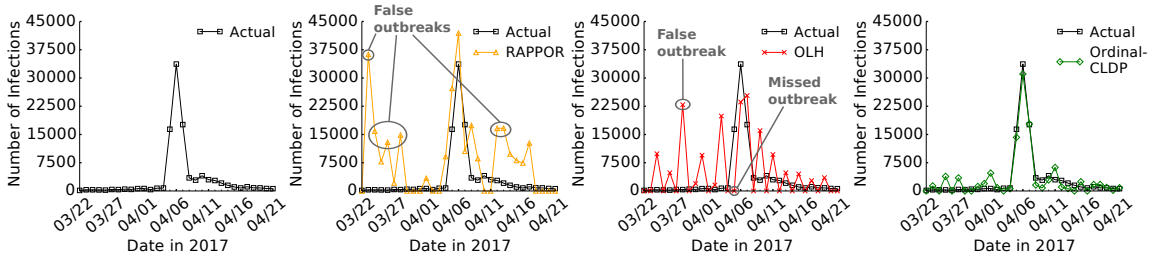


Figure 3.4: Monitoring the infections of ransomware Cerber for one month to detect a potential outbreak. Illustrated in the graphs is the actual number of infections versus infections reported by LDP protocols RAPPOR and OLH, and our Ordinal-CLDP protocol (ε = 1). Ordinal-CLDP enables accurate recovery of daily infection counts and detection of outbreaks without major false positives or negatives.

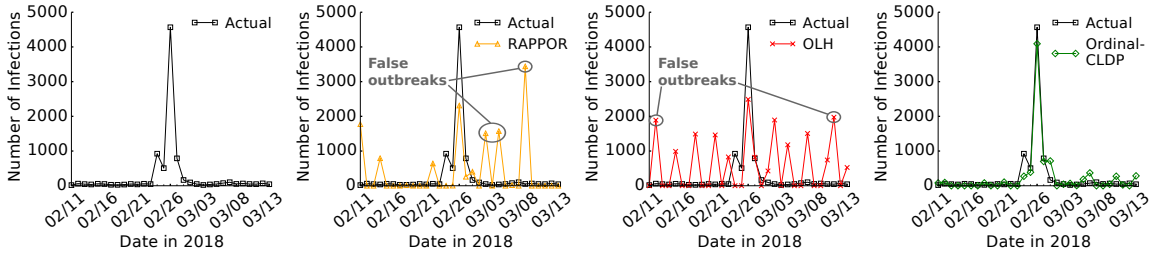


Figure 3.5: Monitoring the infections of ransomware Locky for one month to detect a potential outbreak. Refer to Figure 3.4 for a similar experiment with a different ransomware (ε = 1.0). Same conclusions apply to this figure.

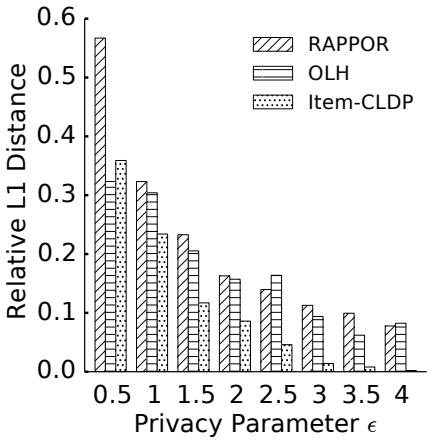
misses the onset of the outbreak happening on April 5 by reporting 0 observed infections whereas in reality there are 16,388 infections. False positives are costly to Symantec since they cause the company to devote resources and response teams to combat a malware outbreak that does not exist. False negatives are also costly since Symantec will not react to an outbreak in a timely manner, losing customer trust. Observing this many false positives and false negatives with LDP methods raises serious concerns. In contrast, using Ordinal-CLDP, Symantec can obtain daily infection counts with high accuracy. Note that in Figures 3.4 and 3.5, there are small discrepancies between the actual infections versus CLDP's predicted infections, which demonstrates that CLDP is not error-free. Nevertheless, using CLDP ransomware outbreaks can be detected in a privacy-preserving manner without major false positives or negatives.

Case Study 2: Ransomware Vulnerability Analysis

Setup: Next, we ask the question: Can we find which operating systems were most infected by ransomware? This would assist Symantec in discovering vulnerable or targeted OSs. When performing this analysis, we focus specifically on the day of outbreak (April 6, 2017 for Cerber and February 26, 2018 for Locky) and the machines reporting infections on this day. We assume Symantec obtains a locally private malware report from these machines including the vendor, specs, and OS version. Upon collecting reports from all machines, Symantec infers how frequently each OS was infected in the population, and ranks OSs in terms of infection frequency.

We conduct two experiments. In the first experiment, we find the actual (non-private) infection frequency of each OS, and compare actual frequencies with the frequencies that would be obtained if RAPPOR, OLH, or Item-CLDP were applied, using L1 distance as measurement of error. We vary ε between $0.5 \leq \varepsilon \leq 4$. In the second experiment, we fix $\varepsilon = 1$, rank the OSs in terms of infection frequency (highest to lowest), and study the top-10 most infected OSs. Due to ethical considerations, we anonymize OS names by renaming according to their actual rank, e.g., top-ranked OS is named os1, 2nd ranked OS is named os2, and so forth. If a lower-ranked OS is a different version of a higher ranked OS, we add the version information to the name, e.g., os1 v2.

Results: We report the results of these experiments on Cerber and Locky in Figures 3.6 and 3.7, respectively. In the tables on the right, those OSs that are correctly discovered by RAPPOR, OLH, and Item-CLDP with correct ranks are depicted in bold. OSs that are correctly discovered but have incorrect rank are depicted in regular font. OSs that privacy methods claim to be among top-10 but in reality are not are depicted with strike-through. We first observe from the tables that the heaviest hitters are correctly discovered in the correct order by all privacy solutions, e.g., top-3 in Cerber. However, as we move lower in the ranking, LDP/CLDP methods start making errors. Particularly for Cerber, RAPPOR and OLH correctly identify only 4 and 5 out of 10 most frequent OSs, respectively, whereas



	Actual	RAP-POR	OLH	Item-CLDP
1	**os1**	**os1**	**os1**	**os1**
2	**os2**	**os2**	**os2**	**os2**
3	**os2 v2**	**os2 v2**	**os2 v2**	**os2 v2**
4	**os3**	~~os2 v5~~	~~os2 v6~~	**os3**
5	**os3 v2**	~~os2 v6~~	os1 v2	os4
6	**os4**	os3	~~os8~~	os5
7	**os5**	~~os2 v7~~	~~os6 v3~~	os3 v2
8	**os2 v3**	~~os6~~	~~os2 v8~~	**os2 v3**
9	**os2 v4**	~~os7~~	~~os2 v5~~	**os2 v4**
10	**os1 v2**	~~os6 v2~~	os3	~~os8~~

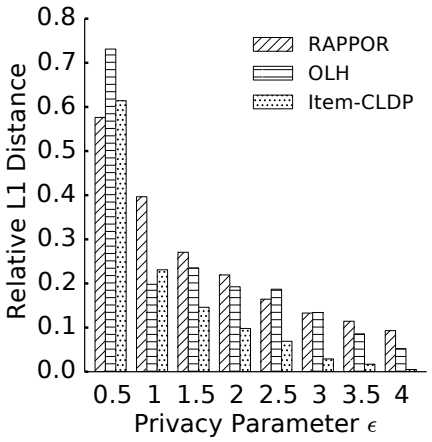
Figure 3.6: Analyzing OS vulnerability for ransomware Cerber. *Left*: L1 distances between actual OS counts versus locally private counts reported by RAPPOR, OLH, and our Item-CLDP protocol. *Right*: Actual and locally private top-10 OS rankings when $\varepsilon = 1$. **Bold** = OS with correct rank, regular font = OS with wrong rank, ~~strike-through~~ = OS not in actual top-10. Item-CLDP better preserves relative rankings and generally has lower L1 errors.

Item-CLDP can identify 9 out of 10. Note that Item-CLDP is missing only the lowest ranked OS (10th), which is arguably the least significant among all ten.

L1 errors in the graphs on the left show that when ε is small (0.5 or 1), LDP is competitive against CLDP in this case study. When ε is higher, CLDP clearly dominates in terms of accuracy. Comparing tabular rankings with L1 scores, we see that CLDP can preserve relative rankings even when its L1 errors are similar to those of LDP. For example, the L1 errors of RAPPOR, OLH, and Item-CLDP are similar when $\varepsilon = 1$. However, studying the top-10 tables shows that Item-CLDP is better at identifying frequent OSs than RAPPOR and OLH.

Case Study 3: Inspecting Suspicious Activity

Setup: In this case study, we consider the sequences of security-related event flags raised by Symantec's behavioral detection engine on each client machine. There are 143 different flags signalling various forms of suspicious activity, ranging from process injection to load point modification. When a flag is raised, it is logged on the client machine with a timestamp, as such the collection of the flagged events constitute a *sequence* over a time



	Actual	RAP-POR	OLH	Item-CLDP
1	**os1**	**os1**	**os1**	**os1**
2	**os2**	**os2**	**os2**	**os2**
3	**os1 v2**	~~os5 v2~~	os3	**os1 v2**
4	**os1 v3**	os1 v5	os4	~~os5 v3~~
5	**os3**	os1 v3	os2 v2	**os3**
6	**os1 v4**	os1 v2	**os1 v4**	os5
7	**os4**	os3	**os4**	~~os6~~
8	**os1 v5**	os5	~~os8~~	**os1 v5**
9	**os5**	~~os6~~	os1 v2	~~os7~~
10	**os2 v2**	~~os7~~	~~os6~~	~~os5 v4~~

Figure 3.7: Analyzing OS vulnerability for ransomware Locky. Refer to Figure 3.6 for a similar experiment with a different ransomware, same details apply to this figure.

period. We investigate the accuracy of collecting these event sequences using Sequence-CLDP. We focus on the same 31-day periods we considered in Case Study #1, and collect locally private event sequences from the same set of machines infected by ransomware Cerber/Locky. Longitudinal analysis of these event sequences enables Symantec to inspect suspicious activities possibly related to the ransomware infection, e.g., chain of anomalous events leading to the infection. This helps in inferring the precursors or consequences of the infection, and Symnatec can update its detection engine based on the findings. In total, we have 23,558 and 5,717 sequences for Cerber and Locky, respectively, with lengths between 2 to 30.

We use n-gram analysis by mining the top-k popular bigram and trigram patterns from the sequences [106, 88]. We mine the actual patterns that would be obtained if no privacy were applied, and the patterns obtained after Sequence-CLDP is applied. Let A denote the set of actual top-k patterns and B denote the set of top-k patterns mined from perturbed data. We measure their similarity using the Jaccard index: $\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$. Jaccard similarity is between 0 and 1, with values close to 1 indicating higher similarity. We do not compare against RAPPOR, OLH or SVIM in this case study, since they are not compatible with sequence perturbation.

Results: The results are shown in Figure 3.8. We make two important observations.

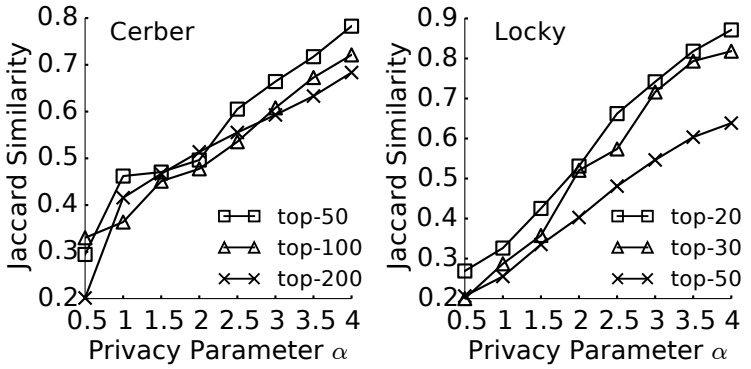


Figure 3.8: Utility preservation of Sequence-CLDP in mining top-k n-gram patterns from security event sequences obtained from ransomware infected machines. Sequence-CLDP allows discovery of a high fraction of frequent patterns, which are useful to analyze suspicious activity on the machines.

First, as we relax the privacy requirement by increasing α, n-grams mined from perturbed sequences become more accurate, as implied by the increase in Jaccard similarity. Second, mining fewer top-k patterns is easier than mining many patterns in general. For example, top-20 in Locky has higher Jaccard similarity score than top-30 and top-50. Similar observation applies to Cerber. This shows Sequence-CLDP preserves the heaviest hitters best, and has higher probability of making errors as n-grams become less and less frequent, which agree with our intuition from Case Study #2. Note that we mine more patterns in the case of Cerber (up to top-200 as opposed to top-50 for Locky) since the Cerber dataset has more input sequences, thus we can find more n-grams with significant support and confidence.

3.5.2 Experiments with Public Datasets

Datasets: We also experimented on two public datasets: POS and Retail. Both are set-valued datasets. We use them to run singleton non-ordinal item experiments as well as set-valued experiments. For the former, we randomly sample an item from each itemset to create a singleton item dataset. For the latter, we run the set-valued adaptation of our Sequence-CLDP protocol and compare it against SVIM, the state-of-the-art set-valued LDP protocol [105].

POS contains several years of market basket sale data from a large electronics retailer [107]. It consists of a total of 515,596 transactions with 1,657 unique items sold.

Retail contains transactions occurring between January 2010 and September 2011 for a UK-based online retail site [108]. After cleaning empty entries, this dataset consists of a total of 540,455 transactions with 2,603 unique items.

Evaluation Metrics: We use the following metrics to evaluate the accuracy of the privacy protocols. Let x denote an item, true(x) denote its true frequency, and est(x) denote its frequency estimated by the privacy protocol. Let $\mathbf{X}_{gt} = \{x_1, x_2, ..., x_k\}$ be the ground truth top-k items where x_j is the j'th most frequent item.

Average Relative Error (AvRE) measures the mean relative error in top-k items' estimated frequencies versus their true frequencies. Formally:

$$\text{AvRE} = \frac{\sum_{x \in \mathbf{X}_{gt}} \frac{abs(\text{est}(x) - \text{true}(x))}{\text{true}(x)}}{k}$$

The *Kendall-tau coefficient* (KT) measures how well the rankings of heavy hitter top-k items are preserved. A pair of items x, $y \in \mathbf{X}_{gt}$ are said to be concordant if their sorted popularity ranks agree, i.e., either of the following hold:

$$\text{true}(x) > \text{true}(y) \wedge \text{est}(x) > \text{est}(y)$$

$$\text{true}(x) < \text{true}(y) \wedge \text{est}(x) < \text{est}(y)$$

They are said to be discordant if neither holds. Then, the Kendall-tau coefficient of correlation can be defined as:

$$\text{KT} = \frac{(\text{\# of concordant pairs}) - (\text{\# of discordant pairs})}{k(k-1)/2}$$

Results of Singleton Experiments: We run experiments in the singleton setting and compare Item-CLDP with LDP protocols RAPPOR and OLH. Each experiment is repeated

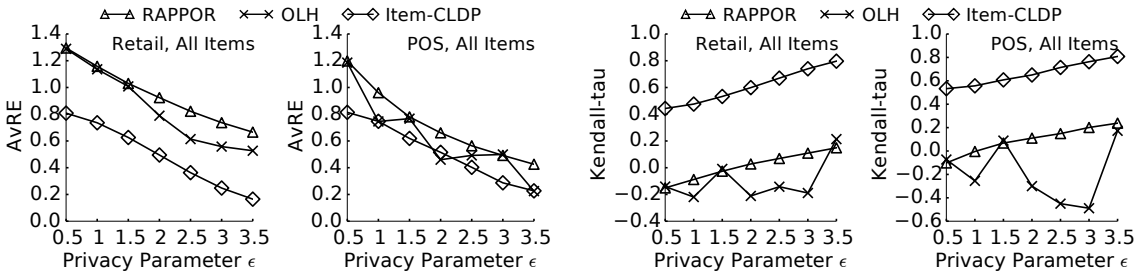


Figure 3.9: AvRE (lower is better) and Kendall-tau scores (higher is better) for singleton experiments across all items in datasets Retail and POS. Item-CLDP provides high utility in estimating item frequencies and rankings across a spectrum of privacy parameter settings.

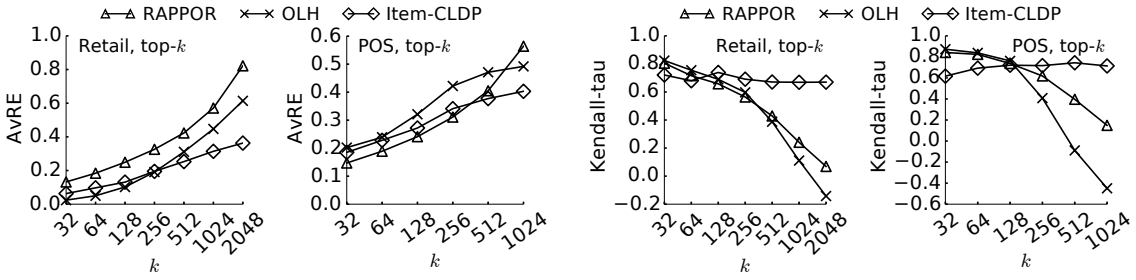


Figure 3.10: AvRE and Kendall-tau scores for top-k singletons in Retail and POS. ε fixed to 2.5, varying k on x-axis. Item-CLDP's accuracy is higher than LDP protocols especially for medium-frequency and infrequent items, e.g., $k \geq 256$.

20 times and results are averaged. In Figure 3.9, we measure AvRE and Kendall-tau across *all* items by setting $k = |\mathcal{U}|$. Results show that as privacy is relaxed (i.e., ε and α increase) AvRE decreases and Kendall-tau increases. In most cases, Item-CLDP provides better accuracy; most noticeably, Kendall-tau scores of Item-CLDP are much higher than those of RAPPOR and OLH. When $\varepsilon \geq 3.5$, Kendall-tau scores indicate almost perfect correlation between actual item rankings and rankings found by Item-CLDP, confirming its high accuracy.

Next, we fix the privacy parameter to $\varepsilon = 2.5$ and vary the k (for top-k) to analyze how the protocols behave with respect to varying popularities of items. The results of this experiment are reported in Figure 3.10. For small k such as $k \leq 64$, there is usually one LDP protocol at least comparable to or better than Item-CLDP. Note that $k = 64$ is a constrained setting covering less than only 6% of the items in the universe. LDP protocols

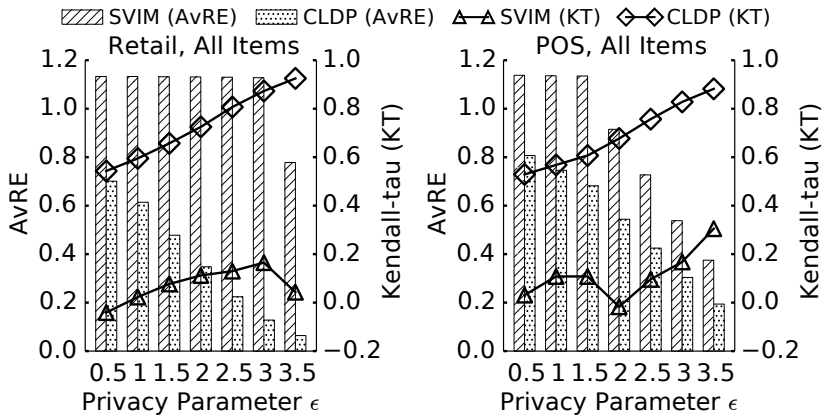


Figure 3.11: AvRE (bars, left y-axis) and Kendall-tau scores (lines, right y-axis) for set-valued experiments across all items. Sequence-CLDP preserves item frequencies and rankings more effectively than SVIM.

are optimized to discover such heavy hitters and therefore, they deliver good results when k is small. However, for larger k, we observe that Item-CLDP can significantly outperform RAPPOR and OLH. In particular, for $k \geq 512$, under LDP protocols there is almost no correlation in frequency rankings (implied by Kendall-tau results near or below 0), whereas in Item-CLDP, a strong correlation is maintained across all k. In short, if the goal is to discover only the top few heavy hitters, LDP protocols offer sufficient accuracy. However, if the goal is to find statistics regarding medium-frequency or infrequent items as well, LDP protocols have inadequate accuracy and we recommend using Item-CLDP.

Results of Set-Valued Experiments: We compare the set-valued adaptation of Sequence-CLDP against the SVIM protocol satisfying LDP [105]. Similar to the singleton experiment, we start by setting $k = |\mathcal{U}|$ and vary ε to study the impact of the privacy budget on accuracy across all items. From the results in Figure 3.11, we observe that Sequence-CLDP offers significant accuracy improvement in terms of both AvRE and Kendall-tau score. For example, the accuracy improvement in terms of AvRE ranges between 30-90% depending on the dataset and the value of the privacy parameter. In Figure 3.12, we fix ε to 2.5 and vary k. As we increase k, the accuracy of the protocols generally decrease, since making estimations regarding infrequent items is often more difficult than estimating only

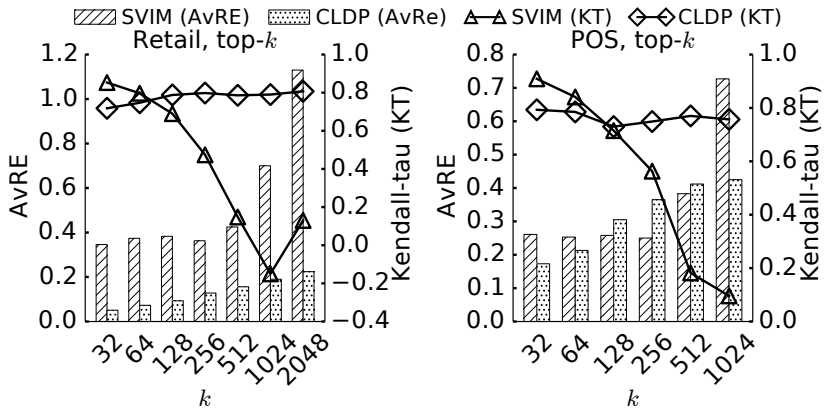


Figure 3.12: AvRE (bars, left y-axis) and Kendall-tau scores (lines, right y-axis) for set-valued experiments over top-k items with $\varepsilon = 2.5$. Sequence-CLDP offers higher accuracy in terms of AvRE and Kendall-tau for majority of k values.

the heavy hitters. For Sequence-CLDP, this accuracy decrease is linear or sub-linear; but for SVIM, when $k > 512$, errors start increasing almost exponentially, reiterating that LDP protocols can be poorly suited to estimate statistics regarding infrequent items. Studying the Kendall-tau scores, for very small k, Kendall-tau of SVIM and Sequence-CLDP are similar, whereas when $k > 128$, Sequence-CLDP's Kendall-tau scores are significantly better.

3.6 Related Work

Differential privacy was initially proposed in the centralized setting in which a trusted central data collector possesses a database containing clients' true values, and noise is applied on the database or queries executed on the database instead of each client's individual value [33, 94]. In contrast, in LDP, each client locally perturbs their data on their device before sending the perturbed version to the data collector [83]. The local setting has seen practical real-world deployment, including Google's RAPPOR as a Chrome extension [14, 85], Apple's use of LDP for spelling prediction and emoji frequency detection [86, 16], and Microsoft's collection of application telemetry [15].

LDP has also sparked interest from the academic community. There have been several

theoretical treatments for finding upper and lower bounds on the accuracy and utility of LDP [83, 95, 109, 110, 111]. From a more practical perspective, Wang et al. [84] showed the optimality of OLH for singleton item frequency estimation. Qin et al. [97] and Wang et al. [105] studied frequent item and itemset mining from set-valued client data. Cormode et al. [112] and Zhang et al. [113] studied the problem of obtaining marginal tables from high-dimensional data. Recently, LDP was considered in the contexts of geolocations [114], decentralized social graphs [115], discovering emerging terms from text [116, 117], and key-value stores [118].

However, there have also been criticisms and concerns regarding the utility of LDP, which motivated recent works proposing relaxations or alternatives to LDP. BLENDER [119] proposed a hybrid privacy model in which only a subset of users enjoy LDP, whereas remaining users act as opt-in beta testers who receive the guarantees of centralized DP. In contrast, our work stays purely in the local privacy model without requiring a trusted data collector (necessary in centralized or hybrid DP) or opt-in clients. Personalized LDP, a weaker form of LDP, was proposed for spatial data aggregation in [114]. Utility-Optimized LDP (ULDP) was proposed in [120], which treats certain client data as more sensitive than others and suggests adaptations of existing perturbation schemes (such as utility-optimized RAPPOR and utility-optimized GRR) that specifically address the more sensitive data. In contrast, our CLDP approach treats all users' data as sensitive (parallel with LDP assumptions) and remains agnostic and extensible with respect to data types.

3.7 Conclusion

In this chapter, we proposed *Condensed Local Differential Privacy (CLDP)* for utility-aware and privacy-preserving data collection, and developed three protocols: Ordinal-CLDP, Item-CLDP, and Sequence-CLDP for handling a variety of data types prevalent in the cybersecurity domain. Our protocols remain accurate for populations that are an order of magnitude smaller than those required by existing LDP protocols to give adequate accu-

racy. Case studies using Symantec datasets and experiments on public datasets demonstrate the utility of our proposed approach.

CHAPTER 4

ANALYZING LOCAL DIFFERENTIAL PRIVACY PROTOCOLS THROUGH A BAYESIAN ADVERSARY LENS

Local Differential Privacy (LDP) is popularly used as a practical standard for privacy-preserving data collection. Numerous LDP protocols have been proposed in the literature which differ in how they provide higher utility in different settings. Yet, few have engaged in analyzing and understanding the privacy relationships of the different protocols with respect to different utility and security settings. In this chapter, we present a principled framework for analyzing the privacy and utility properties of representative LDP protocols with three original contributions. First, we introduce a common Bayesian adversary model and a suite of metrics to formally analyze the privacy relationships of different LDP protocols. We show that different LDP protocols have substantially different responses to the attack effectiveness of a Bayesian adversary, measured in terms of Adversarial Success Rate (ASR). Second, we provide a formal and empirical analysis on a set of privacy and utility-critical factors such as encoding parameters, privacy budget ε, data domain, adversarial background knowledge, and statistical data distribution. We show that different settings of these factors can have significant effects on the privacy and utility properties of LDP protocols, and that no single protocol provides consistently high robustness (low ASR) against Bayesian adversaries. Third, we develop LDPLens, a prototype implementation of our proposed framework, which can recommend an effective protocol when given a data collection scenario, utility requirement and adversary metric; based on analyzing the privacy-utility tradeoffs of candidate LDP protocols. We evaluate the effectiveness of LDPLens using three case studies with real-world datasets. We show that: (i) LDPLens can effectively recommend a different protocol from one case study to another, and no single LDP protocol prevails over others across all three case studies. (ii) In each case study, the

protocol recommended by LDPLens offers up to 2-fold reduction in utility loss or ASR, compared to a random protocol.

4.1 Introduction

In recent years, Local Differential Privacy (LDP) has emerged as a popular notion for privacy-preserving data collection in client-server settings [121, 86, 15, 14, 97, 84]. In LDP, each client locally perturbs their sensitive data on their device before sending the perturbed output to the data collector. Since privacy is achieved on the client device via randomized perturbation, LDP enables privacy-preserving analytics in scenarios where clients do not trust the data collector. LDP has received significant attention from the research community as well as the industry, including Google's RAPPOR for analyzing Chrome browser settings [14], Apple's implementation in iOS to collect popular emojis and trending words for typing recommendation [122, 16, 86], and Microsoft's implementation in Windows 10 for collecting application telemetry [15].

The popularity of the LDP notion has led to the development of several LDP protocols. Common LDP protocols include GRR, BLH, OLH, RAPPOR, OUE and SS [95, 84, 14, 85, 123, 124]. New applications nowadays often use these LDP protocols as building blocks for building more complex systems with richer capabilities. For example, among recently developed applications, [125] uses GRR, RAPPOR and OUE protocols as building blocks, [105] uses GRR and OLH protocols as building blocks, [126] uses the OUE protocol, and [117] uses GRR and OLH protocols. Yet, few have engaged in analyzing the privacy relationships of these protocols across different factors. It is expected that different applications will have varying settings in terms of privacy budget ε, domain $\mathcal{U}$, data encoding parameters, statistical data distribution, and so forth. While existing research has recognized the utility impact of such factors on LDP protocols [84, 121, 105, 127, 126], their privacy impacts have been relatively less studied.

In this chapter, we present a principled framework for analyzing the privacy and utility

properties of LDP protocols and make three contributions. First, we introduce a Bayesian adversary model to formally analyze the privacy relationships of LDP protocols under varying settings. We show that different LDP protocols may have substantially different responses to the attack effectiveness of the Bayesian adversary, measured in terms of Adversarial Success Rate (ASR). Second, we provide a formal and empirical analysis of six popular LDP protocols (GRR, BLH, OLH, RAPPOR, OUE, SS) using a set of privacy and utility-critical factors including encoding parameters, privacy budget ε, data domain $\mathcal{U}$, adversarial background knowledge, and statistical data distribution. We mathematically derive expected ASR under each protocol using expected value analysis and we also perform experimental analysis using four datasets, including real-world URL page visit datasets. Our mathematical derivations and experimental results show that: (i) For protocols that have internal encoding and construction-related parameters, changing the values of these parameters has substantial impact on ASR (3-4 fold or more). (ii) ASR tends to differ from protocol to protocol even with the same ε being used, due to changes in data domain and encoding, protocol suitability, and statistical distribution. (iii) One protocol may yield higher ASR than another protocol under a certain set of factors, but the opposite may be true under a different set of factors. These findings show that no single protocol yields optimal utility and lowest ASR consistently across different combinations of factors.

Motivated by the above findings, the third contribution of this chapter is the design and development of LDPLens, a prototype implementation of our proposed framework. LDPLens leverages its privacy-utility tradeoff analysis to recommend an effective LDP protocol that is suitable under the given setting of factors. Concretely, LDPLens enables its user to specify relevant factors through customizable modules. Its adversary module enables specification of the adversary model and privacy metric (such as our proposed Bayesian adversary and ASR metric). Its utility measurement module enables the specification of a desired application-specific utility metric. Its data characteristics module enables the specification of factors such as data domain $\mathcal{U}$, encoding parameters, and statistical data distri-

bution. Combining these modules and considering the privacy/utility constraints specified by the user, LDPLens enables the user to explore the privacy-utility tradeoffs of each protocol under the chosen metrics, thereby assisting the selection of a suitable protocol under the user's specified setting. In addition to recommending suitable protocols, LDPLens can also help users to select a good ε value that fits their desired privacy/utility constraint.

We demonstrate LDPLens through real-world case studies. For demonstration, we use average L_1 norm error in item frequency estimation as our utility metric, and Bayesian adversary's ASR as our privacy metric. Upon executing LDPLens on the MSNBC dataset with a utility constraint of maximum L_1 error ≤ 0.002, we observe that the recommended protocol achieving lowest ASR (highest privacy) is OLH. The recommended ε here is $\varepsilon \approx 1$. In contrast, upon executing LDPLens on the Kosarak dataset with a privacy constraint of maximum ASR ≤ 0.25, we observe that the recommended protocol achieving lowest L_1 error (highest utility) is GRR. The recommended ε here is $\varepsilon \approx 2$. In both cases, using the protocol recommended by LDPLens rather than a randomly chosen protocol can yield up to 2-fold reduction in utility loss or ASR, which demonstrates the benefits of our LDPLens approach.

We conjecture that LDPLens can be useful for assisting non-expert application designers in selecting LDP protocols and ε privacy budget values suitable for their application-specific settings and privacy/utility constraints. LDPLens can also serve as a unifying platform for evaluating LDP protocols under varying adversary models, privacy metrics, and utility metrics.

The rest of this chapter is organized as follows. In Section 4.2, we review LDP background, briefly describe popular LDP protocols, and state our problem setting and adversary assumptions. In Section 4.3, we formally define our Bayesian adversary $\mathcal{A}$ and mathematically derive the expected ASR of $\mathcal{A}$ under each protocol. In Section 4.4, we perform experimental analysis with our adversary model. In Section 4.5, we describe the design and development of LDPLens. We summarize related work in Section 4.6 and conclude in

Section 4.7.

4.2 Background

4.2.1 Local Differential Privacy

Local Differential Privacy (LDP) is a popular notion for privacy-preserving data collection. It has been deployed in consumer-facing products of companies such as Apple, Google and Microsoft [15, 14, 122]. In a typical LDP scenario, there exist many clients (*users*) and a data aggregator (*server*). To protect privacy, each client perturbs their true value on their local device using a randomized algorithm Ψ, and sends the perturbed output to the aggregator.

Definition 5 (ε-LDP)**.** *A randomized algorithm Ψ satisfies ε-local differential privacy (ε-LDP), where $\varepsilon > 0$, if and only if for any two inputs v_1, v_2 in universe $\mathcal{U}$, it holds that:*

$$\forall y \in Range(\Psi): \quad \frac{Pr[\Psi(v_1) = y]}{Pr[\Psi(v_2) = y]} \leq e^{\varepsilon} \tag{4.1}$$

where $Range(\Psi)$ denotes the set of all possible outputs of Ψ.

ε-LDP ensures that given the perturbed value y, an adversary will not be able to distinguish whether the original value was v_1 or v_2 with probability odds-ratio higher than e^{ε}. Parameter ε is often called the privacy budget. Lower ε yields stronger privacy.

The popularity of the LDP notion has led to the development of several LDP protocols, which implement the LDP notion in different ways. Let $\mathcal{U}$ denote the universe (domain) of possible items, and let $\mathcal{L}$ denote the client population. For client $\ell \in \mathcal{L}$, we say that $v_\ell \in \mathcal{U}$ is ℓ's true value. v_ℓ is encoded and perturbed by the LDP protocol, and the perturbed output is sent to the server. Upon receiving perturbed outputs from many clients, the server performs estimation to recover statistics pertaining to the general client population, by applying a statistical estimator on the aggregate noisy data it has received. For a value $v \in \mathcal{U}$, let $C(v)$ the true count of v, i.e., total number of clients in $\mathcal{L}$ who truly possess

v. We denote by $\bar{C}(v)$ the estimated count of v, i.e., the count estimated by the server following the LDP protocol. Next, we give brief descriptions of popular and representative LDP protocols that are commonly used in the literature. Each protocol is characterized by its two main components: its client-side encoding and perturbation technique, and its server-side estimation technique.

4.2.2 LDP Protocols

We briefly describe six representative LDP protocols: GRR, BLH, OLH, RAPPOR, OUE and SS in terms of how they differ in (1) the mechanisms for encoding and injecting perturbation to client data on client devices to achieve privacy protection, and (2) the mechanisms for aggregating and estimating from perturbed data collected from population $\mathcal{L}$ to recover high-utility estimates.

Generalized Randomized Response (GRR) is a generalization of the randomized response survey technique introduced in [93] to support non-binary $\mathcal{U}$ and arbitrary ε. GRR uses direct encoding such that $\text{ENCODE}_{\text{GRR}}(v_\ell) = v_\ell$. The perturbation algorithm Ψ_{GRR} perturbs the encoded value and outputs $y_\ell \in \mathcal{U}$ with probability:

$$\Pr[\Psi_{\text{GRR}}(v_\ell) = y_\ell] = \begin{cases} p = \frac{e^\varepsilon}{e^\varepsilon + |\mathcal{U}| - 1} & \text{if } y_\ell = v_\ell \\ q = \frac{1}{e^\varepsilon + |\mathcal{U}| - 1} & \text{if } y_\ell \neq v_\ell \end{cases} \tag{4.2}$$

where $|\mathcal{U}|$ denotes the size of the universe. This satisfies ε-LDP since $\frac{p}{q} = e^\varepsilon$. The client sends y_ℓ to the server.

Upon receiving perturbed responses from all clients, the server wants to estimate the true number of clients who possess some value $v \in \mathcal{U}$. To do so, the server first counts $\widehat{C}(v)$: total number of clients who reported v as their perturbed output. Then, the estimate $\bar{C}(v)$ is computed as:

$$\bar{C}(v) = \frac{\widehat{C}(v) - |\mathcal{L}| \cdot q}{p - q} \tag{4.3}$$

Binary Local Hashing (BLH) is inspired by [95], which uses random matrix projection for building Succinct Histograms (SH). Instead of a random matrix projection that can be expensive to construct and perform matrix multiplication, [84] proposed that using a random hash function H from a universal hash function family $\mathcal{H}$ is logically equivalent. It was noted in [127] that Hadamard transform is also similar in essence to BLH. It can be used to speed up BLH in cases where evaluating a Hadamard entry is faster than evaluating hash functions.

Let $\mathcal{H}$ be a universal hash function family such that each hash function $H \in \mathcal{H}$ maps a value from $\mathcal{U}$ into one bit, i.e., $H : \mathcal{U} \to \{0, 1\}$. Each client ℓ first draws a hash function uniformly randomly from $\mathcal{H}$, i.e., $H_\ell \leftarrow_\$ \mathcal{H}$. Then, bit b is computed as $b_\ell = H_\ell(v_\ell)$. Encoding result is the tuple: $\text{ENCODE}_{\text{BLH}}(v_\ell) = \langle H_\ell, b_\ell \rangle$. Perturbation algorithm Ψ_{BLH} perturbs b_ℓ to b'_ℓ such that:

$$\Pr[b'_\ell = 1] = \begin{cases} \frac{e^\varepsilon}{e^\varepsilon + 1} & \text{if } b_\ell = 1 \\ \frac{1}{e^\varepsilon + 1} & \text{if } b_\ell = 0 \end{cases} \tag{4.4}$$

The client sends tuple $\langle H_\ell, b'_\ell \rangle$ to the server.

The server receives tuples of the form $\langle H_\ell, b'_\ell \rangle$ from all clients $\ell \in \mathcal{L}$. To perform estimation for value v, the server computes $Sup(v)$: total number of clients whose reported tuples satisfy the constraint: $b'_\ell = H_\ell(v)$. Then, the estimate $\bar{C}(v)$ is computed as:

$$\bar{C}(v) = \frac{(e^\varepsilon + 1) \cdot (2 \cdot Sup(v) - |\mathcal{L}|)}{e^\varepsilon - 1} \tag{4.5}$$

Optimized Local Hashing (OLH) differs from BLH in that the output space of the hash functions in family $\mathcal{H}$ is no longer binary. Instead, they are g-ary. That is, OLH allows the client to encode v_ℓ into an integer in range $[1, g]$ instead of a single bit, where $g \geq 2$ is an adjustable parameter of the protocol. The rationale is to address BLH's excessive utility loss in cases where binary encoding is suboptimal. OLH's approach was shown to yield

substantial utility improvement compared to BLH when ε and $\mathcal{U}$ are large [84]. The default value of g is $g = e^{\varepsilon} + 1$ as derived and used in [84, 105].

Let $\mathcal{H}$ be a universal hash function family where each $H \in \mathcal{H}$ maps a value from $\mathcal{U}$ into an integer in $[1, g]$, i.e., $H : \mathcal{U} \to [1, g]$. Each client ℓ randomly draws $H_\ell \leftarrow_{\$} \mathcal{H}$ and computes integer x_ℓ as: $x_\ell = H_\ell(v_\ell)$. Encoding result is the tuple: $\text{ENCODE}_{\text{OLH}}(v_\ell) = \langle H_\ell, x_\ell \rangle$. Perturbation algorithm Ψ_{OLH} perturbs x_ℓ to x'_ℓ such that:

$$\forall_{i \in [1,g]} : \quad \Pr[x'_\ell = i] = \begin{cases} \frac{e^{\varepsilon}}{e^{\varepsilon}+g-1} & \text{if } x_\ell = i \\ \frac{1}{e^{\varepsilon}+g-1} & \text{if } x_\ell \neq i \end{cases} \tag{4.6}$$

The client sends tuple $\langle H_\ell, x'_\ell \rangle$ to the server.

The server receives tuples of the form $\langle H_\ell, x'_\ell \rangle$ from all clients $\ell \in \mathcal{L}$. To perform estimation for value v, the server computes $Sup(v)$: total number of clients whose reported tuples satisfy the constraint: $x'_\ell = H_\ell(v)$. Then, the estimate $\bar{C}(v)$ is computed as:

$$\bar{C}(v) = \frac{(e^{\varepsilon} + g - 1) \cdot (g \cdot Sup(v) - |\mathcal{L}|)}{(e^{\varepsilon} - 1) \cdot (g - 1)} \tag{4.7}$$

RAPPOR was developed by Google and implemented in Chrome [14, 85]. While the original version of RAPPOR relies on Bloom filters for encoding string data, variants of RAPPOR were deployed in follow-up works for binary, unary and complex data depending on the application scenario [84, 18, 97, 115]. Here, without loss of generality, we give the version with unary encoding.

Client ℓ initializes a bitvector B_ℓ with length $|\mathcal{U}|$. The client sets $B_\ell[v_\ell] = 1$, and for all remaining positions $j \neq v_\ell$, $B_\ell[j] = 0$. Then, the perturbation step of RAPPOR takes as input B_ℓ and outputs a perturbed bitvector B'_ℓ. Perturbation algorithm Ψ_{RAP} considers each

bit in B_ℓ one-by-one, and either keeps or flips it with probability:

$$\forall_{i\in[1,|\mathcal{U}|]}: \quad \Pr[B'_\ell[i]=1] = \begin{cases} \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1} & \text{if } B_\ell[i]=1 \\ \frac{1}{e^{\varepsilon/2}+1} & \text{if } B_\ell[i]=0 \end{cases} \tag{4.8}$$

The client sends B'_ℓ to the server.

The server receives perturbed bitvectors B'_ℓ from all clients $\ell \in \mathcal{L}$. To perform estimation for value v, $Sup(v)$ is computed as the total number of received bitvectors that satisfy: $B'_\ell[v]=1$. Then, the estimate $\bar{C}(v)$ is computed as:

$$\bar{C}(v) = \frac{Sup(v) + |\mathcal{L}| \cdot (\alpha - 1)}{2\alpha - 1} \tag{4.9}$$

where α is the bit keeping probability: $\alpha = \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1}$.

Optimized Unary Encoding (OUE) is an optimized version of RAPPOR, such that its encoding phase is same as RAPPOR with unary encoding, but its bit keeping and flipping probabilities are different. As opposed to RAPPOR, it treats the 0 and 1 bits asymmetrically to improve accuracy of server-side estimation [84, 126].

Client ℓ initializes bitvector B_ℓ with length $|\mathcal{U}|$ such that $B_\ell[v_\ell] = 1$, and for all remaining positions $j \neq v_\ell$, $B_\ell[j] = 0$. Perturbation algorithm Ψ_{OUE} takes as input B_ℓ and produces perturbed bitvector B'_ℓ such that:

$$\forall_{i\in[1,|\mathcal{U}|]}: \quad \Pr[B'_\ell[i]=1] = \begin{cases} \frac{1}{2} & \text{if } B_\ell[i]=1 \\ \frac{1}{e^{\varepsilon}+1} & \text{if } B_\ell[i]=0 \end{cases} \tag{4.10}$$

The client sends B'_ℓ to the server.

The server receives perturbed bitvectors B'_ℓ from all clients $\ell \in \mathcal{L}$. To perform estimation for value v, $Sup(v)$ is computed as the total number of received bitvectors that satisfy:

$B'_\ell[v] = 1$. Then, the estimate $\bar{C}(v)$ is computed as:

$$\bar{C}(v) = \frac{2 \cdot \left((e^\varepsilon + 1) \cdot Sup(v) - |\mathcal{L}|\right)}{e^\varepsilon - 1} \tag{4.11}$$

Subset Selection (SS) operates by having each client ℓ report a randomly selected subset Z_ℓ of $\mathcal{U}$ to the server [123, 127]. Client's true value v_ℓ has higher probability of being included in their reported Z_ℓ, compared to other values in $\mathcal{U} \setminus \{v_\ell\}$ which are sampled uniformly randomly without replacement. The subset size $k = |Z_\ell|$ is a key parameter of the protocol. In [123, 127], it was noted that the default value of k is $k = \frac{|\mathcal{U}|}{e^\varepsilon + 1}$.

The execution of the SS protocol starts by initializing an empty subset Z_ℓ. Algorithm Ψ_{SS} adds v_ℓ to Z_ℓ with probability $\frac{k \cdot e^\varepsilon}{k \cdot e^\varepsilon + |\mathcal{U}| - k}$. It constructs the remainder of Z_ℓ as follows:

- If v_ℓ was added to Z_ℓ in the previous step, then $k-1$ items are sampled from $\mathcal{U} \setminus \{v_\ell\}$ uniformly randomly without replacement, and they are added to Z_ℓ.
- If v_ℓ was not added to Z_ℓ in the previous step, then k items are sampled from $\mathcal{U} \setminus \{v_\ell\}$ uniformly randomly without replacement, and they are added to Z_ℓ.

The client sends resulting Z_ℓ to the server.

The server receives randomized subsets Z_ℓ from all clients $\ell \in \mathcal{L}$. The server defines g_k and h_k as:

$$g_k = \frac{ke^\varepsilon}{ke^\varepsilon + |\mathcal{U}| - k} \qquad h_k = \frac{(k-1)(ke^\varepsilon) + (|\mathcal{U}| - k)k}{(|\mathcal{U}| - 1)(ke^\varepsilon + |\mathcal{U}| - k)}$$

To perform estimation for value v, $Sup(v)$ is computed as the total number of clients in $\mathcal{L}$ whose reported subset contains v. Then, the estimate $\bar{C}(v)$ is computed as:

$$\bar{C}(v) = \frac{Sup(v) - |\mathcal{L}| \cdot h_k}{g_k - h_k} \tag{4.12}$$

4.2.3 Problem Definition and Threat Model

As shown in the previous section, LDP protocols use different ways of encoding client data and applying a randomized perturbation algorithm Ψ, e.g., GRR uses direct encoding, BLH and OLH use hash encoding, RAPPOR and OUE use bitvector encoding, and SS uses subset construction. Studies have shown that protocols yield varying estimation utility and efficiency under scenarios with differing $\varepsilon, \mathcal{U}$, encoding parameters, and auxiliary knowledge (incl. consistency requirements) [84, 121, 18, 127, 126]. However, what remains unexplored is analyzing whether similar factors may impact the privacy and confidentiality of client data under a fixed, common adversary model. To that end, we introduce a Bayesian adversary model to analyze LDP protocols under the same adversary model while one or more of the factors are varied. Our adversary model borrows ideas from Bayesian techniques in orthogonal contexts such as membership privacy, location privacy, and Bayesian DP on correlated data [91, 34, 92, 26]; but it is modified to adapt to LDP's untrusted client-server data collection setting.

Adversary capability: In LDP, each client perturbs their true data locally on their device and sends the perturbed version to the server. Hence, the client's trust boundary is his/her own device. In accordance with this setting, we assume that the adversary's capability is to only observe the perturbed report sent from each client to the server (passive adversary). The adversary can be the server itself, a man-in-the-middle who observes the communication between the client and the server, or a third-party data analyst with whom the server shares the collected data.

Adversary background knowledge: We consider two adversary flavors: without Background Knowledge (w/o BK) and with Background Knowledge (w/ BK). Adversary w/o BK performs Bayesian inference using only the client's perturbed output, without assuming existence of any auxiliary knowledge. However, an adversary may often have prior knowledge regarding the statistical distribution of client data. For example, node degrees in social

networks and video viewings on Youtube follow power-law distributions [128, 129, 126], browser, homepage, and password-related statistics are often published for research purposes [130, 131, 132], and the likes of geolocation density and city traffic statistics are either available or can be crawled on the Web [8, 26, 23]. The adversary may use such information sources to form his/her BK. Consequently, we assume that the adversary w/ BK has prior knowledge of the statistical data distribution, but does not know the individual true value v_ℓ of any ℓ. It was shown recently that such prior statistical knowledge can be used improve server-side estimation accuracy in LDP [127, 126]. We use it in the context of adversarial power and privacy measurement.

Adversary goal: After observing the perturbed output sent from the client to the server, the adversary's goal is to predict the true value v_ℓ of client ℓ. The adversary can be executed for all clients $\ell \in \mathcal{L}$. Higher prediction success for adversary implies lower privacy for the end clients.

4.3 Bayesian Adversary and Analysis

In this section, we formally describe the adversary model and analyze the six protocols (GRR, BLH, OLH, RAPPOR, OUE, SS) under this adversary. In Section 4.3.1, we formalize the Bayesian adversary, its prediction strategy, and success measurement metric. In Sections 4.3.2 to 4.3.7, we show how the adversary is applied to each protocol and mathematically derive the adversary's expected success rate under each protocol one-by-one. In Section 4.3.8, we summarize and demonstrate the findings of our formal analysis.

4.3.1 Adversary Model $\mathcal{A}$

Let $\mathcal{A}$ denote the adversary. We choose a Bayesian $\mathcal{A}$ to quantify the combined privacy impacts of LDP protocol encoding and perturbation, due to the popularity of Bayesian methods in different contexts such as measurement of membership privacy in centralized databases, location privacy, and Bayesian DP on correlated data [91, 34, 92, 26]. For client

ℓ, let $\mathcal{O}_\ell$ denote the $\mathcal{A}$'s observations of LDP protocol outputs sent from ℓ to the server, e.g., in case of RAPPOR, $\mathcal{O}_\ell = \{B'_\ell\}$; in case of OLH, $\mathcal{O}_\ell = \{\langle H_\ell, x'_\ell \rangle\}$. The goal of $\mathcal{A}$ is to correctly predict v_ℓ given $\mathcal{O}_\ell$. Denoting by v_ℓ^p the adversary's prediction, the optimal Bayesian prediction strategy is:

$$v_\ell^p = \underset{\hat{v} \in \mathcal{U}}{\operatorname{argmax}} \Pr[\hat{v} | \mathcal{O}_\ell] \tag{4.13}$$

$$= \underset{\hat{v} \in \mathcal{U}}{\operatorname{argmax}} \frac{\Pr[\mathcal{O}_\ell | \hat{v}] \cdot \Pr[\hat{v}]}{\Pr[\mathcal{O}_\ell]} \tag{4.14}$$

$$= \underset{\hat{v} \in \mathcal{U}}{\operatorname{argmax}} \Pr[\mathcal{O}_\ell | \hat{v}] \cdot \Pr[\hat{v}] \tag{4.15}$$

The first step follows from Bayes' theorem and the second step is because $\Pr[\mathcal{O}_\ell]$ is constant as the adversary maximizes over variable $\hat{v}$. We say that $\mathcal{A}$ made a correct prediction if and only if $v_\ell = v_\ell^p$ and a false prediction otherwise. For the adversary w/ BK, we insert the background knowledge into Equation 4.15 via $\Pr[\hat{v}]$, i.e., by specifying an informed prior in Bayesian inference.

We use the Adversarial Success Rate (ASR) metric for measuring $\mathcal{A}$'s prediction success. ASR can be defined as the probability that the adversary's prediction is correct, i.e., $\Pr[v_\ell = v_\ell^p]$. It can be measured both empirically and analytically. In a client population $\mathcal{L}$, ASR is empirically measured as the ratio of clients whose true value is correctly predicted by $\mathcal{A}$:

$$ASR = \Pr[v_\ell = v_\ell^p] = \frac{\text{\# of clients } \ell \in \mathcal{L} \text{ such that } v_\ell = v_\ell^p}{|\mathcal{L}|} \tag{4.16}$$

In addition to empirical measurement, it is also possible to derive the *expected* ASR of $\mathcal{A}$ formally through probabilistic expected value analysis:

$$\mathbb{E}[ASR] = \mathbb{E}[\Pr[v_\ell = v_\ell^p]] \tag{4.17}$$

In the remainder of this section, we perform expected ASR analysis using Equation 4.17, by deriving expected ASR under each protocol one-by-one. We perform empirical ASR

analysis in Section 4.4.

4.3.2 Applying $\mathcal{A}$ to GRR

The analysis of GRR is most straightforward among all protocols. Observe from Ψ_{GRR} in Equation 4.2 that $\Pr[y_\ell = v_\ell] > \Pr[y_\ell = v']$ for all $v' \in \mathcal{U} \setminus \{v_\ell\}$. Thus, the Bayes-optimal prediction strategy for $\mathcal{A}$ is to predict v_ℓ^p as: $v_\ell^p = y_\ell$. Then:

$$\mathbb{E}[ASR] = \mathbb{E}[\Pr[v_\ell = v_\ell^p]] = \Pr[v_\ell = y_\ell] = \frac{e^\varepsilon}{e^\varepsilon + |\mathcal{U}| - 1} \tag{4.18}$$

Since GRR uses direct encoding, expected ASR under GRR need not be conditioned on the expected behavior of a hash function (as in BLH/OLH) or item sampling (as in SS). Hence, its expected ASR behavior is closest to its empirical performance.

4.3.3 Applying $\mathcal{A}$ to BLH

BLH uses client-side hashing, hence our analysis must take into account the expected behavior of the hash function H. Since H is picked from a universal family, its average behavior is to hash half of $\mathcal{U}$ into 0 bit whereas the remaining half of $\mathcal{U}$ is hashed to 1 bit. Observe from Ψ_{BLH} in Equation 4.4 that $\Pr[b'_\ell = b_\ell] > \Pr[b'_\ell = \neg b_\ell]$. Thus, the Bayes-optimal prediction strategy for $\mathcal{A}$ is to predict v_ℓ^p by random choice from subset of items which hash to b'_ℓ given H_ℓ. That is: $v_\ell^p \leftarrow_{\$} \mathcal{U}_{H_\ell, b'_\ell}$ where $\mathcal{U}_{H_\ell, b'_\ell}$ is the subset of $\mathcal{U}$ satisfying the condition $\mathcal{U}_{H_\ell, b'_\ell} = \{v | v \in \mathcal{U}, H_\ell(v) = b'_\ell\}$. From this, we derive expected ASR under

BLH as:

$$\mathbb{E}[ASR] = \mathbb{E}[\Pr[v_\ell = v_\ell^p]] = 1 - \mathbb{E}[\Pr[v_\ell \neq v_\ell^p]] \tag{4.19}$$

$$= 1 - \frac{e^\varepsilon}{e^\varepsilon + 1} \cdot \frac{\mathbb{E}[|\mathcal{U}_{H_\ell, b'_\ell}|] - 1}{\mathbb{E}[|\mathcal{U}_{H_\ell, b'_\ell}|]} - \frac{1}{e^\varepsilon + 1} \tag{4.20}$$

$$= 1 - \frac{e^\varepsilon}{e^\varepsilon + 1} \cdot \frac{\frac{|\mathcal{U}|}{2} - 1}{\frac{|\mathcal{U}|}{2}} - \frac{1}{e^\varepsilon + 1} \tag{4.21}$$

$$= 1 - \frac{e^\varepsilon \cdot (|\mathcal{U}| - 2)}{(e^\varepsilon + 1) \cdot |\mathcal{U}|} - \frac{1}{e^\varepsilon + 1} = \frac{2e^\varepsilon}{(e^\varepsilon + 1) \cdot |\mathcal{U}|} \tag{4.22}$$

In BLH's expected ASR, $|\mathcal{U}|$ is a multiplicative factor in the denominator as opposed to GRR's expected ASR in which $|\mathcal{U}|$ is an additive factor. Consequently, $|\mathcal{U}|$ plays a large role in BLH's expected ASR often being lower than GRR's expected ASR.

4.3.4 Applying $\mathcal{A}$ to OLH

Recall that OLH uses g-ary hash encoding. Thus, in OLH, the average behavior of hash function H is to hash $|\mathcal{U}|/g$ values to each hash bucket $x \in [1, g]$. By Equation 4.6, we know that $\Pr[x'_\ell = x_\ell] > \Pr[x'_\ell = i]$ for all $i \neq x_\ell$ and $1 \leq i \leq g$. Consequently, the Bayes-optimal prediction strategy for $\mathcal{A}$ is to predict v_ℓ^p by random choice from subset of items which hash to x'_ℓ, i.e.: $v_\ell^p \leftarrow_\$ \mathcal{U}_{H_\ell, x'_\ell}$ where $\mathcal{U}_{H_\ell, x'_\ell}$ is the subset of $\mathcal{U}$ defined as: $\mathcal{U}_{H_\ell, x'_\ell} = \{v | v \in \mathcal{U}, H_\ell(v) = x'_\ell\}$. Expected ASR is derived as:

$$\mathbb{E}[ASR] = \mathbb{E}[\Pr[v_\ell = v_\ell^p]] = 1 - \mathbb{E}[\Pr[v_\ell \neq v_\ell^p]] \tag{4.23}$$

$$= 1 - \frac{e^\varepsilon}{e^\varepsilon + g - 1} \cdot \frac{\mathbb{E}[|\mathcal{U}_{H_\ell, x'_\ell}|] - 1}{\mathbb{E}[|\mathcal{U}_{H_\ell, x'_\ell}|]} - \frac{g - 1}{e^\varepsilon + g - 1} \tag{4.24}$$

To solve further, we must consider two cases.

Case 1: $|\mathcal{U}|/g \leq 1$: In this case, $\mathbb{E}[|\mathcal{U}_{H_\ell, x'_\ell}|] = 1$ since x'_ℓ is reported. Then, continuing from

Equation 4.24, $\mathbb{E}[ASR]$ becomes:

$$\mathbb{E}[ASR] = 1 - \frac{e^{\varepsilon}}{e^{\varepsilon}+g-1} \cdot \frac{1-1}{1} - \frac{g-1}{e^{\varepsilon}+g-1} = \frac{e^{\varepsilon}}{e^{\varepsilon}+g-1} \tag{4.25}$$

Case 2: $|\mathcal{U}|/g > 1$: In this case, $\mathbb{E}[|\mathcal{U}_{H_\ell, x'_\ell}|] = |\mathcal{U}|/g$. Then, continuing from Equation 4.24, $\mathbb{E}[ASR]$ becomes:

$$\mathbb{E}[ASR] = 1 - \frac{e^{\varepsilon}}{e^{\varepsilon}+g-1} \cdot \frac{\frac{|\mathcal{U}|}{g}-1}{\frac{|\mathcal{U}|}{g}} - \frac{g-1}{e^{\varepsilon}+g-1} \tag{4.26}$$

$$= 1 - \frac{e^{\varepsilon}(|\mathcal{U}|-g)}{|\mathcal{U}|(e^{\varepsilon}+g-1)} - \frac{g-1}{e^{\varepsilon}+g-1} \tag{4.27}$$

$$= \frac{e^{\varepsilon}g}{|\mathcal{U}|(e^{\varepsilon}+g-1)} \tag{4.28}$$

We finish the derivation by combining the final steps of Case 1 and Case 2. By doing so, we find that expected ASR under OLH overall is:

$$\mathbb{E}[ASR] = \frac{e^{\varepsilon}}{(e^{\varepsilon}+g-1) \cdot \max\left\{\frac{|\mathcal{U}|}{g}, 1\right\}} \tag{4.29}$$

Note that expected ASR under OLH found in Equation 4.29 is also dependent on the protocol encoding parameter g. If the default value of g, which is $g = e^{\varepsilon} + 1$ [84, 105], is plugged into Equation 4.29 we obtain:

$$\frac{1}{2} \cdot \frac{1}{\max\left\{\frac{|\mathcal{U}|}{e^{\varepsilon}+1}, 1\right\}} \tag{4.30}$$

From Equation 4.30, we observe that when ε is large enough that $e^{\varepsilon}+1 > |\mathcal{U}|$, then OLH's expected ASR will be upper bounded by $\frac{1}{2}$. For smaller ε, expected ASR under OLH will be smaller than $\frac{1}{2}$.

4.3.5 Applying $\mathcal{A}$ to RAPPOR

RAPPOR uses bitvector encoding, and client ℓ sends perturbed bitvector B'_ℓ to the server. We define a set of pairwise disjoint events E_0, E_1, ..., $E_{|\mathcal{U}|}$ as follows. Event E_0 is the case where the original 1 bit in the client's bitvector is flipped to 0 in B'_ℓ. For all remaining events E_i where $i \in [1, |\mathcal{U}|]$, event E_i is the case where the original 1 bit is kept and $i-1$ other bits that were 0 in the client's original bitvector became 1 in B'_ℓ due to bit flipping. Then:

$$\mathbb{E}[ASR] = \sum_{i=0}^{|\mathcal{U}|} \text{Pr}[v_\ell = v_\ell^p \wedge E_i] \tag{4.31}$$

Below, we calculate each entry in the right hand side summation in Equation 4.31 separately. Finally, we sum them up.

$\mathbb{E}[ASR]$ with Event E_0:

$$\text{Pr}[v_\ell = v_\ell^p \wedge E_0] = \text{Pr}[v_\ell = v_\ell^p | E_0] \cdot \text{Pr}[E_0] \tag{4.32}$$

$$= \frac{1}{|\mathcal{U}|} \cdot \Big(\frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1}\Big)^{|\mathcal{U}|-1} \cdot \frac{1}{e^{\varepsilon/2}+1} \tag{4.33}$$

The rationale is as follows. By definition of Ψ_{RAP}, we have: $\text{Pr}[E_0] = \frac{1}{e^{\varepsilon/2}+1}$. To compute $\text{Pr}[v_\ell = v_\ell^p | E_0]$, we observe that given the original bit was flipped to 0, if any other bit is flipped to 1, $\mathcal{A}$'s optimal prediction strategy is to pick among the indexes with 1 bit. Thus, the only way for $\mathcal{A}$ to predict $v_\ell^p = v_\ell$ is if no original 0 bits are flipped to 1, which happens with probability $\Big(\frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1}\Big)^{|\mathcal{U}|-1}$, and under that scenario, the correct prediction probability is $\frac{1}{|B|} = \frac{1}{|\mathcal{U}|}$. We combine them to arrive at Equation 4.33.

$\mathbb{E}[ASR]$ with Event E_i where $1 \leq i \leq |\mathcal{U}|$:

$$\text{Pr}[v_\ell = v_\ell^p \wedge E_i] = \text{Pr}[v_\ell = v_\ell^p | E_i] \cdot \text{Pr}[E_i] \tag{4.34}$$

$$= \frac{1}{i} \cdot \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1} \cdot \text{Bin}\Big(i-1; |\mathcal{U}|-1, \frac{1}{e^{\varepsilon/2}+1}\Big) \tag{4.35}$$

The rationale is as follows. To compute $\Pr[E_i]$, we observe that the original 1 bit is kept with probability $\frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1}$ by Ψ_{RAP}. Among the $|\mathcal{U}|-1$ bits that were initially 0, the probability of $i-1$ of them being flipped to 1 by Ψ_{RAP} can be modeled by a Binomial distribution with $|\mathcal{U}|-1$ trials, success probability $\frac{1}{e^{\varepsilon/2}+1}$, and exactly $i-1$ successes. This is reflected in the rightmost Binomial entry in Equation 4.35. To compute $\Pr[v_\ell = v_\ell^p | E_i]$, we observe that the Bayes-optimal prediction strategy for $\mathcal{A}$ is random guess among the indexes in B that contain the 1 bit. By definition of E_i, the original 1 bit is kept, and additionally $i-1$ bits that were originally 0 were flipped to 1; thus there are a total of i 1 bits. The success probability of $\mathcal{A}$'s guess is therefore $\frac{1}{i}$. Combining all, we arrive at Equation 4.35.

Total $\mathbb{E}[ASR]$ under RAPPOR: To arrive at RAPPOR's expected ASR, we plug Equation 4.33 and 4.35 into Equation 4.31 to sum up individual ASRs from all individual events. The final result is:

$$\begin{aligned}\mathbb{E}[ASR] =& \frac{1}{|\mathcal{U}|(e^{\varepsilon/2}+1)} \cdot \Big(\frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1}\Big)^{|\mathcal{U}|-1} \\ &+ \sum_{i=1}^{|\mathcal{U}|} \frac{e^{\varepsilon/2}}{(e^{\varepsilon/2}+1)i} \cdot \text{Bin}\Big(i-1; |\mathcal{U}|-1, \frac{1}{e^{\varepsilon/2}+1}\Big) \end{aligned} \tag{4.36}$$

4.3.6 Applying $\mathcal{A}$ to OUE

While OUE uses the same bitvector encoding as unary RAPPOR, its perturbation algorithm Ψ_{OUE} behaves differently. In particular, for any bit that is originally 1, Ψ_{OUE} keeps or flips the bit with equal probability $\frac{1}{2}$. In contrast, for any bit that is originally 0, it is kept with high probability $\frac{e^\varepsilon}{e^\varepsilon+1}$ and flipped with smaller probability $\frac{1}{e^\varepsilon+1}$. For this Ψ_{OUE}, we establish that for any index j, if $B'_\ell[j] = 0$ is observed by $\mathcal{A}$, then in the original bitvector B_ℓ, the probability that $B_\ell[j] = 0$ is higher than the probability that $B_\ell[j] = 1$. Consequently, the Bayes-optimal prediction strategy for $\mathcal{A}$ is picking among the indexes j in B'_ℓ that have: $B'_\ell[j] = 1$.

Similar to RAPPOR, we define a set of pairwise disjoint events E_0, E_1, ..., $E_{|\mathcal{U}|}$ as

follows. Event E_0 is the case where the original 1 bit in the client's bitvector is flipped to 0 in B'_ℓ. For all remaining events E_i where $i \in [1, |\mathcal{U}|]$, event E_i is the case where the original 1 bit is kept and $i-1$ other bits that were 0 in the client's original bitvector became 1 in B'_ℓ. Then:

$$\mathbb{E}[ASR] = \sum_{i=0}^{|\mathcal{U}|} \Pr[v_\ell = v_\ell^p \wedge E_i] \tag{4.37}$$

$\underline{\mathbb{E}[ASR] \text{ with Event } E_0}$:

$$\Pr[v_\ell = v_\ell^p \wedge E_0] = \Pr[v_\ell = v_\ell^p | E_0] \cdot \Pr[E_0] \tag{4.38}$$

$$= \frac{1}{|\mathcal{U}|} \cdot \frac{1}{2} \cdot \left(\frac{e^\varepsilon}{e^\varepsilon + 1}\right)^{|\mathcal{U}|-1} \tag{4.39}$$

The rationale is as follows. By definition of Ψ_{OUE}, we have: $\Pr[E_0] = \frac{1}{2}$. To compute $\Pr[v_\ell = v_\ell^p | E_0]$, we observe that given the original bit was flipped to 0, if any other bit is flipped to 1, $\mathcal{A}$'s optimal strategy is to pick among the indexes with 1 bit. Thus, the only way for $\mathcal{A}$ to predict $v_\ell^p = v_\ell$ is if no original 0 bits are flipped to 1, which happens with probability $\left(\frac{e^\varepsilon}{e^\varepsilon+1}\right)^{|\mathcal{U}|-1}$, and under that scenario, the probability that $\mathcal{A}$'s prediction is successful is $\frac{1}{|\mathcal{U}|}$.

$\underline{\mathbb{E}[ASR] \text{ with Event } E_i \text{ where } 1 \leq i \leq |\mathcal{U}|}$:

$$\Pr[v_\ell = v_\ell^p \wedge E_i] = \Pr[v_\ell = v_\ell^p | E_i] \cdot \Pr[E_i] \tag{4.40}$$

$$= \frac{1}{i} \cdot \frac{1}{2} \cdot \text{Bin}\Big(i-1; |\mathcal{U}|-1, \frac{1}{e^\varepsilon + 1}\Big) \tag{4.41}$$

The rationale is as follows. To compute $\Pr[E_i]$, we observe that the original 1 bit is kept with probability $\frac{1}{2}$ by Ψ_{OUE}. Among the $|\mathcal{U}|-1$ bits that were initially 0, the probability of $i-1$ of them being flipped to 1 by Ψ_{OUE} can be modeled by a Binomial distribution with $|\mathcal{U}|-1$ trials, success probability $\frac{1}{e^\varepsilon+1}$, and exactly $i-1$ successes. Having established that $\mathcal{A}$'s Bayes-optimal prediction strategy is picking among the indexes that contain the 1 bit, to compute $\Pr[v_\ell = v_\ell^p | E_i]$, we observe that the original 1 bit is kept, and additionally $i-1$

bits that were originally 0 were flipped to 1; thus there are a total of i 1 bits. The success probability of $\mathcal{A}$'s guess is therefore $\frac{1}{i}$. Combining all, we arrive at Equation 4.41.

Total $\mathbb{E}[ASR]$ under OUE: To arrive at OUE's expected ASR, we plug Equation 4.39 and 4.41 into Equation 4.37 to sum up individual ASRs from all individual events. The final result is:

$$\mathbb{E}[ASR] = \frac{1}{2|\mathcal{U}|} \cdot \left(\frac{e^{\varepsilon}}{e^{\varepsilon}+1}\right)^{|\mathcal{U}|-1} + \sum_{i=1}^{|\mathcal{U}|} \frac{1}{2i} \cdot \text{Bin}\left(i-1; |\mathcal{U}|-1, \frac{1}{e^{\varepsilon}+1}\right) \tag{4.42}$$

While the analysis of OUE may seem similar to RAPPOR, the final result is semantically very different. A key difference between OUE's final result in Equation 4.42 and RAPPOR's final result in Equation 4.36 is that the latter contains the term $\frac{e^{\varepsilon/2}}{(e^{\varepsilon/2}+1)i}$ whereas the prior contains the term $\frac{1}{2i}$ in its place. Ignoring i in both terms, as ε increases and approaches $+\infty$, RAPPOR's term increases and approaches 1. However, OUE's term is always upper bounded by the constant $\frac{1}{2}$ regardless of how large ε is. We will observe in our analysis and empirical evaluation, that this is a key observation which helps explain why ASR under RAPPOR may increase and approach 1 as ε becomes larger, but ASR under OUE increases very slowly after a certain ε.

4.3.7 Applying $\mathcal{A}$ to SS

In SS, each client ℓ builds and sends a subset Z_ℓ with size $k = |Z_\ell|$ to the server. First, recall that Ψ_{SS} adds v_ℓ to Z_ℓ with probability $\frac{k \cdot e^{\varepsilon}}{k \cdot e^{\varepsilon} + |\mathcal{U}| - k}$. Second, observe that a fake value $v \in \mathcal{U} \setminus \{v_\ell\}$ is added to Z_ℓ by Ψ_{SS} with probability:

$$\frac{ke^{\varepsilon}}{ke^{\varepsilon}+|\mathcal{U}|-k} \cdot \frac{k-1}{|\mathcal{U}|-1} + \frac{|\mathcal{U}|-k}{ke^{\varepsilon}+|\mathcal{U}|-k} \cdot \frac{k}{|\mathcal{U}|-1} \tag{4.43}$$

For this to be larger than the probability that v_ℓ is added to Z_ℓ, the following must be true:

$$\frac{k^2e^{\varepsilon} - ke^{\varepsilon} + k|\mathcal{U}| - k^2}{(ke^{\varepsilon}+|\mathcal{U}|-k) \cdot (|\mathcal{U}|-1)} > \frac{ke^{\varepsilon}}{ke^{\varepsilon}+|\mathcal{U}|-k} \tag{4.44}$$

However, if we solve Equation 4.44, we arrive at: $k > |\mathcal{U}|$ which yields a contradiction because k is the size of a *subset* of $\mathcal{U}$, thus k cannot exceed $|\mathcal{U}|$. Therefore, we find that it is not possible for any fake value's probability of being added to Z_ℓ to be higher than the probability that v_ℓ is added to Z_ℓ. Then, the prediction strategy of $\mathcal{A}$ has to be to predict v_ℓ^p from reported Z_ℓ, since the probability of it being included in Z_ℓ is highest. $\mathcal{A}$'s prediction strategy is therefore: $v_\ell^p \leftarrow_\$ Z_\ell$. Expected ASR under SS can then be derived as:

$$\mathbb{E}[ASR] = \Pr[v_\ell = v_\ell^p] = \Pr[v_\ell = v_\ell^p | v_\ell \in Z_\ell] \cdot \Pr[v_\ell \in Z_\ell] \tag{4.45}$$

$$= \frac{ke^\varepsilon}{ke^\varepsilon + |\mathcal{U}| - k} \cdot \frac{1}{k} = \frac{e^\varepsilon}{ke^\varepsilon + |\mathcal{U}| - k} \tag{4.46}$$

which completes the derivation. Note that expected ASR under SS found in Equation 4.46 is also dependent on the protocol subset size parameter k. If the default value of $k = \frac{|\mathcal{U}|}{e^\varepsilon + 1}$ is plugged in to Equation 4.46, we get: $\frac{e^\varepsilon + 1}{2|\mathcal{U}|}$ as the expected ASR. This confirms the intuitive expectations that: (i) when ε increases, ASR will increase since privacy becomes more relaxed; and (ii) when $|\mathcal{U}|$ increases, ASR will decrease since it is more difficult for $\mathcal{A}$ to predict v_ℓ correctly from a larger domain.

4.3.8 Summary and Analysis of Expected ASR

We have now completed deriving the expected ASR of $\mathcal{A}$ under each protocol. Expected ASR under GRR is given in Equation 4.18, BLH is given in Equation 4.22, OLH is given in Equation 4.29, RAPPOR is given in Equation 4.36, OUE is given in Equation 4.42, and SS is given in Equation 4.46. These capture how successful the Bayesian adversary's prediction is *expected* to be under each protocol.

We now perform a side-by-side comparison of each protocol's expected ASR to analyze their behavior with varying ε and $\mathcal{U}$. Note that ε and $\mathcal{U}$ are the two common factors that appear in the expected ASR equations of all protocols. In Figure 4.1, we plot the expected ASRs of each protocol: on the left, we fix $|\mathcal{U}| = 64$ and vary ε; on the right, we fix

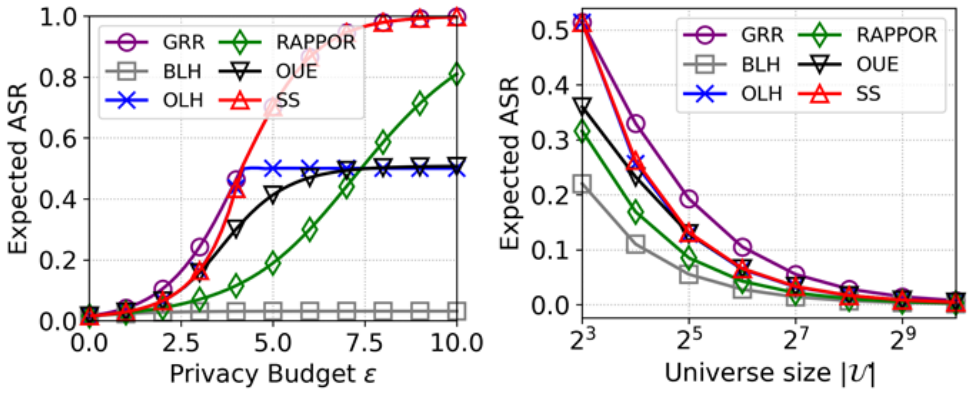


Figure 4.1: Expected ASR of $\mathcal{A}$ under all six protocols; across varying ε and $|\mathcal{U}|$.

$\varepsilon = 2$ and vary $|\mathcal{U}|$. The graphs show some general trends that are apply to all LDP protocols, which agree with the intuitive expectations of the LDP setting. In particular, with increasing ε, the privacy achieved by randomized perturbation decreases, therefore ASR values increase. Also, with a larger domain $\mathcal{U}$, we observe a decrease in ASR which is intuitive given that a larger domain makes $\mathcal{A}$'s predictions more difficult (e.g., the ASR of a completely random guess is $1/|\mathcal{U}|$, which becomes smaller as $|\mathcal{U}|$ becomes larger). The results across all six protocols agree on these general trends.

A key observation is that different LDP protocols may yield substantially different ASR values under the same ε and $\mathcal{U}$. For example, the expected ASR under GRR can be several times higher than BLH under the same ε and $\mathcal{U}$. This leads to our key finding that given a fixed adversary notion and adversary success measurement metric, the adversary's success can be different from one LDP protocol to another. As such, we conclude that there can be factors other than ε in the LDP setting that determine how much practical privacy the clients get. One factor is the encoding type and parameters that the protocols use, which vary from protocol to protocol. For example, GRR uses direct encoding whereas BLH uses binary hash encoding. BLH's mapping of v_ℓ to a binary hash output causes an additional privacy benefit for clients because hash functions are many-to-one and non-invertible. Such added privacy benefit is not applicable to a protocol such as GRR, which uses direct encoding. Consequently, we observe in Figure 4.1 that GRR's expected ASR values are consistently

high whereas BLH's expected ASR values are consistently low across varying ε and $\mathcal{U}$.

Another important observation is that one protocol may have higher ASR over another protocol under certain ε and $\mathcal{U}$ settings, whereas the other protocol may have higher ASR under different ε and $\mathcal{U}$ settings. For example, comparing OLH and RAPPOR in the first figure, OLH's ASR is higher than RAPPOR's ASR when $2.5 \leq \varepsilon \leq 7$, but RAPPOR's ASR exceeds OLH when $\varepsilon \geq 8$. Another example is that SS has ASR close to OUE for small ε, but when $\varepsilon \geq 5$, SS becomes very different than OUE and more similar to GRR. These observations are partly caused by OLH and OUE's ASR being heavily impacted by the $\frac{1}{2}$ terms in their expected ASR, as discussed at the end of Section 4.3.4 and 4.3.6 respectively. Altogether, these examples and observations show that no single protocol provides lowest ASR across all settings and factors. Thus, the protocol recommended to minimize ASR in one setting can be different than the recommended protocol in another setting.

4.4 Experimental Analysis

4.4.1 Setup and Datasets

We conduct experiments with the six LDP protocols under consideration (GRR, BLH, OLH, RAPPOR, OUE, SS) and four datasets: MSNBC, Kosarak, Uniform, and Exponential.

MSNBC contains logs from msnbc.com for the day of September 28, 1999 where each row in the dataset corresponds to one user's page visit sequence [1]. Visits are recorded at the granularity of page category (news, tech, weather, sports, etc.). There are $|\mathcal{U}| = 17$ categories and $|\mathcal{L}| = 989,818$ users. A large number of users either visit one category of pages, or for users that visit multiple categories, their visits are often dominated by one category. Hence, for each user we determine the most visited category and assume the user's true value is equal to that category.

[1] `http://archive.ics.uci.edu/ml/datasets/msnbc.com+anonymous+web+data`

Kosarak contains click stream data from a Hungarian online news portal [2]. Each user is associated with a set of clicked URL IDs. There are around one million users and 41,270 unique URL IDs. Since many URLs are visited few times (e.g., once or twice), we pre-processed the dataset by identifying the top-128 most visited URLs across the whole dataset and removing the remaining URLs from each user's click stream. We discarded those users whose resulting click stream was empty. For users who had more than one URL in their resulting stream, we picked one URL as their true value v_ℓ. In the resulting dataset, we had $|\mathcal{U}| = 128$ and $|\mathcal{L}| = 929,669$.

Uniform: We create synthetic client populations consisting of $|\mathcal{L}|$ clients whose true data, when aggregated, corresponds to a uniform distribution across domain $\mathcal{U}$. Unless otherwise noted, default values are $|\mathcal{L}| = 100,000$ and $|\mathcal{U}| = 40$.

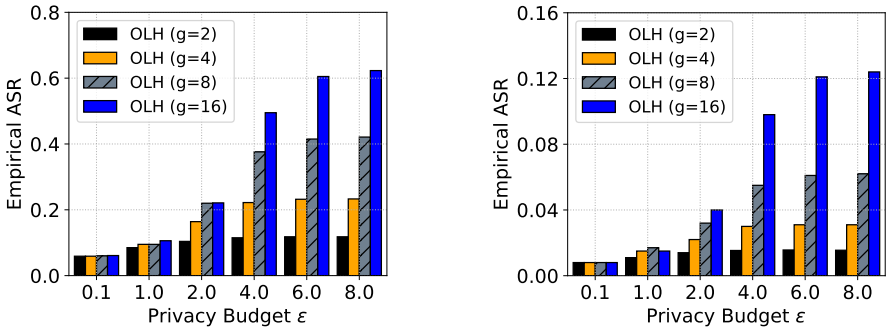
Exponential: We use an Exponential distribution, which has probability density function $f(x;\beta) = \beta^{-1}e^{-x/\beta}$ where β is the scale parameter of the distribution, to generate client data. Lower β means the distribution is more skewed, i.e., denser in the characteristic peak of the Exponential [3]. By default we use $\beta = 3$, but when measuring the impacts of data skewness, we create different client populations with varying data skewness by changing the β parameter and using an Exponential with different β for each population. Each client's true value is a data point sampled from the corresponding Exponential, with samples rounded to the nearest integer. By default, $|\mathcal{L}| = 100,000$ and $|\mathcal{U}| = 50$.

4.4.2 Impact of Protocol Encoding Parameters

Recall that some protocols have internal encoding-related parameters, e.g.: OLH has the g parameter which determines the output space of its hash encoding (OLH uses g-ary encoding) and SS has the k parameter which determines the reported subset size (in SS, clients report randomized subsets of size k to the server). Here, we study the impact of changing g and k parameters on empirical ASR under the OLH and SS protocols, respectively.

[2] `http://fimi.uantwerpen.be/data/`

[3] `https://en.wikipedia.org/wiki/Exponential_distribution`



(a) MSNBC dataset

(b) Kosarak dataset

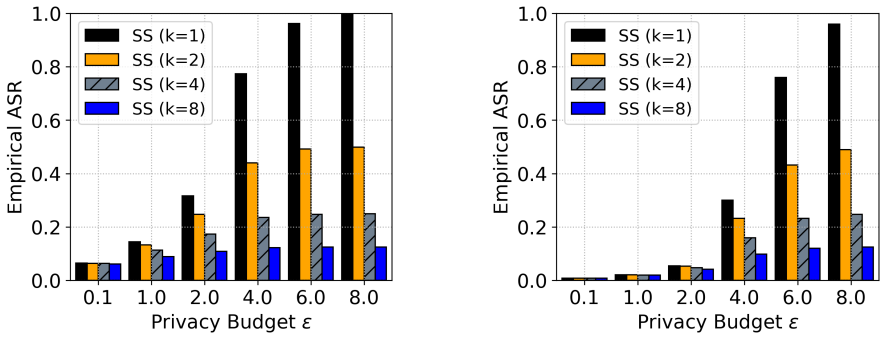
(c) Uniform dataset

(d) Exponential dataset (β=3)

Figure 4.2: Adversarial Success Rate (ASR) changes substantially as OLH protocol's encoding parameter g is varied.

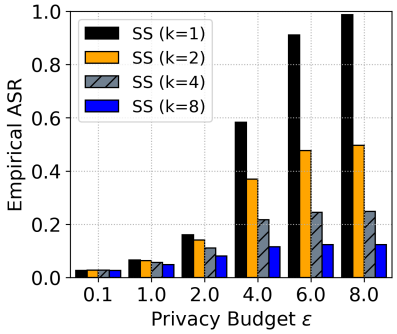
In Figure 4.2, we show how empirical ASR changes under the OLH protocol when its g parameter is varied between 2 and 16. While the differences between varying g values is small for small ε budgets, for larger ε budgets such as $\varepsilon \geq 2$, there starts to be large discrepancies between the ASR values. For $\varepsilon = 8$, the empirical ASR of OLH with $g = 16$ is almost 3-4 fold higher than ASR of OLH with $g = 2$. The results across all four datasets agree with these trends.

The rationale behind observing different ASR under the same ε but different g is as follows. In OLH, the expected number of hash collisions due to g-ary hash encoding is $|\mathcal{U}|/g$. When g is smaller, more collisions are expected. Increased number of collisions provide an added amount of privacy (i.e., added on top of the privacy already achieved by ε-LDP perturbation), thanks to the irreversibility of hash collisions. Even in the extreme case where $\varepsilon = +\infty$ in which no privacy is given by ε-LDP perturbation, hash collisions

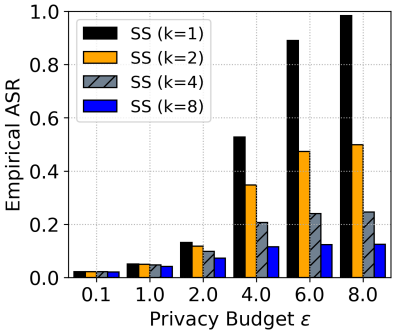


(a) MSNBC dataset

(b) Kosarak dataset



(c) Uniform dataset



(d) Exponential dataset (β=3)

Figure 4.3: Adversarial Success Rate (ASR) changes substantially as SS protocol's subset size parameter k is varied.

will still cause $\mathcal{A}$'s prediction accuracy and ASR to go down, yielding *some* privacy to clients. This supports the observation that especially for large ε such as 8, ASR remains low when g is small (many collisions) compared to $g = 16$ (fewer collisions).

In Figure 4.3, we perform a similar experiment for the SS protocol by varying its k parameter between 1 and 8. The results for SS protocol show that varying k values have less observable impact on ASR when ε is small, but may cause 4-5 fold difference in ASR when ε is large. In particular, for $\varepsilon \geq 4$, the ASR of SS with $k = 1$ is several folds larger than the ASR of SS with $k = 8$. This can be intuitively explained through a crowd-blending privacy perspective as follows. In SS, v_ℓ blends in a crowd of k items (reported subset). When k is small, v_ℓ does not have a large crowd to blend in, making the adversary's prediction easier. In contrast, when k is large, the crowd is larger therefore ASR can remain low due to the existence of a large number of $k - 1$ fake reported items.

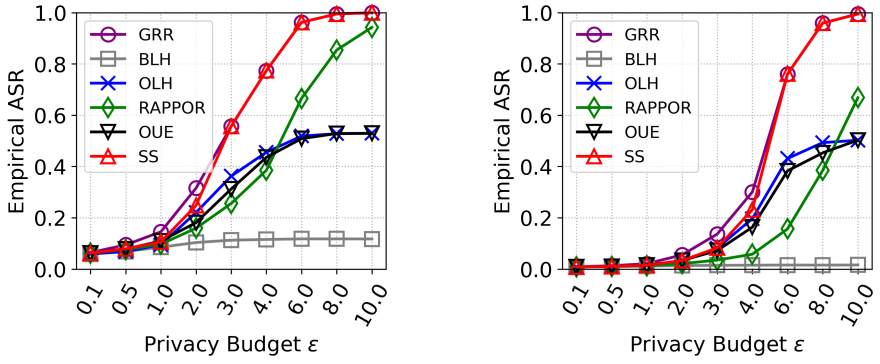
The take-away message from this set of experiments is that for both OLH and SS protocols, their internal parameters (g and k) have substantial impact on adversary's empirical ASR. Thus, when implementing LDP protocols in practice, one should pay close attention to how the individual protocol parameters (such as g and k) are chosen when reasoning about how much end privacy and adversary-resilience the clients will receive from that protocol.

4.4.3 Comparison Between Protocols

Here, we assume that each protocol uses their default parameters that are found in the respective papers and noted in Section 4.2.2. Under this setting, we compare all protocols with one another to observe how ASR changes under different protocols, datasets, and ε.

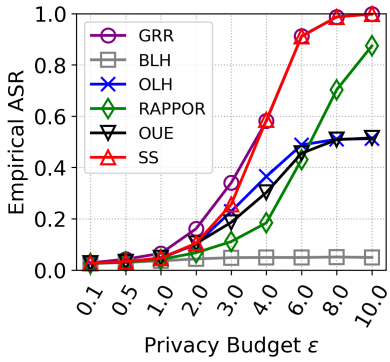
We give the results of this experiment in Figure 4.4. We first observe that these experiment results agree with the theoretical derivations done in Section 4.3 and illustrated in Figure 4.1. It is often the case that GRR's ASR is higher than remaining protocols. SS protocol has lower ASR than GRR for small ε, but its ASR becomes very close to GRR when ε is larger. BLH often has the lowest ASR among all protocols, due to many hash collisions (on average $|\mathcal{U}|/2$ collisions).

Among the remaining protocols, RAPPOR has lower ASR than OLH and OUE when ε is small, but RAPPOR's ASR exceeds OLH and OUE when ε is large. The ε value at which RAPPOR's ASR starts exceeding OLH and OUE changes from dataset to dataset due to different $|\mathcal{U}|$ in different datasets. In the MSNBC dataset in which $|\mathcal{U}|$ is small, this ε value is between 4 and 6. In the Uniform and Exponential datasets in which $|\mathcal{U}|$ is relatively larger, this ε value is between 6 and 8. In the Kosarak dataset in which $|\mathcal{U}|$ is largest among all our datasets, this ε value is between 8 and 10. A key factor why RAPPOR's ASR exceeds OLH and OUE when ε is large is that OLH and OUE's ASR increase very slowly after they hit 0.5. This was observed as part of the theoretical expected ASR derivations, as well as Figure 4.1. The experimental results in Figure 4.4 validate the theoretical finding.

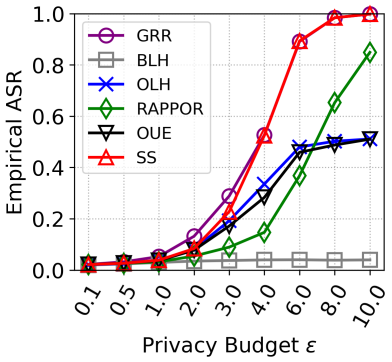


(a) MSNBC dataset

(b) Kosarak dataset



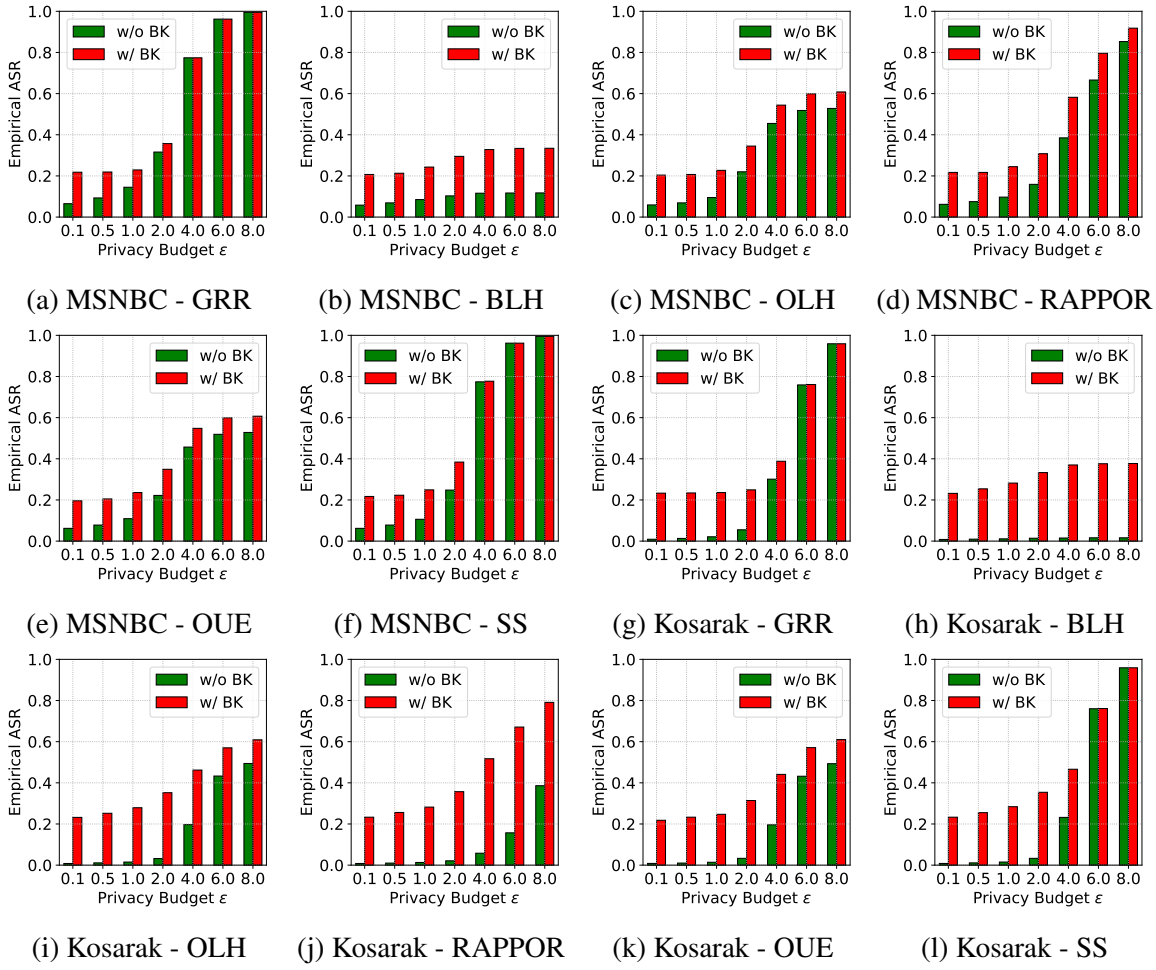
(c) Uniform dataset



(d) Exponential dataset (β=3)

Figure 4.4: Empirical ASR of $\mathcal{A}$ under all six protocols. While the individual ASR values may change from one dataset to another due to data domain and characteristics, overall trends agree across multiple datasets. These empirical results also agree with the trends observed in the theoretical analysis in Section 4.3 and in Figure 4.1.

Finally, Figure 4.4 shows that OLH and OUE's empirical ASR are similar for various values of ε although they differ slightly between $2 \leq \varepsilon \leq 6$. This is also true for their expected ASR, as illustrated in Figure 4.1. An interesting observation here is that OLH and OUE, which use very different encoding styles (g-ary hash versus unary bitvector encoding), can have similar ASR; as opposed to two protocols which use similar encoding principle having substantially different ASR behavior (such as BLH and OLH that both use hash encoding, or RAPPOR and OUE that both use bitvector encoding).



(a) MSNBC - GRR (b) MSNBC - BLH (c) MSNBC - OLH (d) MSNBC - RAPPOR

(e) MSNBC - OUE (f) MSNBC - SS (g) Kosarak - GRR (h) Kosarak - BLH

(i) Kosarak - OLH (j) Kosarak - RAPPOR (k) Kosarak - OUE (l) Kosarak - SS

Figure 4.5: Comparison of adversary with and without background knowledge (w/ BK versus w/o BK). Experiments using two real datasets and six protocols agree that adversarial knowledge of aggregate statistics improves ASR.

4.4.4 Impact of Adversary Knowledge

Recall from Section 4.2.3 and Section 4.3.1 that adversary $\mathcal{A}$ may inject his/her statistical background knowledge (BK) into the Bayesian prediction process through informed priors. We measure the impact of this by comparing the ASR without background knowledge (w/o BK) and with background knowledge (w/ BK).

The results are given in Figure 4.5 for the two real datasets (Kosarak and MSNBC) and for all six protocols. We compare the adversary w/ BK and w/o BK under each scenario in which other factors such as ε, dataset, and protocol are fixed. This comparison shows that,

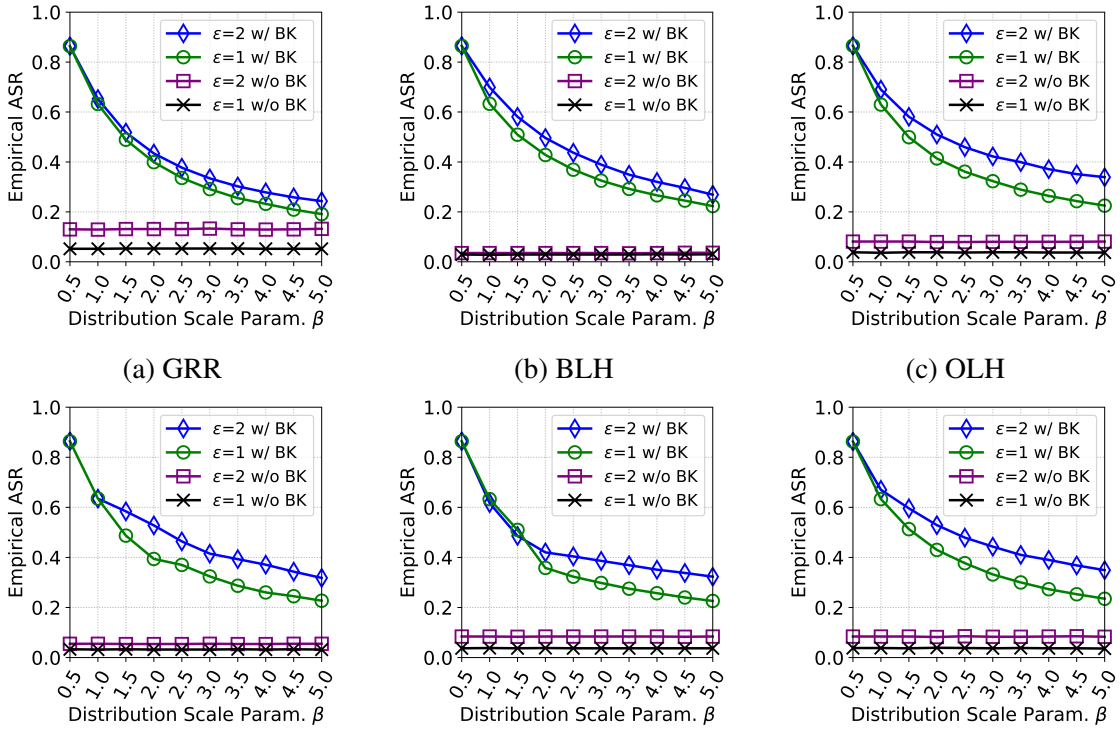
while the individual ASR values may differ in different protocols and datasets, all protocols and datasets agree that the adversary w/ BK outperforms the adversary w/o BK in terms of ASR. We note that in protocols GRR and SS, the adversary w/o BK catches up with the adversary w/ BK when ε is large such as 6 or 8. However, in the remaining protocols, there remains a gap between the adversary w/ BK and w/o BK across varying ε.

The take-away message from this analysis is that in a practical scenario with a realistic adversary model and dataset, an adversary with statistical background knowledge may achieve higher accuracy in predicting clients' true values than an adversary without background knowledge, under the same ε, dataset, and protocol.

4.4.5 Impact of Data Skewness

We now study the impact of data skewness, i.e., statistical distribution of client data. If some values $v \in \mathcal{U}$ are very frequently observed in the client population whereas other values are rarely observed, we say that data distribution is skewed. If the observation frequency of all values in $\mathcal{U}$ is roughly even, we say that the data is uniform (i.e., less skewed or not skewed). We create synthetic client populations with varying data skewness by changing the distribution scale parameter of the Exponential dataset as explained in Section 4.4.1. We measure ASR under each dataset with two scenarios: adversary w/ BK and adversary w/o BK.

The results are given in Figure 4.6. We first observe that varying data skewness does not have observable impact on ASR when the adversary does not have BK. This is confirmed by roughly constant ASR values for the adversary w/o BK while skewness is varied, in all six protocols and $\varepsilon = 1$ and 2. However, for the adversary w/ BK, ASR values change substantially with varying skewness. In cases where there is large data skewness (leftmost end of each graph in Figure 4.6), ASR is much higher than cases that are not skewed (rightmost end of each graph in Figure 4.6). The take-away message from this experiment therefore is that skewness plays a role in ASR when the adversary has background knowledge of the



(a) GRR (b) BLH (c) OLH

(d) RAPPOR (e) OUE (f) SS

Figure 4.6: Analyzing the impact of data skewness on ASR, for adversary with and without background knowledge. The distribution scale parameter β on x axis controls data skewness, where smaller β indicates skewness is higher according to Exponential distribution. Results show that adversary w/o BK is not impacted by change in skewness, but adversary w/ BK enjoys substantially higher ASR on more skewed datasets.

skewed statistics; otherwise, it does not impact ASR.

4.5 LDPLens

4.5.1 Motivation

We have established both mathematically and empirically that under a common adversary model and measurement metric (e.g., our Bayesian adversary $\mathcal{A}$ and ASR metric), different LDP protocols may yield different results, according to factors such as: (i) LDP budget ε, (ii) encoding type and parameters used by the protocol, (iii) the domain of user data, e.g., $\mathcal{U}$, (iv) background knowledge BK, and (v) statistical data distribution and skewness. Furthermore, when these factors are set in a certain way, one protocol may yield lower

ASR than the other; whereas for a different set of factors, the other protocol may yield lower ASR and therefore become more preferable. For example, Figure 4.1 and Figure 4.4 agree that when $\varepsilon = 4$, the RAPPOR protocol has lower ASR compared to OUE, whereas when $\varepsilon = 8$, the OUE protocol has lower ASR compared to RAPPOR.

Since there is no single protocol that yields lowest ASR and highest utility simultaneously across all possible factors, the problem of *protocol selection* becomes an open question: Given a certain setting with a certain set of factors, which LDP protocol is most desirable to use? The protocol selection problem becomes even more challenging when one considers the utility perspective, given the results in recent works showing that one protocol may have higher utility than another for some ε and $\mathcal{U}$, whereas the opposite may be true for other values of ε and $\mathcal{U}$ [84, 105, 121].

In this section, we propose LDPLens to assist researchers and application designers in analyzing the privacy-utility tradeoffs of LDP protocols, and therefore making informed decisions regarding protocol selection and ε budget selection. We outline two example use cases for LDPLens:

(1) Let Alice be a researcher who wants to design a locally differentially private system for her application domain, which could be healthcare, IoT, cybersecurity, etc. Alice knows the data type and $\mathcal{U}$, as well as the statistical characteristics and suitable encoding types for the data in her domain. She also knows the adversary metric and utility metric that are applicable to her domain. Under these settings, Alice wants to learn which LDP protocol and ε value would be most suitable to achieve a good tradeoff between privacy and utility. Alice can provide these metrics and settings to LDPLens to analyze the tradeoff performances of existing LDP protocols, which assists her in choosing a suitable protocol. Furthermore, Alice can specify a utility loss constraint [resp. privacy loss constraint] that she believes is the maximum utility loss [resp. privacy loss] that is tolerable in her application. Alice can then learn from LDPLens which protocols and ε budgets would satisfy her constraint, as well as the ε budget that would be appropriate for her constraint.

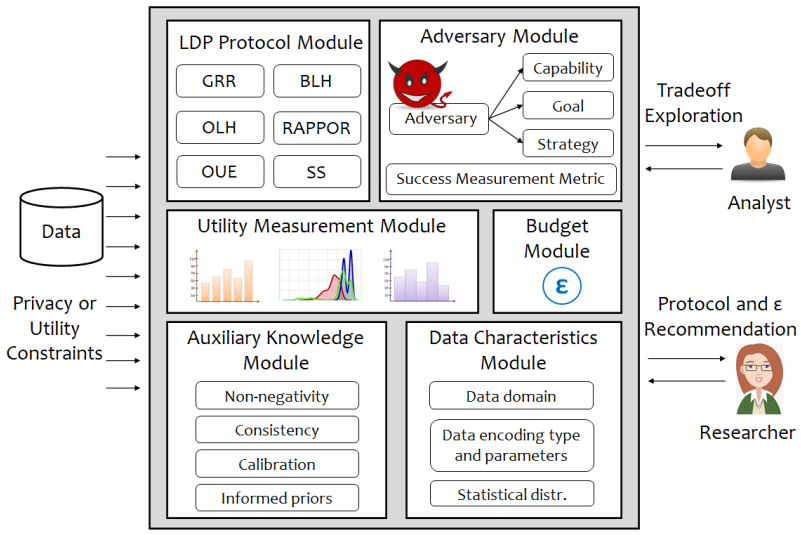


Figure 4.7: Overview of LDPLens design

(2) Another researcher, Bob, may use LDPLens to benchmark existing LDP protocols under various ε, $\mathcal{U}$, and encoding parameters. Bob's results may indicate that existing protocols provide sub-optimal utility under a certain choice of $\varepsilon, \mathcal{U}$ and encoding parameters. Bob can then design a new LDP protocol targeted to achieve higher utility under that particular subset of parameters, and use LDPLens to evaluate his proposed protocol versus existing protocols in terms of privacy and utility under various adversary metrics and utility metrics.

4.5.2 Design of LDPLens

LDPLens is designed to be a customizable system that enables effective exploration of privacy-utility tradeoffs of LDP protocols, according to the privacy and utility metrics specified by users. The design of LDPLens is summarized in Figure 4.7. LDPLens consists of several modules as follows:

LDP Protocol Module: This module contains the implementations of available LDP protocols, e.g.: GRR, BLH, OLH, RAPPOR, OUE, SS. A new protocol named PROT can be added to LDPLens by implementing two functions of this module: one for implementing the client-side encoding and perturbation process Ψ_{PROT}, and one for implementing the

server-side estimation process.

ε **Budget Module:** This module specifies the allowed range of the LDP privacy budget ε. By definition, $\varepsilon > 0$ is a positive float. In the current implementation of LDPLens, we allow ε between 0.01 and 16 as a wide range of possible values.

Adversary Module: This module contains: (i) the adversary or attack model, and (ii) the metric for measuring how successful the adversary/attack is. Our Bayesian adversary $\mathcal{A}$ and Adversarial Success Rate metric (ASR) from Section 4.3.1 constitute one option that is currently implemented in this module. New adversary models and privacy metrics may be added to LDPLens by implementing them in this module. For example, there have been recent pre-prints which propose new attacks to LDP protocols, such as Cao et al.'s data poisoning attacks [133] and Cheu et al.'s manipulation attacks [134]. (Note that these works assume a different, stronger adversary model than ours. The adversary in their attacks are capable of compromising a subset of clients' devices.) Another work by Lopuhaa-Zwakenberg et al. [135] propose information-theoretic metrics for measuring privacy in LDP protocols. Given the recent interest in alternative adversary models and privacy metrics for LDP protocols, we believe that LDPLens can serve as a unifying platform that integrates multiple adversary models and privacy metrics.

Utility Measurement Module: This module contains the metrics for measuring utility loss. Since LDP relies on randomized perturbation, there will be some amount of utility loss caused by the LDP protocol. Depending on the application domain and data type, the utility metric can be different.

One metric that is currently implemented in LDPLens, which we use in our case studies in the next section, is L_p norm error in item frequency estimation. This is a fundamental utility metric that has received significant attention in the LDP literature [84, 120, 118, 125, 18]. It can be formalized as follows. Recall from Section 4.2.2 that $|\mathcal{L}|$ denotes client population size, $C(v)$ denotes the true count of item v, and $\bar{C}(v)$ denotes the estimated count of v, i.e., estimated count recovered by the server at the end of LDP protocol. Then,

v's true frequency is $f(v) = C(v)/|\mathcal{L}|$ and its estimated frequency is $\bar{f}(v) = \bar{C}(v)/|\mathcal{L}|$. Average L_p norm error in item frequency estimation is:

$$\frac{\sum_{v \in \mathcal{U}} \left(|f(v) - \bar{f}(v)| \right)^p}{|\mathcal{U}|} \tag{4.47}$$

In our LDPLens demonstration in the next section, we use $p = 1$.

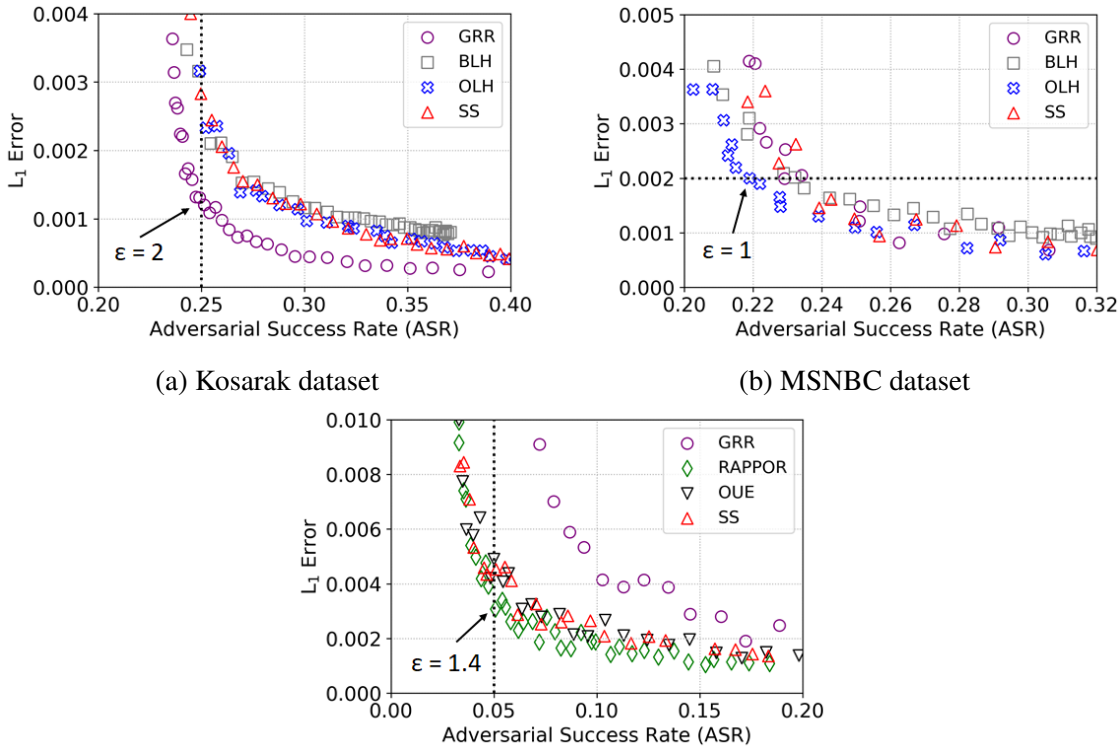
Data Characteristics Module: This module contains information regarding client data, such as the data domain $\mathcal{U}$, suitable data representation and encoding methods (e.g., bitvector or hash), encoding parameters, and statistical distribution or skewness. These factors can either be specified by the LDPLens user, or automatically parsed from sample input data given to LDPLens.

Auxiliary Knowledge Module: This module captures any auxiliary knowledge or background knowledge (BK) that may stem from domain semantics or characteristics. We showed in Section 4.4.4 that such BK may be used by an adversary in increasing ASR. In recent works, it was shown that similar auxiliary knowledge may also improve LDP protocol utility through consistency and non-negativity requirements [127] as well as estimation calibration via post-processing [126]. Since auxiliary knowledge may have dual impact (impacts both privacy and utility), we capture it in LDPLens using its own module.

It should be noted that while LDPLens modules come with default implementations in place, they are extensible and customizable. It is possible to add new LDP protocols, adversary models, utility measurement metrics, data types, and so forth into LDPLens.

4.5.3 Case Studies with LDPLens

We now demonstrate how an analyst or researcher may use LDPLens to explore the privacy-utility tradeoffs of multiple LDP protocols, and find out which protocol is most favorable under their choice of dataset and privacy/utility metrics. We perform three case studies with Kosarak, MSNBC and Uniform datasets. We illustrate their results in Figure 4.8. We



(a) Kosarak dataset

(b) MSNBC dataset

(c) Uniform dataset

Figure 4.8: Exploring tradeoffs between Adversarial Success Rate and L_1 error (utility loss) using LDPLens. Experiments show that for different datasets and privacy/utility constraints, the protocol that yields the most favorable tradeoff may be different, e.g., in Kosarak it is GRR, in MSNBC it is OLH, and in Uniform it is RAPPOR. LDPLens helps to recommend a protocol under a given ASR or L_1 error constraint, as well as to recommend an appropriate ε value for that constraint.

assume the adversary module of LDPLens is invoked with the Bayesian adversary $\mathcal{A}$ and ASR metric from Section 4.3.1, and the utility module of LDPLens is invoked with L_1 norm item frequency estimation error from Equation 4.47. Each data point in the graphs correspond to a protocol execution with one value of ε, and different points are plotted for ε ranging between 0.1 and 4.0 in increments of 0.1. Since lower ASR means higher privacy and lower L_1 estimation error means higher utility, the bottom-left corner of each graph is most desirable. However, as expected, all protocols show tradeoffs between privacy and utility: as L_1 errors decrease, ASRs increase. Yet, some protocols achieve more favorable tradeoffs than others. Next, we analyze their tradeoff behaviors in each case study.

Case Study #1: Consider the comparison between GRR, BLH, OLH and SS protocols

in Figure 4.8a on the Kosarak dataset. Say that Alice wants ASR to be at most 0.25, and wants to choose the protocol and suitable ε value that minimizes L_1 error under this constraint. We first represent Alice's desired ASR constraint via the vertical dotted line in Figure 4.8a. Under this constraint, we observe that the GRR protocol gives the most favorable tradeoff, since for ASR = 0.25, GRR has the lowest L_1 error. The utility loss of all remaining protocols are 1.5-2x higher than GRR. In addition to showing that GRR is the protocol achieving the most favorable tradeoff for this dataset and ASR constraint, LDPLens can also recommend a suitable value of ε, which happens to be $\varepsilon \approx 2$. Finally, we observe that the selection of GRR as the most favorable protocol is robust to small changes in ASR constraint. In fact, GRR is observed to be the most favorable protocol for all ASR values ranging between 0.23 and 0.35.

Case Study #2: Consider the comparison between GRR, BLH, OLH and SS protocols in Figure 4.8b on the MSNBC dataset. Say that Bob wants L_1 error to be at most 0.002, and wants to choose the protocol and suitable ε value that minimizes ASR under this constraint. We represent Bob's desired L_1 error constraint via the horizontal dotted line in Figure 4.8b. Under this constraint, we observe that the OLH protocol gives the most favorable tradeoff since its ASR is lowest compared to other protocols. In addition, the recommended value of ε is $\varepsilon \approx 1$. Similar to the previous example, the selection of OLH as the most favorable protocol is robust to changes in the L_1 error constraint, since OLH remains the most favorable protocol for values of L_1 error between 0.001 and 0.0035.

Case Study #3: Consider the comparison between GRR, RAPPOR, OUE and SS protocols in Figure 4.8c on the Uniform dataset. Say that Charlie wants ASR to be at most 0.05, and wants to choose the protocol and ε value that minimizes L_1 error under this constraint. The most favorable protocol in this case becomes RAPPOR, with a recommended ε value of $\varepsilon \approx 1.4$.

Summary and Discussion: First, we observe that the most favorable protocol in each case study is different. This supports the initial motivation of LDPLens that the most fa-

vorable protocol can change from scenario to scenario, and thus finding the most favorable protocol in a new scenario/dataset is a non-trivial problem. Second, we observe that the choice of the most favorable protocol in a given scenario can be robust, i.e., even if there are small changes in privacy or utility constraints for that scenario, the most favorable protocol remains the same. Finally, there is substantial benefit in using the protocol recommended by LDPLens rather than a randomly chosen protocol. On the Kosarak dataset, there is roughly 2-fold improvement in utility when the GRR protocol (recommended by LDPLens) is used to achieve ASR = 0.25 rather than using BLH, OLH or SS to achieve the same ASR. On the Uniform dataset, RAPPOR's L_1 error is several folds better than GRR under the given ASR constraint. These clearly demonstrate the benefit in using LDPLens to discover a protocol that is "good fit" for a given scenario, rather than randomly choosing a protocol.

4.6 Related Work

LDP has recently received significant attention from the research community as well as the industry [121, 14, 122, 15]. Several LDP protocols were developed and deployed, such as GRR, BLH, OLH, RAPPOR, OUE and SS [95, 84, 14, 85, 123, 124]. These protocols often serve as building blocks for more advanced analytical tasks, e.g., heavy hitter identification [96, 111, 117], mining frequent items and itemsets from set-valued data [97, 105], mining key-value stores [118, 125], computing and releasing marginals from high-dimensional data [112, 136, 113], answering multi-dimensional analytical range queries [137, 138], mean estimation from numerical data [83, 139], privately analyzing decentralized social graphs [115], monitoring evolving data [140], and discovery of frequent terms [116, 85].

There have also been theoretical studies to establish upper and lower bounds on the accuracy of what can be learned in the LDP setting, as well as to study the importance of client-server interactivity in learning [141, 142, 110, 111]. Other studies have shown that estimation accuracy in the LDP setting can be improved via post-processing, such as non-

negativity and consistency requirements [127, 116], prior statistical knowledge [126], and the likes of expectation maximization or maximum likelihood estimation [143, 127, 85].

Recent pre-prints proposed measuring the privacy properties and adversarial resilience of LDP protocols against adversaries with varying power and success metrics. [135] considers information-theoretic metrics for measuring privacy. [133] and [134] study data poisoning and manipulation attacks to LDP protocols, in which the adversary has the capability of compromising a subset of clients' devices. This is a stronger adversary compared to our work. While the adversary power and success metrics may be different, these types of attacks may also be integrated into LDPLens so that LDPLens becomes a unifying platform that integrates multiple adversary models and metrics for convenient benchmarking.

Finally, several relaxations and alternatives to LDP were proposed in the literature. These include PLDP [114], CLDP [18], ULDP [120] and ID-LDP [144]; as well as BLENDER in which only a subset of users enjoy LDP whereas remaining users act as opt-in beta testers receiving the guarantees of centralized DP [119]. Since the mathematical formalism and/or architectural setting of these relaxations are different than LDP, the current version of LDPLens does not include these relaxations. In future work, we will explore principled methods to integrate them into LDPLens.

4.7 Conclusion

In this chapter, we studied the privacy properties of LDP protocols through the lens of a Bayesian adversary model, using both mathematical derivations and empirical measurements. We showed that the Adversarial Success Rate (ASR) changes from protocol to protocol, and that there are multiple factors affecting ASR, including privacy budget ε, domain $\mathcal{U}$, encoding parameters, background knowledge, and data skewness. Consequently, we argued that no single protocol is consistently better than others in terms of ASR and utility across all possible settings. We then designed and developed a prototype system called LDPLens, which considers the aforementioned factors through customizable modules, and

enables the selection of the most suitable LDP protocol for the given set of application-specific settings. We presented 3 case studies with real-world datasets, showing that the best protocol in each case is different. Our results showed that one can gain substantial utility improvement by using the protocol recommended by LDPLens rather than randomly choosing one LDP protocol, which demonstrates the benefit of informed protocol selection using LDPLens.

CHAPTER 5
CONCLUSION, ONGOING AND FUTURE WORK

In today's world, individuals interact with numerous electronic devices and networked services every day, ranging from smartphone apps to smartwatches and to smart home sensors. These interactions lead to large amounts of sensitive data to be generated, including individuals' location traces, web browser settings, and browsing and purchase histories. While these datasets are lucrative sources for understanding and modeling human behavior, individuals' privacy must be respected. This dissertation contributes algorithmic and practical solutions to the problem of data privacy in modern computerized systems from two perspectives: privacy-preserving data collection and privacy-preserving data sharing. In this chapter, we provide a summary of dissertation contributions, discuss our ongoing work, and present directions for future work.

5.1 Summary

In summary, this dissertation makes the following contributions. First, in order to enable privacy-preserving sharing of individuals' location traces, we designed and developed a system for utility-aware synthesis of differentially private and attack-resilient location traces, called AdaTrace. Given a set of real location traces, AdaTrace builds a generative model in four steps: feature extraction from real traces, synopsis construction and learning, noise injection to achieve privacy protection, and generation of synthetic location traces. The location traces synthesized by AdaTrace bear a statistical resemblance to real traces, but they do not leak sensitive information about any individual, due to the enforcement of differential privacy and attack-resilience guarantees. Through experiments on three datasets, we demonstrated that the location traces synthesized by AdaTrace preserve aggregate location trace utility.

Second, we designed and developed privacy-preserving algorithms and protocols for collecting mobile and Web user data from users' IoT devices. We showed that the existing protocols that implement Local Differential Privacy (LDP) for privacy-preserving data collection perform poorly in terms of aggregate utility when data is collected from a small population of users. Motivated by this, we introduced the notion of Condensed Local Differential Privacy (CLDP) and a suite of protocols satisfying CLDP to enable the collection of various types of user data with different syntax. We demonstrated the real-world applicability and utility improvement of our CLDP protocols on real-world item purchase datasets as well as cybersecurity case studies.

Third, we presented a formal and experimental analysis of the privacy properties of LDP protocols through the lens of a Bayesian adversary. We analyzed a set of critical parameters that may cause diverse responses of different LDP protocols to the same Bayesian adversary in terms of Adversarial Success Rate (ASR), including privacy budget ε, data domain $\mathcal{U}$, encoding parameters, data skewness, and background knowledge. We showed that no single protocol is consistently better than others in terms of both ASR and utility across all possible settings of parameters. We then designed and developed LDPLens, a prototype system which allows LDP users to customize and select one of the most suitable LDP protocols for a given application scenario with a set of domain-specific privacy and utility requirements. Tested with three case studies with real-world datasets, we showed that the protocol recommended by LDPLens in each scenario is different, and it offers substantial utility improvement compared to randomly choosing an LDP protocol.

5.2 Ongoing Research Directions

One of our ongoing research directions is the *repeated* and *periodic* collection of user data with local differential privacy. Consider a client-server setting in which clients' data is perturbed with LDP before being collected by the server. In many real-world systems with continuous software monitoring and telemetry monitoring goals, the server collects client

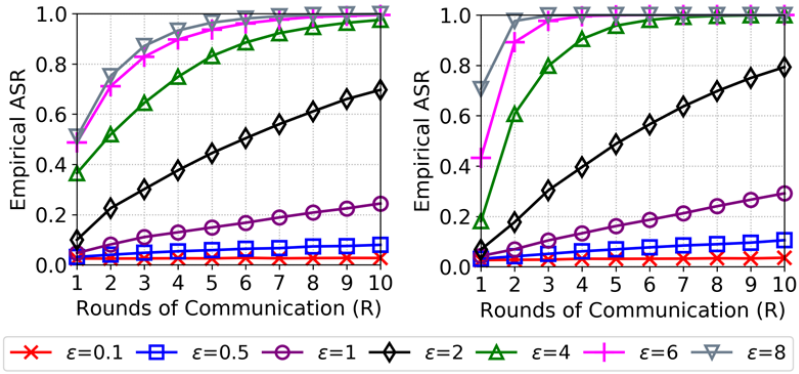


Figure 5.1: Impact of increasing number of data collection rounds on empirical ASR of the Bayesian adversary. The left graph is with the OLH protocol. The right graph is with the RAPPOR protocol.

data repeatedly and periodically over time [145, 146], e.g., Apple's collection of Safari data twice every day and Health data once every day [147], and Microsoft's collection of telemetry statistics once every 6 hours [121]. Furthermore, in iterative and interactive learning systems such as federated learning, clients contribute to the global model in multiple rounds of the learning process [110, 148, 149, 150]. In such scenarios, although a single perturbed response from the client may not violate privacy by itself; accumulation of perturbed responses across multiple rounds, originating from the client's same underlying data, may degrade privacy due to accumulating privacy loss.

We demonstrate the risk of privacy degradation in repeated data collection using the Bayesian adversary $\mathcal{A}$ and Adversarial Success Rate (ASR) metric from Chapter 4. In Figure 5.1, we show how ASR changes with varying number of communication rounds between the client and the server. The number of communication rounds, which is equal to the number of data collections, is denoted by R. The analysis is performed with two popular LDP protocols (RAPPOR and OLH) with varying ε values, using the Uniform dataset from Chapter 4. The results clearly show that ASR increases as data is collected across more rounds (R is increased). This is because increasing the number of observations available to $\mathcal{A}$ also increases the success of Bayesian inference, resulting in higher ASR. Furthermore, while there is positive correlation between ASR and R as well as ASR and ε, the rate of

increase in ASR is different for each protocol and ε. For small ε such as 0.5 or 1, ASR increases either linearly (with small slope) or sub-linearly with respect to R. In contrast, for high ε such as 4 or 6, ASR increases super-linearly between $1 \leq R \leq 5$ and becomes equal to 100% after $R \geq 6$. This shows that the adverse impact of repeated collection on longitudinal privacy with increasing R is also dependent on choice of protocol and ε.

In our ongoing research, we are investigating defense strategies to the problem of privacy degradation in repeated data collection. In particular, we are considering two defense strategies: Memoization (MEM) and Bagging (BAG).

Memoization (MEM) has two parameters: ε_p for controlling the permanent ε_p-LDP guarantee that will hold despite arbitrary R, and ε_t for controlling the instantaneous ε_t-LDP guarantee that will be achieved in each round τ of data collection. In MEM, the client device encodes and perturbs client's true value with ε_p-LDP prior to the start of data collection. This perturbed value is said to be securely *memoized* by the client device. Then, at each collection round τ, the memoized value is re-perturbed with ε_t-LDP. The output of the second perturbation is sent to the server.

Since the outcome observed by the server is the result of two perturbations (one with ε_p-LDP, one with ε_t-LDP), ASR is smaller. In fact, despite arbitrarily large R, MEM can be shown to uphold the ε_p-LDP guarantee. We exemplify this property in Figure 5.2. The graph on the left of Figure 5.2 is with $\varepsilon_p = 2$ and the graph on the right is with $\varepsilon_p = 4$. The dashed black lines represent the ASR in the scenario where ε_p-LDP is enforced but there is no second perturbation. Each of the remaining lines denote MEM with different ε_t, ranging from $\varepsilon_t = 0.5$ to $\varepsilon_t = 6$. It can be observed that despite many rounds of communication (R approaching 50), the ASR values do not exceed the dashed black line. The speed with which they approach the dashed black line, however, changes according to the value of ε_t. That is, with $\varepsilon_t = 1$ it takes around $R = 30$ rounds of communication for ASR to become equal to that of the dashed black line; however with $\varepsilon_t = 4$, it takes only $R = 5$ rounds for ASR to become equal to that of the dashed black line. Overall, the results here show that

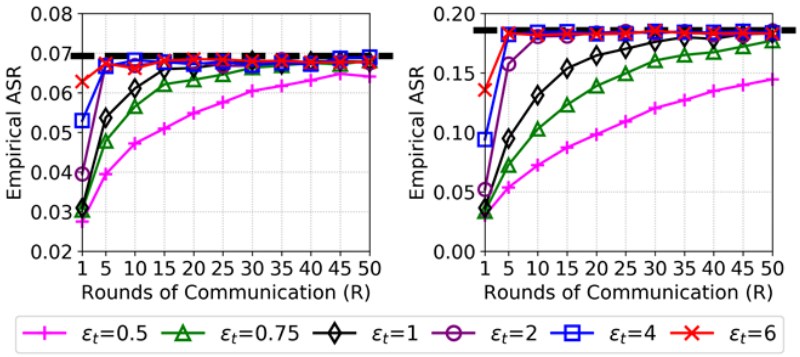


Figure 5.2: Behavior of the Memoization (MEM) defense strategy. In both graphs, RAPPOR protocol and Uniform dataset are used for illustration. $\varepsilon_p = 2$ in the left graph, $\varepsilon_p = 4$ in the right graph. Results show that MEM can successfully achieve an upper bound on ASR despite increasing number of communication rounds.

in MEM, the adversary resilience achieved by ε_p-LDP is upheld regardless of arbitrary R and ε_t; however, resilience will be higher (ASR will be lower) for lower R and ε_t.

On the other hand, the permanent ε_p-LDP guarantee achieved by MEM comes at the cost of increased utility loss. Since data must be perturbed twice on the client device, the expected utility loss in MEM becomes high. Motivated by the goal of achieving a permanent ε_p-LDP guarantee without requiring two perturbations, we propose the Bagging (BAG) strategy.

Bagging (BAG) has two parameters: ε_p for controlling the permanent ε_p-LDP guarantee that will hold for arbitrary R, and κ which denotes the bag size. By definition, κ is an integer $\kappa \geq 1$. In BAG, client's device memoizes κ perturbed responses, each obtained by perturbing the true value with budget $\frac{\varepsilon_p}{\kappa}$, as opposed to one memoized response in MEM. We say that these perturbed responses constitute a *bag of noisy responses*, where κ is the size of the bag. At each round τ of data collection, the client randomly picks one element from the bag and sends it to the server.

Note that in BAG, there is no second perturbation as opposed to MEM. Nevertheless, since the obtainment of the bag of noisy responses satisfies ε_p-LDP, the permanent guarantee of ε_p-LDP upholds. The κ parameter serves to enable a tradeoff between instantaneous privacy and utility. As κ increases and approaches R, the utility of BAG decreases whereas

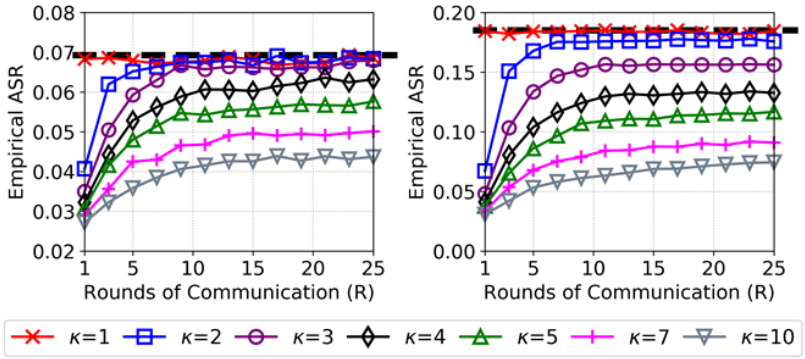


Figure 5.3: Behavior of the Bagging (BAG) defense strategy. In both graphs, RAPPOR protocol and Uniform dataset are used for illustration. $\varepsilon_p = 2$ in the left graph, $\varepsilon_p = 4$ in the right graph. Results show that BAG can successfully implement an upper bound on ASR despite increasing number of communication rounds.

adversarial resilience increases. As κ is closer to 1, utility becomes higher whereas adversarial resilience becomes lower.

We demonstrate BAG's behavior in Figure 5.3. Similar to Figure 5.2, the dashed black line for the graph on the left is with $\varepsilon_p = 2$ and the graph on the right is with $\varepsilon_p = 4$. It can be observed that with $\kappa = 1$, the ASR of BAG is roughly equivalent to that of the black dashed line, and it does not change despite increasing R. With $\kappa \geq 2$, the ASR values increase with increasing R, but they do not exceed the dashed black line. This behavior of BAG empirically demonstrates that the permanent ε_p-LDP guarantee is achieved by BAG.

Given that there are two potential defenses (MEM and BAG), in our ongoing research, we are aiming to identify those cases and factors when BAG becomes preferable to MEM and vice versa. The first factor here is storage size. Since BAG memoizes a bag of responses with size κ whereas MEM memoizes only one response, the storage requirement of BAG is κ times that of MEM. The second factor is the real-time computation requirement. MEM requires the client device to re-perturb the memoized response at each round of data collection; thus, it incurs a real-time computation cost at each collection round. This can be a problem for resource-constrained IoT devices which have to execute an LDP protocol with a costly perturbation step, e.g., consider the bitwise perturbation of RAPPOR protocol, which has complexity linear in the size of the bitvector. In contrast, BAG only re-

quires the client device to pick among pre-computed noisy responses at each round of data collection, which is a much simpler operation. The third and perhaps the most complex factor is data utility. MEM's utility depends on ε_p and ε_t, whereas BAG's utility depends on ε_p and κ. In our ongoing research, we are looking for formal and empirical relationships to relate the utility impact of MEM's ε_t parameter to BAG's κ parameter.

5.3 Open Problems and Future Research Directions

Data privacy is an exciting and growing area of research. In the future, we are interested in extending this dissertation research in several ways. We highlight some open problems and future research directions below.

5.3.1 Scalable and Adaptive Synthesis of Privacy-Preserving Location Traces

In Chapter 2 of this dissertation, we developed AdaTrace for synthesizing privacy-preserving location traces. There are multiple ways in which we are planning to extend AdaTrace in future work. One of our future work directions is to extend AdaTrace to handle incremental updates, to make it compatible with growing and streaming location trace datasets. Currently, when the real location trace dataset changes (which is an input of the AdaTrace system), AdaTrace has to re-execute its feature extraction and synopsis construction steps. For large and/or frequently growing input datasets, re-executions of AdaTrace can be time and resource-consuming. As a result, we are investigating the feasibility of applying incremental updates to AdaTrace's synopsis while still satisfying AdaTrace's desired privacy and attack-resilience guarantees.

Another future work direction for AdaTrace is to design a method for adaptive selection of AdaTrace input parameters for utility-optimized location trace synthesis. AdaTrace has four budget distribution parameters, which determine how much privacy budget each synopsis element is assigned. These budget distribution parameters have a high impact on the privacy and utility level of each synopsis element. Currently, the budget distribution

in AdaTrace is set empirically to give good average utility across several metrics of utility. However, there can be situations in which the AdaTrace user thinks one particular utility metric is more important than others for their end application. To address such situations, in future work we will investigate automated and adaptive parameter selection strategies for AdaTrace so that its synthesis can be optimized towards a user-defined choice of utility metric.

5.3.2 Investigating Adversary and Attack Models in LDP

In Chapter 4 of this dissertation, we showed that different LDP protocols may yield different resilience against a common Bayesian adversary. In practice, the adversary types may differ according to the data type and application domain, e.g., the adversarial knowledge and goal in the healthcare domain could be different from that in the Web log domain or the machine learning domain. Consequently, one of our future work directions will be to investigate the impacts of different adversary and attack types to LDP protocols. We plan to extend the adversary module of LDPLens (Chapter 4) to incorporate new adversary and attack types. The adversary and attack types can also become new benchmarks to establish fair comparisons and relationships between CLDP (Chapter 3) and LDP.

5.3.3 Applying DP, LDP and CLDP to Combat ML Attacks

Machine learning (ML) is revolutionizing many sectors such as computer vision, healthcare, and natural language processing. Recent ML approaches powered by deep neural networks require large training datasets to be successful. This increases the demand to collect and process potentially sensitive subject-specific data such as photos, speech, electronic healthcare records (EHRs), and so forth. It is therefore desirable to apply appropriate privacy countermeasures to ML training and testing processes in order to protect sensitive subject-specific data, such as Differential Privacy (DP), Local Differential Privacy (LDP) or Condensed Local Differential Privacy (CLDP) [151, 152, 153].

Another future work direction is to investigate the impacts of DP, LDP and CLDP on ML attacks. Several ML attacks have been reported in the literature. Popular training-time ML attacks include data poisoning, label manipulation and gradient leakage attacks [154, 155]. Popular test-time ML attacks include membership inference, adversarial examples, and model stealing attacks [156, 10, 157]. While these attacks have been shown to be successful on vanilla ML models, we plan to investigate their success rates on ML models that are protected by DP, LDP and CLDP.

5.3.4 LDP and Fairness

Algorithmic fairness has become an important research topic, following findings that real-world algorithms and ML deployments may contain implicit or explicit biases which may disproportionately impact minority groups in a population. However, currently in the LDP literature, it is assumed that the same LDP protocol is executed on each client's device regardless of data skewness or whether the client belongs to a minority group or not. Thus, there may be several research directions to pursue at the intersection of LDP and fairness.

First, our results in Chapter 4 show that adversarial knowledge of data skewness may increase Adversary Success Rates of predicting clients' true values. This may indicate that skewed client populations or sub-populations may be at higher risk. In future work, we plan to investigate adversary and utility metrics that are specialized towards measuring the LDP protocol impact on skewed sub-populations or minority groups. One concrete approach could be to adapt the ASR definition in Chapter 4 to measure ASR not over the whole client population $\mathcal{L}$, but instead over the skewed sub-population $\mathcal{L}_{skew} \subseteq \mathcal{L}$. Similarly, the utility metric in Chapter 4 can be modified to measure utility for only some values $v \in \mathcal{U}_{skew}$ instead of the whole domain $\mathcal{U}$. The modular and customizable nature of LDPLens design enables the integration of such new adversary and utility metrics seamlessly, which makes it easier to measure the impact of fairness in LDP protocols.

Second, a broader question at the intersection of LDP and fairness remains open: What

is the "right" way of defining the adversary metric to measure the amount of disclosure, i.e., should we measure *average* disclosure in a (sub-)population or should we measure the *worst-case* disclosure for any one individual in the population? Our ASR metric in Chapter 4 is closer to the prior approach since it measures the adversary's success across the whole population $\mathcal{L}$. Our Maximum Posterior Confidence (MPC) metric in Chapter 3 is closer to the latter approach since it is measured as the maximum (worst-case) disclosure the adversary can achieve across all inputs and outputs. In future work, we plan to investigate and measure the difference between the average disclosure approach versus the worst-case disclosure approach to enable clients compare the best-case, average-case, and worst-case risks of different LDP protocols and ε budgets.

5.3.5 Support for Dynamic and Evolving Environments in LDPLens

The current prototype LDPLens system in Chapter 4 performs its tradeoff analysis and LDP protocol recommendations in a *static* setting where the characteristics and properties of the environment are fixed, e.g., the data domain $\mathcal{U}$ is constant, data distribution and encoding are constant, and so forth. However, in scenarios such as smart cities, IoT, and sensor streams, we may have dynamic and evolving environments in which the aforementioned factors may change over time. In future work, we plan to extend LDPLens to support such dynamic environments efficiently.

There are two current advantages regarding how LDPLens may support dynamic and changing environments. First, our case studies in Chapter 4 show that the protocol recommended by LDPLens remains robust despite small changes in ASR and/or L_1 utility loss constraints. Thus, we would expect that even if there are small changes to the environment, LDPLens would still make the same or similar recommendations. Second, in current real-world LDP deployments, data is collected periodically every few hours, e.g., Apple collects Safari data twice every day and Health data once every day [147], and Microsoft collects telemetry statistics once every 6 hours [121]. If there are environmental changes from one

collection to another, it is possible to re-execute the tradeoff analysis of LDPLens during the time between consecutive collections (which is several hours) and obtain updated recommendations. For the representative datasets currently used in our case studies, LDPLens takes less than a few hours to finish its analysis on a commodity laptop, which makes it feasible to re-execute LDPLens in the time between one data collection and another.

Nevertheless, as LDP deployment becomes pervasive, there could be situations in which data is collected frequently or continuously, and environmental changes are common. For example, in a smart city environment, data can be continuously collected with LDP and yet there may exist frequent concept drifts with large novelty. We may need to extend LDPLens to make it more effective and efficient in such situations. Towards this goal, there are two future work directions. First, we plan to simulate concept drift and novelty using skewed data subsets. By proactively simulating different scenarios with varying concept drifts and pre-computing protocol recommendations in each condition with LDPLens, we can ensure that when a concept drift eventually does happen in a real-time system, we can use the pre-computed protocol recommendations and thereby minimize delay in adapting to the new environment. Second, we plan to investigate the possibility of building a supervised ML model using pre-computed recommendations of LDPLens. That is, we will execute LDPLens many times to generate a dataset of diverse LDPLens recommendations to formulate a training dataset, and then train a supervised ML model which mimics the behavior of LDPLens using this training dataset. Then, one can query the ML model in a real-time system to obtain protocol recommendations almost instantaneously, rather than actually executing LDPLens. This strategy of mimicking the behavior of LDPLens using a supervised ML model is similar to the idea behind knowledge distillation and model compression in machine learning [158, 159, 160], which aim to build simpler models that behave similarly to larger, more complex and less efficient models.

5.3.6 Support for Complex Query and Data Types in LDPLens

While the current implementation of LDPLens is a prototype, in future work we intend to extend its scope and applicability. To that end, in future work we plan to perform research in two directions: (i) supporting complex query types and query functionalities, and (ii) supporting high-dimensional and complex data types.

Support for Range Queries: A range query is a popular database operation which retrieves records in a database that have value between the upper and lower bounds specified by the range query. Range query functionality can be added to both the Bayesian adversary model and the utility metric in Chapter 4. First consider the adversary model. Instead of measuring ASR as the ratio of the client population whose true values v_ℓ are exactly correctly predicted by the adversary (i.e., $v_\ell^p = v_\ell$), we can build a range metric such that the adversary's prediction is said to be successful if it is within a small range of v_ℓ, i.e., $v_\ell^p \in [v_\ell - k, v_\ell + k]$ where k is a parameter controlling the size of the range. Next consider the utility metric. Instead of measuring L_1 error in the frequency of each single value $v \in \mathcal{U}$, we can measure errors in range query results such that $q(v_{lo}, v_{hi})$ where v_{lo} and v_{hi} denote the lower and upper bounds of the range query q. Note that both the adversary model and the utility metric are meaningful only when the domain of values $\mathcal{U}$ is ordinal. Adding support for such new adversary models and utility metrics is beneficial in extending the scope of LDPLens to support diverse real-world data analytics functionalities.

Support for Complex Data Types: In our experiments in Chapter 4, we assumed each client has one true value v_ℓ. In complex systems such as multi-dimensional tables, set-valued datasets or key-value stores, the client may have multiple true values or a database of true values following a certain organization (e.g., multiple tables with foreign key relationships or a key-value store). LDPLens can currently support such complex data types in two ways. First, recent LDP applications for complex data types often use the LDP protocols in LDPLens as building blocks, e.g., the approach for handling key-value stores in [125] uses GRR, RAPPOR and OUE protocols as building blocks, the approach for han-

dling set-valued data in [105] uses GRR and OLH protocols as building blocks, and so forth. Choosing a suboptimal building block to begin with would cause errors to propagate to the next stages of a complex application. Thus, LDPLens can be used to recommend suitable building blocks in such applications. Second, complex data types can sometimes be encoded into simpler data structures using application-specific or novel encoding strategies. For example, the neighbor list of a vertex in a graph can be represented as a bitvector, which enables the application of bitvector perturbation protocols such as RAPPOR and OUE. Strings and complex objects can be encoded into bitvectors using a Bloom filter approach [14, 85]. Set-valued data or key-value stores can be down-sampled into a singleton and then a protocol such as GRR and OLH can be applied. In such cases, LDPLens can be executed without further modification. In future work, we plan to investigate other new ways to support complex data types in LDPLens, and/or develop versions of LDPLens targeted towards certain complex data types, such as LDPLens for set-valued data, LDPLens for key-value stores, and so forth.

REFERENCES

[1] *The science behind the netflix algorithms that decide what you'll watch next*, https://www.wired.com/2013/08/qq-netflix-algorithm/.

[2] *Big data analytics in banking market*, https://www.mordorintelligence.com/industry-reports/big-data-in-banking-industry.

[3] *What is the impact of big data on the ecommerce industry: Let's find out*, https://towardsdatascience.com/what-is-the-impact-of-big-data-on-the-ecommerce-industry-lets-find-out-c5ac4b2dfd8.

[4] *Mobile ecosystem forum: Trust-related concerns could hamper consumer adoption of iot*, https://mobileecosystemforum.com/2016/04/07/trust-related-concerns-hamper-consumer-adoption-iot/.

[5] *Facebook - cambridge analytica data scandal*, https://en.wikipedia.org/wiki/Facebook%E2%80%93Cambridge_Analytica_data_scandal.

[6] *The biggest data scandals and breaches of 2018*, https://www.fastcompany.com/90272858/how-our-data-got-hacked-scandalized-and-abused-in-2018.

[7] *Nyc taxi & limousine commission – trip record data*, http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.

[8] *Uber movement: Let's find smarter ways forward*, http://movement.uber.com.

[9] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *IEEE Symposium on Security and Privacy*, IEEE, 2008, pp. 111–125.

[10] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, "Demystifying membership inference attacks in machine learning as a service," *IEEE Transactions on Services Computing*, 2019.

[11] I. Ozalp, M. E. Gursoy, M. E. Nergiz, and Y. Saygin, "Privacy-preserving publishing of hierarchical data," *ACM Transactions on Privacy and Security (TOPS)*, vol. 19, no. 3, pp. 1–29, 2016.

[12] M. E. Gursoy, L. Liu, S. Truex, L. Yu, and W. Wei, "Utility-aware synthesis of differentially private and attack-resilient location traces," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 196–211.

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

[14] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2014, pp. 1054–1067.

[15] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *Advances in Neural Information Processing Systems*, 2017, pp. 3571–3580.

[16] A. G. Thakurta, A. H. Vyrros, U. S. Vaishampayan, G. Kapoor, J. Freudinger, V. V. Prakash, A. Legendre, and S. Duplinsky, *Emoji frequency detection and deep link frequency*, US Patent App. 15/640,266, 2017.

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

[18] M. E. Gursoy, A. Tamersoy, S. Truex, W. Wei, and L. Liu, "Secure and utility-aware data collection with condensed local differential privacy," *IEEE Transactions on Dependable and Secure Computing*, 2019.

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

[20] B. Gedik and L. Liu, "Location privacy in mobile systems: A personalized anonymization model," in *25th IEEE International Conference on Distributed Computing Systems (ICDCS 2005)*, IEEE, 2005, pp. 620–629.

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

[22] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux, "Quantifying location privacy," in *2011 IEEE Symposium on Security and Privacy (S&P)*, IEEE, 2011, pp. 247–262.

[23] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec, "Protecting location privacy: Optimal strategy against localization attacks," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ACM, 2012, pp. 617–627.

[24] N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Optimal geo-indistinguishable mechanisms for location privacy," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2014, pp. 251–262.

[25] H. Ngo and J. Kim, "Location privacy via differential private perturbation of cloaking area," in *2015 IEEE 28th Computer Security Foundations Symposium (CSF)*, IEEE, 2015, pp. 63–74.

[26] L. Yu, L. Liu, and C. Pu, "Dynamic differential location privacy with personalized error bounds," in *Network and Distributed System Security Symposium (NDSS '17)*, 2017.

[27] Z. Montazeri, A. Houmansadr, and H. Pishro-Nik, "Achieving perfect location privacy in wireless devices using anonymization," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2683–2698, 2017.

[28] Y. Xiao and L. Xiong, "Protecting locations with differential privacy under temporal correlations," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2015, pp. 1298–1309.

[29] V. Bindschaedler and R. Shokri, "Synthesizing plausible privacy-preserving location traces," in *2016 IEEE Symposium on Security and Privacy (S&P)*, IEEE, 2016, pp. 546–563.

[30] K. Chatzikokolakis, C. Palamidessi, and M. Stronati, "A predictive differentially-private mechanism for mobility traces," in *International Symposium on Privacy Enhancing Technologies*, Springer, 2014, pp. 21–41.

[31] R. Chen, G. Acs, and C. Castelluccia, "Differentially private sequential data publication via variable-length n-grams," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ACM, 2012, pp. 638–649.

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

[33] C. Dwork, "Differential privacy," in *International Colloquium on Automata, Languages, and Programming*, Springer, 2006, pp. 1–12.

[34] S. P. Kasiviswanathan and A. Smith, "On the'semantics' of differential privacy: A bayesian formulation," *Journal of Privacy and Confidentiality*, vol. 6, no. 1, p. 1, 2014.

[35] 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.

[36] ——, "Pufferfish: A framework for mathematical privacy definitions," *ACM Transactions on Database Systems (TODS)*, vol. 39, no. 1, p. 3, 2014.

[37] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from gps trajectories," in *Proceedings of the 18th International Conference on World Wide Web*, ACM, 2009, pp. 791–800.

[38] T. Brinkhoff, "A framework for generating network-based moving objects," *GeoInformatica*, vol. 6, no. 2, pp. 153–180, 2002.

[39] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, ACM, 2003, pp. 31–42.

[40] R. Shokri, "Privacy games: Optimal user-centric data obfuscation," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 299–315, 2015.

[41] H. Jiang, P. Zhao, and C. Wang, "Roblop: Towards robust privacy preserving against location dependent attacks in continuous lbs queries," *IEEE/ACM Transactions on Networking*, vol. 26, no. 2, pp. 1018–1032, 2018.

[42] Y. Cao, M. Yoshikawa, Y. Xiao, and L. Xiong, "Quantifying differential privacy in continuous data release under temporal correlations," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 7, pp. 1281–1295, 2018.

[43] O. Abul, F. Bonchi, and M. Nanni, "Never walk alone: Uncertainty for anonymity in moving objects databases," in *2008 IEEE 24th International Conference on Data Engineering*, IEEE, 2008, pp. 376–385.

[44] M. E. Nergiz, M. Atzori, B. Guc, and Y. Saygin, "Towards trajectory anonymization: A generalization-based approach," *Transactions on Data Privacy*, vol. 2, no. 1, pp. 47–75, 2009.

[45] R. Yarovoy, F. Bonchi, L. V. Lakshmanan, and W. H. Wang, "Anonymizing moving objects: How to hide a mob in a crowd?" In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, ACM, 2009, pp. 72–83.

[46] R. Chen, B. C. Fung, N. Mohammed, B. C. Desai, and K. Wang, "Privacy-preserving trajectory data publishing by local suppression," *Information Sciences*, vol. 231, pp. 83–97, 2013.

[47] Y.-A. De Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Scientific Reports*, vol. 3, p. 1376, 2013.

[48] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[49] M. Douriez, H. Doraiswamy, J. Freire, and C. T. Silva, "Anonymizing nyc taxi data: Does it matter?" In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE, 2016, pp. 140–148.

[50] C. Y. Ma, D. K. Yau, N. K. Yip, and N. S. Rao, "Privacy vulnerability of published anonymous mobility traces," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 3, pp. 720–733, 2013.

[51] H. Zang and J. Bolot, "Anonymization of location data does not work: A large-scale measurement study," in *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, ACM, 2011, pp. 145–156.

[52] A. Gervais, R. Shokri, A. Singla, S. Capkun, and V. Lenders, "Quantifying web-search privacy," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2014, pp. 966–977.

[53] A. Juels and R. L. Rivest, "Honeywords: Making password-cracking detectable," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2013, pp. 145–160.

[54] S. T. Peddinti and N. Saxena, "On the privacy of web search based on query obfuscation: A case study of trackmenot," in *International Symposium on Privacy Enhancing Technologies*, Springer, 2010, pp. 19–37.

[55] T.-H. You, W.-C. Peng, and W.-C. Lee, "Protecting moving trajectories with dummies," in *2007 International Conference on Mobile Data Management*, IEEE, 2007, pp. 278–282.

[56] R. Kato, M. Iwata, T. Hara, A. Suzuki, X. Xie, Y. Arase, and S. Nishio, "A dummy-based anonymization method based on user trajectory with pauses," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, ACM, 2012, pp. 249–258.

[57] C. Clifton and T. Tassa, "On syntactic anonymity and differential privacy," *Transactions on Data Privacy*, vol. 6, no. 2, pp. 161–183, 2013.

[58] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *USENIX Security Symposium*, 2014, pp. 17–32.

[59] G. Cormode, "Personal privacy vs population privacy: Learning to attack anonymization," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2011, pp. 1253–1261.

[60] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017, pp. 3–18.

[61] A. Pyrgelis, C. Troncoso, and E. De Cristofaro, "Knock knock, who's there? membership inference on aggregate location data," in *Network and Distributed System Security Symposium (NDSS) 2018*, 2018.

[62] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*, Springer, 2008, pp. 1–19.

[63] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *48th Annual IEEE Symposium on Foundations of Computer Science, (FOCS) 2007*, IEEE, 2007, pp. 94–103.

[64] F. D. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, ACM, 2009, pp. 19–30.

[65] *Analyzing 1.1 billion nyc taxi and uber trips, with a vengeance*, `http://toddwschneider.com/posts/analyzing-1-1-billion-nyc-taxi-and-uber-trips-with-a-vengeance/`, 2018.

[66] G. Acs and C. Castelluccia, "A case study: Privacy preserving release of spatio-temporal density in paris," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2014, pp. 1679–1688.

[67] B. Bamba, L. Liu, P. Pesti, and T. Wang, "Supporting anonymous location queries in mobile environments with privacygrid," in *Proceedings of the 17th international conference on World Wide Web*, ACM, 2008, pp. 237–246.

[68] W. Qardaji, W. Yang, and N. Li, "Differentially private grids for geospatial data," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, IEEE, 2013, pp. 757–768.

[69] H. To, K. Nguyen, and C. Shahabi, "Differentially private publication of location entropy," in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ACM, 2016, p. 35.

[70] W.-Y. Day and N. Li, "Differentially private publishing of high-dimensional data using sensitivity control," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*, ACM, 2015, pp. 451–462.

[71] M. Hay, V. Rastogi, G. Miklau, and D. Suciu, "Boosting the accuracy of differentially private histograms through consistency," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1021–1032, 2010.

[72] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu, "Differentially private spatial decompositions," in *2012 IEEE 28th International Conference on Data Engineering (ICDE)*, IEEE, 2012, pp. 20–31.

[73] 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.

[74] L. Zhang, S. Luo, and H. Xia, "An investigation of intra-urban mobility pattern of taxi passengers," *arXiv preprint arXiv:1612.08378*, 2016.

[75] M. Gaboardi, H.-W. Lim, R. M. Rogers, and S. P. Vadhan, "Differentially private chi-squared hypothesis testing: Goodness of fit and independence testing," in *ICML*, 2016, pp. 2111–2120.

[76] K. Toohey and M. Duckham, "Trajectory similarity measures," *SIGSPATIAL Special*, vol. 7, no. 1, pp. 43–50, 2015.

[77] B.-K. Yi, H. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *14th International Conference on Data Engineering (ICDE 1998)*, IEEE, 1998, pp. 201–208.

[78] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358–386, 2005.

[79] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *ACM SIGMOD Record*, ACM, vol. 29, 2000, pp. 427–438.

[80] G. H. Orair, C. H. Teixeira, W. Meira Jr, Y. Wang, and S. Parthasarathy, "Distance-based outlier detection: Consolidation and renewed bearing," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1469–1480, 2010.

[81] H. Li, L. Xiong, and X. Jiang, "Differentially private synthesization of multi-dimensional data using copula functions," in *International Conference on Extending Database Technology (EDBT)*, NIH Public Access, vol. 2014, 2014, p. 475.

[82] R. Chen, B. C. Fung, S. Y. Philip, and B. C. Desai, "Correlated network data publication via differential privacy," *The VLDB Journal*, vol. 23, no. 4, pp. 653–676, 2014.

[83] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE, 2013, pp. 429–438.

[84] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *Proceedings of the 26th USENIX Security Symposium*, 2017, pp. 729–745.

[85] G. Fanti, V. Pihur, and Ú. Erlingsson, "Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 3, pp. 41–61, 2016.

[86] A. G. Thakurta, A. H. Vyrros, U. S. Vaishampayan, G. Kapoor, J. Freudiger, V. R. Sridhar, and D. Davidson, *Learning new words*, US Patent 9,594,741, 2017.

[87] D. H. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos, "Polonium: Tera-scale graph mining and inference for malware detection," in *Proceedings of the 2011 SIAM International Conference on Data Mining*, SIAM, 2011, pp. 131–142.

[88] N. Karampatziakis, J. W. Stokes, A. Thomas, and M. Marinescu, "Using file relationships in malware classification," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Springer, 2012, pp. 1–20.

[89] A. Tamersoy, K. Roundy, and D. H. Chau, "Guilt by association: Large scale malware detection by mining file-relation graphs," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2014, pp. 1524–1533.

[90] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Computing Surveys (CSUR)*, vol. 44, no. 2, p. 6, 2012.

[91] N. Li, W. Qardaji, D. Su, Y. Wu, and W. Yang, "Membership privacy: A unifying framework for privacy definitions," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & Communications Security*, ACM, 2013, pp. 889–900.

[92] B. Yang, I. Sato, and H. Nakagawa, "Bayesian differential privacy on correlated data," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ACM, 2015, pp. 747–762.

[93] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.

[94] A. Inan, M. E. Gursoy, and Y. Saygin, "Sensitivity analysis for non-interactive differential privacy: Bounds and efficient algorithms," *IEEE Transactions on Dependable and Secure Computing*, 2017.

[95] R. Bassily and A. Smith, "Local, private, efficient protocols for succinct histograms," in *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, ACM, 2015, pp. 127–135.

[96] R. Bassily, U. Stemmer, A. G. Thakurta, *et al.*, "Practical locally private heavy hitters," in *Advances in Neural Information Processing Systems*, 2017, pp. 2288–2296.

[97] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren, "Heavy hitter estimation over set-valued data with local differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2016, pp. 192–203.

[98] K. Chatzikokolakis, M. E. Andres, N. E. Bordenabe, and C. Palamidessi, "Broadening the scope of differential privacy using metrics," in *International Symposium on Privacy Enhancing Technologies (PETS)*, Springer, 2013, pp. 82–102.

[99] M. Hay, C. Li, G. Miklau, and D. Jensen, "Accurate estimation of the degree distribution of private networks," in *9th IEEE International Conference on Data Mining (ICDM)*, IEEE, 2009, pp. 169–178.

[100] Z. Ding, Y. Wang, G. Wang, D. Zhang, and D. Kifer, "Detecting violations of differential privacy," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2018, pp. 475–489.

[101] P. Kairouz, S. Oh, and P. Viswanath, "The composition theorem for differential privacy," *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 4037–4049, 2017.

[102] C. Liu, X. He, T. Chanyaswad, S. Wang, and P. Mittal, "Investigating statistical privacy frameworks from the perspective of hypothesis testing," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 233–254, 2019.

[103] J. Soria-Comas, J. Domingo-Ferrer, D. Sánchez, and S. Martínez, "Enhancing data utility in differential privacy via microaggregation-based k-anonymity," *The VLDB Journal – The International Journal on Very Large Data Bases*, vol. 23, no. 5, pp. 771–794, 2014.

[104] D. Sánchez, M. Batet, D. Isern, and A. Valls, "Ontology-based semantic similarity: A new feature-based approach," *Expert Systems with Applications*, vol. 39, no. 9, pp. 7718–7728, 2012.

[105] T. Wang, N. Li, and S. Jha, "Locally differentially private frequent itemset mining," in *IEEE Symposium on Security and Privacy (SP)*, IEEE, 2018.

[106] S. B. Mehdi, A. K. Tanwani, and M. Farooq, "Imad: In-execution malware analysis and detection," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, ACM, 2009, pp. 1553–1560.

[107] *Pos dataset*, `https://github.com/cpearce/HARM/blob/master/datasets/BMS-POS.csv`, 2015.

[108] D. Chen, S. L. Sain, and K. Guo, "Data mining for the online retail industry: A case study of rfm model-based customer segmentation using data mining," *Journal of Database Marketing & Customer Strategy Management*, vol. 19, no. 3, pp. 197–208, 2012.

[109] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," in *Advances in Neural Information Processing Systems*, 2014, pp. 2879–2887.

[110] A. Smith, A. Thakurta, and J. Upadhyay, "Is interaction necessary for distributed private learning?" In *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017, pp. 58–77.

[111] M. Bun, J. Nelson, and U. Stemmer, "Heavy hitters and the structure of local privacy," in *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, ACM, 2018, pp. 435–447.

[112] G. Cormode, T. Kulkarni, and D. Srivastava, "Marginal release under local differential privacy," in *Proceedings of the 2018 International Conference on Management of Data*, ACM, 2018, pp. 131–146.

[113] Z. Zhang, T. Wang, N. Li, S. He, and J. Chen, "Calm: Consistent adaptive local marginal for marginal release under local differential privacy," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2018, pp. 212–229.

[114] R. Chen, H. Li, A. Qin, S. P. Kasiviswanathan, and H. Jin, "Private spatial data aggregation in the local setting," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, IEEE, 2016, pp. 289–300.

[115] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2017, pp. 425–438.

[116] N. Wang, X. Xiao, Y. Yang, T. D. Hoang, H. Shin, J. Shin, and G. Yu, "Privtrie: Effective frequent term discovery under local differential privacy," in *IEEE International Conference on Data Engineering (ICDE)*, 2018.

[117] T. Wang, N. Li, and S. Jha, "Locally differentially private heavy hitter identification," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[118] Q. Ye, H. Hu, X. Meng, and H. Zheng, "Privkv: Key-value data collection with local differential privacy," in *2019 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2019, pp. 317–331.

[119] B. Avent, A. Korolova, D. Zeber, T. Hovden, and B. Livshits, "BLENDER: Enabling local search with a hybrid differential privacy model," in *26th USENIX Security Symposium (USENIX Security 17)*, Vancouver, BC: USENIX Association, 2017, pp. 747–764, ISBN: 978-1-931971-40-9.

[120] T. Murakami and Y. Kawamoto, "Utility-optimized local differential privacy mechanisms for distribution estimation," in *Proceedings of the 28th USENIX Security Symposium*, 2019, pp. 1877–1894.

[121] G. Cormode, S. Jha, T. Kulkarni, N. Li, D. Srivastava, and T. Wang, "Privacy at scale: Local differential privacy in practice," in *Proceedings of the 2018 International Conference on Management of Data*, ACM, 2018, pp. 1655–1658.

[122] *Apple – learning with privacy at scale*, `https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf`, 2020.

[123] S. Wang, L. Huang, P. Wang, Y. Nie, H. Xu, W. Yang, X.-Y. Li, and C. Qiao, "Mutual information optimally local private discrete distribution estimation," *arXiv preprint arXiv:1607.08025*, 2016.

[124] J. Acharya, Z. Sun, and H. Zhang, "Hadamard response: Estimating distributions privately, efficiently, and with little communication," in *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019, pp. 1120–1129.

[125] X. Gu, M. Li, Y. Cheng, L. Xiong, and Y. Cao, "Pckv: Locally differentially private correlated key-value data collection with optimized utility," in *To appear in 2020 USENIX Security Symposium*, 2020.

[126] J. Jia and N. Z. Gong, "Calibrate: Frequency estimation and heavy hitter identification with local differential privacy via incorporating prior knowledge," in *IEEE International Conference on Computer Communications (INFOCOM)*, IEEE, 2019, pp. 2008–2016.

[127] T. Wang, M. Lopuhaä-Zwakenberg, Z. Li, B. Skoric, and N. Li, "Locally differentially private frequency estimation with consistency," *NDSS*, 2020.

[128] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, 2007, pp. 1–14.

[129] A. Clauset, C. R. Shalizi, and M. E. Newman, "Power-law distributions in empirical data," *SIAM Review*, vol. 51, no. 4, pp. 661–703, 2009.

[130] J. Blocki, A. Datta, and J. Bonneau, "Differentially private password frequency lists," in *NDSS*, vol. 16, 2016, p. 153.

[131] R. Jansen and A. Johnson, "Safely measuring tor," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1553–1567.

[132] *Welcome to tor metrics*, `https://metrics.torproject.org/`, [Online], 2020.

[133] X. Cao, J. Jia, and N. Z. Gong, "Data poisoning attacks to local differential privacy protocols," *arXiv preprint arXiv:1911.02046*, 2019.

[134] A. Cheu, A. Smith, and J. Ullman, "Manipulation attacks in local differential privacy," *arXiv preprint arXiv:1909.09630*, 2019.

[135] M. Lopuhaä-Zwakenberg, B. Škorić, and N. Li, "Information-theoretic metrics for local differential privacy protocols," *arXiv preprint arXiv:1910.07826*, 2019.

[136] X. Ren, C.-M. Yu, W. Yu, S. Yang, X. Yang, J. A. McCann, and S. Y. Philip, "Lopub: High-dimensional crowdsourced data publication with local differential

privacy," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2151–2166, 2018.

[137] G. Cormode, T. Kulkarni, and D. Srivastava, "Answering range queries under local differential privacy," *Proceedings of the VLDB Endowment*, vol. 12, no. 10, pp. 1126–1138, 2019.

[138] T. Wang, B. Ding, J. Zhou, C. Hong, Z. Huang, N. Li, and S. Jha, "Answering multi-dimensional analytical queries under local differential privacy," in *Proceedings of the 2019 International Conference on Management of Data*, ACM, 2019, pp. 159–176.

[139] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. C. Hui, H. Shin, J. Shin, and G. Yu, "Collecting and analyzing multidimensional data with local differential privacy," in *IEEE 35th International Conference on Data Engineering (ICDE)*, IEEE, 2019, pp. 638–649.

[140] M. Joseph, A. Roth, J. Ullman, and B. Waggoner, "Local differential privacy for evolving data," in *Advances in Neural Information Processing Systems*, 2018, pp. 2381–2390.

[141] A. Beimel, K. Nissim, and E. Omri, "Distributed private data analysis: Simultaneously solving how and what," in *Annual International Cryptology Conference*, Springer, 2008, pp. 451–468.

[142] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What can we learn privately?" *SIAM Journal on Computing*, vol. 40, no. 3, pp. 793–826, 2011.

[143] T. Murakami, H. Hino, and J. Sakuma, "Toward distribution estimation under local differential privacy with small samples," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 3, pp. 84–104, 2018.

[144] X. Gu, M. Li, L. Xiong, and Y. Cao, "Providing input-discriminative protection for local differential privacy," *arXiv preprint arXiv:1911.01402*, 2019.

[145] J. Tang, A. Korolova, X. Bai, X. Wang, and X. Wang, "Privacy loss in apple's implementation of differential privacy on macos 10.12," *arXiv preprint arXiv:1709.02753*, 2017.

[146] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, "Amplification by shuffling: From local to central differential privacy via anonymity," in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, 2019, pp. 2468–2479.

[147] *Apple – differential privacy overview*, `https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf`, [Online], 2017.

[148] M. Joseph, J. Mao, S. Neel, and A. Roth, "The role of interactivity in local differential privacy," in *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE, 2019, pp. 94–105.

[149] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[150] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[151] M. E. Gursoy, A. Inan, M. E. Nergiz, and Y. Saygin, "Differentially private nearest neighbor classification," *Data Mining and Knowledge Discovery*, vol. 31, no. 5, pp. 1544–1575, 2017.

[152] S. Truex, L. Liu, M. E. Gursoy, W. Wei, and L. Yu, "Effects of differential privacy and data skewness on membership inference vulnerability," *arXiv preprint arXiv:1911.09777*, 2019.

[153] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2019, pp. 332–349.

[154] O. Suciu, R. Marginean, Y. Kaya, H. Daume III, and T. Dumitras, "When does machine learning fail? generalized transferability for evasion and poisoning attacks," in *27th USENIX Security Symposium*, 2018, pp. 1299–1316.

[155] W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, and Y. Wu, "A framework for evaluating gradient leakage attacks in federated learning," *arXiv preprint arXiv:2004.10397*, 2020.

[156] W. Wei, L. Liu, M. Loper, S. Truex, L. Yu, M. E. Gursoy, and Y. Wu, "Adversarial examples in deep learning: Characterization and divergence," *arXiv preprint arXiv:1807.00051*, 2018.

[157] K.-H. Chow, L. Liu, M. E. Gursoy, S. Truex, W. Wei, and Y. Wu, "Tog: Targeted adversarial objectness gradient attacks on real-time object detection systems," *arXiv preprint arXiv:2004.04320*, 2020.

[158] J. Ba and R. Caruana, "Do deep nets really need to be deep?" In *Advances in Neural Information Processing Systems*, 2014, pp. 2654–2662.

[159] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[160] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017.

VITA

M. Emre Gursoy was born and raised in Istanbul, Turkey. He received the BS degree in Computer Science and Engineering from Sabanci University, Turkey in 2013. He received the MS degree in Computer Science from University of California Los Angeles in 2015. He has been working towards the PhD degree in the School of Computer Science at Georgia Institute of Technology. Emre's research interests include privacy, security, machine learning, big data analytics, and Internet of Things. He has numerous publications in these areas in prestigious conferences and journals, including CCS, S&P, TDSC, TOPS, TMC, TSC, and DMKD. He has received three merit-based scholarships during his undergraduate education, two conference travel/registration grants during his PhD, and a best paper award from EdgeSys. Emre regularly serves as an invited reviewer for IEEE and ACM Transactions journals. He is currently also serving as the Information Director of ACM Transactions on Internet Technology (TOIT).