Policy Driven Data Sharing with Provable Privacy Guarantees

by

Xi He

Department of Computer Science
Duke University

Date: __________________
Approved:

________________________________
Ashwin Machanavajjhala, Supervisor

______________________________
Jun Yang

______________________________
Landon Cox

______________________________
Divesh Srivastava

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science in the Graduate School of Duke University
2018
Policy Driven Data Sharing with Provable Privacy Guarantees

by

Xi He

Department of Computer Science
Duke University

Date: __________________________
Approved: ______________________

Ashwin Machanavajjhala, Supervisor

Jun Yang

Landon Cox

Divesh Srivastava

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science in the Graduate School of Duke University 2018
Abstract

Companies such as Google or Facebook collect a substantial amount of data about their users to provide useful services. The release of these datasets for general use can enable numerous innovative applications and scientific research. However, such data contains sensitive information about users, and simple anonymization techniques have been shown to be ineffective to ensure users’ privacy. These privacy concerns have motivated the development of algorithms that share data with provable privacy guarantees including differential privacy. However, the focus of differentially private algorithm design has been on simplified problem settings. Real world applications must satisfy to complex privacy policies (beyond whether an individual is in or out of the dataset) and adhere to complex constraints, which hinders the deployment of differentially private algorithms.

This dissertation presents a novel policy-driven approach to design provable privacy guarantees for complex settings. This policy-driven approach results in a useful class of provable privacy definitions, named as Blowfish privacy, (a) generalize differential privacy to handle complex privacy preferences and constraints, (b) unify several variants of differential privacy that are used in practice, and (c) allow the creation of new well founded privacy definitions that allow flexible trade-offs between privacy, accuracy, and performance, based on the application’s requirements. The usefulness of this approach are shown in two use cases of data sharing: (1) analyzing location data which involves complex data types and privacy preferences, and (2)
scaling private record linkage which involves secure computations between multiple parties. This work concludes with directions for future privacy research in private data analysis.
## Contents

Abstract iv

List of Tables xi

List of Figures xii

1 Introduction 1

1.1 Problem Setting . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.2 Research Question . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

1.3 Primary Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . 6

1.4 Thesis Organization . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2 Background of Provable Privacy Guarantees 9

2.1 Notations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.2 Desiderata of Provable Privacy Guarantees . . . . . . . . . . . . . . 10

2.3 Differential Privacy . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

2.3.1 Definition of Differential Privacy . . . . . . . . . . . . . . . . . 13

2.3.2 Properties of Differential Privacy . . . . . . . . . . . . . . . . . 14

2.3.3 Algorithms for Differential Privacy . . . . . . . . . . . . . . . . 15

2.4 Pufferfish Privacy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

2.4.1 Definition of Pufferfish Framework . . . . . . . . . . . . . . . . 17

2.4.2 Relation to Differential Privacy . . . . . . . . . . . . . . . . . . 20

2.4.3 Algorithms for Pufferfish Privacy . . . . . . . . . . . . . . . . . 21
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>Hierarchical RSS Model for DPT</td>
<td>85</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Notation and Background</td>
<td>85</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Discretization and Reference Systems</td>
<td>87</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Hierarchical Reference Systems</td>
<td>87</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Hierarchical RSS Model</td>
<td>92</td>
</tr>
<tr>
<td>4.4.5</td>
<td>Sampling Trajectories</td>
<td>93</td>
</tr>
<tr>
<td>4.5</td>
<td>Private Model Learning for DPT</td>
<td>95</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Differential Privacy</td>
<td>96</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Private Model Selection</td>
<td>97</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Learning Transitions</td>
<td>100</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Direction Weighted Sampling</td>
<td>103</td>
</tr>
<tr>
<td>4.6</td>
<td>Empirical Evaluation for DPT</td>
<td>104</td>
</tr>
<tr>
<td>4.6.1</td>
<td>Datasets and Utility Metrics</td>
<td>104</td>
</tr>
<tr>
<td>4.6.2</td>
<td>Design Choices</td>
<td>107</td>
</tr>
<tr>
<td>4.6.3</td>
<td>End-to-End Evaluation</td>
<td>107</td>
</tr>
<tr>
<td>4.6.4</td>
<td>Component Utility Evaluation</td>
<td>109</td>
</tr>
<tr>
<td>4.7</td>
<td>VisDPT: Visual Exploration of DP Trajectories</td>
<td>113</td>
</tr>
<tr>
<td>4.7.1</td>
<td>Privacy Metrics</td>
<td>114</td>
</tr>
<tr>
<td>4.7.2</td>
<td>Utility Metrics</td>
<td>116</td>
</tr>
<tr>
<td>4.7.3</td>
<td>Dashboard and Demo</td>
<td>117</td>
</tr>
<tr>
<td>4.8</td>
<td>Summary</td>
<td>120</td>
</tr>
<tr>
<td>5</td>
<td>Case Study II: Scaling Private Record Linkage</td>
<td>122</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>122</td>
</tr>
<tr>
<td>5.2</td>
<td>Problem Setting &amp; Statement</td>
<td>125</td>
</tr>
<tr>
<td>5.2.1</td>
<td>The Private Record Linkage Problem</td>
<td>126</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Table of Notation ......................................................... 11
4.1 Notation Summary for DPT ............................................. 85
4.2 Datasets Summary for DPT .............................................. 106
4.3 Design choices in DPT .................................................... 107
4.4 Monotonicity of Taxi dataset in Section 4.5.2 ...................... 110
5.1 Summary of Prior Work ................................................. 128
# List of Figures

1.1 Problem Setting ................................................................. 2

3.1 K-Means: Error under Laplace mechanism v.s Blowfish privacy for different discriminative graphs. ................................................................. 43

3.2 Ordered Hierarchical Mechanism. 3.2(a) gives an example of $OH$, where $\theta = 4$. 3.2(b) and 3.2(c) shows privacy-utility trade-offs for range query, using $G^{d,\theta}$ for sensitive information. .................................................. 49

3.3 Policy graph $G_P = (V_P, E_P)$ of databases with three attributes $A_1 = \{a_1, a_2\}$, $A_2 = \{b_1, b_2\}$, and $A_3 = \{c_1, c_2, c_3\}$ subject to count query constraint $Q = \{q_1, q_2, q_3, q_4\}$ and full-domain sensitive information . 57

4.1 Comparing synthetic trajectories generation using seq (at varying granularities) vs DPT ................................................................. 82

4.2 DPT Framework Overview .......................................................... 83

4.3 Map trajectory with different granularity .................................................. 88

4.4 Prefix tree $T_2$ for $\Sigma_{v_2}$ .................................................. 90

4.5 Utility of DPT v.s seq for Taxi, at $\epsilon = 1.0$ .................................................. 109

4.6 Differentially private model selection v.s. other models .................................................. 110

4.7 Differentially private pruning methods .................................................. 111

4.8 Direction-weighted sampling .................................................. 111

4.9 Outline of VisDPT framework .................................................. 113

4.10 Snapshot of VisDPT Dashboard .................................................. 117

4.11 Privacy Game: guess DP model .................................................. 119

4.12 Utility Game: guess $\epsilon$ .................................................. 120

xii
5.1 Neighboring databases for (a) DP, and (b) IND-S2PC for Example 67. 137

5.2 The average log (cost) of LP, APC, PSI+X and non-private matching (np) for the Taxi and AB datasets vs log (data size). LP give lower costs than the baselines PSI+X and APC for all values of $\epsilon = 0.1, 0.4, 1.6$ and $\delta = 10^{-5}$, and scales near linearly. 153

5.3 The average cost with standard deviation of LP protocols with five settings: (1) ‘basic’ – the basic LP protocol in Algorithm 3, (2) ‘basic+SP’ – the basic LP with sort & prune step in S 5.4.3, (3) ‘basic+GMC’ – the basic protocol with greedy match & clean step in Algorithm 5, (4) ‘basic+GMC+S’ – the previous protocol with addition sorting step described in S 5.4.3, and (5) ‘basic+GMC+SP’ – the protocol stops at recall less than full recall. 154

5.4 LP with varying privacy settings plotted over the default Taxi datasets and AB datasets. Each trade-off line between recall and the efficiency $(\text{cost(LP)}/\text{cost(APC)})$ corresponds to the default LP at a privacy setting $(\epsilon, \delta)$. Figure 5.4(a) varies $\epsilon$ and Figure 5.4(b) varies $\delta$. 157

A.1 The average log(cost) vs log(data size) 182
Introduction

But are my footsteps silent? Are they just holes within the sand? Or does another hear them, follow, And find me where I am? – L.S. Hartfield

“What is privacy?” From the earliest philosophical concept on “the oikos” raised by Aristotle for the private sphere of the family to the recent controversial discussion on whether companies like Facebook should have to get user permission to sell or share user data by law, and from the famous “the right to be let alone” concept proposed by Samuel Warren and Louis Brandeis in 1890 to the recent General Data Protection Regulation (GDPR) passed in EU, the understanding and the enforcement of privacy has undergone drastic changes and developments. One of the main driving forces for these changes is the revolution of computing technology – the line between public life and private life is becoming blurred on social networks; and the tension between privacy and utility is growing in the prevalent data-driven applications and scientific research. Despite the many challenges unleashed by the power of computing technology in defining and realizing privacy today, we as computer scientists who play an important role in this domain would like to tackle these challenges by taking
technologies further to make privacy possible in the current and future world.

Computing technology has enabled massive of digital traces of our personal life (e.g. visited places, financial transactions, and health status) to be recorded, stored, and analyzed. For instance, location sensing technology (like GPS) and the rising popularity of location-based applications has resulted in a wealth of data about the movements of individuals and population. This, in turn, has sparked renewed interest in studying large-scale human mobility patterns [13, 89, 52] with applications in city/traffic planning, epidemiology, and location-driven advertising, and has revolutionized the transportation system with smart and fully connected data-driven devices and systems, such as self-driving cars. Similar smart devices and systems that can manage and analyze data at individual levels for useful services in healthcare, financial sector, and education have become increasingly affordable and popular, and they will become inevitably ubiquitous everywhere such as home, workplace and even public spaces in the near future. While the society might enjoy the benefits of living in a smart environment and gain insightful knowledge from these massive and rich collections of data, it is not yet fully understood how much privacy users are losing in exchanging for these direct or indirect benefits.
1.1 Problem Setting

To understand this question, we consider the following problem setting in this thesis. As shown in Figure 1.1, data curators such as Census, AT&T, and Google collect data from individuals with sensitive information. Besides individuals and data curators, there is a third party, data analysts, who are interested in the data for improving services or exploring innovative solutions for existing problems and hence request the sharing and using of these highly sensitive data. Sometimes, the data analyst is the data curator. However, any sharing or using of this data for analysis may leak sensitive information of the individuals. Unlike information disclosure due to the intrusion by hackers due to poor system security and wrong network administration, the theft of physical devices, or dishonest insiders, the sharing and using of this highly detailed personal data usually begins with the good intention of accelerating value and innovation but ends with a privacy crisis.

For instance, in August 2006, logs of search queries collected from users were released by the Internet company America OnLine (AOL) to aid information retrieval research in academia. Though these logs are anonymous, individuals were still identifiable with their detailed private lives exposed to the public [1]. Similarly, it has shown that participants in genomic research studies may be uncovered from the publication of aggregated research [62]. Though these data curators have made a tremendous effort to avoid privacy breaches in this data sharing process, they still failed. Naive anonymization by merely removing identifiers or coarsening the data are not enough to hide a person, mainly due to the high uniqueness of the digital traces of individuals for locations, transactions, and search logs [36, 35, 1] and the availability of the side information. Simply releasing aggregate statistics also lacks privacy guarantees as composing too many accurate statistics about the data can reconstruct the database and hence lead to a drastic decay in privacy guarantee [37].
It is also impossible to rely on manual efforts to examine all possible scenarios over data of unprecedented scales and unseen structures. Hence, what is needed is a strong promise for data curator to share sensitive data of individuals with mathematical rigorous guarantees that avoid the aforementioned vulnerabilities due to side information, naive-anonymization, and lack of composability. We name this promise **provable privacy guarantee**.

This guarantee specifies the behavior of algorithms that run on the data. Algorithms that can be shown provably to satisfy this specification are called privacy mechanisms or mechanisms with a provable privacy guarantee. A data curator just needs to choose a good privacy definition wisely and identify a privacy mechanism $M$ that satisfies this definition so that the sensitive information is protected. What is considered as a good provable privacy definition? In “Designing Statistical Privacy for Your Data” [85], Machanavajjhala and Kifer summarize a set of good practices from the literature which can be used as a guideline for selecting and designing robust provable privacy definitions for different applications.

One of the state-of-the-art provable privacy definitions for sharing sensitive data is **differential privacy**. Differential privacy (DP) [39], first proposed by the academic community, has been very effective in protecting individuals’ sensitive information and is gradually being adopted by the industry (e.g., Google and Apple) and government (e.g., the U.S. Census Bureau). Differential privacy permits the release of aggregate statistics from a dataset while not revealing the presence or absence of a single record in the data. The strength of this privacy notion is that this guarantee is true regardless of the availability of other sources of data about the subject.

Differential privacy is formally defined as follows. A randomized mechanism $M$ with domain $\mathbb{N}^{\mid X\mid}$ satisfies $\epsilon$-differential privacy if for all databases $x, y \in \mathbb{N}^{\mid X\mid}$ such that $\lVert x - y \rVert_1 \leq 1$, the output distributions of $M(x)$ and $M(y)$ are similar, i.e., $\Pr[M(x) \in S] \leq e^\epsilon \Pr[M(y) \in S]$ for all output set $S \subseteq \text{Range}(M)$. Intuitively, this
definition guarantees that the presence or absence of an individual will not affect the final output of the mechanism significantly. The privacy parameter $\epsilon$ is also known as the privacy budget to indicate the privacy loss. The smaller $\epsilon$ is, the stronger the privacy guarantee. Hence, verifying the fulfillment of a privacy promise before releasing or sharing data becomes simply a matter of proving the formalized notions. Therefore, differential privacy obviates the need for manual data review, which is typically tedious and error-prone. Moreover, the privacy loss incurred by the use of multiple differential privacy algorithms can be algebraically reasoned using composition rules avoiding aforementioned vulnerabilities.

1.2 Research Question

Over the past decade, there has been extensive work by the academic community on designing sophisticated differentially private algorithms. However, the focus of differentially private algorithm design has been on simplified problem settings. Real world applications must satisfy to complex privacy policies (beyond whether an individual is in or out of the dataset) and adhere to complex constraints, which hinders the deployment of differentially private algorithms. An issue with this formulation is that the fine-grained privacy preferences of an individual or the different privacy requirements between individuals cannot be expressed directly. For instance, many location-based applications such as OpenPaths [96] and Airbnb [4] state in their policies that user’s location information will only be shared or collected at coarse granularity. Another example is that: while people are generally willing to keep their medical status private, those with severe health conditions may be more open to sharing data about themselves; hoping that this may lead to medical innovations that improve their health condition. Each application has its own unique privacy requirements, hence we would like to have customizable privacy definitions with provable desiderata like differential privacy for sharing data accurately and efficiently.
Therefore, what is needed is a principled approach to share sensitive data that (a) generalizes differential privacy to handle complex privacy preferences and constraints, (b) unifies several variants of differential privacy that are used in practice, and (c) allows the creation of new well founded privacy definitions that allow flexible trade-offs between privacy, accuracy, and performance, based on the application’s requirements.

1.3 Primary Contributions

In this section, we summarize the primary contributions of this work.

**Policy-driven privacy class for data sharing.** Inspired by a semantic privacy framework known as Pufferfish [75], we proposed a class of definitions called Blowfish privacy [59] to encode the privacy requirements of different applications using privacy policies that are represented by graphs. Blowfish privacy also provides the same properties as differential privacy such as composition rules, and has inspired new privacy definitions such as Employer-Employee (ER-EE) privacy [54] used for the ER-EE data by the U.S. Census. Moreover, many seemingly different privacy notions for location data or graph data can be described as instances with different policies under Blowfish privacy for comparison and analysis [83, 84, 82]. It also has been shown by Haney et al. [55] that mechanisms that achieve a new Blowfish privacy instance can be easily built from existing mechanisms for differential privacy instead of starting from scratch, based on a transformational equivalence between differential privacy and Blowfish.

Most importantly, mechanisms designed under Blowfish privacy can achieve higher utility than differential privacy when privacy policies indicate that not all properties of an individual need to be kept secret as shown by us in typical data analysis tasks like answering range queries on data and k-means clustering. We further illustrate
the rich set of tuning knobs provided by this policy-driven approach for the trade-offs between privacy, accuracy, and efficiency with two complex settings in data sharing (1) analyzing location data and (2) scaling private record linkage, as described next.

**Analyzing location data with provable privacy guarantees.** The first use case of Blowfish privacy considers a complex data type, location data. There are many seemingly different privacy notions for location data in the literature, such as user-DP, \( w \)-event privacy, event-DP, geo-indistinguishability, and \( \delta \)-location set based DP. All these location privacy can be described as instances with different policies under Blowfish privacy for comparison and analysis [83, 84, 82].

Under the strongest privacy guarantee, user-DP, we designed and built DP Trajectory Synthesis (DPT) [58] to enable privacy-preserving analysis on trajectory data of the users. This is a challenging problem as an individual’s raw trajectories are typically very detailed and unique; it is highly unlikely for two different users to have similar trajectories. DPT is the first scalable end-to-end system that generates synthetic trajectories that remain faithful to the behavior of the original trajectories while providing provable privacy guarantee on any single user.

Based on DPT, we also designed a system, called VisDPT [61], that provides an interactive visualization of (i) a rich set of utility queries that are commonly used in mobility data analysis and (ii) a rich class of privacy metrics to allow comprehensive examination of privacy risks of both raw and synthetic trajectories without writing repeated low-level queries. This system can help data curators understand the impact on utility brought by the privacy-preserving mechanisms, and received the best demo award in 2016 VLDB Conference.

**Scaling private record linkage with provable privacy guarantees.** The second use case explored the privacy needs under complex settings involving secure computations where data analysis requires the integration of datasets held by differ-
ent parties that do not trust one another. In this setting, We considered a common data integration task, private record linkage (PRL), which identifies pairs of matching records from datasets held by different parties while revealing nothing else to the opposite parties. For the first time, we showed that cross-composing different provable privacy guarantees such as differential privacy and secure multi-party computations are not always trivial, especially when their goals are in conflict [60]. Prior work that adopted a hybrid privacy approach to speed up PRL protocols failed to provide any end-to-end privacy guarantees. In light of this deficiency, we proposed a novel privacy model, called output constrained differential privacy [60], a weaker but provable end-to-end privacy notion with a policy that encodes the different protections over matching and non-matching records, and developed more efficient and highly accurate protocols for this data integration process.

1.4 Thesis Organization

The remainder of this dissertation proceeds as follows. Chapter 2 provides notations and the background including the desiderata and the state-of-the-art of provable privacy guarantees. Chapter 3 presents Blowfish privacy, the general policy-driven approach to design provable privacy guarantees and its applications. Chapter 4 and Chapter 5 further illustrate the usefulness of this policy-driven approach in two complex settings. Chapter 6 concludes with a summary and future directions.
Preparing data for public release requires significant attention to fundamental principles of privacy. [85] – Machanavajjhala and Kifer

A commonly used privacy notion for releasing data to the public is $k$-anonymity [117]. Formally, an algorithm satisfies $k$-anonymity if it outputs only $k$-anonymous tables, where a table is $k$-anonymous if every record in it has the same quasi-identifier values as $k - 1$ other records. Just like many other privacy notions in the literature, these algorithms were designed to defend against one type of attack and soon they were found vulnerable towards others [123, 36] and hence were amended into new definitions which can be broken again. How can this vicious cycle be stopped?

Many best practices for designing robust privacy definitions have been recognized in the last decade and are summarized by Machanavajjhala and Kifer in “Designing Statistical Privacy for Your Data” [85]. These practices are also the desired properties of provable privacy guarantees.

In this chapter, we first describe the notations in Section 2.1 and summarize these best practices in Section 2.2. Then we illustrate the desiderata of provable privacy
guarantees with existing provable privacy guarantees including differential privacy and Pufferfish privacy framework in Section 2.3 and Section 2.4 respectively.

2.1 Notations

Let $\mathcal{I}$ be the set of all possible database instances, and let each database instance be a collection of record values/tuples. The variable $r$ is used to represent a record and is associated with an individual $h_i$ in the population $\mathcal{H}$. Let $\mathcal{T}$ be the domain for the record variable $r$, and a tuple $t \in \mathcal{T}$ be a value taken by a record. The data curator will choose a privacy definition and a privacy mechanism (algorithm) $M$ that satisfies that privacy definition. Then the data curator will apply $M$ to the data to obtain a sanitized output $\omega = M(\mathsf{Data})$, where $\mathsf{Data}$ is the random variable representing the true database instance owned by the data curator which is unknown to the adversary. We use $\mathsf{records}(\mathsf{Data})$ to denote the set of records in $\mathsf{Data}$ and $t(\mathsf{Data})$ to denote the record values (tuples). These notations and the key notations from the rest of this chapter is summarized in Table 2.1.

2.2 Desiderata of Provable Privacy Guarantees

The provable guarantee of a robust privacy definition specifies the behavior of the mechanism. Algorithms that can be shown provably to satisfy this specification are called privacy mechanisms or mechanisms with a provable privacy guarantee. A good provable privacy definition should consider the following desired properties [85]:

- *Security without Obscurity*: The privacy guarantee of a mechanism $M$ is not achieved via obscurity (by hiding the details of the mechanism). This desiderata first ensures a strong privacy even against smart adversaries who might know these details of $M$, except the specific random bits used in $M$; and second allows the output of $M$ to be useful for analysis and research.
Table 2.1: Table of Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{I}$</td>
<td>The set of possible database instances.</td>
</tr>
<tr>
<td>$D$</td>
<td>A database instance belonging to $\mathcal{I}$.</td>
</tr>
<tr>
<td>$\mathcal{T}$</td>
<td>The domain of tuples/record values in the database.</td>
</tr>
<tr>
<td>$t$</td>
<td>A tuple/record value, a value in $\mathcal{T}$.</td>
</tr>
<tr>
<td>$\mathcal{H}$</td>
<td>The set of all individuals. $\mathcal{H} = {h_1, h_2, \ldots}$</td>
</tr>
<tr>
<td>$r_i$</td>
<td>The record associated with individual $h_i$</td>
</tr>
<tr>
<td>$\mathbf{Data}$</td>
<td>A random variable representing the true dataset (which is unknown to the adversary).</td>
</tr>
<tr>
<td>tuples($\mathbf{Data}$)</td>
<td>The tuples in the database (record values without explicit reference to the identities of individuals).</td>
</tr>
<tr>
<td>records($\mathbf{Data}$)</td>
<td>The identities and record values of individuals in the data.</td>
</tr>
<tr>
<td>$M$</td>
<td>A privacy mechanism: a deterministic or randomized algorithm (often used in the context of a privacy definition).</td>
</tr>
<tr>
<td>$N_{np}$</td>
<td>The set of neighboring databases for unbounded differential privacy</td>
</tr>
<tr>
<td>$N_{np}^n$</td>
<td>The set of neighboring databases for bounded differential privacy</td>
</tr>
<tr>
<td>$\sigma_i$</td>
<td>$r_i \in \text{records}(\mathbf{Data})$: The statement that the record $r_i$ belonging to individual $h_i$ is in the data.</td>
</tr>
<tr>
<td>$\mathcal{S}$</td>
<td>Set of potential secrets. Revealing $s$ or $\neg s$ may be harmful if $s \in \mathcal{S}$</td>
</tr>
<tr>
<td>$\mathcal{S}_{\text{pairs}}$</td>
<td>Discriminative pairs, $\mathcal{S}_{\text{pairs}} \subseteq \mathcal{S} \times \mathcal{S}$</td>
</tr>
<tr>
<td>$\mathcal{D}$</td>
<td>The set of evolution scenarios: a conservative collection of plausible data generating distributions.</td>
</tr>
<tr>
<td>$\theta$</td>
<td>A probability distribution. The probability, under $\theta$, that the data equals $D_i$ is $\Pr(\mathbf{Data} = D_i</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>The spatial domain with distance metric $d(\cdot)$</td>
</tr>
</tbody>
</table>

- **Post-processing:** The provable privacy guarantee should provide a future proof for the mechanisms. If $M$ is such a mechanism that satisfies a privacy guarantee, running another algorithm $A$ that can be applied to the output of $M$ should not change the privacy guarantee of the mechanism $M$. This is known as post-processing. For example, $M$ creates a synthetic data from its inputs, and then $A$ builds a learning model on the synthetic data, which should result in no more worse privacy loss than $M$. With this post-processing property, after releasing the synthetic data using mechanism $M$, the data curator does
not have to worry about what other mechanisms can be used by adversary to break this guarantee.

- **Composition**: When a sequence of privacy-preserving mechanisms that satisfy the same privacy definition are applied to a dataset, it is desired to have a graceful degradation of privacy protection.

Unfortunately, $k$-anonymity as a popular privacy guarantee is not closed under post-processing. Some mechanisms that satisfy $k$-anonymity are susceptible to an attack called minimality attack [123]. This attack post-processes the $k$-anonymous output based on the knowledge of the mechanism to recover records with unique quasi-identifier values. This makes the dataset released under $k$-anonymity vulnerable again to linkage attacks, a type of attacks which $k$-anonymity mainly is designed for. Moreover, $k$-anonymous releases do not guarantee $k$-anonymity under composition. It is possible to learn the sensitive information of an individual exactly by combining two $k$-anonymous releases [85]. Hence, $k$-anonymity fails the aforementioned properties and is not an optimal privacy choice in practice. Next, we will present several examples that can achieve these properties.

2.3 Differential Privacy

Differential privacy was first introduced in 2006 [39] as a promise to ensure the private information of an individual while allowing the learning of useful information about a population. This promise has quickly arisen as the state of the art privacy definition with a rich class of mechanisms satisfying it. Unlike anonymization, this privacy guarantee specifies a provable property of the privacy-preserving mechanisms and satisfies many of the privacy desiderata discussed in the previous section.
2.3.1 Definition of Differential Privacy

An algorithm satisfies differential privacy if adding, removing or changing a record in terms of the input does not significantly alter the output of the algorithm. More formally:

**Definition 1** (Differential Privacy [39, 40]). *Given a privacy parameter $\epsilon > 0$, a randomized algorithm $M$ satisfies $\epsilon$-differential privacy if for any outputs $\omega \in \text{range}(M)$ and all pairs of datasets $D$ and $D'$ in $\mathcal{I}$ that differ in one record (i.e. $D$ can be derived from $D'$ by either adding or deleting one record), the following holds:

$$\Pr[M(D) = \omega] \leq \exp(\epsilon) \Pr[M(D') = \omega],$$

(2.1)

where the probability only depends on the randomness in $M$.*

In this definition, the number of individuals in the database are unknown, and hence this definition is also known as *unbounded* differential privacy. When the number of the individuals is known in the database, the neighboring databases are defined as a pair of databases that differ in the value of only one individual's record, and the remaining individuals all have the same record values. This is also known as *bounded* differential privacy or *indistinguishability*. We represent the set of neighbors for unbounded differential privacy by $N_{dp}$, and the set of neighbors for unbounded differential privacy by $N_{dp}^n$, where $n$ is the number of records in the database.

Intuitively, changing an individual’s record value to the database for bounded differential privacy (or adding or removing an individual’s record for unbounded differential privacy) has little impact on the distribution of the output of a randomized algorithm. The parameter $\epsilon$ is usually known as the privacy budget. When $\epsilon$ is small, the output distributions of $M$ are similar regardless of whether an individual’s record value was used in the computation. The definition only applies to randomized
algorithms, since it is easy to see that deterministic algorithms cannot satisfy this definition.

2.3.2 Properties of Differential Privacy

An important property of differentially private algorithms is that their composition also satisfies differential privacy.

**Theorem 2** (Sequential Composition [39, 40]). Let \( D \in \mathcal{I} \) be an input database. Let \( M_1(\cdot) \) and \( M_2(\cdot, \cdot) \) be algorithms with independent sources of randomness that satisfy \( \epsilon_1 \)- and \( \epsilon_2 \)-differential privacy, resp. Then an algorithm that outputs both \( M_1(D) = \omega_1 \) and \( M_2(\omega_1, D) = \omega_2 \) satisfies \((\epsilon_1 + \epsilon_2)\)-differential privacy.

If the second algorithm \( M_2 \) does not access the raw data \( (\epsilon_2 = 0) \), but only applies on the output of the first algorithm, the provable privacy guarantee of the first algorithm after applying \( M_2 \) is unchanging. Formally,

**Theorem 3** (Post-Processing [39, 40]). Let \( D \in \mathcal{I} \) be an input database. Let \( M_1(\cdot) \) be an algorithm that satisfies \( \epsilon \)-differential privacy. Then if an algorithm \( M_2 \) is applied to the output of \( M_1(\cdot) \), then the overall mechanism \( M_2 \circ M_1(\cdot) \) also satisfies \( \epsilon \)-differential privacy.

All steps in the post-processing algorithm do not access the raw data, and hence they do not affect the privacy analysis. While it seems intuitive that postprocessing the output of a privacy algorithm should not result in additional privacy loss, there are some privacy metrics, like k-anonymity, that do not satisfy the postprocessing theorem.

**Theorem 4** (Parallel Composition [39, 40]). Let \( D \in \mathcal{I} \) be an input database. Let \( \mathcal{H}_1, \ldots, \mathcal{H}_p \) be disjoint subsets of individuals \( \mathcal{H} ; D \cap \mathcal{H}_i \) denotes the dataset restricted to the individuals in \( \mathcal{H}_i \). Let \( M_i \) be mechanisms that each ensure \( \epsilon_i \)-differential privacy. Then the sequence of \( M_i(D \cap \mathcal{H}_i) \) ensures \((\max_i \epsilon_i)\)-differential privacy.
These composition properties are very useful in proving the privacy guarantees of complex algorithms. Sequential composition theorem allows us to decompose an algorithm into a few sequential components, and then analyze each component separately. The parallel composition theorem enables us to analyze an algorithm that works on disjoint partitions of the data. The postprocessing theorem ensures that we only need to analyze the steps in the algorithm that actually touch the private database. Then the overall privacy guarantee of an algorithm over the entire database can be established with the two theorems above. These composition theorems are also very important, as they can address the impossibility result by Dinur and Nissim [123] that a database of size $n$ can be reconstructed with high accuracy from the answers to $n \log(n)^2$ statistical queries even if each answer is perturbed with up to $o(\sqrt{n})$ error. Differential privacy conforms to this negative result as the privacy guarantee degrades as the number of sequential accesses to the data increases (according to the sequential composition result). Nevertheless, unlike k-anonymity, the privacy degradation is gradual and can be theoretically quantified. We would like to note that the sequential composition theorem holds (a) in the worst case and (b) even when the next query or differential private mechanism in the sequence is chosen adversarially and adaptively based on the answers to the previous queries. There are more sophisticated but advanced composition theorems [42]. When the queries are not adaptively chosen, tighter bounds on the privacy loss are known [42, 56, 79].

There are other privacy axioms which differential privacy and other good provable privacy notions can satisfy [85]. These properties make differential privacy an appealing choice for many privacy-aware applications and research.

2.3.3 Algorithms for Differential Privacy

Laplace mechanism is an important building block for designing differentially private algorithms.
**Definition 5** (The Laplace Mechanism). Given any function \( f : \mathcal{I} \to \mathbb{R}^k \), the Laplace mechanism is defined as:

\[
M_L(D, f(\cdot), \epsilon) = f(D) + (\eta_1, \ldots, \eta_k),
\]

where \( \eta_i \) are i.i.d random variables drawn from \( \text{Lap}(\Delta f/\epsilon) \), and \( \Delta f \) is the \( l_1 \)-sensitivity of the query \( f \).

The \( l_1 \)-sensitivity of the query \( f \) is a key concept for the Laplace mechanism, defined as the maximum difference in the query output between any two neighboring databases. Formally,

**Definition 6** (\( l_1 \)-sensitivity). The \( l_1 \)-sensitivity of a function \( f : \mathcal{I} \to \mathbb{R}^k \) is

\[
\Delta f = \max_{D, D' \in \mathcal{I}, (D, D') \in \mathcal{N}_{dp} \text{ or } \mathcal{N}_{dp}' } || f(D) - f(D') ||_1,
\]

where \( ||x - y||_1 \) denotes the \( l_1 \) norm of the difference between vectors \( x \) and \( y \), and is defined as \( \sum_i |x[i] - y[i]| \).

Intuitively, the Laplace mechanism adds noise that is large enough to hide the maximum difference in the query output between any two neighboring databases such that adversaries cannot distinguish the neighboring databases from the noisy output. Besides Laplace mechanism, there are many other algorithmic building blocks for differential privacy. Readers may refer to [42] for more details.

### 2.4 Pufferfish Privacy

Pufferfish privacy [75] is a framework that generalizes differential privacy, helps understand the privacy semantics underlying privacy definitions, and create new privacy definitions customized to the requirements of an application. In section 2.4.1, we present how the Pufferfish framework defines privacy in terms of secrets and
(the prior knowledge available to) adversaries, rather than neighboring databases. In Section 2.4.2, we show that differential privacy is an instantiation of the Pufferfish framework. Finally, Section 2.4.3 describes an algorithm for general Pufferfish privacy definitions.

### 2.4.1 Definition of Pufferfish Framework

Pufferfish framework requires domain expert to specify three components: (1) a set of potential secrets \( S \), (2) a set of discriminative pairs \( S_{\text{pairs}} \subseteq S \times S \), and (3) a collection of data evolution scenarios \( D \). The specification of these three components in this framework gives a rich class of privacy definitions.

- **The set of potential secrets** \( S \) represents the information that data curator would like to protect. A secret can be specified as a statement such as “Bob is at location \( l \in T \)”, “Bob is not at location \( l \in T \)”. In general, a domain expert should add a statement \( s \) to the potential secrets \( S \) if either the claim that \( s \) is true or the claim that \( s \) is false can be harmful. The resulted \( S \) forms a domain for the discriminative pairs, a subset of \( S \times S \).

- **The set of discriminative pairs** \( S_{\text{pairs}} \), is a subset of \( S \times S \). The role of \( S_{\text{pairs}} \) is to tell how to protect the potential secrets \( S \). For any discriminative pair \( (s_i, s_j) \in S_{\text{pairs}} \), we would like to guarantee that adversaries are unable to distinguish between the case where \( s_i \) is true of the actual data and the case where \( s_j \) is true of the actual data. For this reason, \( s_i \) and \( s_j \) must be mutually exclusive, but not necessarily exhaustive (it could be the case that neither is true). One example of a discriminative pair is (“Bob is at location \( l_1 \)”, “Bob is at location \( l_2 \)”), where \( l_1 \neq l_2 \), or (“Bob is at location \( l_1 \)”, “Bob is not at location \( l_1 \)”), where \( l_1, l_2 \in T \). The set of changes for neighboring databases shown in Section 4.2 are examples for the set of discriminative pairs.
This specification allows highly customizable privacy guarantees. For instance, many location-based applications such as OpenPaths [96] and Airbnb [4] state in their policies that user’s location information will only be shared or collected at coarse granularity. This property can be specified by pairs of secrets, such as (“Bob is at location \( l_1 \in \mathcal{T} \)” , “Bob is at location \( l_2 \in \mathcal{T} \)”), where \( l_1 \) is 21 miles away from \( l_2 \). Or if users are fine with releasing their location at city-level, but not at any street level within a city, this privacy preference can be expressed via a set of discriminative pairs \( S_{\text{pairs}} \) that exclude pairs of secrets like (“Bob is at Durham”, “Bob is at New York”), but includes pairs of secrets with nearby places, such as (“Bob is at a cafe in Durham”, “Bob is at home in Durham”).

- **The evolution scenarios** \( \mathcal{D} \) can be viewed as a set of conservative assumptions how the data evolved (or were generated) and about knowledge of potential adversaries. Note that assumptions are absolutely necessary – privacy definitions that can provide privacy guarantees without making any assumptions provide little utility beyond the default approach of releasing nothing at all [41, 72]. In order to release useful information about the database, the domain expert should be able to identify a reasonable set of assumptions. In many cases, they already do this informally [107]. Formally, \( \mathcal{D} \) is represented as a set of probability distributions over \( \mathcal{I} \) (the possible database instances). Each probability distribution \( \theta \in \mathcal{D} \) corresponds to an adversary that we want to protect against and represents that adversary’s belief in how the data were generated (incorporating any background knowledge and side information). For \( D \in \mathcal{I} \), we use the notation \( \Pr(\mathbf{Data} = D | \theta) \) to represent the probability, under \( \theta \), that the true database is \( D \). Below we give some examples of possible choices of \( \mathcal{D} \) and their interpretations.

**Example 7** (No assumptions). \( \mathcal{D} \) can consist of all possible probability distri-
butions over database instances (i.e. including those with arbitrary correlations between records). This corresponds to making no assumptions.

**Example 8** (Independent individuals but Markov model-based events). Several work [121, 125] consider that individuals are independent, but the events per individual are correlated by Markov model. The individuals in the database are independent of each other, that is, \( \mathcal{D} \) consists of all \( \theta \) for which

\[
\Pr[\text{Data} = \{r_1, \ldots, r_n|\theta\}] = f_1(r_1) \times f_2(r_2) \times \ldots \times f_n(r_n) \tag{2.4}
\]

for arbitrary \( f_1, f_2, \ldots, f_n \). The correlation within the events of an individual is modeled by a transition matrix or a class of transition matrices \( P_\theta \) for \( \mathcal{D} \), where each \( (l_1, l_2) \)th entry of this matrix specifies the probability an individual \( h_i \) being at location \( l_2 \) at time stamp \( j \) given the previous \( (j - 1) \)th event, i.e. \( \Pr(r_i[j] = l_2|r_{j-1} = l_1) \).

Readers may refer to [75] for more examples of data evolution scenarios.

To use the Pufferfish framework, the domain expert simply does what he or she does best, and is no longer required to be a privacy expert. After specifying the assumptions explicitly, the corresponding Pufferfish privacy instance is formally stated as follows.

**Definition 9** (Pufferfish Privacy [75]). Given a set of potential secrets \( \mathcal{S} \), a set of discriminative pairs \( S_{\text{pairs}} \), a set of data evolution scenarios \( \mathcal{D} \), and a privacy parameter \( \epsilon > 0 \), a potentially randomized algorithm \( M \) satisfies \( \epsilon \)-Pufferfish \( (\mathcal{S}, S_{\text{pairs}}, \mathcal{D}) \) privacy if (i) for all possible outputs \( \omega \in \text{range}(M) \), (ii) for all pairs \( (s_i, s_j) \in S_{\text{pairs}} \) of potential secrets, (iii) for all distributions \( \theta \in \mathcal{D} \) for which \( \Pr(s_i|\theta) \neq 0 \) and \( \Pr(s_j|\theta) \neq 0 \), the following holds:

\[
\Pr[M(\text{Data}) = \omega|s_i, \theta] \leq \epsilon \Pr[M(\text{Data}) = \omega|s_j, \theta] \tag{2.5}
\]

\[
\Pr[M(\text{Data}) = \omega|s_j, \theta] \leq \epsilon \Pr[M(\text{Data}) = \omega|s_i, \theta], \tag{2.6}
\]
where \( \text{Data} \) is a random variable representing the true dataset (which is unknown to the adversaries).

2.4.2 Relation to Differential Privacy

Recall that the definition of differential privacy is based on neighboring databases by changing an individual’s record. This definition is a condition of a randomized algorithm – the output of the randomized algorithm is insensitive to the change of an individual’s record to the database. In this definition, there is no mention or assumption of data evolution scenarios known by the adversaries. In this section, we would like to show how to understand and analyze differential privacy in the framework of Pufferfish.

Consider the following specifications. Let \( \mathcal{H} = \{h_1, h_2, \ldots, h_N\} \) be the set of all individuals in a population of size \( N \). Define \( \sigma_i \) be the statement \( r_i \in \text{records}(\text{Data}) \) (i.e. “records \( r_i \) belonging to individual \( h_i \) is in the data”), and let \( \sigma_{(i,t)} \) be the statement \( r_i \in \text{records}(\text{Data}) \land r_i = t \) (i.e. “record \( r_i \) belonging to individual \( h_i \) has value \( t \) and is in the data”). Let the set of secrets and the set of discriminative secret pairs be specified respectively as

\[
\mathbb{S} = \{\sigma_{i,t} : h_i \in \mathcal{H}, t \in \mathcal{T}\} \cup \{\neg \sigma_i : h_i \in \mathcal{H}\} \quad (2.7)
\]

\[
S_{\text{pairs}} = \{(\sigma_{i,t}, \neg \sigma_i) : h_i \in \mathcal{H}, t \in \mathcal{T}\} \quad (2.8)
\]

This specification of secret pairs aim to prevent an adversary from distinguishing whether the record \( r_i \) associating with \( h_i \) is in the data and has the value \( t \) v.s. the record about individual \( h_i \) is not in the data, for any individual \( h_i \) in the population \( \mathcal{H} \), and any possible tuple value \( t \in \mathcal{T} \). Consider the data evolution scenario \( \mathcal{D} \) where all individuals are independent (including their presence/absence in the data and their tuple values if present in the data). This distribution can be specified as

\[
\Pr[\text{Data}|\theta] = \prod_{r_i \in \text{records}(\text{Data})} f_i(r_i) \Pr[\sigma_i] \prod_{r_i \notin \text{records}(\text{Data})} (1 - \Pr[\sigma_i]), \quad (2.9)
\]
where \( f_i(r_i) \) is the distribution for the value taken by record \( r_i \) of an individual \( h_i \), and \( \Pr[\sigma_i] \) is the probability of the record of an individual being in the data.

**Theorem 10 ([75]).** With the choices of \( S \) and \( S_{\text{pairs}} \) defined in Eqns. 2.7 and 2.8, and the set of data evolution scenarios \( \mathbb{D} \) as specified in Eqn. 2.9, unbounded \( \epsilon \)-differential privacy is equivalent to \( \epsilon \)-Pufferfish \( (S, S_{\text{pairs}}, \mathbb{D}) \).

2.4.3 Algorithms for Pufferfish Privacy

In the Laplace mechanism for differential privacy, the noise to the query output is proportional to the \( l_1 \)-sensitivity defined in Eqn. 2.3, which is the worst case distance between \( f(D_1) \) and \( f(D_2) \) where \( D_1 \) and \( D_2 \) are neighboring databases that differ in the value of a single individual. The corresponding concept for a pair of neighboring databases in Pufferfish framework are all possible pairs of databases that differ in a given pair of discriminative secrets \( (s_i, s_j) \in S_{\text{pairs}} \). Hence, Wang et al. [121] consider the two distributions given a secret pair \( (s_i, s_j) \in S_{\text{pairs}} \), i.e.

\[
\mu_{i,\theta} = \Pr(f(D) = \cdot|s_i, \theta) \\
\mu_{j,\theta} = \Pr(f(D) = \cdot|s_j, \theta)
\]

and apply Wasserstein distance to measure the relevant distance between distributions \( \mu_{i,\theta} \) and \( \mu_{j,\theta} \). Wasserstein distance is formally defined as below.

**Definition 11** (\( \infty \)-Wasserstein Distance [121]). Let \( (\mathcal{X}, d) \) be a Radon space, and \( \mu, \nu \) be two probability distribution on \( \mathcal{X} \) with finite \( p \)-th moment. The \( \infty \)-Wasserstein distance between \( \mu, \nu \) with \( d(x, y) = |x - y| \):

\[
W_{\infty}(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu, \nu)} \max_{(x, y) \in A} |x - y|,
\]

where \( A = \{(x, y) | \gamma(x, y) \neq 0 \} \) is the support of \( \gamma \), and \( \Gamma(\mu, \nu) \) is the set of all couplings \( \gamma \) over \( \mu \) and \( \nu \).
Intuitively, $\gamma \in \Gamma(\mu, \nu)$ is a way to shift probability mass between $\mu$ and $\nu$, and $W_\infty(\mu, \nu)$ can be interpreted as the maximum “distance” that any probability mass moves while transforming $\mu$ to $\nu$ in the most optimal way. Wang et al. [121] proposed a general mechanism for Pufferfish framework Pufferfish framework $(S, S_{pairs}, D)$ with privacy budget $\epsilon$ and query $f$. This mechanism first computes the generalized sensitivity, defined as

$$\Delta W_\infty(f, S, S_{pairs}, D) = \max_{(s_i, s_j) \in S_{pairs}} \max_{\theta \in D} \max_{\Pr(s_i(\theta) \neq 0, \Pr(s_j(\theta) \neq 0} W_\infty(\mu_{i, \theta}, \mu_{j, \theta})$$

This generalized sensitivity iterates over all possible secret pairs in $S_{pairs}$ and data evolution scenarios $\theta \in D$ and computes the inf-Wasserstein distance between the distributions given each secret and $theta$. Similar to Laplace mechanism, Wasserstein mechanism adds noise that is proportional to the general sensitivity of the given function can guarantee $\epsilon$-Pufferfish privacy. Here is the formal statement.

**Definition 12** (Wasserstein Mechanism). Given any function $f : I \rightarrow \mathbb{R}^k$, the Wasserstein mechanism is defined as :

$$M_W(D, f(\cdot), S, S_{pairs}, D, \epsilon) = f(D) + (\eta_1, \ldots, \eta_k),$$

where $\eta_i$ are i.i.d random variables drawn from $Lap(\Delta W_\infty(f, S, S_{pairs}, D)/\epsilon)$.

Wang et al. [121] showed that Wasserstein mechanism provides $\epsilon$-Pufferfish privacy in the framework $(S, S_{pairs}, D)$. This mechanism is also shown with a smaller sensitivity parameter than the $l_1$-sensitivity of query $f$ under group differential privacy if $f$ is $L$-Lipschitz query [121], and hence can result in higher accuracy for query $f$.

Though Pufferfish framework provides a wide variety of privacy definitions, the domain experts are required to specify adversarial knowledge as sets of complex probability distributions, and this framework does not always result in composable
privacy definitions [75]. Hence, we introduce a simple but useful class of privacy
definitions named Blowfish privacy [59] that addresses limitations of the general
framework of Pufferfish. We will present the definition, properties, and applications
of Blowfish privacy in the following chapters.
3

Blowfish Privacy: Designing Privacy with Policies

3.1 Introduction

With the increasing popularity of “big-data” applications which collect, analyze and disseminate individual level information in literally every aspect of our life, ensuring that these applications do not breach the privacy of individuals is an important problem. The last decade has seen the development of a number of privacy definitions and mechanisms that trade-off the privacy of individuals in these databases for the utility (or accuracy) of data analysis (see [24] for a survey). Differential privacy [39] has emerged as a gold standard not only because it is not susceptible to attacks that other definition can’t tolerate, but also since it provides a simple knob, namely $\epsilon$, for trading off privacy for utility.

While $\epsilon$ is intuitive, it does not sufficiently capture the diversity in the privacy-utility trade-off space. For instance, recent work has shown two seemingly contradictory results. In certain applications (e.g., social recommendations [88]) differential privacy is too strong and does not permit sufficient utility. Next, when data are correlated (e.g., when constraints are known publicly about the data, or in social
network data) differentially private mechanisms may not limit the ability of an attacker to learn sensitive information [73]. Subsequently, Kifer and Machanavajjhala [74] proposed a semantic privacy framework, called Pufferfish, which helps clarify assumptions underlying privacy definitions – specifically, the information that is being kept secret, and the adversary’s background knowledge. They showed that differential privacy is equivalent to a specific instantiation of the Pufferfish framework, where (a) every property about an individual’s record in the data is kept secret, and (b) the adversary assumes that every individual is independent of the rest of the individuals in the data (no correlations). We believe that these shortcomings severely limit the applicability of differential privacy to real world scenarios that either require high utility, or deal with correlated data.

Inspired by Pufferfish, we seek to better explore the trade-off between privacy and utility by providing a richer set of “tuning knobs”. We explore a class of definitions called Blowfish privacy. In addition to $\epsilon$, which controls the amount of information disclosed, Blowfish definitions take as input a privacy policy that specifies two more parameters – which information must be kept secret about individuals, and what constraints may be known publicly about the data. By extending differential privacy using these policies, we can hope to develop mechanisms that permit more utility since not all properties of an individual need to be kept secret. Moreover, we also can limit adversarial attacks that leverage correlations due to publicly known constraints.

In this chapter, we introduce and formalize sensitive information specifications, constraints, policies and Blowfish privacy. We consider a number of realistic examples of sensitive information specification, and focus on count constraints. We show how to adapt well known differential privacy mechanisms to satisfy Blowfish privacy, and using the example of k-means clustering illustrate the gains in accuracy for Blowfish policies having weaker sensitive information specifications. We propose the ordered mechanism, a novel strategy for releasing cumulative histograms and answering range
queries. We show analytically and using experiments on real data that, for reasonable sensitive information specifications, the ordered hierarchical mechanism is more accurate than the best known differentially private mechanisms for these workloads. We study how to calibrate noise for policies expressing count constraints, and its applications in several practical scenarios.

The rest of this chapter is organized as follows. Section 3.2 formalizes privacy policies. We define Blowfish privacy, and discuss composition properties and its relationship to prior work in Section 3.3. We define the policy specific global sensitivity of queries in Section 3.4. We describe mechanisms for kmeans clustering (Section 3.4.1), and releasing cumulative histograms & answering range queries (Section 3.4.2) under Blowfish policies without constraints and empirically evaluate the resulting privacy-utility trade-offs on real datasets. We show how to release histograms in the presence of count constraints in Section 3.5 and then conclude in Section 3.6.

3.2 Policy Driven Privacy

In this section, we describe an abstraction called a policy that helps specify which information has to be kept secret and what background knowledge an attacker may possess about the correlations in the data.

3.2.1 Sensitive Information

As indicated by the name, Blowfish privacy is inspired by the Pufferfish privacy framework [74]. In fact, we will show later (in Section 3.3.3) that Blowfish privacy is equivalent to specific instantiations of semantic definitions arising from the Pufferfish framework.

Like Pufferfish, Blowfish privacy also uses the notions of secrets and discriminative

\footnote{Pufferfish and Blowfish are common names of the same family of marine fish, Tetraodontidae.}
pairs of secrets. We define a secret to be an arbitrary propositional statement over the values in the dataset. For instance, the secret $s : t.id = 'Bob' \land t.Disease = 'Cancer'$ is true in a dataset where Bob has Cancer. We denote by $S$ a set of secrets that the data publisher would like to protect. As we will see in this section each individual may have multiple secrets. Secrets may also pertain to sets of individuals. For instance, the following secret $s : t_1.id = 'Alice' \land t_2.id = 'Bob' \land t_1.Disease = t_2.Disease$ is true when Alice and Bob have the same disease. However, in this paper, we focus on the case where each secret is about a single individual.

We call a pair of secrets $(s, s') \in S \times S$ discriminative if they are mutually exclusive. Each discriminative pair describes properties that an adversary must not be able to distinguish between. One input to a policy is a set of discriminative pairs of secrets $S_{\text{pairs}}$.

We now present a few examples of sensitive information specified as a set of discriminative secrets.

- **Full Domain**: Let $s^i_x$ be the secret $(t.id = i \land t = x)$, for some $x \in T$. We define $S_{\text{pairs}}^{\text{full}}$ as:

  $$S_{\text{pairs}}^{\text{full}} = \{(s^i_x, s^j_y) | \forall i, \forall (x, y) \in T \times T\} \quad (3.1)$$

  This means that for every individual, an adversary should not be able to distinguish whether that individual’s value is $x$ or $y$, for all $x, y \in T$.

- **Attributes**: Let $x \in T$ denote a multidimensional value. Let $x[A]$ denote value of attribute $A$, and $x[\bar{A}]$ the value for the other attributes. Then a second example of sensitive information is:

  $$S_{\text{pairs}}^{\text{attr}} = \{(s^i_x, s^j_y) | \forall i, \exists A, x[A] \neq y[A] \land x[\bar{A}] = y[\bar{A}]\} \quad (3.2)$$

  $S_{\text{pairs}}^{\text{attr}}$ ensures that an adversary should not be able to sufficiently distinguish between any two values for each attribute of every individual’s value.
• **Partitioned:** Let $\mathcal{P} = \{P_1, \ldots, P_p\}$ be a partition that divides the domain into $p$ disjoint sets ($\cup_i P_i = \mathcal{T}$ and $\forall 1 \leq i, j \leq p, P_i \cap P_j = \emptyset$). We define partitioned sensitive information as:

$$\mathcal{S}_{\text{pairs}}^\mathcal{P} = \{(s^i_x, s^i_y) | \forall i, \exists j, (x, y) \in P_j \times P_j\} \quad (3.3)$$

In this case, an adversary is allowed to deduce whether an individual is in one of two different partitions, but can’t distinguish between two values within a single partition. This is a natural specification for location data – an individual may be OK with releasing his/her location at a coarse granularity (e.g., a coarse grid), but location within each grid cell must be hidden from the adversary.

• **Distance Threshold:** In many situations there is an inherent distance metric $d$ associated with the points in the domain (e.g., $L_1$ distance on age or salary, or Manhattan distance on locations). Rather than requiring that an adversary should not be able to distinguish between any pairs of points $x$ and $y$, one could require that each pair of points that are close are not distinguishable. So, for this purpose, the set of discriminative secrets is:

$$\mathcal{S}_{\text{pairs}}^{d, \theta} = \{(s^i_x, s^i_y) | \forall i, d(x, y) \leq \theta\} \quad (3.4)$$

Under this policy, the adversary will not be able to distinguish any pair of values with certainty. However, the adversary may distinguish points that are farther apart better that points that are close.

All of the above specifications of sensitive information can be generalized using the **discriminative secret graph**, defined below. Consider a graph $G = (V, E)$, where $V = \mathcal{T}$ and the set of edges $E \subseteq \mathcal{T} \times \mathcal{T}$. The set of edges can be interpreted as values in the domain that an adversary must not distinguish between; i.e., the set of discriminative secrets is $\mathcal{S}_{\text{pairs}}^G = \{(s^i_x, s^i_y) | \forall i, \forall (x, y) \in E\}$. The above examples correspond to the following graphs: $G^{\text{full}}$ corresponds to a complete graph on all the
elements in $\mathcal{T}$. $G^{\text{attr}}$ corresponds to a graph where two values are connected by an edge when only one attribute value changes. $G^P$ has $|\mathcal{P}|$ connected components, where each component is a complete graph on vertices in $P_i$. Finally, in $G^{d,\theta}$, $(x, y) \in E$ iff $d(x, y) \leq \theta$.

We would like to note that a policy could have secrets and discriminative pairs about sets of individuals. However, throughout this chapter, we only consider secrets pertaining to a single individual, and thus discriminative pairs refer to two secrets about the same individual. Additionally, the set of discriminative pairs is the same for all individuals. One can envision different individuals having different sets of discriminative pairs. For instance, we can model an individual who is privacy agnostic and does not mind disclosing his/her value exactly by having no discriminative pair involving that individual. Finally note that in all of the discussion in this section, the specification of what is sensitive information does not depend on the original database $D$. One could specify sensitive information that depends on $D$, but one must be wary that this might leak additional information to an adversary. In this chapter, we focus on data-independent discriminative pairs, uniform secrets and secrets that only pertain to single individuals.

Throughout this chapter, we will assume that the adversary knows the total number of tuples in the database (i.e., the set of possible instances is $\mathcal{I}_n$). Hence, we can limit ourselves to considering changes in tuples (and not additions or deletions). We can in principle relax this assumption about cardinality, by adding an additional set of secrets of the form $s^i_\perp$ which mean “individual $i$ is not in dataset”. All of our definitions and algorithms can be modified to handle this case by adding $\perp$ to the domain and to the discriminative secret graph $G$. We defer these extensions to future work.
3.2.2 Auxiliary Knowledge

Recent work [73] showed that differentially private mechanisms could still lead to an inordinate disclosure of sensitive information when adversaries have access to publicly known constraints about the data that induce correlations across tuples. This can be illustrated by the following example. Consider a table $D$ with one attribute $R$ that takes values $r_1, \ldots, r_k$. Suppose, based on publicly released datasets the following $k - 1$ constraints are already known: $c(r_1) + c(r_2) = a_1$, $c(r_2) + c(r_3) = a_2$, and so on, where $c(r_i)$ is the number of records with value $r_i$. This does not provide enough information to always reconstruct the counts in $D$ (k unknowns but $k - 1$ linear equations). However, if we knew the answer to some $c(r_i)$, then all counts can be reconstructed – in this way tuples are correlated.

Differential privacy allows answering all the count queries $c(r_i)$ by adding independent noise with variance $2/\epsilon^2$ to each count. While these noisy counts $\tilde{c}(r_i)$ themselves do not disclose information about any individual, they can be combined with the constraints to get very precise estimates of $c(r_i)$. That is, we can construct $k$ independent estimators for each count as follow. For $r_1$, $\tilde{c}(r_1)$, $a_1 - \tilde{c}(r_2)$, $a_1 - a_2 + \tilde{c}(r_3)$, $\ldots$ each equal $c(r_1)$ in expectation and have a variance of $2/\epsilon^2$. By averaging these estimators, we can predict the value of $c(r_i)$ with a variance of $2/(k\epsilon^2)$. For large $k$ (e.g., when there are $2^d$ values in $R$), the variance is small so that the table $D$ is reconstructed with very high probability, thus causing a complete breach of privacy.

Therefore, our policy specification also takes into account auxiliary knowledge that an adversary might know about the individuals in the private database. In Blowfish, we consider knowledge in the form of a set of deterministic constraints $Q$ that are publicly known about the dataset. We believe these are easier to specify than probabilistic correlation functions for data publishers. The effect of the constraints in $Q$ is to make only a subset of the possible database instances $I_Q \subset I_n$ possible;
or equivalently, all instances in \( \mathcal{I}_n \setminus \mathcal{I}_Q \) are impossible. For any database \( D \in \mathcal{I}_n \), we denote by \( D \models Q \) if \( D \) satisfies the constraints in \( Q \); i.e., \( D \in \mathcal{I}_Q \). Examples of deterministic constraints include:

- **Count Query Constraints**: A count query on a database returns the number of tuples that satisfy a certain predicate. A count query constraints is a set of (count query, answer) pairs over the database that are publicly known.

- **Marginal Constraints**: A marginal is a projection of the database on a subset of attributes, and each row counts the number of tuples that agree on the subset of attributes. The auxiliary knowledge of marginals means these database marginals are known to the adversary.

### 3.2.3 Policy

We are now ready to define a policy.

**Definition 13** (Policy). A policy is a triple \( P = (\mathcal{T}, G, \mathcal{I}_Q) \), where \( G = (V, E) \) is a discriminative secret graph with \( V \subseteq \mathcal{T} \). In \( P \), the set of discriminative pairs \( S^G_{\text{pairs}} \) is defined as the set \( \{(s^i_x, s^i_y) \mid \forall i \in \_id, \forall (x, y) \in E\} \), where \( s^i_x \) denotes the statement: \( t.\_id = i \land t = x \). \( \mathcal{I}_Q \) denotes the set of databases that are possible under the constraints \( Q \) that are known about the database.

Note that the description of the policy can be exponential in the size of the input dataset. We will use shorthand to describe certain types of sensitive information (e.g., full domain, partition, etc), and specify the set of possible databases \( \mathcal{I}_Q \) using the description of \( Q \).

### 3.3 Blowfish Privacy

Like differential privacy, Blowfish uses the notion of neighboring datasets. The key difference is that the set of neighbors in Blowfish depend on the policy \( P \) – both
on the set of discriminative pairs as well as on the constraints known about the database.

3.3.1 Definition of Blowfish Privacy

**Definition 14 (Neighbors).** Let \( P = (T, G, I_Q) \) be a policy. For any pair of datasets \( D_1, D_2 \), let \( T(D_1, D_2) \subseteq S_{pairs}^G \) be the set of discriminative pairs \((s^i_x, s^i_y)\) such that the \( i^{th} \) tuples in \( D_1 \) and \( D_2 \) are \( x \) and \( y \), resp. Let \( \Delta(D_1, D_2) = D_1 \setminus D_2 \cup D_2 \setminus D_1 \). \( D_1 \) and \( D_2 \) are neighbors with respect to a policy \( P \), denoted by \((D_1, D_2) \in N(P)\), if:

1. \( D_1, D_2 \in I_Q \) (i.e., both the datasets satisfy \( Q \)).
2. \( T \neq \emptyset \). (i.e., \( \exists (s^i_x, s^i_y) \in S_{pairs}^G \) such that the \( i^{th} \) tuples in \( D_1 \) and \( D_2 \) are \( x \) and \( y \), resp).
3. There is no database \( D_3 \vdash Q \) such that
   (a) \( T(D_1, D_3) \subset T(D_1, D_2) \), or
   (b) \( T(D_1, D_3) = T(D_1, D_2) \land \Delta(D_3, D_1) \subset \Delta(D_2, D_1) \).

When \( P = (T, G, I_n) \) (i.e., no constraints), \( D_1 \) and \( D_2 \) are neighbors if some individual tuples value is changed from \( x \) to \( y \), where \( (x, y) \) is an edge in \( G \). Note that \( T(D_1, D_2) \) is non-empty and has the smallest size (of 1). Neighboring datasets in differential privacy correspond to neighbors when \( G \) is a complete graph.

For policies having constraints, conditions 1 and 2 ensure that neighbors satisfy the constraints (i.e., are in \( I_Q \)), and that they differ in at least one discriminative pair of secrets. Condition 3 ensures that \( D_1 \) and \( D_2 \) are minimally different in terms of discriminative pairs and tuple changes.

**Definition 15 (Blowfish Privacy).** Let \( \epsilon > 0 \) be a real number and \( P = (T, G, I_Q) \) be a policy. A randomized mechanism \( M \) satisfies \((\epsilon, P)\)-Blowfish privacy if for every pair of neighboring databases \((D_1, D_2) \in N(P)\), and every set of outputs
Note that Blowfish privacy takes in the policy $P$ in addition to $\epsilon$ as an input, and is different from differential privacy in only the set of neighboring databases $N(P)$. For $P = (T, G, \mathcal{I}_a)$ (i.e., no constraints), it is easy to check that for any two databases that arbitrarily differ in one tuple $(D_1 = D \cup \{x\}, D_2 = D \cup \{y\})$, and any set of outputs $S$,

$$Pr[M(D_1) \in S] \leq e^\epsilon Pr[M(D_2) \in S]$$  \hfill (3.5)

where $d_G(x, y)$ is the shortest distance between $x, y$ in $G$. This implies that an attacker may better distinguish pairs of points farther apart in the graph (e.g., values with many differing attributes in $S_{\text{pairs}}^{\text{attr}}$), than those that are closer. Similarly, an attack can distinguish between $x, y$ with probability 1, when $x$ and $y$ appear in different partitions under partitioned sensitive information $S_{\text{pairs}}^P$ ($d_G(x, y) = \infty$).

### 3.3.2 Composition Properties

Composition [48] is an important property that any privacy notion should satisfy in order to be able to reason about independent data releases. Sequential composition ensures that a sequence of computations that each ensure privacy in isolation also ensures privacy. This allows breaking down computations into smaller building blocks. Parallel composition is crucial to ensure that too much error is not introduced on computations occurring on disjoint subsets of data. We can show that Blowfish satisfies sequential composition, and a weak form of parallel composition.

**Theorem 16** (Sequential Composition). Let $P = (T, G, \mathcal{I}_Q)$ be a policy and $D \in \mathcal{I}_Q$ be an input database. Let $M_1(\cdot)$ and $M_2(\cdot, \cdot)$ be algorithms with independent sources of randomness that satisfy $(\epsilon_1, P)$ and $(\epsilon_2, P)$-Blowfish privacy, resp. Then an algorithm
that outputs both \( M_1(D) = \omega_1 \) and \( M_2(\omega_1, D) = \omega_2 \) satisfies \((\epsilon_1 + \epsilon_2, P)\)-Blowfish privacy.

**Proof.** (sketch) Let \( M_{M_1, M_2} \) denote the mechanism that outputting the results of \( M_1 \) and \( M_2 \) sequentially. As \( M_1 \) satisfies \((\epsilon_1, P)\)-Blowfish privacy, for every pair of neighboring databases \( (D_a, D_b) \in N(P) \), and every result \( r_1 \in \text{range}(M_1) \), we have

\[
Pr[M_1(D_a) = r_1] \leq \epsilon^1 Pr[M_1(D_b) = r_1] \tag{3.7}
\]

The result of \( M_1 \) is outputted before the result of \( M_2 \), so \( r_1 \) will turn out to be another input of \( M_2 \), together with the original dataset. As \( M_1 \) satisfies \((\epsilon_1, P)\)-Blowfish privacy, for every pair of neighboring databases \( (D_a, D_b) \in N(P) \) coupling with the same \( r_1 \), and for every result \( r_2 \in \text{range}(M_2) \), we have

\[
Pr[M_2(D_a, r_1) = r_2] \leq \epsilon^1 Pr[M_2(D_b, r_1) = r_2] \tag{3.8}
\]

Therefore, for every pair of neighboring databases \( (D_a, D_b) \in N(P) \), and every set of output sequence \((r_1, r_2)\), we have

\[
Pr[M_{M_1, M_2}(D_a) = (r_1, r_2)] = Pr[M_1(D_a) = r_1]Pr[M_2(D_a, r_1) = r_2] \leq \epsilon^1 Pr[M_1(D_b) = r_1]e^{\epsilon_2}Pr[M_2(D_b, r_1) = r_2] \leq \epsilon^{\epsilon_1 + \epsilon_2} Pr[M_1(D_b) = r_1]Pr[M_2(D_b, r_1) = r_2] = e^{\epsilon_1 + \epsilon_2} Pr[M_{M_1, M_2}(D_b) = (r_1, r_2)] \tag{3.9}
\]

\[\square\]

**Theorem 17.** (Parallel Composition with Cardinality Constraint). Let \( P = (T, G, I_n) \) be a policy where the cardinality of the input \( D \in I_n \) is known. Let \( S_1, \ldots, S_p \) be disjoint subsets of \( \text{ids} \); \( D \cap S_i \) denotes the dataset restricted to the individuals in \( S_i \). Let \( M_i \) be mechanisms that each ensure \((\epsilon_i, P)\)-Blowfish privacy. Then the sequence of \( M_i(D \cap S_i) \) ensures \((\max_i \epsilon_i, P)\)-Blowfish privacy.

34
Proof. (sketch) For every pair of neighboring databases \((D_a, D_b) \in N(P)\) with the cardinality constraint or with disjoint subsets of constraints \(Q_1, \ldots, Q_p\), there is only one subset of \(\text{id}s\), let’s say \(S^\ast\), with different values in \(D_a\) and \(D_b\) while \(D_a \cap S_i = D_b \cap S_i\) for all \(i \neq i^\ast\). Hence, for every set of output sequence \(r\),

\[
Pr[M(D_a) = r] = \prod_i Pr[M_i(D_a \cap S_i) = r_i]
\]

\[
\leq e^{\epsilon^\ast} Pr[M_i^\ast(D_b \cap S_i^\ast) = r_i^\ast] \prod_{i,i \neq i^\ast} Pr[M_i(D_b \cap S_i) = r_i]
\]

\[
\leq e^{\max_i \epsilon_i} \prod_i Pr[M_i(D_b \cap S_i) = r_i]
\]

\[
= e^{\max_i \epsilon_i} Pr[M(D_b) = r]
\]  \hspace{1cm} (3.10)

Reasoning about parallel composition in the presence of general constraints is non-trivial. Consider two neighboring datasets \(D_a, D_b \in N(P)\). For instance, suppose one of the attributes is gender, we know the number of males and females in the dataset, and we are considering full domain sensitive information. Then there exist neighboring datasets such that differ in two tuples \(i\) and \(j\) that are alternately male and female in \(D_a\) and \(D_b\). If \(i\) and \(j\) appear in different subsets \(S_1\) and \(S_2\) resp., then \(D_a \cap S_1 \neq D_b \cap S_2\) and \(D_a \cap S_1 \neq D_b \cap S_2\). Thus the sequence \(M_i^\ast(D \cap S_i)\) does not ensure \((\max_i \epsilon_i, P)\)-Blowfish privacy. We generalize this observation below.

Define a pair of secrets \((s, s')\) to be critical to a constraint \(q\) if there exist \(D_s, D_{s'}\) such that \(T(D_s, D_{s'}) = (s, s')\), and \(D_s \vdash q\), but \(D_{s'} \not\vdash q\). Let \(\text{crit}(q)\) denote the set of secret pairs that are critical to \(q\). Next, consider \(S_1, \ldots, S_k\) disjoint subsets of ids. We denote by \(SP(S_i)\) the set of secret pairs that pertain to the ids in \(S_i\). We say that a constraint \(q\) affects \(D \cap S_i\) if \(\text{crit}(q) \cap SP(S_i) \neq \emptyset\). We can now state a sufficient condition for parallel composition.

35
Theorem 18 (Parallel Composition with General Constraints). Let $P = (\mathcal{T}, G, \mathcal{I}_Q)$ be a policy and $S_1, \ldots, S_p$ be disjoint subsets of ids. Let $M_i$ be mechanisms that each ensure $(\epsilon_i, P)$-Blowfish privacy. Then the sequence of $M_i(D \cap S_i)$ ensures $(\max_i \epsilon_i, P)$-Blowfish privacy if there exist disjoint subsets of constraints $Q_1, \ldots, Q_p \subseteq Q$ such that all the constraints in $Q_i$ only affects $D \cap S_i$.

We conclude this section with an example of parallel composition. Suppose $G$ contains two disconnected components on nodes $S$ and $T \setminus S$. The set of all secret pairs correspond to pairs of values that come either from $S$ or from $T \setminus S$. Suppose we know two count constraints $q_S$ and $q_{T \setminus S}$ that count the number of tuples with values in $S$ and $T \setminus S$, respectively. It is easy to see that $\text{crit}(q_S) = \text{crit}(q_{T \setminus S}) = 0$. Therefore, running an $(\epsilon, (\mathcal{T}, G, \{q_S, q_{T \setminus S}\}))$-Blowfish private mechanism on disjoint subsets results in no loss of privacy.

3.3.3 Relation to Other Definitions

In this section, we relate Blowfish privacy to existing notions of privacy. We discuss variants of differential privacy [39] (including restricted sensitivity [15]), the Pufferfish framework [74], privacy axioms [71], and a recent independent work on extending differential privacy with metrics [23].

Differential Privacy [39] One can easily verify that a mechanism satisfies $\epsilon$-differential privacy (Definition 66) if and only if it satisfies $(\epsilon, P)$-Blowfish privacy, where $P = (\mathcal{T}, K, \mathcal{I}_n)$, and $K$ is the complete graph on the domain. Thus, Blowfish privacy is a generalization of differential privacy that allows a data curator to trade-off privacy vs utility by controlling sensitive information $G$ (instead of $K$) and auxiliary knowledge $\mathcal{I}_Q$ (instead of $\mathcal{I}_n$) in the policy.

Pufferfish Framework [74, 75] Blowfish borrows the sensitive information specification from Pufferfish. Pufferfish defines adversarial knowledge using a set of data generating distributions, while Blowfish instantiates the same using publicly known
constraints. We can show formal relationships between Blowfish and Pufferfish instantiations.

**Theorem 19.** Let $S_{\text{pairs}}$ be the set of discriminative pairs corresponding to policy $P = (\mathcal{T}, G, \mathcal{I}_n)$. Let $D$ denote the set of all product distributions $\{p_i(\cdot)\}_i$ over $n$ tuples. $p_i(\cdot)$ denotes a probability distribution for tuple $i$ over $\mathcal{T}$. Then a mechanism satisfies $(\epsilon, S_{\text{pairs}}, D)$-Pufferfish privacy if and only if it satisfies $(\epsilon, P)$-Blowfish privacy.

**Theorem 20.** Consider a policy $P = (\mathcal{T}, G, \mathcal{I}_Q)$ corresponding to a set of constraints $Q$. Let $S_{\text{pairs}}$ be defined as in Theorem 19. Let $D_Q$ be the set of product distributions conditioned on the constraints in $Q$; i.e.,

$$P[D = x_1, \ldots, x_k] \propto \begin{cases} 
\prod_i p_i(x_i) & \text{if } D \in \mathcal{I}_Q \\
0 & \text{otherwise}
\end{cases}$$

A mechanism $M$ that satisfies $(\epsilon, S_{\text{pairs}}, D_Q)$-Pufferfish privacy also satisfies $(\epsilon, P)$-Blowfish privacy.

Theorem 19 states that Blowfish policies without constraints are equivalent to Pufferfish instantiated using adversaries who believe tuples in $D$ are independent (proof follows from Theorem 6.1 [75]). Theorem 20 states that when constraints are known, Blowfish is a necessary condition for any mechanism that satisfies a similar Pufferfish instantiation with constraints (we conjecture the sufficiency of Blowfish as well). Thus Blowfish privacy policies correspond to a subclass of privacy definitions that can be instantiated using Pufferfish.

Both Pufferfish and Blowfish aid the data publisher to customize privacy definitions by carefully defining sensitive information and adversarial knowledge. However, Blowfish improves over Pufferfish in three key aspects. First, there are no general algorithms known for Pufferfish instantiations. In this chapter, we present algorithms for various Blowfish policies. Thus, we can’t compare Blowfish and Pufferfish
experimentally. Second, all Blowfish privacy policies result in composable privacy definitions. This is not true for the Pufferfish framework. Finally, we believe Blowfish privacy is easier to understand and use than the Pufferfish framework for data publishers who are not privacy experts.\(^2\) For instance, one needs to specify adversarial knowledge as sets of complex probability distributions in Pufferfish, while in Blowfish policies one only needs to specify conceptually simpler publicly known constraints.

**Other Privacy Definitions:** Kifer and Lin [71] stipulate that every “good” privacy definition should satisfy two axioms – transformation invariance, and convexity. We can show that Blowfish privacy satisfy both these axioms.

Recent papers have extended differential privacy to handle constraints. Induced neighbor privacy [73, 74] extends the notion of neighbors such that neighboring databases satisfy the constraints and are minimally far apart (in terms of tuple changes). Blowfish extends this notion of induced neighbors to take into account discriminative pairs of secrets and measures distance in terms of the set of different discriminative pairs. Restricted sensitivity [15] extends the notion of sensitivity to account for constraints. In particular, the restricted sensitivity of a function \(f\) given a set of constraints \(Q\), or \(RS_f(Q)\), is the maximum \(|f(D_1) - f(D_2)|/d(D_1, D_2)\), over all \(D_1, D_2 \in \mathcal{I}_Q\). However, tuning noise to \(RS_f(Q)\) may not limit the ability of an attacker to learn sensitive information. For instance, if \(\mathcal{I}_Q = \{0^n, 1^n\}\), then the restricted sensitivity of releasing the number of 1s is 1. Adding constant noise does not disallow the adversary from knowing whether the database was 0\(^n\) or 1\(^n\).

A very recent independent work suggests extending differential privacy using a metric over all possible databases [23]. In particular, given a distance metric \(d\) over instances, they require an algorithm to ensure that \(P[M(X) \subseteq S] \leq e^{d(X,Y)}P[M(Y) \subseteq S]\), for all sets of outputs \(S\) and all instances \(X\) and \(Y\). Thus differential privacy

\(^2\) We have some initial anecdotal evidence of this fact working with statisticians from the US Census.
corresponds to a specific distance measure – Hamming distance. The sensitive information specification in Blowfish can also be thought of in terms of a distance metric over tuples. In addition we present novel algorithms (ordered mechanism) and allow incorporating knowledge of constraints. We defer a more detailed comparison to future work.

3.4 Blowfish without Constraints

Given any query $f$ that outputs a vector of reals, we can define a policy specific sensitivity of $f$. Thus, the Laplace mechanism with noise calibrated to the policy specific sensitivity ensures Blowfish privacy.

**Definition 21.** (Policy Specific Global Sensitivity). Given a policy $(T, G, I_Q)$, $S(f, P)$ denotes the policy specific global sensitivity of a function $f$ and is defined as

$$\max_{(D_1, D_2) \in N(P)} ||f(D_1) - f(D_2)||_1.$$

**Theorem 22.** Let $P = (T, G, I_Q)$ be a policy. Given a function $f : I_Q \rightarrow \mathbb{R}^d$, outputting $f(D) + \eta$ ensures $(\epsilon, P)$-Blowfish privacy if $\eta \in \mathbb{R}^d$ is a vector of independent random numbers drawn from $\text{Lap}(S(f, p)/\epsilon)$.

When policies do not have constraints $(P = (T, G, I_n))$, $(\epsilon, P)$-Blowfish differs from $\epsilon$-differential privacy only in the specification of sensitive information. Note that every pair $(D_1, D_2) \in N(P)$ differ in only one tuple when $P$ has no constraints. Therefore, the following result trivially holds.

**Lemma 23.** Any mechanism $M$ that satisfies $\epsilon$-differential privacy also satisfies $(\epsilon, (T, G, I_n))$-Blowfish privacy for all discriminative secret graphs $G$.

The proof follows from the fact that $\epsilon$-differential privacy is equivalent to $(\epsilon, (T, K, I_n))$-Blowfish privacy, where $K$ is the complete graph.
In many cases, we can do better in terms of utility than differentially privacy mechanisms. It is easy to see that $S(f, P)$ is never larger than the global sensitivity $S(f)$. Therefore, just using the Laplace mechanism with $S(f, P)$ can provide better utility.

For instance, consider a linear sum query $f_w = \sum_{i=1}^{n} w_i x_i$, where $w \in \mathbb{R}^n$ is a weight vector, and each value $x_i \in \mathcal{T} = [a, b]$. For $G^{\text{full}}$, the policy specific sensitivity is $(b-a) \cdot (\max_i w_i)$ the same as the global sensitivity. For $G^{d,\theta}$, where $d(x, y) = |x - y|$, the policy specific sensitivity is $\theta \cdot (\max_i w_i)$, which can be much smaller than the global sensitivity when $\theta \ll (b-a)$.

As a second example, suppose $\mathcal{P}$ is a partitioning of the domain. If the policy specifies sensitive information partitioned by $\mathcal{P}$ ($G^{\mathcal{P}}$), then the policy specific sensitivity of $h_\mathcal{P}$ is 0. That is, the histogram of $\mathcal{P}$ or any coarser partitioning can be released without any noise.

However, for histogram queries, the policy specific sensitivity for most reasonable policies (with no constraints) is 2, the same as global sensitivity. Thus, it cannot significantly improve the accuracy for histogram queries.

Next, we present examples of two analysis tasks – $k$-means clustering (Section 3.4.1), and releasing cumulative histograms (Section 3.4.2) – for which we can design mechanisms for Blowfish policies without constraints with more utility (lesser error) than mechanisms that satisfy differential privacy. In $k$-means clustering we will see that using Blowfish policies helps reduce the sensitivity of intermediate queries on the data. In the case of the cumulative histogram workload, we can identify novel query answering strategies given a Blowfish policy that helps reduce the error.

3 The one exception is partitioned sensitive information.
3.4.1 K-means Clustering

K-means clustering is widely used in many applications such as classification and feature learning. It aims to cluster proximate data together and is formally defined below.

**Definition 24 (K-means clustering).** Given a data set of $n$ points $(t_1, \ldots, t_n) \in \mathcal{T}^n$, $k$-means clustering aims to partition the points into $k \leq n$ clusters $S = \{S_1, \ldots, S_k\}$ in order to minimize

$$
\sum_{i=1}^{k} \sum_{t_j \in S_i} ||t_j - \mu_i||^2,
$$

(3.11)

where $\mu_i = \frac{1}{|S_i|} \sum_{t_j \in S_i} t_j$, and $||x - y||$ denotes $L_2$ distance.

The non-private version of $k$-means clustering initializes the means/centroids $(\mu_1, \ldots, \mu_k)$ (e.g. randomly) and updates them iteratively as follows: 1) assign each point to the nearest centroid; 2) recompute the centroid of each cluster, until reaching some convergence criterion or a fixed number of iterations.

The first differentially private $k$-means clustering algorithm was proposed by Blum et al. [16] as SuLQ $k$-means. Observe that only two queries are required explicitly: 1) the number of points in each new cluster, $q_{\text{size}} = (|S_1|, \ldots, |S_k|)$ and 2) the sum of the data points for each cluster, $q_{\text{sum}} = (\sum_{t_j \in S_1} t_j, \ldots, \sum_{t_j \in S_k} t_j)$, to compute the centroid. The sensitivity of $q_{\text{size}}$ is 2 (same as a histogram query). Let $d(\mathcal{T})$ denote the diameter of the domain, or the largest $L_1$ distance ($||x - y||_1$) between any two points $x, y \in \mathcal{T}$. The sensitivity of $q_{\text{sum}}$ could be as large as the diameter $2 \cdot d(\mathcal{T})$ since a tuple from $x$ to $y$ can only change the sums for two clusters by at most $d(\mathcal{T})$.

Under Blowfish privacy policies, the policy specific sensitivity of $q_{\text{sum}}$ can be much smaller than $|\mathcal{T}|$ under Differential privacy (i.e. complete graph $G^{\text{full}}$ for Blowfish
policies). Since $q_{\text{size}}$ is the histogram query, the sensitivity of $q_{\text{size}}$ under Blowfish is also 2.

**Lemma 25.** Policy specific global sensitivities of $q_{\text{sum}}$ under the attribute $G^{\text{attr}}$, $L_1$-distance $G^{(L_1, \theta)}$, and partition $G^P$ discriminative graphs (from Section 3.2) are smaller than the global sensitivity of $q_{\text{sum}}$ under differential privacy.

**Proof.** First, in the attribute discriminative graph $G^{\text{attr}}$, edges correspond to $(x, y) \in T \times T$ that differ only in any one attribute. Thus, if $|A|$ denotes maximum distance between two elements in $A$, then the policy specific sensitivity of $q_{\text{sum}}$ under $G^{\text{attr}}$ is $\max_A(2 \cdot |A|) < 2 \cdot d(T)$. Next, suppose we use $G^{(L_1, \theta)}$, where $x, y \in T$ are connected by an edge if $||x - y||_1 \leq \theta$. Thus, policy specific sensitivity of $q_{\text{sum}}$ is $2 \theta$.

Finally, consider the policy specified using the partitioned sensitive graph $G^P$, where $P = \{P_1, P_2, \ldots, P_k\}$ is some data independent partitioning of the domain $T$. Here, an adversary should not distinguish between an individual’s tuple taking a pair of values $x, y \in T$ only if $x$ and $y$ appear in the same partition $P_i$ for some $i$. Under this policy the sensitivity of $q_{\text{sum}}$ is at most $\max_{P \in P} 2 \cdot d(P) < 2 \cdot d(T)$. \qed

Thus, by Theorem 22, we can use the SULQ $k$-means mechanism with the appropriate policy specific sensitivity for $q_{\text{sum}}$ (from Lemma 25) and thus satisfy privacy under the Blowfish policy while ensuring better accuracy.

**Empirical Evaluation.** We empirically evaluate the accuracy of $k$-means clustering for $(\epsilon, (T, G, L_n))$-Blowfish privacy on three data sets. The first two datasets are real-world datasets – TWITTER and SKIN SEGMENTATION\(^4\). The TWITTER data set consists of a total of 193563 tweets collected using Twitter API that all contained a latitude/longitude within a bounding box of 50N, 125W and 30N, 110W (western USA) – about 2222 × 1442 square km. By setting the precision of latitude/longitude

coordinates to be 0.05, we obtain a 2D domain of size $400 \times 300$. The skin segmentation data set consists of 245057 instances. Three ordinal attributes are considered and they are B, G, R values from face images of different classes. Each of them has a range from 0 to 255. To understand the effect of Blowfish policies on datasets of different sizes, we consider the full dataset skin, as well as a 10% and 1% sub-sample (skin10, skin01) of the data. The third dataset is a synthetic dataset where we generate 1000 points from $p^0, 1^q 4$ with $k$ randomly chosen centers and a Gaussian noise with $\sigma(0, 0.2)$ in each direction.

In Figures 3.1(a)-3.1(d), we report the ratio of the mean of the objective value in Eqn. (3.11) between private clustering methods including Laplace mechanism and Blowfish privacy with $S_{\text{pairs}}^{d,\theta}$, and the non-private k-means algorithm, for various values of $\epsilon = \{0.1, 0.2, \ldots, 0.9, 1.0\}$. For all datasets, $d(\cdot)$ is $L_1$ (or Manhattan) distance. The number of iterations is fixed to be 10 and the number of clusters is $k = 4$. Each experiment is repeated 50 times to find mean, lower and upper quartile. Figure 3.1(a) clusters according to latitude/longitude of each tweet. We consider five different
policies: \( G^{\text{full}} \) (Laplace mechanism), \( G^{\text{d},2000\text{km}} \), \( G^{\text{d},1000\text{km}} \), \( G^{\text{d},500\text{km}} \), \( G^{\text{d},100\text{km}} \). Here, \( \theta = 100\text{km} \) means that the adversary cannot distinguish locations within a 20000 square km region. Figure 3.1(b) clusters the 1\% subsample skin01 based on three attributes: B, G, R values and considers 5 policies as well: \( G^{\text{full}} \), \( G^{\text{d},256} \), \( G^{\text{d},128} \), \( G^{\text{d},64} \) and \( G^{\text{d},32} \). Lastly, we also consider five policies for the synthetic dataset in Figure 3.1(c): \( G^{\text{full}} \), \( G^{\text{d},1.0} \), \( G^{\text{d},0.5} \), \( G^{\text{d},0.25} \), \( G^{\text{d},0.1} \).

From Figures 3.1(a)-3.1(c), we observe that the objective value of Laplace mechanism could deviate up to 100 times away from non-private method, but under Blowfish policies objective values could be less than 5 times that for non-private k-means. Moreover, the error introduced by Laplace mechanism becomes larger with higher dimensionality – the ratio for Laplace mechanism in Figure Figure 3.1(c) and 3.1(b) (4 and 3 dimensional resp.) is much higher than that in the 2D twitter dataset in Figure 3.1(a). From Figure 3.1(b), we observe that the error introduced by private mechanisms do not necessarily reduce monotonically as we reduce Blowfish privacy protection (i.e. reduce \( \theta \)). The same pattern is observed in Figure 3.1(a) and Figure 3.1(c). One possible explanation is that adding a sufficient amount of noise could be helpful to get out of local minima for clustering, but adding too much noise could lead to less accurate results.

To study the interplay between dataset size and Blowfish, we plot (Figure 3.1(d)) for skin, skin10 and skin01 the ratio of the objective value attained by the Laplace method to the objective value attained by one of the Blowfish policies: \( G^{\text{d},128} \). In all cases, we see an improvement in the objective under Blowfish. The improvement in the objective is smaller for larger \( \epsilon \) and larger datasets (since the Laplace mechanism solution is close to the non-private solutions on skin).

Finally, Figures 3.1(e) and 3.1(f) summarize our results on the \( G^{\text{attr}} \) and \( G^{\text{P}} \) discriminative graphs. Figure 3.1(e) shows that under the \( G^{\text{attr}} \) Blowfish policy, the error decreases by an order of magnitude compared to the Laplace mechanism for
SKIN01 and the synthetic dataset due to higher dimensionality and small dataset size. On the other hand, there is little gain by using $G^{attr}$ for the larger 2D TWITTER dataset.

Figure 3.1(f) shows ratio of the objective attained by the private methods to that of the non-private k-means under $G^P$, for partitions $\mathcal{P}$ of different sizes. In each case, the 300x400 grid is uniformly divided; e.g., in partition\textmid100, we consider a uniform partitioning in 100 coarse cells, where each new cell contains 30x40 cells from the original grid. Thus an adversary will not be able to tell whether an individual’s location was within an area spanned by the 30x40 cells (about 36,300 sq km). partition\textmid120000 corresponds to the original grid; thus we only protects pairs of locations within each cell in the original grid (about 30 sq km). We see that the objective value for Blowfish policies are smaller than the objective values under Laplace mechanisms, suggesting more accurate clustering. We also note that under partition\textmid120000, we can do the clustering exactly, since the sensitivity of both $q_{size}$ and $q_{sum}$ are 0.

To summarize, Blowfish policies allow us to effectively improve utility by trading off privacy. In certain cases, we observe that Blowfish policies attain an objective value that is close to 10 times smaller than that for the Laplace mechanism. The gap between Laplace and Blowfish policies increases with dimensionality, and reduces with data size.

### 3.4.2 Answering Cumulative Histograms

In this section, we develop novel query answering strategies for two workloads – cumulative histograms and range queries.

**Definition 26** (Cumulative Histogram). Consider a domain $\mathcal{T} = \{x_1, ..., x_{|\mathcal{T}|}\}$ that has a total ordering $x_1 \leq ... \leq x_{|\mathcal{T}|}$. Let $c(x_i)$ denote the number of times $x_i$ appears in the database $D$. Then, the cumulative histogram of $\mathcal{T}$, denoted by $S_{\mathcal{T}}(\cdot)$ is a
sequence of cumulative counts

\[
\left\{ s_i \mid s_i = \sum_{j=1}^{i} c(x_j), \forall i = 1, \ldots, |T| \right\}
\]  

(3.12)

Throughout this section, we will use Mean Squared Error as a measure of accuracy/error defined in Def 27.

**Definition 27.** Let \( M \) be a randomized algorithm that privately computes a function \( f : \mathcal{I}_n \rightarrow \mathbb{R}^d \). The expected mean squared error of \( M \) is given by:

\[
\mathcal{E}_M(D) = \sum_i \mathbb{E}(f_i(D) - \tilde{f}_i(D))^2
\]  

(3.13)

where \( f_i(\cdot) \) and \( \tilde{f}_i(\cdot) \) denote the \( i \)-th component of the true and noisy answers, respectively.

Since we know the total size of the dataset \(|D| = n\), dividing each cumulative count in \( S_T(\cdot) \) by \( n \) gives us the cumulative distribution function (CDF) over \( T \). Releasing the CDF has many applications including computing quantiles and histograms, answering range queries and constructing indexes (e.g. k-d tree). This motivates us to design a mechanism for releasing cumulative histograms.

The cumulative histogram has a global sensitivity of \(|T| - 1\) because all the counts in cumulative histogram except \( s_{|T|} \) will be reduced by 1 when a record in \( D \) changes from \( x_1 \) to \( x_{|T|} \). Similar to k-means clustering, we could reduce the sensitivity of cumulative histogram \( S_T(\cdot) \) by specifying the sensitive information, such as \( S_{\text{pairs}}^{d,\theta} \) and \( S_{\text{pairs}}^{d,\theta} \). For this section, we focus on \( S_{\text{pairs}}^{d,\theta} \), where \( d(\cdot) \) is the \( L1 \) distance on the domain and we assume that all the domains discussed here have a total ordering.

**Ordered Mechanism.** Let us first consider a policy \( P_\theta = (T, G^{d,\theta}, \mathcal{I}_n) \) with \( \theta = 1 \). The discriminative secret graph is a line graph, \( G^{d,1} = (V, E) \), where \( V = T \) and \( E = \{(x_i, x_{i+1}) \mid \forall i = 1, \ldots, |T| - 1\} \). This means that only adjacent domain values
(x_i, x_{i+1}) \in \mathcal{T} \times \mathcal{T} can form a secret pair. Therefore, the policy specific sensitivity of S_T(\cdot) for a line graph is 1. Based on this small sensitivity, we propose a mechanism, named Ordered Mechanism \text{M}_G^{d,1} to perturb cumulative histogram S_T(\cdot) over line graph G^{d,1} in the following way. For each s_i, we add \eta_i \sim Laplace(\frac{1}{\epsilon}) to get \bar{s}_i to ensure (\epsilon, P)-Blowfish privacy. Each \bar{s}_i has an error with an expectation equals to \frac{2}{\epsilon^2}. Note that Theorem 22 already ensures that releasing \bar{s}_i’s satisfies (\epsilon, P_1)-Blowfish privacy. Furthermore, observe that the counts in S_T(\cdot) are in ascending order. Hence, we can boost the accuracy of \hat{S}_T(\cdot) using constrained inference proposed by Hay et al. in [57]. In this way, the new cumulative histogram, denoted by \hat{S}_T(\cdot), satisfies the ordering constraint and has an error \mathcal{E}_{\hat{S}} = O(\frac{p \log |\mathcal{T}|}{\epsilon^2}), where p represents the number of distinct values in S_T(\cdot) [57]. Note that, if we additionally enforce the constraint that s_1 > 0, then all the counts are also positive. In particular, when p = 1, \mathcal{E}_{\hat{S}} = O(\frac{\log |\mathcal{T}|}{\epsilon^2}) and when p = |\mathcal{T}|, \mathcal{E}_{\hat{S}} = O(\frac{|\mathcal{T}|}{\epsilon^2}). Many real datasets are sparse, i.e. the majority of the domain values have zero counts, and hence have fewer distinct cumulative counts, i.e. p \ll |\mathcal{T}|. This leads to much smaller \mathcal{E}_{\hat{S}} compared to \mathcal{E}_{\bar{S}}. The best known strategy for releasing the cumulative histogram is using the hierarchical mechanism [57], which results in a total error of O(\frac{|\mathcal{T}| \log^3 |\mathcal{T}|}{\epsilon^2}). Moreover, the SVD bound [80] suggests that no strategy can release the cumulative histogram with O(\frac{|\mathcal{T}|}{\epsilon^2}) error. Thus under the line graph policy, the Ordered Mechanism is a much better strategy for cumulative histogram.

One important application of cumulative histogram is answering range query, defined as follows.

**Definition 28** (Range Query). Let D has domain \mathcal{T} = \{x_1, ..., x_{|\mathcal{T}|}\}, where \mathcal{T} has a total ordering. A range query, denoted by q[x_i, x_j] counts the number of tuples falling within the range [x_i, x_j] where x_i, x_j \in \mathcal{T} and x_i \leq x_j.

Range queries can be directly answered using cumulative histogram \hat{S}_T(\cdot), q[x_i, x_j] =
\( \hat{s}_j - \hat{s}_{i-1} \). As each range query requires at most two noisy cumulative counts, it has an error smaller than \( 2 \cdot \frac{2}{\epsilon^2} \) (even without constrained inference). Hence, we have the following theorem.

**Theorem 29.** Consider a policy \((\mathcal{T},G^d,\mathcal{I}_n)\), where \(G^d\) is a line graph. Then the expected error of a range query \(q[x_i, x_j]\) for Ordered Mechanism is given by:

\[
\mathcal{E}_{q[x_i, x_j], MO_{G^d, 1}} \leq 4/\epsilon^2
\]

This error bound is independent of \(|\mathcal{T}|\), much lower than the expected error using hierarchical structure with Laplace mechanism to answer range queries, \(\mathcal{E}_{q[x_i, x_j], lap} = \log^3 |\mathcal{T}| / \epsilon^2\). Again, the SVD bound [80] suggests that no differentially private strategy can answer each range query with \(O(1/|\mathcal{T}|)\) error. Other applications of \(S_T(\cdot)\) including computing quantiles and histograms and constructing indexes (e.g. \(k\)-d tree) could also use cumulative histogram in a similar manner as range query to obtain a much smaller error by trading utility with privacy under \(G^d\). Next, we describe the **ordered hierarchical mechanism** that works for general graphs, \(G^{d, \theta}\).

**Ordered Hierarchical Mechanism.** For a more general graph \(G^{d, \theta} = (V, E)\), where \(V = \mathcal{T}\) and \(E = \{(x_i, x_{i+1}), ..., (x_i, x_{i+\theta})|\forall i = 1, ..., |\mathcal{T}|\}\), the sensitivity of releasing cumulative histogram \(S_T(\cdot)\) becomes \(\theta\). The Ordered Mechanism would add noise from \(Lap(\frac{4}{\epsilon^2})\) to each cumulative counts \(s_i\). The total error in the released cumulative histogram and range queries would still be asymptotically smaller than the error achieved by any differentially private mechanism for small \(\theta\). However, the errors become comparable as the \(\theta\) reaches \(\log |\mathcal{T}|\), and the Ordered Mechanism’s error exceeds the error from the hierarchical mechanism when \(\theta = O(\log^{3/2} |\mathcal{T}|)\). In this section, we present a hybrid strategy for releasing cumulative histograms (and hence range queries), called Ordered Hierarchical Mechanism, that always has an error less than or equal to the hierarchical mechanism for all \(\theta\).
Various hierarchical methods have been proposed in the literature [57, 124, 78, 126, 103]. A basic hierarchical structure is usually described as a tree with a regular fan-out $f$. The root records the total size of the dataset $D$, i.e. the answer to the range query $q[x_1, x_T]$. This range is then partitioned into $f$ intervals. If $\delta = \lceil \frac{|T|}{f} \rceil$, the intervals are $[x_1, x_\delta], [x_{\delta+1}, x_{2\delta}], \ldots, [x_{|T|-\delta+1}, x_T]$ and answers to the range queries over those intervals are recorded by the children of the root node. Recursively, the interval represented by the current node will be further divided into $f$ subintervals. Leaf nodes correspond to unit length interval $q[x_i, x_i]$. The height of the tree is $h = \lceil \log_f |T| \rceil$. In the above construction, the counts at level $i$ are released using the Laplace mechanism with parameter $\frac{2}{\epsilon}$, and $\sum_i \epsilon_i = \epsilon$. Prior work has considered distributing the $\epsilon$ uniformly or geometrically [33]. We use uniform budgeting in our experiments.

Inspired by ordered mechanism for line graph, we propose a hybrid structure,
called Ordered Hierarchical Structure \( OH \) for \((\epsilon, (T, G^d, \mathcal{I}_n))\)-Blowfish privacy. As shown in Figure 3.2(a), \( OH \) has two types of nodes, \( S \) nodes and \( H \) nodes. The number of \( S \) nodes is \( k = \lceil \frac{n}{\theta} \rceil \), which is dependent on the threshold \( \theta \). In this way, we could guarantee a sensitivity of 1 among the \( S \) nodes. Let us represent \( S \) nodes as \( s_1, \ldots, s_k \), where 
\[
s_1 = q[x_1, x_{\theta}], \ldots, s_{k-1} = q[x_1, x_{(k-1)\theta}], s_k = q[x_1, x_{T}].
\]
Note that the \( s_i \) nodes here are not the same as the count for cumulative histogram, so we will use range query \( q[x_1, x_i] \) to represent the count \( s_i \) in a cumulative histogram. The first \( S \) node, \( s_1 \) is the root of a subtree consisting of \( H \) nodes. This subtree is denoted by \( H_1 \) and is used for answering all possible range queries within this interval \([x_1, x_{\theta}]\). For all \( 1 < i \leq k \), \( s_i \) has two children: \( s_{i-1} \) and the root of a subtree made of \( H \) nodes, denoted by \( H_i \). Similarly, it also has a fan-out of \( f \) and represents counts for values \([((i-1)\theta + 1), i\theta]\). We denote the height of the subtree by \( h = \lceil \log_f \theta \rceil \). Using this hybrid structure, we could release cumulative counts in this way: 
\[
q[x_1, x_{l\theta}] + q[x_{l\theta+1}, x_j],
\]
where \( l\theta \leq j < (l+1)\theta \). Here \( q[x_1, x_{l\theta}] \) is answered using \( s_l \) and \( q[x_{l\theta+1}, x_j] \) is answered using \( H_l \). Then any range query could be answered as 
\[
q[x_i, x_j] = q[x_1, x_j] - q[x_1, x_{i-1}].
\]

**Privacy Budgeting Strategy.** Given total privacy budget \( \epsilon \), we denote the privacy budget assigned to all the \( S \) nodes by \( \epsilon_S \) and to all the \( H \) nodes by \( \epsilon_H \). When a tuple change its value from \( x \) to \( y \), where \( d_G(x, y) \leq \theta \), at most one \( S \) node changes its count value and at most \( 2h \) \( H \) nodes change their count values. Hence, for \( i = 2, \ldots, k \), we add Laplace noise drawn from \( Lap(\frac{1}{\epsilon_S}) \) to each \( s_i \) and we add Laplace noise drawn from \( Lap(\frac{2h}{\epsilon_H}) \) to each \( H \) node in the subtree \( H_i \). As \( S_1 \) is the root of \( H_1 \), we assign \( \epsilon = \epsilon_S + \epsilon_H \) to the tree \( H_1 \) and hence we add Laplace noise drawn from \( Lap(\frac{2h}{\epsilon_H + \epsilon_S}) \) to each \( H \) node in \( H_1 \), including \( s_1 \). In this way, we could claim that this \( OH \) tree satisfies \((\epsilon, (T, G^d, \mathcal{I}_n))\)-Blowfish privacy. When \( \theta = |T| \), \( H_1 \) is going to be the only tree to have all the privacy budget. This is equivalent to the hierarchical mechanism.
for differential privacy.

**Theorem 30.** Consider a policy $P = (T, G^{d, \theta}, I_n)$. (1) The Ordered Hierarchical structure satisfies $(\epsilon, P)$-Blowfish privacy. (2) The expected error of releasing a single count in cumulative histogram or answering a range query this structure over $T$ is,

$$\mathcal{E}_{q[x_i, x_j], M_{G^{d, \theta}}} = O\left(\left|T\right| - \theta + \frac{(f - 1) \log^2 f}{\epsilon^2} \right)$$

(3.15)

Proof. (sketch) 1) Increasing count of $x$ by 1 and decreasing count of $y$ by 1, where $d(x, y) < \theta$ only affect the counts of at most one $S$ node and $2h$ $H$ nodes. Since we draw noise from $Lap(\frac{1}{\epsilon_S})$ for $S$ nodes and from $Lap(\frac{2h}{\epsilon_H})$ for $H$ nodes (where $\epsilon = \epsilon_S + \epsilon_H$), we get $(\epsilon, P)$-Blowfish privacy based on sequential composition.

2) Consider all the counts in cumulative histogram, and there are $|T|$ of them, only $|T| - \theta$ requires $S$ nodes. Each $S$ node has an error of $\frac{2}{\epsilon_S}$. This gives the first fraction in Eqn. (3.15). On average, the number of $H$ nodes used for each count in cumulative histogram is bounded by the height of the $H$ tree and each $H$ node has an error of $\frac{8h^2}{\epsilon^2_H}$, which explains the second fraction in Eqn. (3.15).

Each range query $q[x_i, x_j]$ requires at most 2 counts from cumulative histogram and its exact form of expected error is shown below,

$$\mathcal{E}_{q[x_i, x_j], M_{G^{d, \theta}}} = \frac{c_1}{\epsilon^2_S} + \frac{c_2}{\epsilon^2_H},$$

(3.16)

where $c_1 = \frac{4|\left|T\right| - \theta}{|T| + 1}$ and $c_2 = \frac{8(f - 1) \log f \theta^3 |T|}{|T| + 1}$. Given $\theta$ and $f$, at $\epsilon^*_S = \frac{c_1^{1/3} + c_2^{1/3}}{c_1^{1/3} + c_2^{1/3}} \epsilon$, we obtain the minimum

$$\mathcal{E}^*_{q[x_i, x_j], M_{G^{d, \theta}}} = \left(\frac{c_1^{1/3} + c_2^{1/3}}{\epsilon^2}\right)^3$$

(3.17)

In particular, when $\theta = |T|$, $c_1 = 0$, this is equivalent to the classical hierarchical mechanism without $S$ nodes and we have $\mathcal{E}_{q[x_i, x_j], M_{G^{d, \theta}}} = O\left(\frac{|T|}{\epsilon^2}\right)$. When
θ = 1, c_2 = 0, this is the pure Ordered Mechanism without using H nodes and $\mathcal{E}_{q[x_i,x_j], M_{OH}} = O(\frac{1}{\epsilon^2})$.

**Complexity Analysis.** The complexity of construction of the hybrid tree OH and answering range query are $O(|T|)$ and $O(\log \theta)$ respectively, where $\theta \leq |T|$, which is not worse than the classical hierarchical methods.

**Empirical Evaluation.** We empirically evaluate the error of range queries for the Ordered Hierarchical Mechanism with $(\epsilon, (T, G^{d,\theta}, \mathcal{L}_n))$-Blowfish privacy on two real-world datasets – ADULT and TWITTER. The ADULT data set\(^5\) consists of Census records of 48842 individuals. We consider the ordinal attribute capital loss with a domain size of 4357. The TWITTER data set is the same dataset used for k-means clustering (Sec 3.4.1). Here, in order to have a total ordering for the dataset, we project the TWITTER data set on its L with a domain size of 400, around 2222 km. The fan-out \(f\) is set to be 16 and each experiment is repeated 50 times. Figure 3.2(b) shows the mean square error $\mathcal{E}$ of 10000 random range queries for various values of $\epsilon = \{0.1, 0.2, \ldots, 0.9, 1.0\}$. Seven threshold values $\theta = \{\text{full}, 1000, 500, 100, 50, 10, 1\}$ are considered. For ADULT, for example, $\theta = 100$ means the adversary cannot distinguish between values of capital loss within a range of 100 and $\theta = \text{full}$ means the adversary cannot distinguish between all the domain values (same as differential privacy). Figure 3.2(c) considers 4 threshold values $\theta = \{\text{full}, 500km, 50km, 5km\}$ for TWITTER. When $\theta = 1$ (ADULT) or $\theta = 5km$ (TWITTER), the ordered hierarchical mechanism is same as the ordered mechanism. From both figures, we see that as the $\theta$ increases, $\mathcal{E}$ decreases and orders of magnitude difference in error between $\theta = 1$ and $\theta = |T|$.

In [59, 55], Blowfish private mechanisms were designed to answer aggregate queries under different policy graphs. Each policy graph can instantiate a new notion

---

\(^5\) http://mlr.cs.umass.edu/ml/datasets/Adult
of neighboring databases. Rather than re-designing a new algorithm for each notion of neighboring databases, [55] showed a transformational equivalence between a large class of Blowfish private algorithms and standard differential privacy algorithms and for many policy graphs. This equivalence can be stated as follows: for policy graph $G$, there exists a transformation of the workload and database $(W, x) \rightarrow (W_G, x_G)$ such that $Wx = W_G x_G$, and a mechanism $M$ is an $(\epsilon, G)$-Blowfish private mechanism for answering workload $W$ on input $x$ if and only if $M$ is also an $\epsilon$-differentially private mechanism for answering $W_G$ on $x_G$. This result does not hold in general, but [55] showed that under a class of mechanism called matrix mechanism, transformational equivalence holds for any policy graph.

### 3.4.3 Equivalence for Matrix Mechanism

Matrix mechanism framework was designed for optimally answering a workload of linear queries [78]. Some workloads $W$ have a high sensitivity, but they can be answered with low error by answering a different strategy query workload $A$ such that (a) $A$ has a low sensitivity $\Delta_A$, and (b) rows in $W$ can be reconstructed using a small number of rows in $A$.

In particular, let $A$ be a $p \times k$ matrix, and $A^+$ denote its Moore-Penrose pseudo-inverse, such that $WAA^+ = W$. The matrix mechanism is given by the following:

$$M_A(W, x) = Wx + W A^+ \text{Lap}(\Delta_A/\epsilon)^p$$  \hspace{1cm} (3.18)

where, $\text{Lap}(\lambda)^p$ denotes $p$ independent random variables drawn from the Laplace distribution with scale $\lambda$. The corresponding Blowfish specific sensitivity of a workload, $\Delta_w(G)$ is defined as follows:

**Definition 31** (Policy Specific $l_1$ Sensitivity). The $l_1$ policy specific sensitivity of a query matrix $W$ with respect to policy graph $G$ is

$$\Delta_w(G) = \max_{x, x' \in N(G)} ||Wx - Wx'||_1$$  \hspace{1cm} (3.19)
Let $P_G$ be a matrix that satisfies the following properties.

- $P_G$ has $|V| - 1$ rows and $|E|$ columns.
- Let $W_G = WP_G$. Then $\Delta_w(G) = \Delta_{W_G}$. i.e. the sensitivity of workload $W$ under Blowfish policy $G$ is the same as the sensitivity of $W_G$ under differential privacy.
- $P_G$ has full row rank (and therefore a right inverse $P_G^{-1}$). For vector $x$, we let $x_G$ denote $P_G^{-1}x$.

Given such a $P_G$, the following theorem is true.

**Theorem 32.** [55] Let $G$ be a Blowfish policy graph and $W$ be a workload. Suppose $P_G$ exists with the properties given above. Then the matrix mechanism given by Eqn. 3.18 is both a $(\epsilon, G)$-Blowfish private mechanism for answering $W$ on $x$ and an $\epsilon$-differentially private algorithm for answering $W_G$ on $x_G$. Since $Wx = W_Gx_G$, the mechanism has the same error in both instances.

This theorem allows the problem of answering queries under a large class of Blowfish privacy definitions to be turned to one of query answering under standard differential privacy without starting from scratch.

### 3.5 Blowfish with Counting Constraints

In this section, we consider query answering under Blowfish policies with constraints $P = (\mathcal{T}, G, \mathcal{I}_Q)$, where $(\mathcal{I}_Q \subseteq \mathcal{I}_n)$. In the presence of general deterministic constraints $Q$, pairs of neighboring databases can differ in any number of tuples, depending on structures of $Q$ and the discriminative graph $G$. Computing the policy specific sensitivity in this general case is a hard problem, as shown next.

**Theorem 33.** Given a function $f$ and a policy $P = (\mathcal{T}, G, \mathcal{I}_Q)$. Checking whether $S(f, P) > 0$ is NP-hard. The same is true for the complete histogram query $h$. 

54
Proof. (sketch) The proof follows from the hardness of checking whether 3SAT has at least 2 solutions. The reduction uses $I = \{0, 1\}^n$, constraints corresponding to clauses in the formula, and $\{s_i^0, s_i^1\}_i$ as secret pairs. □

Theorem 33 implies that checking whether $S(f, P) \leq z$ is co-NP-hard for general constraints $Q$. In fact, the hardness result holds even if we just consider the histogram query $h$ and general count query constraints.

Hence, in the rest of this section, we will focus on releasing histograms under a large subclass of constraints called sparse count query constraints. In Section 3.5.1 we show that when the count query constraint is “sparse”, we can efficiently compute $S(h, P)$, and thus we can use the Laplace mechanism to release the histogram. In Section 3.5.2, we will show that our general result about $S(h, P)$ subject to sparse count query constraints can be applied to several important practical scenarios.

3.5.1 Global Sensitivity for Sparse Constraints

A count query $q_\phi$ returns the number of tuples satisfying predicate $\phi$ in a database $D$, i.e., $q_\phi(D) = \sum_{t \in D} 1_{\phi(t) = \text{true}}$. The auxiliary knowledge we consider here is a count query constraint $Q$, which can be expressed as a conjunction of query-answer pairs:

$$q_{\phi_1}(D) = \text{cnt}_1 \wedge q_{\phi_2}(D) = \text{cnt}_2 \wedge \ldots \wedge q_{\phi_p}(D) = \text{cnt}_p. \quad (3.20)$$

Since the answers $\text{cnt}_1, \text{cnt}_2, \ldots, \text{cnt}_p$ do not affect our analysis, we denote the auxiliary knowledge or count query constraint as $Q = \{q_{\phi_1}, q_{\phi_2}, \ldots, q_{\phi_p}\}$. Note that this class of auxiliary knowledge is already very general and commonly seen in practice. For example, marginals of contingency tables, range queries, and degree distributions of graphs can all be expressed in this form.

Even for this class of constraints, calculating $S(h, P)$ is still hard. In fact, the same hardness result in Theorem 33 holds for count query constraints (using a reduction from the Vertex Cover problem).
Sparse Auxiliary Knowledge. Consider a secret pair \((s^i_x, s^i_y) \in \mathcal{S}^G_{\text{pairs}}\) about a tuple \(t\) with \(t_{.id} = i\), and a count query \(q_{\phi} \in Q\). If the tuple \(t \in D\) changes from \(x\) to \(y\), there are three mutually exclusive cases about \(q_{\phi}(D)\): i) increases by one \((\neg \phi(x) \land \phi(y))\), ii) decreases by one \((\phi(x) \land \neg \phi(y))\), or iii) stays the same (otherwise).

Definition 34 (Lift and Lower). A pair \((x, y) \in \mathcal{T} \times \mathcal{T}\) is said to lift a count query \(q_{\phi}\) iff \(\phi(x) = \text{false} \land \phi(y) = \text{true}\), or lower \(q_{\phi}\) iff \(\phi(x) = \text{true} \land \phi(y) = \text{false}\).

Note that one pair may lift or lower many count queries simultaneously. We now define sparse auxiliary knowledge.

Definition 35 (Sparse Knowledge). The auxiliary knowledge \(Q = \{q_{\phi_1}, q_{\phi_2}, \ldots, q_{\phi_p}\}\) is sparse w.r.t. the discriminative secret graph \(G = (V, E)\), iff each pair \((x, y) \in E\) lifts at most one count query in \(Q\) and lowers at most one count query in \(Q\).

Example 36. (Lift, Lower, and Sparse Knowledge) Consider databases from domain \(\mathcal{T} = A_1 \times A_2 \times A_3\), where \(A_1 = \{a_1, a_2\}\), \(A_2 = \{b_1, b_2\}\), and \(A_3 = \{c_1, c_2, c_3\}\) and count query constraint \(Q = \{q_1, q_2, q_3, q_4\}\) as in Figure 3.3(a). With full-domain sensitive information, any pair in \(\mathcal{T} \times \mathcal{T}\) is a discriminative secret and thus the discriminative secret graph \(G\) is a complete graph. A pair \(((a_1, b_1, c_1), (a_2, b_2, c_2))\) lifts \(q_4\) and lowers \(q_1\); and a pair \(((a_1, b_2, c_1), (a_1, b_2, c_2))\) neither lifts nor lowers a query. We can verify every pair either (i) lifts exactly one query in \(Q\) and lowers exactly one in \(Q\), or (ii) lifts or lowers no query in \(Q\). So \(Q\) is sparse w.r.t. the discriminative secret graph \(G\).

We will show that when the auxiliary knowledge \(Q\) is sparse w.r.t. the discriminative secret graph \(G = (V, E)\) in a policy \(P = (\mathcal{T}, G, \mathcal{I}_Q)\), it is possible to analytically bound the policy specific global sensitivity \(S(h, P)\). To this end, let’s first construct a directed graph called policy graph \(\mathcal{G}_P = (V_P, E_P)\) from \(P\), the count queries in \(Q\) forming the vertices, and the relationships between count queries and secret pairs forming edges.
\[ q_1 : t.A_1 = a_1 \land t.A_2 = b_1 \\
q_2 : t.A_1 = a_1 \land t.A_2 = b_2 \\
q_3 : t.A_1 = a_2 \land t.A_2 = b_1 \\
q_4 : t.A_1 = a_2 \land t.A_2 = b_2 \]

(a) \( Q = \{ q_1, q_2, q_3, q_4 \} \) 

(b) \( \mathcal{G}_P = (\mathcal{V}_P, \mathcal{E}_P) \)

Figure 3.3: Policy graph \( \mathcal{G}_P = (\mathcal{V}_P, \mathcal{E}_P) \) of databases with three attributes \( A_1 = \{a_1, a_2\}, A_2 = \{b_1, b_2\}, \) and \( A_3 = \{c_1, c_2, c_3\} \) subject to count query constraint \( Q = \{q_1, q_2, q_3, q_4\} \) and full-domain sensitive information

**Definition 37 (Policy Graph).** Given a policy \( P = (\mathcal{T}, G(V, E), \mathcal{I}_Q) \), and a sparse count constraint \( Q \), the policy graph \( \mathcal{G}_P = (\mathcal{V}_P, \mathcal{E}_P) \) is a directed graph, where

- \( \mathcal{V}_P = Q \cup \{ v^+, v^- \} \): Create a vertex for each count query \( q_\phi \in Q \), and two additional special vertices \( v^+ \) and \( v^- \).

- \( \mathcal{E}_P \): i) add a directed edge \( (q_\phi, q'_\phi) \) iff there exists a secret pair \( (x, y) \in E \) lifting \( q_\phi \) and lowering \( q'_\phi \); ii) add a directed edge \( (v^+, q_\phi) \) iff there is a secret pair in \( E \) lifting \( q_\phi \) but not lowering any other \( q_\phi' \); iii) add a directed edge \( (q_\phi, v^-) \) iff there is a secret pair in \( E \) lowering \( q_\phi \) but not lifting any other \( q_\phi' \); and iv) add edge \( (v^+, v^-) \).

Let \( \alpha(\mathcal{G}_P) \) denote the length (number of edges) of the longest simple cycle in \( \mathcal{G}_P \). \( \alpha(\mathcal{G}_P) \) is defined to be 0 if \( \mathcal{G}_P \) has no directed cycle. Let \( \xi(\mathcal{G}_P) \) be the length (number of edges) of a longest simple path from \( v^+ \) to \( v^- \) in \( \mathcal{G}_P \).

**Example 38.** (Policy Graph) Followed by Example 36, since the count query constraint \( Q \) is sparse w.r.t. the discriminative secret graph, we have its policy graph \( \mathcal{G}_P = (\mathcal{V}_P, \mathcal{E}_P) \) as in Figure 3.3(b). For example, a pair \((a_1, b_1, c_1), (a_2, b_2, c_2)\) in \( \mathcal{T} \times \mathcal{T} \) lifts \( q_4 \) and lowers \( q_1 \), so there is an edge \( (q_1, q_4) \). There is no edge from \( v^+ \) or to \( v^- \), except \( (v^+, v^-) \), because every pair in \( \mathcal{T} \times \mathcal{T} \) either lifts one query and lowers one, or lifts/lowers no query. In this policy graph \( \mathcal{G}_P \), we have \( \alpha(\mathcal{G}_P) = 4 \) and \( \xi(\mathcal{G}_P) = 1 \).
Theorem 39. Let $h$ be the complete histogram query. In a policy $P = (T, G, \mathcal{I}_Q)$, if the auxiliary knowledge $Q$ is sparse w.r.t. $G$, then we have:

$$S(h, P) \leq 2 \max\{\alpha(G_P), \xi(G_P)\},$$

If there exist two neighboring databases $(D_1, D_2) \in N(P)$ s.t. $||h(D_1) - h(D_2)||_1 = 2|T(D_1, D_2)|$ when $|T(D_1, D_2)| = \max_{(D', D'') \in N(P)} |T(D', D'')|$, then we have the equality:

$$S(h, P) = 2 \max\{\alpha(G_P), \xi(G_P)\}.$$

Proof. (sketch) Recall that the policy specific global sensitivity is defined as

$$S(h, P) = \max_{(D_1, D_2) \in N(P)} ||h(D_1) - h(D_2)||_1.$$

Direction I ($S(h, P) \leq 2 \max\{\alpha(G_P), \xi(G_P)\}$). It suffices to prove that for any two databases $D_1, D_2 \in \mathcal{I}_Q$, if $|T(D_1, D_2)| > \max\{\alpha(G_P), \xi(G_P)\}$, there must exist another database $D_3 \in \mathcal{I}_Q$ s.t. $T(D_1, D_3) \subseteq T(D_1, D_2)$, i.e., $(D_1, D_2) \notin N(P)$; and thus for any two databases $(D_1, D_2) \in N(P)$, we have $||h(D_1) - h(D_2)||_1 \leq 2|T(D_1, D_2)| \leq 2 \max\{\alpha(G_P), \xi(G_P)\}$ which implies $S(h, P) \leq 2 \max\{\alpha(G_P), \xi(G_P)\}$.

To complete the proof, we consider two databases $D_1, D_2 \in \mathcal{I}_Q$ with $|T(D_1, D_2)| > \max\{\alpha(G_P), \xi(G_P)\}$, and show how to construct the $D_3$ defined above.

First of all, for any secret pair $(s^i_x, s^i_y) \in T(D_1, D_2)$, it must lift and/or lower some count query $q_\phi \in Q$; otherwise, we can construct $D_3$ by changing the value of tuple $t$ with $t_{-id} = i$ in $D_1$ into its value in $D_2$.

To construct $D_3$, now let’s consider a directed graph $G_{D_1|D_2} = (V_{D_1|D_2}, E_{D_1|D_2})$, where $V_{D_1|D_2} \subseteq V_P$ and $E_{D_1|D_2}$ is a multi-subset of $E_P$ (i.e., an edge in $E_P$ may appear multiple times in $E_{D_1|D_2}$). $E_{D_1|D_2}$ is constructed as follows: for each $(s^i_x, s^i_y) \in T(D_1, D_2)$, i) if $(x, y)$ lifts $q_\phi$ and lowers $q_\phi$, add a directed edge $(q_\phi, q_\phi')$ into $E_{D_1|D_2}$; ii) if $(x, y)$ lifts $q_\phi$ but not lowering any other $q_\phi$, add an edge $(v^+, q_\phi)$; and iii) if...
$(x, y)$ lowers $q_\phi$ but not lifting any other $q_{\phi'}$, add an edge $(q_\phi, v^-)$. $\mathcal{V}_{D_1|D_2}$ is the set of count queries involved in $\mathcal{E}_{D_1|D_2}$.

$\mathcal{G}_{D_1|D_2}$ is Eulerian, i.e., each vertex has the same in-degree as out-degree except $v^+$ and $v^-$ (if existing in $\mathcal{G}_{D_1|D_2}$), because of the above construction and the fact that $D_1, D_2 \in \mathcal{I}_Q$. As $|\mathcal{E}_{D_1|D_2}| = |T(D_1, D_2)| > \max\{\alpha(\mathcal{G}_P), \xi(\mathcal{G}_P)\}$ (i.e., $\mathcal{G}_{D_1|D_2}$ is larger than any simple cycle or simple $v^+-v^-$ path in $\mathcal{G}_P$) and $\mathcal{G}_{D_1|D_2}$ is Eulerian, $\mathcal{G}_{D_1|D_2}$ must have a proper subgraph which is either a simple cycle or a simple $v^+-v^-$ path. Let $\mathcal{E}_{D_1\rightarrow D_2}$ be the edge set of this simple cycle/path. Construct $D_3$ that is identical to $D_1$, except that for each secret pair $(s^i_x, s^i_y)$ associated with each edge in $\mathcal{E}_{D_1\rightarrow D_2}$, the value of tuple $t$ with $t_{-id} = i$ is changed from $x$ to $y$. We can show that $D_3$ satisfies its definition, and thus the proof for Direction I is completed.

**Direction II** ($S(h, P) \geq 2 \max\{\alpha(\mathcal{G}_P), \xi(\mathcal{G}_P)\}$). Let’s first prove a weaker inequality:

$$\max_{(D_1, D_2) \in N(P)} |T(D_1, D_2)| \geq \max\{\alpha(\mathcal{G}_P), \xi(\mathcal{G}_P)\}.$$ (3.21)

It implies $S(h, P) \geq 2 \max\{\alpha(\mathcal{G}_P), \xi(\mathcal{G}_P)\}$ if the condition in (ii) of the theorem holds. Combined with Direction I, we can conclude $S(h, P) = 2 \max\{\alpha(\mathcal{G}_P), \xi(\mathcal{G}_P)\}$.

To prove (3.21), it suffices to show that for any simple cycle/$v^+-v^-$ path in $\mathcal{G}_P$, we can construct two databases $D_1$ and $D_2$ s.t. $(D_1, D_2) \in N(P)$ and $|T(D_1, D_2)|$ is its length. Consider a simple cycle $q_{\phi_1}, q_{\phi_2}, \ldots, q_{\phi_i}, q_{\phi_{i+1}} = q_{\phi_1}$. Starting with any database $D \in \mathcal{I}_Q$, let $D_1 \leftarrow D$ and $D_2 \leftarrow D$ initially. For each edge $(q_{\phi_i}, q_{\phi_{i+1}})$, from the definition of policy graphs, we can find a secret pair $(x, y) \in E(G)$ s.t. $(-q_{\phi_i}(x) \land q_{\phi_i}(y)) \land (q_{\phi_{i+1}}(x) \land -q_{\phi_{i+1}}(y))$; create two new tuples: $t_{1-id} = t_{2-id} = i$, $t_1 = x$, and $t_2 = y$; and then let $D_1 \leftarrow D_1 \cup \{t_1\}$ and $D_2 \leftarrow D_2 \cup \{t_2\}$. It is not hard to verify that finally we get two databases $D_1$ and $D_2$ s.t. $(D_1, D_2) \in N(P)$ and $|T(D_1, D_2)| = \text{cycle length}$. The proof is similar for a simple $v^+-v^-$ path. \hfill \Box

For count query constraint $Q$ that is sparse with respect to policy $P = (\mathcal{T}, G, \mathcal{I}_Q)$,
we have the following immediate corollary about an upper bound.

**Corollary 40.** In a policy $P = (T, G, \mathcal{I}_Q)$, if $Q$ is sparse w.r.t. $G$, then $S(h, P) \leq 2 \max\{|Q|, 1\}$.

Thus, drawing noise from $\text{Laplace}(2 \max\{|Q|, 1\}/\epsilon)$ suffices (but may not be necessary) for releasing the complete histogram while ensuring $(\epsilon, P)$-Blowfish privacy.

### 3.5.2 Applications

The problem of calculating $\alpha(G_P)$ and $\xi(G_P)$ exactly in a general policy graph $G_P$ is still a hard problem, but becomes tractable in a number of practical scenarios. We give three such examples: i) the policy specific global sensitivity $S(h, P)$ subject to auxiliary knowledge of one marginal for full-domain sensitive information; ii) $S(h, P)$ subject to auxiliary knowledge of multiple marginals for attribute sensitive information; and iii) $S(h, P)$ subject to auxiliary knowledge of range queries for distance-threshold sensitive information.

**Marginals and Full-domain Secrets.** *Marginals* are also called *cuboids* in data cubes. Intuitively, in a marginal or a cuboid $C$, we project the database of tuples onto a subset of attributes $[C] \subseteq \{A_1, A_2, \ldots, A_k\}$ and count the number of tuples that have the same values on these attributes. Here, we consider the scenario when the adversaries have auxiliary knowledge about one or more marginals, i.e., the counts in some marginals are known.

**Definition 41 (Marginal).** Given a database $D$ of $n$ tuples from a $k$-dim domain $T = A_1 \times A_2 \times \ldots \times A_k$, a $d$-dim marginal $C$ is the (exact) answer to the query:

```sql
SELECT A_{i_1}, A_{i_2}, \ldots, A_{i_d}, COUNT(*) FROM D
GROUP BY A_{i_1}, A_{i_2}, \ldots, A_{i_d}
```

Let $[C]$ denote the set of $d$ attributes $\{A_{i_1}, A_{i_2}, \ldots, A_{i_d}\}$. 
A marginal \([C] = \{A_{i_1}, A_{i_2}, \ldots, A_{i_d}\}\) is essentially a set of count queries \(C^q = \{q_0\}\), where the predicate \(\phi(t) := (t.A_{i_1} = a_{i_1}) \land (t.A_{i_2} = a_{i_2}) \land \ldots \land (t.A_{i_d} = a_{i_d})\), for all possible \((a_{i_1}, a_{i_2}, \ldots, a_{i_d}) \in A_{i_1} \times A_{i_2} \times \ldots \times A_{i_d}\).

Let \(\mathcal{A} = \{A_1, A_2, \ldots, A_k\}\) be the set of all attributes. For a marginal \(C\), define 
\[
\text{size}(C) = \prod_{A_i \in [C]} |A_i|,
\]
where \(|A_i|\) is the cardinality of an attribute \(A_i\). So size\((C)\) is the number of possible rows in the marginal \(C\), or the number of the count queries in \(C^q\) constructed as above.

Suppose a marginal with \([C] \subseteq \mathcal{A}\) is known to the adversary. Let \(\mathcal{I}_{Q(C)}\) denote the set of databases with marginal \(C\) equal to certain value. Recall that we want to publish the complete histogram \(h\) of a database \(D\) from a domain \(\mathcal{T} = A_1 \times A_2 \times \ldots \times A_k\). For the full-domain sensitive information, using Theorem 39, we have the global sensitivity equal to \(2 \times \text{size}(C)\).

**Theorem 42.** Let \(h\) be the complete histogram. For a policy \(P = (\mathcal{T}, G, \mathcal{I}_{Q(C)})\), where \(G\) represents the full-domain sensitive information \(S^\text{full}_{\text{pairs}}\) and \([C] \subseteq \mathcal{A}\) is a marginal, we have 
\[
S(h, P) = 2 \times \text{size}(C).
\]

**Proof.** (sketch) Consider the set of count queries \(C^q\). It is not hard to show that \(C^q\) is sparse w.r.t. the complete graph \(G\). So we can construct a policy graph from \((\mathcal{T}, G, \mathcal{I}_{C^q})\), which is a complete graph with vertex set \(C^q\). From Theorem 39, we have 
\[
S(f, P) = 2|C^q| = 2 \times \text{size}(C).
\]
The upper bound \(S(h, P) \leq 2|C^q|\) is directly from Theorem 39, and it is not hard to construct two neighboring databases to match this upper bound as \([C] \subseteq \mathcal{A}\). ∎

**Example 43.** Continuing with Example 38, note that the constraints in Figure 3.3(a) correspond to the marginal \([C] = \{A_1, A_2\}\). So from (i) in Theorem 39, we have 
\[
S(h, P) \leq 2 \times 4 = 8.
\]
The worst case \(S(h, P) = 8\) can be verified by considering the two neighboring databases \(D_1\) and \(D_2\), each with four rows: \(a_1b_1c_1\) (in \(D_1\))/\(a_1b_2c_2\) (in \(D_2\)), \(a_1b_2c_1/a_2b_1c_2\), \(a_2b_1c_1/a_2b_2c_2\), and \(a_2b_2c_1/a_1b_1c_2\).
Marginals and Attribute Secrets. Now suppose a set of $p$ marginals $C_1, \ldots, C_p$ with $[C_1], \ldots, [C_p] \subseteq \{A_1, A_2, \ldots, A_k\}$ are auxiliary knowledge to the adversary. Let $I_{Q(C_1,\ldots,C_p)}$ be the set of databases with these $p$ marginals equal to certain values. For the attribute sensitive information, if the $p$ marginals are disjoint, using Theorem 39 the global sensitivity is $2 \max_{1 \leq i \leq p} \text{size}(C_i)$.

**Theorem 44.** Let $h$ be the complete histogram. Consider a policy $P = (\mathcal{T}, G^{\text{attr}}, I_{Q(C_1,\ldots,C_p)})$, where $[C_i] \subseteq \mathcal{A}$ for any marginal $C_i$, and $[C_i] \cap [C_j] = \emptyset$ for any two $C_i$ and $C_j$. Then we have $S(h, P) = 2 \max_{1 \leq i \leq p} \text{size}(C_i)$.

**Proof.** (sketch) Consider the set of count queries $Q = C_1^q \cup \ldots \cup C_p^q$, it is not hard to show that $Q$ is sparse w.r.t. $G^{\text{attr}}$. The policy graph from $(\mathcal{T}, G^{\text{attr}}, Q)$ is the union of $p$ cliques with vertex sets $C_1^q, \ldots, C_p^q$. From Theorem 39, we have $S(h, P) = 2 \max_i |C_i^q| = 2 \max_i \text{size}(C_i)$. The upper bound $S(h, P) \leq 2 \max_i |C_i^q|$ is directly from Theorem 39, and it is not hard to construct two neighboring databases to match this upper bound as $[C_i] \neq \mathcal{A}$. \qed

Grid and Distance-threshold Secrets. Our general theorem about $S(f, P)$ can be also applied to databases with geographical information.

Consider a domain $\mathcal{T} = [m]^k$, where $[m] = \{1, 2, \ldots, m\}$. When $k = 2$ or $3$, $\mathcal{T}$ can be used to approximately encode a 2-dim plane or a 3-dim space. For two points $x, y \in \mathcal{T}$, we define distance $d(x, y)$ to be the $L^p$ distance $||x - y||_p$. For two point sets $X, Y \subseteq \mathcal{T}$, we define $d(X, Y) = \min_{x \in X, y \in Y} d(x, y)$. A geographical database $D$ consists of $n$ points, each of which is drawn from the domain $\mathcal{T}$ and may represent the location of an object.

Define a rectangle $R = [l_1, u_1] \times [l_2, u_2] \times \ldots \times [l_k, u_k]$, where $l_i \in [m], u_i \in [m]$, and $l_i \leq u_i$. A range count query $q_R$ returns the number of tuples whose locations fall into the rectangle $R$. $R$ is called a point query if $l_i = u_i$ for all $i$. 

62
In this scenario, suppose the answers to a set of $p$ range count queries are known to the adversary. So we can represent the auxiliary knowledge as $Q = \{q_{R_1}, q_{R_2}, \ldots, q_{R_p}\}$. Also, suppose we aim to protect the distance-threshold sensitive information $S_{\text{pairs}}^{d,\theta} = \{(s_x^i, s_y^i) \mid d(x, y) \leq \theta\}$ while the publishing complete histogram $h$.

Using Theorem 39, we can calculate the global sensitivity if all rectangles are disjoint, i.e., $R_i \cap R_j = \emptyset$ for any $i \neq j$, as follows. Construct a graph $G_R(Q) = (V_R, E_R)$ on the set of rectangles in $Q$: i) create a vertex in $V_R$ for rectangle $R_i$ in each range count query $q_{R_i}$ in $Q$; and ii) add an edge $(R_i, R_j)$ into $E_R$ iff $d(R_i, R_j) \leq \theta$.

We can prove that the policy specific global sensitivity equals to $2(\max\text{comp}(Q) + 1)$ when there are no point query constraints, where $\max\text{comp}(Q)$ is the number of nodes in the largest connected component in $G_R(Q)$. Note that $\max\text{comp}(Q)$ (and hence $S(h, P)$) can be computed efficiently.

**Theorem 45.** Let $h$ be the complete histogram. For a policy $P = (T, G, I_Q)$, where $T = [m]^k$, $G$ represents the distance-threshold sensitive information $S_{\text{pairs}}^{d,\theta}$ ($\theta > 0$), and $Q$ is a set of disjoint range count queries $\{q_{R_1}, q_{R_2}, \ldots, q_{R_p}\}$ with $R_i \cap R_j = \emptyset$ for $i \neq j$. We have $S(h, P) \leq 2(\max\text{comp}(Q) + 1)$. If none of the constraints are point queries, then $S(h, P) = 2(\max\text{comp}(Q) + 1)$.

### 3.6 Summary

We propose a new class of privacy definitions, called Blowfish privacy, with the goal of seeking better trade-off between privacy and utility. The key feature of Blowfish is a policy, where users can specify sensitive information that needs to be protected and knowledge about their databases which has been released to potential adversaries. Such a rich set of "tuning knobs" in the policy enable users to improve the utility by customizing sensitive information and to limit attacks from adversaries with auxiliary knowledge. Using examples of kmeans clustering, cumulative histograms
and range queries, we show how to tune utility using reasonable policies with weaker specifications of privacy. For the latter, we develop strategies that are more accurate than any differentially private mechanism. Moreover, we study how to calibrate noise for Blowfish policies with count constraints when publishing histograms, and the general result we obtain can be applied in several practical scenarios.
Case Study I: Analyzing Location Data

4.1 Introduction

The advancement of location-sensing technology such as GPS together with mobile devices has brought forth numerous location-based applications to track, record and share individuals’ locations. Long sequences of detailed location records about individuals are passively collected by organizations or actively shared by individuals. Analysis of this giant collection of location data for the benefits of individuals, business and society has been the focus of many research studies and applications. For instance, the human location patterns learned from taxi trips can help the discovery of important crossroads in a road network [127] and encourage vehicle pooling [108]. Location data has also been found predictive of human purchasing behavior [7], emergency behavior following large-scale disaster [114], and epidemiological patterns [122], and hence improves existing prediction models in many fields. These location-based studies can potentially enhance our understanding of human behavior and foster the development of tools to facilitate our life. To realize these potentials, location data has to be made available to the interested researchers or analysts.
However, this data releasing process may reveal sensitive properties of individuals.

What are the special properties about location data of individuals compared to general tabular databases that make privacy protection challenging? The first property is that individuals’ location data are highly identifiable. Montjoye et al. [36] showed using mobility data of 1.5 million individuals over a period of fifteen months that approximately 95% of the individuals in this dataset can be uniquely identified by 4 spatio-temporal points, and the uniqueness of the location data decays insignificantly as their spatial and temporal resolution coarsens, using mobility data of 1.5 million individuals over a period of fifteen months. The second property of location data is that individual’s location patterns exhibit high predictability. Song et al. [113] found a more than 90% potential predictability in the future whereabouts of each individual despite heterogeneous travel patterns among the population.

Based on these properties, what kind of privacy guarantees can we hope to achieve when releasing location trajectories? A good notion of privacy for location data should consider adversaries with background knowledge. Even if the adversary knows a small set of location points visited by an individual, these points can help the adversary uniquely identify this individual from the location data, and infer the other sensitive locations visited by this individual. Moreover, since location trajectories are highly predictable, adversaries can leverage correlations between points in a user’s trajectory to infer sensitive information even if the locations are coarsened [3, 63, 93, 131], or perturbed [8]. For instance, sensitive locations such as home and work addresses can be discerned easily based on the frequency with which locations are visited [47, 11], and perturbed locations can be reconstructed based on temporal correlations within the sequence of locations [125]. Next, linking these location records of individuals to public information can further reveal more about these individuals such as their health status based on their visits to hospitals. Ma et al. [81] showed that an adversary can infer an extended view of a user including
the true identity in an anonymous trace with a small amount of side information with high probability. Thus, it is important to consider adversaries with background knowledge (about points in the trajectory as well as other side information) when quantifying the privacy loss of a method for sharing location trajectories. Finally, many applications (especially in upcoming IoT applications) require users to share their locations multiple times, or even periodically. For every release to be useful, more information about individuals must be disclosed each time. Hence, any method for sharing locations must be able to provide privacy guarantees across multiple releases and not just one release. A graceful degradation of privacy protection is highly desired over multiple releases.

Traditional location privacy preserving practices are mainly based on anonymization. For instance, $k$-anonymity removes identifiers and coarsens data values such that each individual is indistinguishable from $k - 1$ others. However, these practices fail to achieve the privacy desiderata discussed above. It is well known that $k$-anonymous releases do not protect against adversaries with background knowledge [87]. Besides, $k$-anonymous releases do not guarantee privacy under composition; i.e., two $k$-anonymous release can be combined to learn the sensitive locations of an individual exactly. Moreover, many anonymization algorithms are susceptible to attacks like the minimality attack [123], where the decisions made by the anonymization algorithm reveal information to the attacker. Therefore, in this work, we will present a formal framework that allows the releases or analysis of location data of individuals with provable privacy guarantees that can achieve these desiderata.

In this chapter, we first present several variants of differential privacy for location data and corresponding algorithms for these variants and explain the privacy semantics underlying these notions with Pufferfish privacy and Blowfish Privacy in Section 4.2. The rest of this chapter describes an end-to-end system, DP Trajectory Synthesis (DPT), and its visual support (VisDPT), built by us to enable location
data sharing under the strongest privacy guarantee.

4.2 Provable Privacy Guarantees for Location Data

Differential privacy has arisen as a popular choice for privacy sensitive applications that use location data. We map the differential privacy definition to location data as follows. Consider $\Sigma$ as the spatial domain for the location data with a distance metric, denoted by $d(\cdot)$. The spatial domain is usually a set of latitude-longitude coordinates, or a discretized 2-dimensional space, e.g. a uniform grid over a map. A location database $D \in \mathcal{D}$ consists of individuals with their location data. Each individual $h_i \in \mathcal{H}$ has a variable $r_i$ to represent his or her location trajectories. If the events are recorded at regular time intervals, known as regular trajectories, $r_i[j]$ for $j = 1, 2, \ldots$, represents the $j$th event of individual $h_i$ which takes a location value from the spatial domain $\Sigma$ at time point $j$. Otherwise, each event has a temporal dimension in addition to the space domain, where privacy notions and techniques for regular trajectories can be adapted accordingly.

4.2.1 Variants of Differential Privacy

We will focus on regular trajectories and bounded differential privacy in this section. Neighboring databases for bounded differential privacy $N_{np}$ differ in the record value/tuple for the record $r_i$ of a single individual $h_i \in \mathcal{H}$ in the database. We can define neighboring databases in multiple ways for location data of individuals, and they result in distinct privacy notions with different levels of privacy protection. We describe these in detail below: (1) $r_i$ can differ in one event with two different location values; or (2) differ completely in all the events of a single individual; or (3) differ in a short window of consecutive events. Therefore, these choices result in three key variants of differential privacy with details shown below.
• **Event-differential privacy (Event-DP)**: In event-differential privacy, neighboring databases differ in only one single location (at a single time) of a single individual. Intuitively this definition ensures that the output of an algorithm is insensitive to changing one location at one time point. More formally,

**Definition 46** (Neighboring databases for Event-DP). Databases $D$ and $D'$ are neighbors for event-DP if they differ in a single record $r_i$ which takes values $t$ in $D$ and $t'$ in $D'$ such that

$$|t| = |t'|, \text{ and if } t[j^*] \neq t'[j^*], \text{ then } \forall j \neq j^*, t[j] = t'[j]$$

(4.1)

Algorithms designed under this privacy notion are commonly applied in the scenarios where each individual has one or few sensitive events in the database.

• **User-differential privacy (User-DP)**: In user-differential privacy, neighboring databases differ in the record of a single individual. Intuitively, this definition ensures that the output of an algorithm is insensitive to changing locations of an individual at any time point. More formally,

**Definition 47** (Neighboring databases for User-DP). Databases $D$ and $D'$ are neighbors for user-DP if they differ in a single record $r_i$ which takes values $t$ in $D$ and $t'$ in $D'$ such that

$$t \neq t'$$

(4.2)

This protection is applicable to scenarios where the entire location sequences are released [86, 90, 135, 58, 26].

• **Window-differential privacy (w-event privacy/Window-DP)**: This window-level protection takes in a privacy parameter $w$ to specify how the neighboring databases differ. Intuitively, this definition ensures that the output of an algorithm is insensitive to changing of a window of $w$ consecutive events of a single individual. More formally,
**Definition 48** (Neighboring databases for $w$-event privacy). Databases $D$ and $D'$ are neighbors for window-DP (or $w$-event privacy) if they differ in a single record $r_i$ which takes values $t$ in $D$ and $t'$ in $D'$ such that

$$\forall j_1 < j_2, \text{ if } t[j_1] \neq t'[j_1] \text{ and } t[j_2] \neq t'[j_2], \text{ then } j_2 - j_1 + 1 \leq w$$

Hence, any pairs of neighbors that differ in an event window of length at most $w$ are considered window-level neighbors. This variant is typically used in the streaming setting [70]. When the window size $w$ is 1, this definition is equivalent to the event-level differential privacy.

In summary, these variants of differential privacy ensure the output of an algorithm is insensitive to different levels of changes in location data. Based on the levels of changes, user-DP offers the strongest privacy protection, following by $w$-event privacy and event-DP. User-DP is preferred over other variants when we would like to protect the properties of the entire location trajectory, for instance, to protect the home locations of an individual since it can reappear many times, or to protect the routines of an individual. $w$-event privacy suits the scenarios where short activities of an individual such as in a day or an hour require protection. Event-DP is applicable for one-time release of a single event of an individual. Moreover, by the sequential composition theorem, an event-DP algorithm simultaneously ensures both $w$-event DP and user-DP, albeit with a much larger value of $\epsilon$. If an algorithm ensures $\epsilon$-event DP, then it also satisfies $(w \cdot \epsilon, w)$-event DP, and $(T \cdot \epsilon)$-user-DP where $T$ is the maximum possible number of events per individual in the database. Finally, user-DP implies that only a finite number of queries can be answered, while a potentially infinite number of queries could be answered under event/window-DP.
The variants of differential privacy for location data described in Section 4.2 can all be described under this framework. For instance, event-DP can be shown to be equivalent to a Pufferfish instantiation where: (i) the set of secrets are properties of an individual’s location at a single time point; and (ii) adversaries may know arbitrary prior knowledge about an individual’s location at each time point, but do not know correlations across time points (or across trajectories). On the other hand, user-DP can be shown to be equivalent to an instantiation where: (i) the set of secrets are properties of the entire trajectory; and (ii) adversaries may have arbitrary prior knowledge about a user’s trajectory (including correlations across time points), but assume that there are no correlations across trajectories.

This means that event-DP algorithms are susceptible to attacks when adversaries know constraints or correlations between consecutive locations in a trajectory. Consider a single user’s location sequence, and consider adversaries who know that the individual stayed at the same location for a long period of time, e.g. at home in the evening. Event-DP that adds noise with standard deviation about $1/\epsilon$ to the histogram counts of locations over time cannot hide the evidence of that location. While user-DP does protect against such attacks, it may be an overkill. We can use Pufferfish to design new privacy definitions that match such adversaries. For instance, if one wants to hide properties of individual time points, but handle correlations, one could use the same secrets as Event-DP, but handle more complex adversaries as defined in Example 8. This privacy is named as Event-MarkovAdversary-Privacy. There are algorithms like the Markov Quilt Mechanism (Section 2.4.3) for general Pufferfish privacy, that can ensure more privacy than event-DP, and more accuracy than user-DP, helping us better tradeoff privacy and utility.

Example 49 (Event-DP). Recall event-DP in Section 4.2 considers neighboring
databases differ in a single event. The set of discriminative pairs for event-level neighbors can be specified as

\[ S_{\text{event}}^{\text{pairs}} = \{(r_i = t, r_i = t')| h_i \in \mathcal{H}; \ t, t' \in \Sigma^*, \ |t| = |t'|, \ \forall j^*, \text{if } t[j^*] \neq t'[j^*], \text{ then } \forall j \neq j^*, \ t[j] = t'[j] \}. \] (4.3)

Hence, the policy graph of blowfish privacy considers all possible sequence of events as vertices \( V \), and adds an edge to any pair of event sequences with the same length differing in one event. This policy graph results in a set of neighboring databases for user-DP.

**Example 50** (Geo-indistinguishability). This is a special case of event-DP, where discriminative pairs differ in only one event. Additionally, secrets form pairs if the location they differ in are close to each other. More formally,

\[ S_{\text{event}, \theta}^{\text{pairs}} = \{(r_i = t, r_i = t')| h_i \in \mathcal{H}; \ t, t' \in \Sigma^*, \ |t| = |t'|, \ \forall j^*, \text{if } d(t[j^*], t'[j^*]) \leq \theta, \text{ then } \forall j \neq j^*, \ t[j] = t'[j] \}. \] (4.4)

This captures variants of event DP proposed in prior work like Geo-indistinguishability [8] and \((\theta, \epsilon)\)-location privacy [43], each discriminative secret pair differ not only in a single event, but the difference in location value of the event is bounded by a given distance \( \theta \). Compared to event-level discriminative pairs \( S_{\text{pairs}}^{\text{event}} \), \( S_{\text{pairs}}^{\text{event}, \theta} \) protects a smaller set of discriminative secret pairs with the same privacy guarantee. This results in a sparser policy graph, as fewer pairs of secrets are connected by an edge. It is easy to see that the policy graph for \( S_{\text{pairs}}^{\text{event}, \theta} \) is a subgraph of the policy graph for \( S_{\text{pairs}}^{\text{event}} \). Correspondingly, the set of neighbors protected by geo-indistinguishability is a subset of neighbors protected by event-DP. Hence, geo-indistinguishability provides a weaker guarantee than event-DP.

Besides \( G \), Blowfish privacy policy in [59] includes \( I_Q \) which denotes the set of databases that are possible under the public constraints \( Q \) that are known about the
database. The constraints in $Q$ makes a subset of the possible database instances impossible, and the rest of possible database instances are denoted by $I_Q$. The presence of the constraints will make some neighboring databases no longer possible. For instance, due to temporal constraints, certain sequences of locations are impossible. Below is an example that considers such temporal constraints.

**Example 51** ($\delta$-location set based differential privacy). The privacy definition proposed by Xiao et al [125] considers temporal constraints in the database, and these constraints are also known as the data evolution scenarios in the Pufferfish framework. The data generation model $D$ is represented by a hidden Markov model (HMM) which consists of a single transition matrix $P_\theta$ and an emission probability $P_\theta$. The prior distribution for an individual $h_i$ being at location $l$ at timestamp $j$ given the previous $(j-1)$ events $\Pr(r_i[j] = l|l_{j-1} \ldots l_1)$ can be derived from $P_\theta$ and $P_\theta^e$, and can eliminate unlikely secrets from $S$. The remaining possible locations are specified by a new term called $\delta$-location set. Formally, for any $j \in [1, 2, \ldots]$, the $\delta$-location set at time point $j$, is defined as a set containing minimum number of locations that have prior distribution sum no less than $1 - \delta$, i.e.

$$\Delta X_j = \min\{|l| \sum_l \Pr(r_i[j] = l|l_{j-1} \ldots l_1) \geq 1 - \delta\}. \quad \text{(4.5)}$$

At any time point $j$, a randomized mechanism $M$ satisfies $\epsilon$-differential privacy on $\delta$-location set $\Delta X_j$, if for any output $\omega_j$ and any two locations $l_1$ and $l_2$ in $\Delta X_j$, the following holds: $\Pr(M(l_1) = \omega) \leq e^\epsilon \Pr(M(l_2) = \omega)$.

This privacy definition is also known as $\delta$-location set based differential privacy, and can be perceived as a special case of Blowfish privacy at each timestamp, where the neighboring databases at each timestamp differ. The $\delta$-location set removes all impossible database instances based on the data evolution scenarios, and hence guarantees a stronger privacy against an adversary who knows this evolution scenario.
than an event-DP algorithm which assumes no correlation between events.

4.2.3 Algorithms Overview

We have seen three variants of differential privacy for location data. Given the same query, algorithms that satisfy these different privacy guarantees can result in different utilities. Thus, in addition to tuning $\epsilon$, one can navigate the privacy-utility tradeoff space by using these variants of differential privacy. We illustrate with the example of point queries. A point counting query across time asks for the number of events in $D$ that occur at location $l \in \Sigma$, denoted by $f(D, l)$, (across all time), where $\Sigma$ is the spatial domain \(^1\). Under event-DP, the $l_1$-sensitivity of $f(D, l)$ is just 1 as neighboring databases differ in at most one location of an individual, and hence the count for location $l$ is affected by at most 1. Adding noise drawn from $\text{Lap}(1/\epsilon)$ satisfies $\epsilon$-event-DP and the error in terms of the $l_2$ norm of the difference between the noisy answer and the true count is $\sqrt{2}/\epsilon$ in expectation. On the other hand, under user-DP, the $l_1$-sensitivity of $f(D, l)$ is $T$, where $T$ is the maximum possible number of events per individual in the database. To ensure $\epsilon$-user-DP, the noise added to the query is drawn from $\text{Lap}(T/\epsilon)$, and hence the answer has an expected error of $\sqrt{2T}/\epsilon$. Similarly, to ensure $w$-event privacy, adding noise from $\text{Lap}(w/\epsilon)$ to $f(D, l)$ is sufficient. This noise results in an expected error $\sqrt{2w}/\epsilon$, which is smaller than the error under user-DP, but larger than event-DP.

There are many interesting queries for location data, but we will focus on three important settings: (a) answering counting queries on a single snapshot in time; (b) answering counting queries in a streaming fashion; (c) synthesizing location trajectory databases. We will present corresponding algorithms for each setting.

**Answering counting queries on a single snapshot in time.** In a snapshot location database, each individual has a single location. Hence, all the three variants

\(^1\) The domain is assumed to be discrete, otherwise it can be discretized.
of differential privacy described in Section 4.2 provide equivalent privacy protections. In this setting, besides point counting queries, range counting queries are commonly asked. A range counting query asks for the number of individuals in \( D \) within rectangle \( R \subseteq \Sigma \), denoted by \( f(D, R) \). We represent a set of counting queries by \( \{f(D, l_i)\}_i \), where \( l_i \in \Sigma \), and represent a set of range counting queries by \( \{f(D, R_i)\}_i \), where \( R_i \subseteq \Sigma \). A naive way to answer all possible point and range counting queries is to first obtain a differentially private answer to all point counting queries, i.e, \( \{f(D, l)\}_{l \in \Sigma} \), using the Laplace mechanism. Then each range counting query can be answered by adding up all noisy counts of points falling into the rectangle \( R \). The summation step is a post-processing step which does not require the original data, and hence does not change the privacy guarantee. However, this approach injects too much noise to query answers. The expected error for this algorithm is \( \sqrt{\frac{8|R|}{\epsilon^2}} \), and it can easily dominate the true count, especially when the range queries span large sparse rectangles.

To improve the query accuracy, many prior work [57, 33, 105, 104, 115, 135] proposed quad-tree based solutions. A quad-tree denoted by \( T \) is built from the partitioned spatial domain, where each node of the tree, \( v \), is associated to a sub-region, denoted by \( \text{dom}(v) \) and a noisy count for the number of events in \( D \) falling into that region, denoted by \( \tilde{c}(v) \). The set of children of a node \( v \) is a partition of the region associated to \( v \). The counts stored in \( T \) can be used to answer all possible range counting queries. Given range rectangle \( R \), we first traverse \( T \) from the root and initiate the query answer 0. As traversing downwards, each node \( v \) is examined whether its associated region intersects with the query rectangle \( R \). The count of \( v \) is considered only when \( \text{dom}(v) \) intersects with \( R \). If \( \text{dom}(v) \) is fully contained in \( R \), the answer is incremented by \( \tilde{c}(v) \). If \( \text{dom}(v) \) partially intersects \( R \) and \( v \) is not a leaf node, then every child of \( v \) that is not disjoint with \( R \) will be visited. If \( \text{dom}(v) \)
partially intersects $q$ and $v$ is a leaf node, then we inspect the data points in $\text{dom}(v)$, and the answer is incremented by the number of points contained in $q$. In this way, a range counting query associated with a large rectangle can be answered with few nodes and hence this approach gives a smaller amount of noise.

Given a fixed tree height $h$, the $l_1$ sensitivity for answering all counts in $T$ is $2(h + 1)$. By the Laplace mechanism, adding noise drawn from $\text{Lap}(2(h + 1)/\epsilon)$ to the count of each node in $T$ satisfies $\epsilon$-DP. As both sensitivity and hence the amount of noise depends on the tree height $h$, existing work has made tremendous effort in improving the accuracy by exploring privacy budgeting strategy [33], correlations between noisy counts [57, 33] and pruning tree nodes [105, 104, 115] where the maximum tree height $h$ is given, or by designing algorithms independent of $h$ [135].

- **Optimizations with a given maximum tree height $h$.** Most of the prior work proposed algorithms [57, 33, 105, 104, 115] with the maximum tree height given. The first key optimization with a given maximum tree height is to distribute different privacy budget to each level of the tree [33] by applying the sequential and parallel composition of DP mechanisms. The intuition behind is that the nodes at a higher level have larger counts and hence are more resistant to perturbation while nodes at a lower level with smaller counts are less prone to noise. The baseline method that adds noise drawn from $\text{Lap}(2(h + 1)/\epsilon)$ to all nodes is equivalent to uniform budgeting by split $\epsilon$ uniformly across each level. Cormode et al.[33] aim to improve the total error injected to the tree $T$ with given height $h$, by considering the total error as the sum of the node variances. The variance of the Laplace mechanism with parameter $\epsilon_i$ is $\text{Var}(\text{Lap}(\epsilon_i)) = 2/\epsilon_i^2$. Since the noise is independently generated in each node, the total variance is $\text{Err}(q) = \sum_{i=0}^{h} 2n_i/\epsilon_i^2$, where $n_i$ is the number of nodes at height $i$ contributed to the query $q$ and $\epsilon_i$ is the privacy budget assigned to the
nodes at height \(i\). This error is minimized with the constraint that \(\sum_{i=0}^{h} \epsilon_i = \epsilon\) when \(\epsilon_i = 2^{(h-i)/3} \epsilon \frac{2^{1/3}-1}{2^{h+1/3}-1}\). This strategy corresponds to a geometric budgeting strategy where nodes at higher level receive smaller budgets, and the budget increases geometrically downwards the tree. Another popular optimization technique considers the consistency correlation between the noisy counts, thus to further reduce the total variance of the noise injected to the tree counts [57, 33]. The last optimization [105, 104, 115] is to prune nodes that have small counts and hence their descendants in the tree. This approach can introduce biased noise, but can reduce the total amount of noise with respect to the true counts.

- **Private partition without the maximum tree height.** Another research direction explores spatial partition without the specification of the maximum tree height. A recent work [135] adds a constant amount of noise (regardless of the maximum tree height) to a bias count of each node. If this noisy count is bigger than the threshold, this node will be further partitioned. After obtaining this partition, only the leaf nodes are published with their noisy counts. The intermediate nodes obtain their counts by summing up the counts of all the leaf nodes under them. This approach is the first algorithm that does not require the maximum tree height as an input. The constant amount of noise instead of height-dependent noise largely improves the accuracy of the query answer. The details of this algorithm can be referred to [135].

There are other data-dependent methods, such as kd-tree to partition the spatial domain based on other mechanisms. These algorithms can be referred to [33].

**Answering counting queries in a streaming fashion.** For infinite sequences of locations, stream counting queries have been well studied and are defined as a sequence of counting queries \((f(D[j], l))_{j=1,2,...}\) for location \(l \in \Sigma\) in database \(D\) at
time stamp $j = 1, 2, \ldots$. Event-DP is a special case of $w$-event privacy where $w = 1$. User-DP does not apply here, as there is no bound on the maximum possible length of the sequence. However, an $(\epsilon/2)$-user-DP mechanism can be applied to disjoint subsequences of the stream prefix, where each subsequence has a length $w$. This ensures $(\epsilon, w)$-event privacy, but this approach is not optimal.

Kellaris et al. [70] proposed a sliding window methodology. The overall mechanism denoted by $M$ which takes an input stream with prefix $D[1 : j]$ can be decomposed into $j$ sub-mechanisms $M_1, \ldots, M_j$. Let each $M_k$ for $k \in [1 : j]$ generate independent randomness to achieve $\epsilon_k$-event-DP. If for all $0 \leq j_2 - j_1 \leq w$, $\sum_{k=j_1}^{j_2} \epsilon_k \leq \epsilon$, then the overall mechanism $M$ satisfies $w$-event privacy. This means that if the sum of privacy budgets per sliding $w$-window is no more than $\epsilon$, the overall mechanism ensures $(\epsilon, w)$-event privacy.

With this sliding window methodology, two baselines were given by [70]: (1) uniformly allocating $\epsilon/w$ budget to each event so that the sum of the budget per sliding window is always $\epsilon$; (2) publishing a single event with privacy budget $\epsilon$ for every $w$ timestamps. The first baseline does not work well if $w$ is large as each event is given a small budget. The second baseline approximates the other unpublished counts from the released one. If the released count is very different from the others, the overall estimation is very poor. In order to address these shortcomings, the same work [70] proposed to skip publications of counts that are similar to previously released ones.

In the new solutions, each sub-mechanism $M_j$ has two parts $M_{j,1}$ and $M_{j,2}$. The first part $M_{j,1}$ differentially privately computes the distance between the current count and the preceding released counts. If the distance is small, the publication of this count is skipped; otherwise then the second part $M_{j,2}$ releases the noisy count with part of the remaining budget available for the current sliding window. There are two ways to deal with the privacy budget for events within a window: (1) budget
distribution, and (2) budget absorption. Both schemes assign some budget $\epsilon_1/w$ to $M_{j,1}$ for computing distance differentially privately at time stamp $j$, and use the remaining budget $\epsilon_2$ for publishing counts, where $\epsilon_2 = \epsilon - \epsilon_1$. We will see how the publication budget $\epsilon_2$ is spent within each sliding window.

- **Budget Distribution**: The publication budget $\epsilon_2$ is distributed in an exponentially decreasing fashion to the timestamps where a publication is decided to occur. Formally, at each timestamp, remaining budget is computed as $\epsilon_{rm} = \epsilon_2 - \sum_{k=j-w+1}^{j-1} \epsilon_{k,2}$, where $\epsilon_{k,2}$ is the privacy budget assigned to each of the last $w-1$ aggregated statistics. Then a Laplace noise with a budget of $\epsilon_{rm}/2$ is added to the query output. If a publication is skipped, its budget is saved and spent in timestamps falling outside the active window. If there are $m$ publications per window, the sequence of budget can be $\epsilon_2/2, \epsilon_2/4, \ldots, \epsilon_2/2^m$.

- **Budget Absorption**: This scheme uniformly distributes the publication budget to all timestamps. If it decides not to publish at a timestamp based on the noisy distance, the corresponding budget becomes available for future publication. If it decides to publish at a timestamp, it absorbs all the budget that became available from the previous skipped publications. This allows higher accuracy for the current statistics. To ensure the total budget within a window not exceeding the maximum $\epsilon_2$, after the absorption of budgets from previous timestamps, the same amount of budget must be nullified from the immediate succeeding timestamps.

Both mechanisms satisfy $w$-event privacy. There are no theoretic guarantees that they can do better than the baseline methods, but the experiments in [70] show their superiority over the baselines in most of their settings. In general, this sliding window methodology highly depends on the choice of $w$ for both privacy and utility. It remains an interesting question that what $w$ should be set for each application.
The algorithms discussed so far can achieve \(w\)-event privacy (and hence event-DP) in a streaming setting. There are more event-DP algorithms [22, 17, 21, 28] developed for streaming setting. Among them, only PeGaSus [28] can simultaneously support a variety of stream processing tasks – counts, sliding windows, event monitoring – over multiple resolutions of the stream, and outperform the other solutions specialized to individual queries. These event-DP algorithms can also be extended to user-DP algorithms when each user has a limited number of contributions to the streaming data. If each user contributes a count of 1 at most \(l\) times to the entire streaming setting, then an \(\epsilon\)-event-DP algorithm can automatically guarantee \(l\epsilon\)-user-DP. This assumption is valid for certain scenarios. For instance, in a hotel, most customers stay there for a few days. There are also algorithms that summarize or sample user’s information so that their contributions to the streaming data is bounded [44], but may result in poorer data quality.

**Synthesizing location trajectory databases.** Synthetic location databases are important for applications and research in city/traffic planning, epidemiology, and location-driven advertising, especially when the analysis cannot be limited to a set of counting queries. The synthetic data also keeps the same format of the true data such that data analysts do not have to adapt to a new tool for exploring the private data. Synthesizing location databases corresponds to a non-interactive setting.

To ensure user-DP, prior approaches add noises drawn from a Laplace distribution to parts of the prefix tree \(T\) [26, 135]. These prior work performed well for small domain, and can be applied to continuous spatial domains by discretizing locations (e.g. via a uniform coarse grid). However, they failed to scale to realistic location sequences that span large geographical regions. Though a sufficiently fine discretization of the spatial domain can capture all the mobility patterns in the data, this discretization results in very large domain sizes (of several tens of thousands),
and hence making the model fitting procedure very slow and overfitting the data. Moreover, the amount of noise added to ensure differential privacy also grows with the number of nodes in the tree. On the other hand, if a coarse discretization of the space is used for a small prefix tree, then much of the spatial correlation information in the original trajectories is lost. Hence, we proposed an end-to-end system, named Differentially Private Trajectories (DPT) [58] to address these challenges, as described in the following sections.

4.3 Overview of Differentially Private Trajectories (DPT)

The input to our system is a database of regular trajectories, or sequences of (latitude, longitude) pairs sampled at some uniform rate, corresponding to the movements of a set of individuals. Synthetic trajectories are generated by first fitting a probabilistic model on the data from which new trajectories can be sampled. A popular choice of model for sequential data is the order $k$ Markov process that generates the next location of trajectory based on the previous $k$ locations [66]. The sufficient statistics of this model are given by a prefix tree of depth $k+1$, where nodes at the $i^{th}$ level of the tree record the number of times sequences of $i$ consecutive locations corresponding to the path from the root of the tree up to the node at level $i$ appear in the database. Recent work has proposed a differentially private approach to fit a Markov model by carefully adding noise drawn from a Laplace distribution to parts of the prefix tree [26]. This approach was shown to work well when the domain of locations is small, and can be applied to continuous spatial domains by discretizing locations (e.g., via a uniform coarse grid).

However, the aforementioned approach fails to scale to realistic trajectory databases that span even moderately large geographical regions. Note that the size of the prefix tree is $O(|\Sigma|^{k+1})$, where $\Sigma$ is the domain of possible locations. A sufficiently fine discretization of the spatial domain that captures all the mobility patterns in the
data could result in very large domain sizes (of several tens of thousands) making the model fitting procedure not only very slow, but also overfitting the data. As the number of nodes in the tree increases, the amount of noise added to ensure differential privacy also grows. On the other hand, while a coarse discretization of the space results in a small prefix tree, much of the spatial correlation information in the original trajectories is lost.

This is quantified in Figure 4.1. We considered the movement of taxi cabs in a large metropolitan area [2] spanning a geographical region of roughly 34km × 40km. We discretized this space using uniform grids with varying resolutions of 200m, 400m, 800m, 1600m and 3200m. The number of cells in the discretized domain under the 200m grid is 34000 while the size under the 3200m grid is less than 150. We synthesized trajectories under each of these discretizations using the method of Chen et al. [26] (which we call seq), and measured the Jensen Shannon divergence (JSD) between the distribution of the diameter, i.e., the distance traveled by trajectories, in the original database to that of the trajectories in the synthetic database. In all cases, the divergence is in the range [0.33, 0.59] (a high value considering that the
maximum possible JSD is \( \ln(2) = 0.69 \). Furthermore, for an input of over 4 million trajectories, the time taken to compute the relevant prefix trees and then reconstruct a synthetic trajectory database is at least 3 hours!

Our system Differentially Private Trajectories (DPT) is a scalable end-to-end system for synthesizing regular trajectories with provable privacy properties. DPT can synthesize trajectories spanning large geographical areas with significantly more utility (JSD error on the diameter of less than 0.05) than Chen et al.’s method and is orders of magnitude faster (fitting the model and sampling trajectories take on the order of 100 seconds for an input of over 4 million trajectories). Figure 4.2 gives a schematic overview of the steps of DPT, which incorporates the following key novel ideas:
• **Hierarchical Reference Systems:** DPT discretizes the spatial domain at multiple resolutions (see Step 1 in Figure 4.2) using a hierarchy of reference systems and maintains one prefix tree for each resolution (Step 2). Different reference systems capture movements at different speeds. Within each reference system individuals are restricted to move from each point to only a small number of neighboring points in one step. Thus, while there are a larger number of prefix trees, each tree has a much smaller branching factor resulting in an exponential reduction in the number of counts maintained by the model.

• **Model Selection:** The set of reference systems (corresponding to resolutions at which the spatial domain is discretized) for which prefix trees are materialized, and the depth of the prefix trees are hard to set without looking at the trajectory database. Materializing trees with too large a depth or at resolutions that do not correspond to viable speeds can significantly hurt the accuracy of the system. Hence, DPT uses a novel model selection algorithm (Step 3) to set these parameters in a differentially private manner. As in prior work, noise drawn from the Laplace distribution is added to the chosen prefix trees (Step 4), and these noisy trees are pruned using an adaptive strategy (Step 5) to further improve utility.

• **Direction Weighted Sampling:** Most real world trajectories have an inherent directionality. This is captured to some extent by the Markov model, and trajectories sampled from the true model tend to retain this property. However, this directionality could be lost due to the noise added to the counts of the prefix trees in the private model. In line with prior work on enforcing constraints in differentially private query answering [57], we present a novel postprocessing strategy to restore directionality while sampling synthetic trajectories from the noisy model.
Table 4.1: Notation Summary for DPT

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$</td>
<td>Trajectory database</td>
</tr>
<tr>
<td>$PT$</td>
<td>Personal Trajectory table of an individual</td>
</tr>
<tr>
<td>$t$</td>
<td>A single trajectory</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>A set of latitude-longitude coordinates</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Latitude-longitude coordinate of a location</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>Reference system, a set of anchor points</td>
</tr>
<tr>
<td>$a$</td>
<td>Anchor point in reference system $\Sigma$</td>
</tr>
<tr>
<td>$\top$</td>
<td>Starting symbol</td>
</tr>
<tr>
<td>$\bot$</td>
<td>Ending symbol</td>
</tr>
<tr>
<td>$\Sigma^k$</td>
<td>k-gram pattern</td>
</tr>
<tr>
<td>$c(t, x)$</td>
<td>The number of occurrences of $x \in \Sigma^k$ in $t$</td>
</tr>
<tr>
<td>$c(PT, x)$</td>
<td>The number of occurrences of $x \in \Sigma^k$ in $PT$</td>
</tr>
<tr>
<td>$c(D, x)$</td>
<td>The number of occurrences of $x \in \Sigma^k$ in $D$</td>
</tr>
</tbody>
</table>

Section 4.4 presents our hierarchical reference system set model. Section 4.5 describes our differentially private techniques learning the model from the data, including model selection and direction weighted sampling. We present a thorough experimental evaluation in Section 4.6, using real and model-generated data sets.

4.4 Hierarchical RSS Model for DPT

We next describe a Markov model for mobility trajectories and build up to our novel hierarchical reference system set (RSS) model.

4.4.1 Notation and Background

Given a spatial domain $\Sigma$, a regular trajectory $t$ is a sequence of locations ($\sigma_i$) observed and recorded at regular time intervals, where $\sigma_i \in \Sigma$ for $i = 1, 2, \ldots$. The spatial domain is usually a set of latitude-longitude coordinates. We consider a trajectory database $D$ consisting of $|D|$ individuals. Each individual has a personal trajectory table, denoted by $PT$ with each tuple representing a regular trajectory, $t$. It is possible for an individual to have multiple trajectories, of varying lengths, where length is defined as the number of points in the trajectory. We call any sequence $x \in \Sigma^k$ of $k$ locations as a $k$-gram. If $x$ is a $k$-gram and $y$ is an $\ell$-gram,
then we denote by $xy$ the $(k + \ell)$-gram obtained by appending the sequence $y$ to sequence $x$. We denote by $c(t, x)$ the number of occurrences of $x$ in $t$, $c(PT, x)$ the total number of occurrences of $x$ in the trajectory table $PT$, i.e. $c(PT, x) = \sum_{t \in PT} c(t, x)$, and $c(D, x)$ the total number of occurrences of $x$ in the database $D$, i.e. $c(D, x) = \sum_{PT \in D} c(PT, x)$. We summarize the notation of the paper in Table 4.1.

Markov processes [95] to model correlations between contiguous locations in a regular mobility trajectory are defined as follows.

**Definition 52** (Markov Process). A regular trajectory $(\sigma_1 \sigma_2 \cdots \sigma_n) \in \Sigma^n$ is said to follow an order $\ell$ Markov process if for every $\ell \leq i < n$, $\sigma \in \Sigma$

\[
\Pr[\sigma_{i+1} = \sigma | \sigma_1 \cdots \sigma_i] = \Pr[\sigma_{i+1} = \sigma | \sigma_{i-\ell+1} \cdots \sigma_i] \quad (4.6)
\]

We refer to the probability $\Pr[\sigma_{i+1} = \sigma | \sigma_{i-\ell+1} \cdots \sigma_i]$ as a transition probability of the Markov process. The collection of transition probabilities for all $x = \sigma_{i-\ell+1} \cdots \sigma_i \in \Sigma^\ell$ can be estimated using the set of all $\ell$- and $\ell+1$-gram counts, i.e.

\[
\Pr[\sigma_{i+1} = \sigma | \sigma_{i-\ell+1} \cdots \sigma_i] = \frac{c(D, x\sigma)}{c(D, x)} \quad (4.7)
\]

Typically, a starting symbol ($\top$) and a stopping symbol ($\bot$) are prepended and appended (respectively) to the original trajectories so that the starting and stopping probabilities are captured in the model (and are treated as special locations in $\Sigma$).

A synthetic trajectory is initialized with a starting symbol ($\top$) and is continuously sampled by picking the next location from the Markov process till reaching the stopping symbol ($\bot$). Formally, let $x = (\top \sigma_1 \cdots \sigma_i)$ denote the prefix of a trajectory sampled so far and $x'$ the longest suffix of $x$ of length $\ell \leq (k - 1)$, where $(k - 1)$ is the maximum order of Markov process considered. The next location is chosen using the transition probability given $x'$ in the order $\ell$ model. Thus, sampling synthetic trajectories requires us to maintain all $\ell$-gram counts for $1 \leq \ell \leq k$. These counts
can be stored in a prefix tree $T$ of height $k$ — nodes in $T$ are $\Sigma^1 \cup \cdots \cup \Sigma^k$, and edges connect each $\ell$-gram $x$ to an $(\ell + 1)$-gram $x\sigma$, for all $\sigma \in \Sigma$.

4.4.2 Discretization and Reference Systems

A trajectory is usually recorded as a sequence of points in some continuous domain $\Sigma$, e.g. latitude-longitude coordinates. One common way to analyze trajectories on a continuous domain (and to limit the size of the model) is via discretization of the space using a reference system [116].

**Definition 53** (Reference System). A reference system (RS) is a set of anchor points $\Sigma \subset \Sigma$, associated with a mapping function $f : \Sigma \to \Sigma$.

In this work, we use a reference system constructed by imposing a uniform grid over the space and choosing the centroids as anchor points. We denote such a reference system by $\Sigma_v$, where $v$ denotes the resolution or the length of the side of each grid cell. Varying the resolution $v$ results in different reference systems; coarse grids (large $v$) correspond to fewer anchor points, and fine grids (small $v$) result in a larger number of anchor points. The function considered here for the reference system $\Sigma_v$ is represented by $f_v$ and maps a point in $\Sigma$ to its closest anchor point in $\Sigma_v$ (if a point is equidistant to more than one anchor point, ties are broken in a consistent way).

There are of course other methods for instantiating reference systems apart from grids, such as using points of interest (POI) or intersections on a road network as anchors. These methods can also be adopted within the DPT system directly, but for clarity we frame the subsequent discussion in terms of grids.

4.4.3 Hierarchical Reference Systems

Even with the use of a reference system $\Sigma$, the number of parameters (transition probabilities) to specify a Markov model can be very large. The model requires
\[|\Sigma|^{k+1}\] counts (for large \(|\Sigma|\)) in a prefix tree \(T\) of height \(k\) (i.e., \(\ell\)-gram counts for \(1 \leq \ell \leq k\)). For instance, if \(\Sigma\) is a coarse \(16 \times 16\) grid partitioning the space into 256 cells, then for \(k = 3\), \(|\Sigma|^{k+1} \approx 4 \cdot 10^9\). Yet, grids of much finer resolution are needed to model all the transitions in the data.

Therefore, rather than building a Markov model using a single reference system ([26, 27, 102, 116]), we adopt a novel approach in DPT. First, given a reference system \(\Sigma\), we choose to estimate transition probabilities of \(\ell\)-grams, \(x \in \Sigma^\ell\) to only the anchor points that are “close” to the last observed location in \(x\). In the context of a uniform grid based reference system \(\Sigma_v\), we only consider transitions from a grid cell \(a\) to either itself or to its 8 adjacent cells. We call them the \textit{neighboring} cells. All other transition probabilities are forced to be 0. Under this constraint, we only need to maintain \(|\Sigma_v|\) 1-gram counts, \(9 \cdot |\Sigma_v|\) 2-gram counts, \(9^2 \cdot |\Sigma_v|\) 3-gram counts and so on, thus reducing the total number of counts in the prefix tree \(T\) from \(|\Sigma_v|^{k+1}\) to \(O(9^k \cdot |\Sigma_v|)\).

However, restricting transitions to only neighboring anchor points cannot capture all the transitions in the original trajectories under any single reference system. This is because, even within a single trajectory, objects tend to move at different speed ranges. For instance, a taxi may travel at greater speeds on a highway, but with
much slower speeds in downtown, and both types of driving can occur in a single trip. Hence, we use a set of reference systems to capture motion at different speed ranges, and transitions in each of these reference systems occur between anchor points that are close.

**Definition 54** (Hierarchical Reference Systems). Let $\Sigma_v$ denote the set of centroids of cells in a uniform grid of resolution $v$. The reference system set $RSS = \{\Sigma_{v_1}, \ldots, \Sigma_{v_M}\}$ is a set of hierarchical reference systems, where $v_1 < \cdots < v_M$. For any anchor point $a \in \Sigma_{v_m}$, we define (1) its parent in $\Sigma_{v_{m'}}$, where $m' > m$ as the closest point to $a$ in $\Sigma_{v_{m'}}$,

$$\text{par}(a, \Sigma_{v_{m'}}) = \arg\min_{a' \in \Sigma_{v_{m'}}} d(a', a)$$

and (2) its children in $\Sigma_{v_{m'}}$, where $m' < m$ as

$$\text{children}(a, \Sigma_{v_{m'}}) = \{a' \in \Sigma_{v_{m'}} | \text{par}(a', \Sigma_{v_{m'}}) = a\}.$$

If multiple anchor points can be the parent of a single anchor point, ties are broken in a consistent way. In this paper we consider a set of reference systems with geometrically increasing resolutions. That is, we consider $\{\Sigma_{v_1}, \ldots, \Sigma_{v_M}\}$ such that for all $1 < m \leq M$, $v_m/v_{m-1} = 2$.

This set of hierarchical reference systems $RSS = \{T_{v_1}, \ldots, T_{v_M}\}$ can capture motions at different speed ranges in the raw trajectory $t$ — long steps (pairs of consecutive points which are far away) are mapped with coarser reference systems while shorter steps are mapped with finer reference systems. We assume the longest step in the whole database is less than $v_M$. Given an RSS, we can map a raw trajectory $t$ to an RSS-based trajectory $t_{rs}$ that contains anchor points from the RSS, such that any pair of consecutive anchor points $a_{i-1}a_i$ in the output trajectory $t_{rs}$ are either (1) neighboring cells or the same cell if they are from the same reference system $\Sigma_{v_m}$, i.e. $d(a_{i-1}, a_i) \leq v_m$, where $d(\cdot, \cdot)$ is the $L_\infty$-norm of the coordinates difference between $a_{i-1}$ and $a_i$; or (2) in a parent-child relationship if they are from different reference
systems, i.e. \( a_{i-1} = \text{par}(a_i, \Sigma_{v_m}) \) or \( a_i = \text{par}(a_{i-1}, \Sigma_{v_m}) \). In this way, the number of transitions maintained is controlled by the number of neighboring cells in each reference system, and changes in speed are captured by the transitions across different reference systems. The trajectory \( t_{rs} \) then becomes a sequence of segments, where a segment is defined as contiguous anchor points from the same reference system. We introduce a new start symbol \( T_{vm} \) as a special location in each reference system \( \Sigma_{v_m} \) to indicate the start of a segment falling into that reference system.

**Example 55.** Figure 4.3 considers four grid-based reference systems \( \Sigma_{v_1}, \Sigma_{v_2}, \Sigma_{v_3}, \Sigma_{v_4} \), where \( v_1 = v_2/2 = v_3/4 = v_4/8 \). Each point is represented in the form of \((x, y)_m\), where \( m \) indicates that the reference system is \( T_{vm} \), and \( x \) and \( y \) are the horizontal and vertical indexes in \( T_{vm} \). The trajectories shown here refer to the same trajectory. The first four points in \( t \) are mapped to anchor points in \( \Sigma_{v_2} \) as \((3,0)_2(4,0)_2(4,1)_2(5,1)_2\) (neighboring cells in \( \Sigma_{v_2} \)), as each step crosses more than one cell in \( \Sigma_{v_1} \). Similarly, the next step after \((5,1)_2\) crosses more than one cell in \( \Sigma_{v_2} \), and hence a coarser reference system \( \Sigma_{v_3} \) is used for the mapping, resulting in the anchor point \((2,0)_3\). Though the next step after \((2,0)_3\) could be mapped to a finer reference system \( \Sigma_{v_2} \) as
(7, 1)\(2\)(8, 1)\(2\), this is not preferable, since the trajectory is not slowing down. Greedily changing the reference system to a finer resolution is not favored. Hence, we adopt the rule that only when at least 3 consecutive points are mapped to the same cell of the current reference system do we change to a finer reference system. This gives more contiguous points in \(\Sigma_{v_3}\): \((2, 0)_3(3, 0)_3(4, 0)_3(5, 0)_3(6, 1)_3(6, 1)_3(6, 1)_3\). As the last three points remain in the same cell of \(\Sigma_{v_3}\), this implies slowing down, and so the following points are mapped to a finer reference system \(\Sigma_{v_2}\) as \((13, 3)_2(13, 4)_2(13, 5)_2(14, 6)_2\).

This results in three segments: 

(1)\(\top\)(3, 0)\(2\)(4, 0)\(2\)(2, 1)\(2\)(2, 0)\(3\);  
(2)\(\top\)\(v_3\)(2, 0)\(3\)(3, 0)\(3\)(4, 0)\(3\)(5, 0)\(3\)(6, 1)\(3\)(6, 1)\(3\)(6, 1)\(3\)(13, 3)_2;  
(3)\(\top\)\(v_2\)(13, 3)\(2\)(13, 4)\(2\)(13, 5)\(2\)(14, 6)\(2\)\(\bot\).

Algorithm 1: RSS-based Transformation

| Input: | \(t = \sigma_1 \sigma_2 ... \sigma_{|t|}\): regular trajectory,  
| RSS = \{\(\Sigma_{v_1}, \ldots, \Sigma_{v_M}\)\}: \(M\) hierarchical reference systems. |
| Output: | \(t_{rs}\): sequence of anchor points in RSS |

1: Initialize \(t_{rs} = (\top)\) // start symbol  
2: Initialize optimal reference system index \(m^* = 1\)  
3: for \(i = 2 : |t|\) do  
4: \(m' \leftarrow\) finest reference system \(\Sigma_{v_m}\) s.t.  
\[
\delta(f_{v_m}(\sigma_{i-1}), f_{v_m}(\sigma_i)) \leq v_m
\]
5: if \(v_{m'} \geq v_{m^*}\) then  
6: \(m^* \leftarrow m'\) // coarser or the same reference system  
7: else if \(f_{v_m^*}(\sigma_i) = f_{v_m^*}(\sigma_{i-1}) = f_{v_m^*}(\sigma_{i-2})\) then  
8: // stay in the same cell (assume \(\sigma_0 = \sigma_1\))  
9: \(m^* \leftarrow m'\) // finer reference system  
10: Add \(f_{v_m^*}(\sigma_{i-1})\) to \(t_{rs}\)  
11: Add \(f_{v_m^*}(\sigma_{|t|})\) and \(\bot\) to \(t_{rs}\) // last location and end symbol  
12: return \(t_{rs}\)

Algorithm 1 summarizes our proposed transformation process. This process starts with adding start symbol \(\top\) to indicate the beginning of a trajectory in Line 1. \(m^*\) represents the optimal reference system used for the current mapping when iterating through the sequence of points in Line 3-10. Line 4 specifies that every step (pair of consecutive points) in the raw trajectory is mapped to neighboring cells \((a_1, a_2)\) within the same reference system at the most detailed resolution. This ensures that
the finest resolution at the same or coarser level is used (lines 5-6). However, this greedy mapping may lead to many short segments. Hence, in Lines 7-9, before updating the optimal reference system $\Sigma_{v_m}$ to a finer reference system $\Sigma_{v_{m'}}$, we check if both current and previous steps are sufficiently small (the three consecutive anchor points in the previous optimal reference system are the same). The intuition for this is that if the travelling speed is halved to the previous speed range $v_{m'}$, at least 2 consecutive steps may fall into the same cell in the current reference system because of the geometrically increasing setting of RSS, i.e. $v_m/v_{m-1} = 2$.

### 4.4.4 Hierarchical RSS Model

We now present a probabilistic generative model for trajectories based on hierarchical reference systems $RSS = \{\mathcal{T}_{v_1}, \ldots, \mathcal{T}_{v_M}\}$. Let $x = (a_1 \cdots a_i)$ denote the prefix of a trajectory that has been mapped to RSS using Algorithm 1. Thus $x$ may contain anchor points from $\mathcal{T}_{v_1} \cup \cdots \cup \mathcal{T}_{v_{M}}$. Given $x$, there are two ways to extend $x$ to the next anchor point $a_{i+1}$ where (1) $a_{i+1}$ is a neighboring cell of $a_i$ in the same reference system; or (2) $a_{i+1}$ is the parent or a child of $a_i$ in a different reference system. The count for each extension is maintained in a set of prefix trees $\{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$, where each $\mathcal{T}_m$ has a height $k$, similar to the basic Markov process described in Section 4.4.1. We name this set of prefix trees as a forest $\mathcal{F}$ or forest counting query as defined in Definition 56. Nodes in the prefix tree $\mathcal{T}_m$ correspond to $\ell$-grams in $\Sigma_{v_m}$ for $1 \leq \ell \leq k$, as well as $\ell$-grams appended with a move to a different reference system $m' \neq m$. Thus, every node in $\mathcal{T}_m$ has 9 children representing moves to neighboring nodes in the same reference system (same speed range), and $M$ children representing moves to different reference systems including $\bot$ as $\mathcal{T}_{v_0}$ (change speed range). In Example 57, we give more details on $\mathcal{F}$ built from the trajectory shown in Example 55. Then we will continue illustrating how to sample trajectories from $\mathcal{F}$ in the next section (Section 4.4.5).
Definition 56 (Forest Counting Query $F(\cdot)$). The term ‘Forest counting query’ refers to the set of counting queries on the database $D$ to build model $F$, denoted by

$$F(D) = \{c(D, x) | x \in \mathcal{T}, \mathcal{T} \in F\}.$$  

(4.8)

Example 57. Given the hierarchical reference systems in Example 55, $RSS = \Sigma_{v_1}, \Sigma_{v_2}, \Sigma_{v_3}, \Sigma_{v_4}$, we build a forest consisting of 4 prefix trees: $F = \{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4\}$. In Figure 4.4, we materialize some of the nodes in $\mathcal{T}_2$ whose counts are partially contributed by the segments resulted from the transformation of trajectory $t$ in Figure 4.3. The highest level of $\mathcal{T}_2$ consists of all the anchor points in $\Sigma_{v_2}$, the starting symbol $\top$, and the start of a segment $\top_{v_2}$. The 1-grams listed in Figure 4.3 add counts to their corresponding nodes in $\mathcal{T}_2$. For instance, $(4,1)_2$ increments the count of node $(4,1)$. All the nodes except $\top$ and $\top_{v_2}$ have two types of children: those that remain with the same step size $v_2$, and those with a different step size from their parent. For instance, the 2-gram $(4,1)_2(5,1)_2$ corresponds to the node $(4,1)(5,1)$ which stays in the same reference system $\Sigma_{v_2}$; the 2-gram $(14,6)_{2\bot}$ corresponds to the node $(14,6)_{\Sigma_{v_0}}$ which changes from $\Sigma_{v_2}$ to $\Sigma_{v_0}$ (a stop) and the 3-gram $(4,1)_2(5,1)_2(2,0)_3$ corresponds to the node $(4,1)(5,1)v_3$ which changes from $\Sigma_{v_2}$ to $\Sigma_{v_3}$. $\top$ indicates the start of a trajectory and $\top_{v_2}$ indicates the start of a segment (changing from another reference system). They could be appended by any anchor points in $\Sigma_{v_2}$. The 2-gram $\top(3,0)_2$ and $\top_{v_2}(13,3)_2$ increment the counts of nodes $\top(3,0)$ and $\top_{v_2}(13,3)$ in $\mathcal{T}_2$ respectively.

4.4.5 Sampling Trajectories

Given a forest $F = \{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$ of height $k$ built upon $RSS = \{\mathcal{T}_{v_1}, \ldots, \mathcal{T}_{v_M}\}$, trajectories are sampled independently of each other. We sample the starting reference system $\mathcal{T}_{v_m}$ using probabilities proportional to the count of start symbol $\top$ in each $\mathcal{T}_m$. Let $x = (\top a_1 \cdots a_t)$ denote the prefix of the trajectory generated thus far, and
Given $x'$, the next location is chosen in two steps: (1) choose a reference system $\mathcal{T}_{vm}$ from RSS or stopping symbol $\perp$ (represented by $\mathcal{T}_{v0}$), and then (2) transition to a neighboring cell of $a_i$ in the chosen $\mathcal{T}_{vm}$. Both probabilities depend on the immediate history (of $\ell \leq k$ events) in the same reference system as $a_i$. Thus, the first probability can be estimated from $\mathcal{F}$ as:

$$\Pr[\mathcal{T}_{vm} | x] = \frac{c(D, (x'\mathcal{T}_{vm}))}{c(D, x')}$$

(4.9)

c($D, (x'\mathcal{T}_{vm})$) is the number of trajectories containing $x'$ followed by a move in reference system $\mathcal{T}_{vm}$, for $m \leq M$. Given $x'$, we first sample the next reference system $\mathcal{T}_{vm}$ based on the transition probability stated in Eqn. (4.9).

For a neighboring cell $a$ in the new reference system $\mathcal{T}_{vm}$, the second probability can be estimated as:

$$\Pr[a | x, \mathcal{T}_{vm}] = \frac{c(D, (x'a))}{c(D, x')}$$

(4.10)

If a move to the same reference system is sampled, sampling continues using the first 9 children of $x'$; if a move to a coarser reference system is sampled, sampling continues using the ancestor of the last location in $x'$; if a move to a finer reference system is sampled, sampling continues using one of the children of the last location in $x'$. For the last case, rather than uniformly sampling a child of the last location $a_i$, we sample $a_{i+1}$ from the children of $a_i$ in the new reference system $\Sigma_{vm}$ using probabilities proportional to the count of $\perp_{vm}a_{i+1}$ in $\mathcal{T}_m$. The trajectory generation is stopped when a maximum length $L_{max}$ is reached, or if a stopping symbol $\perp$ is sampled as the reference system $\Sigma_{v0}$.
4.5 Private Model Learning for DPT

Two aspects of the RSS model need to be learnt: (i) the structure of the model, i.e., the set of reference systems to be included in the model, and the maximum height of the prefix trees for the chosen reference systems, and (ii) the transition probabilities, within and across the chosen reference systems. Transition probabilities are learnt from the data by estimating counts in corresponding prefix trees. Differentially private methods for prefix tree construction have been studied in prior work [26, 27], and our prefix tree construction algorithm (Section 4.5.3) follows the approach in prior work with a few notable additions. We normalize the counts in the original trajectory table of each individual to bound the sensitivity (see Defn 59) of the forest counting query. Adding noise to subtrees of the prefix tree with low counts can lead to low signal to noise ratios. We propose an adaptive pruning strategy to eliminate such subtrees, which outperforms prior prefix tree pruning strategies.

The main privacy contributions of our work, however, are (a) a novel differentially private model selection algorithm for learning the structure (Section 4.5.2), and (b) a direction weighted sampling strategy that recovers the global directional properties from the data that might be lost due to the noisy transition probability learning phase (Section 4.5.4). Structure learning is crucial, since the set of appropriate reference systems is not known apriori. A range of candidate reference systems that capture most of the transitions can be learnt purely using domain knowledge (of typical speeds) and without access to the data. However, as we show in our experiments, even within this range, only a few of the reference systems may have sufficient data support. Hence, structure learning based on the data can significantly improve the signal to noise ratio, and thus the utility of the learnt model.

In this section, we summarize the model of differential privacy and describe our novel model selection algorithm, followed by a brief overview of our methods for
estimating transition probabilities privately. We then outline direction weighted sampling. We conclude with an end-to-end privacy evaluation of the system.

4.5.1 Differential Privacy

We define \( \epsilon \)-differential privacy [40] in the context of trajectory databases. Let \( D_1, D_2 \) be two neighboring trajectory databases, i.e. \( D_1 \) and \( D_2 \) differ in only one individual’s trajectory table \( PT \), written as \( \|D_1 - D_2\| = 1 \). This paper consistently takes this to mean that \( PT \) is present in only one of the two databases.

Definition 58 (Differential Privacy). Let \( D_1, D_2 \) be two neighboring databases. Let \( M \) denote a randomized algorithm over databases, and \( O \) be a possible output of \( M \). Mechanism \( M \) is said to be \( \epsilon \)-differentially private, for \( \epsilon > 0 \), if

\[
\Pr[M(D_1) = O] \leq \exp(\epsilon) \Pr[M(D_2) = O]. \tag{4.11}
\]

In our setting, differential privacy guarantees that no individual’s trajectory table can significantly affect the released information — the output distribution generated by \( M \) is nearly the same, whether or not the individual’s trajectory table is present in the database. We make use of the concept of global sensitivity [40].

Definition 59 (Global Sensitivity). The global sensitivity of a function \( q : D \to \mathbb{R}^d \), denoted by \( S(q) \) is defined as the largest L1 difference \( \|q(D_1) - q(D_2)\|_1 \), where \( D_1 \) and \( D_2 \) are neighboring databases that differ in one personal trajectory table. More formally,

\[
S(q) = \max_{D_1, D_2 : \|D_1 - D_2\| = 1} \|q(D_1) - q(D_2)\|_1 \tag{4.12}
\]

One common technique used to achieve differential privacy is the Laplace mechanism proposed in [40]:

Definition 60 (Laplace Mechanism). The Laplace mechanism privately computes a function: \( q : D \rightarrow \mathbb{R}^d \) by computing \( q(D) + \eta \), where \( \eta \in \mathbb{R}^d \) is a vector of independent
Algorithm 2: Private Model Selection

**Input:** $D$: database, $F_{\text{full}}$: full model of size $M$, $\epsilon$: total privacy budget, $\epsilon_s$: privacy budget for model selection, $k_{\text{max}}$: the maximum tree height

**Output:** $(F^{**}, k)$: a subset of $F_{\text{full}}$ of order $k$

1: Count moves in each reference system: $(\bar{c}_1(D), \ldots, \bar{c}_M(D))$
2: $(\tilde{c}_1(D), \ldots, \tilde{c}_M(D)) \leftarrow (c_1(D), \ldots, c_M(D)) + \text{Lap}(1/\epsilon_s)^M$
3: $M^* \leftarrow$ largest index $m$ in $\{T_1, \ldots, T_M\}$ s.t. $\tilde{c}_m(D) \geq \frac{2}{\epsilon^2}$
4: Initialize $k = k_{\text{max}}$, $F^{**} = \{T_1, \ldots, T_M\}$
5: $\epsilon_r = \epsilon - \epsilon_s$
6: while $k \geq 2$ do
7: Compute $N_m(\epsilon_r, k)$, $\tilde{C}_m(D, k)$ $\forall T_m \in F^{**}$
8: $F^{**}(\epsilon_r, D) = \text{argmin}_{F^{**}} \text{Error}(F^{**}, D)$
9: if $F^{**}(\epsilon_r, D) = \emptyset$ then
10: return $(F^{**} \cup \{T_{M^*}\}, k)$
11: else
12: $F^{**} = F^{**}(\epsilon_r, D)$
13: $k = k - 1$
14: return $(F^{**} \cup \{T_{M^*}\}, k)$

random variables, where $\eta_i$ is drawn from the Laplace distribution with parameter $S(q)/\epsilon$. That is, $\Pr[\eta_i = z] = x e^{-x \epsilon / S(q)}$.

4.5.2 Private Model Selection

We would like our hierarchical reference system set to be full, i.e., every move in the trajectory dataset $D$ can be captured as a move between neighboring cells in some reference system. Based on domain knowledge one can construct a full reference system set $\{\Sigma_{v_1}, \ldots, \Sigma_{v_M}\}$ in a data independent fashion (for instance, by picking $v_m$'s as powers of 2 that capture the smallest and largest step sizes). We denote the forest $\mathcal{F}$ built on the full model by $\mathcal{F}_{\text{full}}$. Moreover, we would also like the trees in the forest to have height $k$ large enough to cover high order correlations.

However, choosing the full model or large $k$ is not desirable under the differential privacy regime: the total variance of the counts in the noisy forest grows linearly in the number of trees and quadratically in the height $k$. Thus, the variance introduced by the Laplace mechanism can drown out the signal for large forests.

Our approach to solve this problem is to drop some of the trees (resulting in
\( \mathcal{F}^+ \subset \mathcal{F}_{\text{full}} \) and/or choose a small height \( k \) such that the least bias is introduced into the model (since some of the transitions may no longer be accurately captured). If by introducing a little bias, we can dramatically reduce the variance, the total error in the noisy forest \( \mathcal{F}^+ \) will still be smaller than the error in \( \mathcal{F}_{\text{full}} \).

Hence, we would like to identify \( \mathcal{F}^+ \) and \( k \) which minimize the sum of the variance and bias. The bias of a model arises from the set of trees not selected in the model. We represent this set by \( \mathcal{F}^- \), i.e. \( \mathcal{F}^- = \mathcal{F}_{\text{full}} - \mathcal{F}^+ \). Though no noise is added to \( \mathcal{F}^- \), the moves in \( \mathcal{F}^- \) from the full model are rendered as stationary or coarser moves in \( \mathcal{F}^+ \), losing information. We approximate the information loss in each tree \( T_m \in \mathcal{F}^- \) of height \( k \) by the squared counts of moves from the corresponding reference system, since we are effectively representing these counts by zeros (i.e. the bias). Let \( c_m(D) \) be the number of transitions in \( D \) that occur in reference system \( \Sigma v_m \) (the sum of 1-gram counts in \( T_m \)). Since \( c_m(D)^2 \) is an upper bound for the squared counts of \( i \)-grams for \( i > 1 \), it can be used as an estimate of the information loss at level \( i \) in the tree \( T_m \). Formally, we express this loss term \( C_m(D, k) \) as

\[
C_m(D, k) = k \cdot c_m(D)^2, 
\]

where \( k \) is the height of the tree. We use a small part of the privacy budget \( \epsilon_s \) to compute the noisy \( c_m \).

The variance for tree \( T_m \in \mathcal{F}^+ \) of height \( k \) can be computed as:

\[
N_m(\epsilon_r, k) = E \left[ \sum_{x \in T_m} (c_{\mathcal{F}^+}(D, x) - \tilde{c}_{\mathcal{F}^+}(D, x))^2 \right] 
= \sum_{x \in T_m} \text{Var}(\epsilon_r, k, x) 
\]

where, \( \epsilon_r = \epsilon - \epsilon_s \) is the remaining privacy budget that will be used to learn the transition probabilities (in the next section). \( \text{Var}(\epsilon_r, k, x) = \frac{2}{\epsilon_r(x)^2} \), where \( \epsilon_r(x) \) is the privacy budget used to compute the noisy count for \( x \) and is dependent on \( \epsilon_r \) and \( k \).
Summing up the variance and bias terms specified in Eqns. (4.13) and (4.14), the total error of $F^+$ of height $k$ is

$$\text{Error}(F^+, \epsilon_r, k, D) = \sum_{T_m \in F^+} N_m(\epsilon_r, k) + \sum_{T_m \in F^-} C_m(D, k), \quad (4.15)$$

Our goal is to find the optimal model $F^{+*}$ of height $k$ which minimizes the overall error specified in Eqn. (4.15) with respect to $F^+$ and $k$, i.e.

$$F^{+*}(\epsilon_r, D) = \arg\min_{F^+ \subseteq F_{\text{full}}, k \geq 2} \text{Error}(F^+, \epsilon_r, k, D). \quad (4.16)$$

Eqn. (4.16) satisfies an interesting, and intuitive monotonicity property of the optimal reference systems as a function of privacy budget $\epsilon_r$, and data size $|D|$: (1) given fixed data size $|D|$, a larger $\epsilon_r$ allows either more reference systems to be chosen, or trees to have larger heights; (2) given fixed privacy budget $\epsilon_r$, a larger dataset allows either more reference systems to be chosen, or trees to have larger heights. We omit formal statement and proof of these claims for brevity. This monotonicity property suggests that given fixed dataset $D$ and privacy budget $\epsilon_r$,

$$\arg\min_{F^+ \subseteq F_{\text{full}}} \text{Error}(F^+, \epsilon_r, k + 1, D) \subseteq \arg\min_{F^+ \subseteq F_{\text{full}}} \text{Error}(F^+, \epsilon_r, k, D) \quad (4.17)$$

Hence, we propose Algorithm 2 to search for the optimal $F^{+*}$ and also $k$ by decrementing the value of $k$ from its maximum possible value. In this way, the search space of the optimal model at $k$ is not the subsets of $F_{\text{full}}$ any more, but the subsets of the optimal model at $k + 1$ (as shown in Line 8). In Algorithm 2, Lines 1-2 use a privacy budget of $\epsilon_s$ to compute the noisy bias for each reference system if the corresponding system is dropped. Line 3 chooses the coarsest possible reference system $\Sigma_{v_M^*}$, which has a noisy root count square greater than the variance of the noise $\frac{2}{\epsilon_s^2}$. Lines 4-5 initialize the full model’s maximum possible tree height $k$ and remaining budget $\epsilon_r$. Then the variance in $F^{+*}$ is estimated based on the remaining budget...
\( \epsilon_r \). Lines 6-14 identify the best model with the lowest Error\((F^+, \epsilon_r, k, D)\) given the height \( k \). The coarsest reference system is included in final output (in Line 10 and 14) in order to record any large steps which the optimal model fails to capture. This algorithm may not guarantee the optimal solution for Eqn. (4.16), but reduces the searching space based on Eqn. (4.17) and is shown to be effective in our evaluation (Section 4.6.4).

### 4.5.3 Learning Transitions

**Noisy Forest Counts.** The transition probabilities in the optimal model \((F^{**}, k)\) can be learned by adding Laplace noise to the counts of the prefix trees corresponding to \(F^{**}\). Each transition maps to exactly one prefix tree in \(F^{**}\), and affects the counts of exactly one node at each of the \( k \) levels of the tree. Thus, adding or removing a user’s trajectory table with at most \( L \) transitions affects at most \( L \cdot k \) nodes. However, there is no apriori bound on \( L \), resulting in an unbounded sensitivity for the forest counting query.

Rather than resorting to truncate a user’s trajectories \([26, 27]\) or sampling a bounded number of transitions from a user’s trajectory table \([18, 128]\), we choose to normalize the counts such that the total weight contributed by all transitions in a user’s trajectory table is 1. That is, if \( PT \) has \( L \) transitions, then each transition will contribute a weight of \( 1/L \) to the counts in the prefix tree. Thus, a user’s trajectory table contributes at most weight 1 to level \( i \) of all prefix trees, resulting in a sensitivity of \( k \) for the forest counting query.

We follow Cormode et al.’s \([34]\) strategy of splitting the privacy budget \( \epsilon_r \) geometrically across the different levels of the prefix trees. The roots of the prefix trees are assigned a privacy budget \( \epsilon_0 \). Level \( i \) nodes receive a budget of \( \epsilon_i = \epsilon_0 \cdot \gamma^i \), where \( \gamma = 2^{\frac{1}{3}} \) is a multiplicative factor that increases the budget (resulting in smaller noise) as we move down the prefix tree (to account for lower counts). \( \epsilon_0 \) is set such that
For a node at level $i$ in a prefix tree, a noisy count $\tilde{c}(D, x) = c(D, x) + \eta$ is released, where $\eta$ is drawn from Laplace($1/\epsilon_i$).

**Adaptive Pruning.** Model selection prunes away prefix trees $T$ with low support and chooses a height $k$. However, in each remaining tree $T$, a large fraction of the nodes may have a count of 0. Releasing noisy counts for nodes with 0 counts increases the size of the tree and dilutes the signal to noise ratio. Hence, like in prior work [26], after adding Laplace noise to the nodes in the prefix tree, (including nodes with a count of 0), we prune subtrees that could result in very low signal to noise ratios. Our pruning strategy is different from prior work: Nodes in tree $T$ are traversed breadth-first. With each node $x \in T$, we associate a threshold $\tilde{\theta}(x)$ based on signal to noise ratio. If the noisy count $\tilde{c}(x)$ is greater than the threshold $\tilde{\theta}(x)$, we release the noisy count and continue. Otherwise, we prune away all the descendants of $x$. If the privacy budget ($\epsilon'$) that would have been used for a descendant of $x$ is greater than the privacy budget for node $x$, then we recompute the noisy count for $x$ using $\epsilon'$.

**Choice of Threshold $\tilde{\theta}(x)$:** There are many choices for setting the thresholds $\tilde{\theta}(x)$. A simple data independent method, which we call fixed pruning, is to apply a fixed threshold $\tilde{\theta}$ for all the nodes in the tree. When $\tilde{\theta}$ is too small, we still add noise to many nodes with a true count of 0. If $\tilde{\theta}$ is too large, we may prune away nodes with large true counts. Both scenarios result in poor utility.

Hence, we propose an adaptive pruning strategy to select different thresholds for different nodes such that pruning results in a reduction in total error with high probability. The intuition behind adaptive pruning is the same as that for our model selection strategy: if the count of a node $x$ is much smaller than the number of its descendants, then the bias introduced by pruning the subtree below $x$ is much smaller than the variance due to noisy counts (if the subtree is not pruned). This is
formalized in the following lemma:

**Lemma 61.** Let $\mathcal{T}$ be an input tree of height $k$ and let $x$ be a node at level $\ell$. Let $\text{desc}(x,i)$ denote the number of descendants of $x$ in level $i$. Let $\tilde{\mathcal{T}}$ be the noisy tree resulting from the Laplace mechanism (with geometric budgeting). Let $\tilde{T}_{\text{pruned}}$ be the result of pruning the subtree under $x$ whenever

$$c(D,x) < \theta(x) = \sqrt{\frac{2}{k-\ell+1} \left( \sum_{i=\ell+1}^{k} \frac{\text{desc}(x,i)}{\epsilon_i^2} + \frac{1}{\epsilon(x)^2} - \frac{1}{\epsilon^2} \right)}$$

(4.18)

where $\epsilon_i$ is the privacy budget allocated to nodes at level $i$ by geometric budgeting, and $\epsilon' = \epsilon_r - \sum_{x'}$ is a prefix of $\epsilon(x')$. Then, $\tilde{T}_{\text{pruned}}$ has lower error than $\tilde{T}$.

Note that for regular trees (where all nodes have the same degree), nodes in the same level of the tree are assigned the same threshold $\theta(x)$. However, adaptive pruning works after noise has been added to a node, and hence does not have access to the true count. We use $\tilde{\theta}(x) = \theta(x) - \ln(1/\delta)/\epsilon(x)$, and prune $x$’s descendants when the noisy count $\tilde{c}(D,x) < \tilde{\theta}(D,x)$. This is because with probability $1-\delta$ if the noisy count $\tilde{c}(D,x)$ is smaller than $\tilde{\theta}(x)$ then true count $c(D,x)$ is smaller than the threshold $\theta(x)$. This pruning choice seems to work well on real datasets in our evaluation.

**Stochastic Transition Probabilities.** In order for a prefix tree to represent stochastic transition probabilities, the counts in the tree should be non-negative. Additionally, the counts should satisfy prefix consistency i.e., the count at a node $x$ must equal to the sum of the counts at its children. Consistency is ensured via constrained inference using least squares minimization [57]. We modify the original algorithm and proofs to work for the non-regular trees that result from pruning.

Non-negativity is ensured by (a) setting negative counts to 0 (if a node is not a leaf node, all the nodes in its subtree are set to 0 as well); (b) normalizing the
counts among children of the same parent node to obtain transition probabilities from the parent to the children; (c) computing the counts in a top-down approach from the original root count (obtained by consistency), given the stochastic transition probabilities from (b).

4.5.4 Direction Weighted Sampling

The order $\ell$ Markov process captures global properties of the trajectories like direction, since the next value depends on the previous $\ell$ values. However, the noise added for ensuring privacy can mask this behavior. We introduce a novel direction weighted sampling scheme to recover the directionality in the data.

We encode the neighboring cells in a grid relative to the current cell. For instance, in Figure 4.3, the cell $(12, 7)_1$ in reference system $\Sigma_{v_1}$ has eight neighboring cells, one in each compass direction: $(11, 6)_1$ in the southwest direction, $(12, 6)_1$ in the south direction, etc. In order to avoid sudden unrealistic changes of direction, we modify our sampling procedure to remember the recent trend. Each direction, $\text{dir}$, has an equal weight when we start sampling a new trajectory, and the weight is updated along with the sampling process. Given the prefix of the trajectory generated so far, denoted by $x$, we set the weight of the direction $\text{dir}$ to $w_{\text{dir}}(x) = \alpha^{c(\text{dir}, x)}$, where $\alpha$ is a weighting factor greater than 1. We count the number of times $x$ has followed direction $\text{dir}$ by $c(\text{dir}, x)$. However, in order to allow a long trajectory to turn locally, we use only the window of the last $\omega$ moves of $x$, denoted by $x_{[\omega]}$, to compute the weight of direction $\text{dir}$, i.e. $w_{\text{dir}}(x) = \alpha^{c(\text{dir}, x_{[\omega]})}$. Sampling with window size $\omega = 0$ corresponds to the basic sampling method without considering the direction of the prefix of the trajectory. We show in our experiments that direction-weighted sampling with a reasonable window size $\omega$ can improve the utility of synthetic trajectories.
4.6 Empirical Evaluation for DPT

In this section, we empirically evaluate DPT over two large trajectory datasets. The evaluation results are presented in two parts:

- Synthetic trajectories released using DPT are more useful and scalable compared to prior work in our end-to-end evaluations. We show that the synthetic trajectories mirror the original trajectories on three utility metrics – distribution of diameter (i.e., distance traveled), conditional distributions of destinations given starting regions, and frequent patterns.

- The individual algorithms powering DPT (namely differentially private model selection, adaptive pruning and direction weighted sampling) outperform alternate solutions derived from prior work.

We first describe our datasets and utility metrics in Section 4.6.1, then list the design choices for DPT in Section 4.6.2. Next, we present empirical end-to-end scalability and utility results in Section 4.6.3 followed by utility evaluation on components of DPT in Section 4.6.4.

4.6.1 Datasets and Utility Metrics

Table 4.2 summarizes the two large trajectory datasets used in our experiments. Given $\Sigma_v$, a full set of hierarchical reference systems of size 6 are considered for both datasets, i.e. $\Sigma_{v_1}, \Sigma_{v_2}, \ldots, \Sigma_{v_6}$, where $v_m = v_{m+1}/2$. We will define $\Sigma_{v_1}$ for each dataset below.

**Taxi dataset:** A set of GPS trajectories was recorded by 8602 taxi cabs in Beijing, China, during May 2009 [2]. The trajectories cover the region of Beijing within the bounding box (39.788N, 116.148W) and (40.093N, 116.612W) – approximately 34 km × 40 km. The finest resolution $v_1$ considered is 100 m × 100 m. The raw sampling rate of these trajectories ranges from 30 seconds to 5 minutes. The dataset
consists of approximately 4.3 million trips with passengers. Each trip is linearly interpolated into a sequence of location points at 30-second sampling intervals.

**Network dataset**: A set of trajectories synthesized by Thomas Brinkhoff’s network-based generator [19] of moving objects. The data consists of 300 files of 50000 individuals’ trajectories of length up to 1000 time units, sampled at equal time intervals. This data covers the region of Oldenburg, a city in Germany, within a bounding box of approximately 9 km × 10 km. The finest resolution $v_1$ considered is about 50 m × 50 m.

We developed three utility metrics $Q_d, Q_t, Q_m$ to evaluate whether synthetic trajectories $D_{syn}$ preserve aggregate properties of the original trajectories $D_{raw}$. We sampled 5 sets of 50000 trajectories from the DPT model for $D_{syn}$ and evaluated against 5 sets from $D_{raw}$ each of which consists of 50000 randomly selected raw trajectories. We then reported the relevant mean error or accuracy with its standard deviation. The three metrics are described below.

**Metric I: $Q_d$ - Diameter Distribution.**

The diameter for a trajectory $t = \sigma_1 \cdots \sigma_n$ is the maximum distance between any pair of locations in $t$, i.e. $\max_{i,j} d(\sigma_i, \sigma_j)$ $\forall i, j = 1, \ldots, n$. Let $Q_d(D)$ denote the empirical distribution of the diameter on trajectory database $D$, where the diameter is quantized into 25 equal width buckets $\{[0, x), [x, 2x), \ldots, [24x, 25x)\}$. $x$ is set to 800m for Taxi dataset, and 400m for Network dataset (both correspond to the resolution $\Sigma_{v_4}$). The error in $Q_d$ is measured by:

$$E_{\text{diameter}} = JSD(Q_d(D_{syn}), Q_d(D_{raw})), \tag{4.19}$$

where $JSD(\cdot, \cdot)$ is the Jensen Shannon divergence with range $[0, \ln 2]$.

**Metric II: $Q_t$ - Trip Distribution.**

Given a coarse grid $\Sigma_v$, a starting region $a_s$ is a square consisting of 3 × 3 cells from
Let $D_{a \to a}$ be the set of trajectories in $D$ which starts from the region $a_s$ and ends at the cell $a \in \Sigma_v$. Trip distribution $Q_t$ measures the distribution of the ending point $a$ in $D_{a \to a}$ given $a_s$, i.e. $P(\cdot | a_s, D)$. A set of 1000 starting regions $S$ are randomly chosen. $v$ is set to 4km for Taxi and 1km for Network dataset. The error in $Q_t$ is measured as

$$E_{\text{trip}} = \frac{1}{|S|} \sum_{a_s \in S} JSD(P(\cdot | a_s, D_{\text{syn}}), P(\cdot | a_s, D_{\text{raw}})),$$

where $JSD(\cdot, \cdot)$ is the Jensen Shannon divergence with range $[0, \ln 2]$.

**Metric III: $Q_p^m$ - Frequent Patterns.**

We first project the path taken by each trajectory in $D$ on a grid $\mathcal{T}_v_m$, where the path is formed by connecting consecutive points with a straight line. We obtain the sequence of cells the trajectory passes through, and represent this sequence of cells with their corresponding anchor points in $\mathcal{T}_v_m$, for $m \in \{2, \ldots, 6\}$. The query $Q_p^m$ takes in $D$ and outputs the top $p$ patterns from $\Sigma^k_{v_m}$ with the highest count, for $k = 2, \ldots, 6$. We consider $p$ to be 1000 and 10000 in our evaluations. The utility in $Q_p^m$ of $D_{\text{raw}}$ w.r.t. $D_{\text{syn}}$ is measured as

$$A_{\text{pattern}} = F_1(Q_p^m(D_{\text{raw}}), Q_p^m(D_{\text{syn}})),$$

where $F_1(\cdot, \cdot)$ is the $F_1$ score with range $[0, 1]$ (i.e., harmonic mean of precision and recall) which measures the similarity between two item sets.
4.6.2 Design Choices

We consider alternative implementations of DPT by varying the structure of the RSS model (namely the reference system set used), the pruning strategy and the sampling method as shown in Table 4.3. In the default setting \([\text{opt, adapt, dir}]\), \text{opt} refers to the model output by private model selection, \text{adapt} refers to adaptive pruning, and \text{dir} refers to direction weighted sampling. Two alternate models are presented to \text{opt}: \text{full} and \(\Sigma_v^5\), where \text{full} refers to the full model consisting of all the reference systems from \(\Sigma_v^1\) to \(\Sigma_v^5\). \(\Sigma_v^6\) is not in \text{full} as we do not expect to see transitions in \(\Sigma_v^6\) since transitions in \(\Sigma_v^6\) corresponds to movements at the speed of greater than 240 miles per hour. Alternatives to the pruning strategy are \text{fixed} which uses a fixed threshold of 0 (described in Section 4.5.3), and \text{ngram}, the pruning strategy used in [26]. \text{np} refers to the non-private model without noise. Lastly, direction-weighted sampling is studied by varying the window size \(\omega\). \text{nodir} refers to the case when \(\omega = 0\).

4.6.3 End-to-End Evaluation

In the end-to-end evaluation, we compare DPT with the most relevant prior work [26] denoted by \text{seq}. This prior work computes a noisy prefix tree on the entire domain by adding Laplace noise followed by pruning (sanitization phase), and then uses the noisy prefix tree to reconstruct a database of sequences (reconstruction phase).

For \text{seq}, we transform the first \(l_{\text{max}}\) sampled points of each trajectory with a single reference system \(\Sigma_v^m\), for \(m = 1, \ldots, 6\), where \(l_{\text{max}}\) is the truncation length used in \text{seq} to bound sensitivity as the input. Then this input is processed by the original

<table>
<thead>
<tr>
<th>settings</th>
<th>default</th>
<th>alt</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>rss</td>
<td>opt</td>
<td>full; (T_{\text{vs}})</td>
<td>order (k): 2,3,4,5,8</td>
</tr>
<tr>
<td>pruning</td>
<td>adapt</td>
<td>fixed; (n_{\text{pp}})</td>
<td>privacy budget (\epsilon): 2.0,1.0,0.5,0.3,0.1; model selection budget: (\epsilon_s = 0.1); pruning parameter (\delta): 0.1</td>
</tr>
<tr>
<td>sampling</td>
<td>dir</td>
<td>nodir</td>
<td>factor (\alpha): 1.2; window size (\omega): 0.4,\ldots,32</td>
</tr>
</tbody>
</table>

Table 4.3: Design choices in DPT
code of seq provided by [26], resulting in a differentially private synthetic trajectory database. For end-to-end evaluations, we use the default setting of DPT– [opt, adapt, dir].

**Scalability.** We run seq on a single machine with 2.40GHz 30MB Cache (256GB RAM - 48 hyperthreaded cores) for 5 times, and report the average time for sanitization and reconstruction. Similarly, we run DPT with its default setting for 5 times and report the average time for sanitization and sampling of 50000 trajectories. In Figure 4.1 on page 82, we show the runtime of seq with increasing resolution from $v_6$ to $v_2$ (a $4 \times$ increase in domain size each time) for $l_{\text{max}} = 20$ and $k$-grams for $k \leq 4$ when $\epsilon = 1.0$. The time for sanitization increases exponentially with the domain size, i.e. $O(|\Sigma|^k)$, since seq considers the transitions between all pairs of points in the domain. In comparison, DPT uses the optimal model $\{\Sigma_{v_4}, \Sigma_{v_5}\}$ with a fanout of constant size, which takes orders of magnitude less time to complete the sanitization process. In addition, seq aims to reconstruct a database of all possible trajectories by joining shorter trajectories to form longer ones. When a database has a large size, this approach takes a very long time. When resolution is $\Sigma_{v_2}$, the reconstruction takes more than 5 hours to complete. We observe that the reconstruction time decreases from $v_2$ to $v_3$ and then increases again from $v_3$ to $v_6$. This is due to the fact that many consecutive points are mapped to the same cell indices in the coarse reference system. This allows the reconstruction process to form longer trajectories from shorter trajectories of repeated points. For instance, the maximum length of trajectories reconstructed in $\Sigma_{v_6}$ reaches 20, while the maximum length in $\Sigma_{v_2}$ is only 5. Therefore, a sampling approach is preferred as it can quickly produce a set of good representative trajectories with various lengths and can be parallelized easily.

Results for seq at resolution $v_1$ are not reported as the program does not terminate even after a day.
Utility. In Figure 4.5, we show the JSD error for diameter distribution $Q_d$, and trip distribution $Q_t$, and $F_1$ score for frequent patterns $Q_p^m$ respectively when $\epsilon = 1.0$. For both error metrics, as the resolution gets finer (from $v_6$ to $v_2$), the error gets worse. DPT improves upon seq by $> 80\%$ on $Q_d$ error and $> 60\%$ on $Q_t$ error. For $Q_p^m$, given the resolution at which seq is processed, we report the accuracy score of the top 1000 frequent patterns at the same resolution: both trajectories generated by seq and DPT are projected on $\Sigma_v$ for the evaluation of frequent patterns. The comparison shows that DPT obtains better $F_1$ score at resolutions $v_6$ to $v_3$, with 30\% to 200\% improvements, even though DPT is not necessarily generated at $\Sigma_v$. At $v_2$, DPT performs poorly because its optimal model selected is $\{v_4, v_5\}$, and hence it is unlikely to capture information at fine resolution. Experiments at smaller $\epsilon$ also show that DPT outperforms seq. We omit the figures due to space constraints.

![Figure 4.5: Utility of DPT v.s seq for Taxi, at $\epsilon = 1.0$](image)

4.6.4 Component Utility Evaluation

In this section, we report the evaluation on three components: 1) private model selection, 2) adaptive pruning and 3) direction-weighted sampling on both Taxi (T) and Network (N) datasets. For each component, we will consider alternate choices defined in Section 4.6.2. The setting [full, np, dir] is chosen as the benchmark for the
Component I: Model Selection. Table 4.4 shows the optimal model selected by private model selection (Algorithm 2) on Taxi dataset. As $k$ increases and $\epsilon$ decreases, the signal to noise ratio gets weaker, and only a smaller subset of reference systems can be supported. This validates the monotonicity property from Section 4.5.2. We observe similar trends for Network dataset.

In addition, we have found that the optimal model performs considerably better than its alternatives. In Figure 4.6, we present two obvious alternate models: the private full model $[\text{full}, \text{adapt}, \text{dir}]$ and the private coarsest reference system $[v_5, \text{adapt}, \text{dir}]$. The optimal model $\text{opt}$ approached the high accuracy of the non-private full model $[\text{full}, \text{np}, \text{dir}]$ as $\epsilon$ increases. At $\epsilon = 0.1$ $\text{opt}$ achieves accuracy/error within 85%
of [full, np, dir] for Taxi dataset.

Though the non-private full model preserves good utility, the optimal model consistently beats the private version of the full model [full, adapt, dir] as the private full model incurs too much noise. Moreover, opt outperforms $\Sigma_{v_5}$ at most settings. When $\epsilon$ is large opt significantly outperforms $\Sigma_{v_5}$. For instance, for the Network dataset, the optimal model for $\epsilon = 1$ is $\{\Sigma_{v_2}, \Sigma_{v_3}, \Sigma_{v_5}\}$, and thus captures a lot more finer details than $\Sigma_{v_5}$, resulting in low error. On the other hand, at small $\epsilon$, $\Sigma_{v_5}$ may be the optimal model (but opt picks a different model due to noise). In those cases, $\Sigma_{v_5}$ can outperform opt. Furthermore, the good performance of $\Sigma_{v_5}$ for $Q_p^5$ and $Q_t$ can be explained by the fact that $Q_p^5$ and $Q_t$ are measured at the same resolution as
Component II: Adaptive pruning. We substitute the default setting of the second component (sanitizing method) — geometric noise with adaptive pruning methods (adapt), by its alternatives — geometric noise with fixed threshold pruning at threshold 0 (fixed), and n-gram methods (ngram) [26]. In Figure 4.7, we present the evaluation results on synthetic trajectories generated from these three settings [opt, adapt, dir], [opt, fixed, dir], [opt, ngram, dir]. We observe that adapt performs just as well as ngram for $Q_{p}$, $Q_{p}$ on Taxi dataset, but gives better accuracy than ngram for other metrics. We conjecture two reasons for the poor performance of ngram: First, ngram can effectively estimate the right subtree height at each interval node and prune away subtrees with low support when the given tree height is large. However, our private model selection has considered the optimal tree height in opt and hence, the height estimation in ngram is no longer useful for opt. Second, the privacy budget is split equally amongst nodes on the root-to-leaf path which survive after ngram pruning. Uniform budget splits have been shown to be less accurate than geometric noise in [34]. Furthermore, fixed seems to perform as well as adapt. We further studied the noisy prefix trees output by the three pruning strategies. We find that fixed adds the most number of false nodes (nodes with true count 0) to the noisy trees leading to zigzag trajectories when sampled without dir. Direction-weighted sampling, however, lifts the accuracy of fixed.

Component III: Direction-weighted Sampling. The intuition behind direction-weighted sampling is to minimize unnecessary local turns in the synthetic trajectories. In this evaluation, we vary the direction setting (dir) in the default setting of DPT [opt, adapt, dir], by changing the window size $\omega$ from 0 to 32. $\omega = 0$ refers to no-direction weighted sampling (nodir). Default window sizes are set to 25% of the average length of raw trajectory dataset (6 and 16 for Taxi and Network respectively).
As direction-weighted sampling has a larger impact on longer trajectories, we filter Taxi synthetic trajectories consisting of more than 15 sampled points for evaluation, where 15 is the median number of location points in the ground truth. For Network dataset has an average length greater than the maximum window size evaluated, we use all synthetic trajectories for evaluation. As shown in Figure 4.8, direction-weighted sampling re-enforces the directionality by smoothing the local turns, and hence results in lower errors and higher $F_1$ scores. For non-private model (np), the improvement by this direction constraint is not as significant as in the private models. For $\epsilon = 0.5, 0.1$, the utility of private models improves as window size increases and stabilizes at a window size slightly greater than 4 (the order of the Markov process). The JSD errors for $Q_d$ and $Q_t$ are reduced up to $20\%$ and the $F_1$ scores are improved up to $30\%$.

4.7 VisDPT: Visual Exploration of DP Trajectories

Based on DPT, we also designed a system, called VisDPT [61], that provides an interactive visualization of (i) a rich set of utility queries that are commonly used in mobility data analysis and (ii) a rich class of privacy metrics to allow comprehensive examination of privacy risks of both raw and synthetic trajectories without writing
repeated low-level queries. This system can help data curators understand the impact on utility brought by the privacy-preserving mechanisms. Figure 4.9 shows the outline of VisDPT system which consists of four modules – 1) model generation, 2) sampling, 3) query processing, and 4) visualization. Given a database of taxi trips $D_{\text{raw}}$, VisDPT builds a probabilistic model $H$ and sanitizes it to generate a private model $\tilde{H}$ with a given privacy budget $\epsilon$. The sampling phase uses $H$ and $\tilde{H}$ to generate synthetic trajectories $D_{\text{syn}}$ and private trajectories $D_{\text{syn}}^\epsilon$ respectively. DPT is used for modeling and sampling trajectories. However, this visualization framework can be easily extended to support multiple modeling and sampling techniques to compare their accuracy and privacy guarantee. The query processing module can execute various privacy and utility queries (e.g., finding unique trajectories, calculating trip distribution, etc.) on original as well as sampled databases. The results of these queries are presented by the visualization module which displays quantitative results (such as statistics and graphs) together with three different map views (one for each database). The map views help users visually inspect and compare the results by overlaying output trips or distribution. Next, we explain the privacy and utility queries supported in VisDPT.

4.7.1 Privacy Metrics

The notion of “uniqueness” has been developed as one of the privacy risk assessments on raw trajectory database [36]. The test based on this notion approximates the number of points required to uniquely identify the trajectory of an individual. However, synthetic data built from the raw database has no real individuals and hence the re-identifying test on the ground truth may not be directly applicable for the synthetic data. Therefore, we first develop a class of privacy metrics for raw trajectory database, and extend these metrics to synthetic database.

Metrics for Raw Database. The vulnerability of $D_{\text{raw}}$ to re-identification attacks
and the uniqueness of a trajectory can be well reflected by this class of privacy metrics. Each privacy metric is associated with a similarity function, denoted by \( I_t(t') \) for \( t, t' \in D_{\text{raw}} \) which outputs 0 if \( t' \) is similar to \( t \), otherwise it outputs 1. The uniqueness of a trajectory \( t \in D_{\text{raw}} \) depends on the set of trajectories similar to \( t \) in \( D_{\text{raw}} \), denoted by \( S_t = \{ t' \in D_{\text{raw}} | I_t(t') = 0 \} \). We say \( t \) is unique if \( |S_t| = 1 \), i.e., \( S_t \) only contains \( t \) itself. We denote the set of unique trajectories in \( D_{\text{raw}} \) by \( T_I(D_{\text{raw}}) \). Then, the vulnerability of a raw database \( D_{\text{raw}} \) to re-identification attacks is defined as the fraction of unique trajectories in the raw database. Instances of this class of privacy metrics using two types of similarity functions are presented below.

Point based similarity. The point based similarity \( I_{p,\theta}^p(t, t') \) computes the similarity between trajectories based on the similarity of locations at fixed \( p \) positions. Let \( \{\sigma_1, \ldots, \sigma_p\} \subset t \) and \( \{\sigma'_1, \ldots, \sigma'_p\} \subset t' \), are locations at \( p \) positions in respective trajectories. Then, \( I_{p,\theta}^p(t, t') = 0 \), if all these \( p \) pairs of points are within distance \( \theta \), i.e., \( ||\sigma_i - \sigma'_i||_2 < \theta \quad \forall i = 1, \ldots, p \), otherwise it is 1.

Distance based similarity. The distance based similarity \( I_{d,\theta}^d(t, t') \) computes the similarity between trajectories using a given distance function \( d(\cdot, \cdot) \). Examples of such distance functions are edit distance [25] and Fréchet distance [9]. We say \( I_{d,\theta}^d(t, t') = 0 \), if \( d(t, t') < \theta \), otherwise it is 1.

**Metrics for Synthetic Database.** From the privacy perspective, the synthetic dataset should not have any trajectories that can be exploited for re-identification attacks. One class of such trajectories are the ones that are similar to the unique trajectories in the raw database, i.e., \( T_I(D_{\text{raw}}) \). The set of such trajectories in the synthetic dataset, \( T_I(D_{\text{syn}}) \), is computed as follows. For each \( t \in T_I(D_{\text{raw}}) \), we compute the set of trajectories in \( D_{\text{syn}} \) that are similar to \( t \). We denote this set as \( S_t(D_{\text{syn}}) = \{ t' \in D_{\text{syn}} | I_t(t') = 0 \} \). Then, \( T_I(D_{\text{syn}}) = \cup_{t \in T_I(D_{\text{raw}})} S_t(D_{\text{syn}}) \). It is important to know how similar these trajectories are to the unique trajectories.
Hence, for all trajectories $T_i(D_{syn})$ we compute the distribution of similarity (with the corresponding unique trajectories in the raw database) using a distance function $d(\cdot, \cdot)$ mentioned above.

4.7.2 Utility Metrics

VisDPT presents a rich class of utility metrics for popular taxi data analysis, such as patterns [127], paths taken by taxi [133, 134] and trip distribution [45, 109].

**Diameter.** The diameter for a trajectory $t = (\sigma_1, \ldots, \sigma_n)$ is the maximum Euclidean distance between any pair of locations in $t$, i.e., $\max_{i,j} d(\sigma_i, \sigma_j) \forall i, j = 1, \ldots, n$. $Q_d(D)$ denotes the empirical distribution of the diameter on the trajectory database $D$.

For rest of the queries, we map all trajectories onto a uniform grid $T_v$ with resolution $v$ and compute the distribution at a cell level, to handle the sparsity issue of databases. For every trajectory $t = (\sigma_1, \ldots, \sigma_n)$ in $D$, we find the mapped sequence of cells $(c_1, \ldots, c_n) \in T_v^n$, where $\sigma_i \in c_i$, i.e., location $\sigma_i$ is in the cell $c_i$.

**Origin/Destination.** Given a uniform grid $T_v$, the origin distribution $Q_s(D, v)$ denotes the empirical distribution of the starting points of $D$ over $T_v$. The destination distribution $Q_e(D, v)$ is defined similarly for the ending points of trajectories in $D$.

**Trip/Path.** Trip distribution measures the distribution of ending points of trajectories over $T_v$ that are originating from a given cell $c_s \in T_v$, denoted by $Q_t(D, c_s)$. The path query $Q_p(D, c_s, c_e)$ denotes the set of trajectories starting at $c_s$ and ending in $c_e$.

**Frequent Patterns.** Frequent patterns is a popular metric to analyze the location sequences that are visited most often. The top $l$ $k$-patterns, denoted by $Q_k^l(D)$ are the $l$ most frequent $k$-grams (sequences of $k$ cells) appearing across all mapped trajectories in $D$.

If query results are in the form of distribution, their empirical accuracy is evaluated as $JSD(Q(D_{raw}), Q(D_{syn}))$, where $JSD(\cdot, \cdot)$ is the Jensen Shannon divergence.
with range \([0, \ln 2]\). If query results are in the form of set such as \(Q_f\), their accuracy is measured as \(F_1(Q(D_{\text{raw}}), Q(D_{\text{syn}}))\), where \(F_1(\cdot, \cdot)\) is the \(F_1\) score with range \([0, 1]\) (i.e., harmonic mean of precision and recall) – a similarity measure between item sets.

4.7.3 Dashboard and Demo

In the demo of VisDPT, we expect the attendees to gain insights on – 1) how and to what extent the sensitive information (such as unique trajectories) is protected in a differentially private output; and 2) the impact of noise on the utility of the perturbed dataset. The dashboard for the demo comprises of a control panel (left) and a viewing area (right) as shown in Figure 4.10. The control panel allows users to select samples from databases \(D_{\text{raw}}, D_{\text{syn}}\) and \(D'_{\text{syn}}\) and to perform various analysis on these databases. The results are displayed in three different map windows (one for each database) in the viewing area. Each map window displays a visualization of the query result such as heat maps or trajectory paths and allows operations such as point selection. The overall quantitative results (e.g., statistics, graphs, etc.) are displayed below for various \(\epsilon\) values. The attendee can select \(\epsilon\) value, to change the
For the purpose of this demo, we use a dataset of approximately 4.3 million taxi trips recorded by 8602 taxi cabs in Beijing, China, during May 2009 [2]. The trajectories cover the region of Beijing within the bounding box (39.788N, 116.148W) and (40.093N, 116.612W) – approximately 34 km × 40 km. The raw sampling rate of these trajectories ranges from 30 seconds to 5 minutes. In the interest of time, we precompute all synthetic databases and queries for the demo. The demo comprises of two tasks designed as games that can be played by the attendees. First, we explain them the aim of the game followed by a quick demonstration of how to play (and win) the game. Then, the attendees are given a chance to explore the system via these games as described below.

**Privacy Game.** This game aims to differentiate private results from non-private results based on the existence of unique trajectories. The demonstrators first show how to explore unique trajectories in the ground truth $D_{raw}$ by selecting one of the privacy metrics and adjusting the corresponding threshold $\theta$ for its similarity function. Corresponding unique trajectories are be displayed on the first window. Next, demonstrator selects one of the displayed unique trajectories by clicking on it. The other two windows show synthetic trajectories in $D_{syn}$ and $D_{\epsilon, syn}$ (fixed $\epsilon$) which are close to the selected trajectory (i.e., having distance $< \theta$) and their corresponding similarity (distance) values. The attendee can observe that the differentially private model is unlikely to output trajectories similar to any unique trajectories in the ground truth, while non-perturbed output preserves unique trajectories.

After this learning, three windows (Figure 4.11) shows to the attendee, where the first window displays trajectories in $D_{raw}$, but the positions of the other two windows for $D_{syn}$ and $D_{\epsilon, syn}$ are at random and are unknown to the attendee. The goal of attendee is to guess which window (results) are from $D_{\epsilon, syn}$ by querying on
the three databases. If he chooses the correct window then he wins, otherwise, the window positions are shuffled and he retries. We measure the number of attempts in terms of the number of queries made by an attendee on $D_{\text{raw}}$ which includes adjustment of $\theta$ and selection of a trajectory. One can think of this information as the background information available to the attendee to achieve the goal.

**Utility Game.** The aim of this game is to provide deeper understanding of the trade-off between the aggregate utility of synthetic databases and privacy budget $\epsilon$ in differential privacy. The demonstrators first select the utility query in the control panel and the results are visualized on the corresponding three windows accompanied by an error plot of queries over $D_{\text{syn}}^\epsilon$ w.r.t to $D_{\text{raw}}$ for different $\epsilon$. Sliding bar for $\epsilon$ can be adjusted to explain the relationship between query outputs and $\epsilon$ – smaller $\epsilon$ yields poor utility. Three windows are shown in this game (Figure 4.12), with the first window showing results on $D_{\text{raw}}$. The other two windows show results on $D_{\text{syn}}^{\epsilon_1}$ and $D_{\text{syn}}^{\epsilon_2}$, with a randomly chosen $\epsilon_1$ and $\epsilon_2$ in the range $[0, 10]$. The attendee has to identify the window with the smaller $\epsilon$ by visually comparing the perturbed outputs with the output of the first window (ground truth). The attendee wins if the window is correctly identified, otherwise, the window positions are reshuffled and the attendee retries.

The attendees of this demo has given very positive feedback on the effectiveness of differential private trajectories on their privacy and utility guarantees and the
demonstration of VisDPT received the best demo award in 2016 VLDB Conference. More tools like VisDPT are in need for data curators to make informed decisions and to support the release of data with provable privacy guarantees.

4.8 Summary

In this chapter, we described variants of differential privacy for location data and interpreted them under Blowfish and Pufferfish framework. In practice, there are a number of algorithms known for geo-indistinguishability and event-DP, but these approaches constitute the weakest privacy guarantees. More work is needed to identify practical solutions for location-based applications under stronger privacy notions.

With DPT, we introduce the first system devoted to doing so for natural, speed-varying trajectories under user-level differential privacy, a strong variant of differential privacy. Our analysis and empirical study demonstrate that this is an effective way to reveal information, while protecting privacy both in theory and in practice. Based on DPT, we also designed a system, called VisDPT [61], that provides an interactive visualization of (i) a rich set of utility queries that are commonly used in mobility data analysis and (ii) a rich class of privacy metrics to allow comprehensive examination of privacy risks of both raw and synthetic trajectories without writing repeated low-level queries. This system can help data curators understand the impact on utility brought by the privacy-preserving mechanisms.
There is much opportunity for extending this line of work. While effective in practice, we have yet to show strong analytic guarantees for the utility of data released via DPT. We have focused on reference systems based on multi-resolution grids, but other systems could lead to better results for certain trajectories where paths are more constrained. Moreover, we focused on trajectories within the (two-dimensional) plane. As airborne sensors and unmanned aerial vehicles become more prevalent, it may be necessary to extend to three-dimensional trajectories, where the number of possible cells increases while the data density further decreases. Last but not least, user-level differential privacy only considers correlations within a single trajectory. There are also inherent correlations across trajectories of individuals, but these correlations are acknowledged and handled by little work. For instance, it is known that when the location trajectories of two individuals are similar, then they are highly likely to have strong social connections [29]. Whether techniques like the Markov Quilt mechanism [121] (Section 2.4.3) for handling correlations will be applied to such cases is an interesting open question.
5.1 Introduction

Organizations are increasingly collecting vast amounts of data from individuals to advance science, public health, and resource management and governance. In a number of scenarios, different organizations would like to collaboratively analyze their data in order to mine patterns that they cannot learn from their individual datasets. For instance, hospitals or health workers in neighboring cities might want to identify HIV positive patients who have sought care in multiple cities to quantify the mobility patterns of patients, and hence the spread of the virus. This requires finding patients who occur in multiple databases even though the patient records might not have the same primary key across databases. This problem is called record linkage, and has been well studied for the last several decades [30, 38, 49]. In a collaborative analysis across organizations, privacy is always a concern. In particular, one of the collaborating parties, say Hospital A, should not be able to tell whether or not a record is in the database of the other party, say Hospital B, if that record does not appear in the match output. Privacy constraints arise due to
concerns from individuals who provide their data, such as hospital patients, or due to contractual or legal obligations that organizations have to the individuals in their data. This has led to a field of research called private record linkage (PRL).

Traditional PRL techniques aim to solve the linkage problem with a strong privacy goal – no information should be leaked beyond (a) the sizes of the datasets, and (b) the set of matching records. However, this strong privacy goal (which we call S2PC) \cite{51} comes with a high cost. Existing techniques that achieve this goal either require cryptographically secure comparisons of all pairs of records (and hence are inefficient), or are restricted to equi-joins (and thus have very low recall). Hence, we formalize our problem as follows: given private databases $D_A$ and $D_B$ held by two semi-honest parties, and a matching rule $m$, design a protocol $\Pi$ that outputs pairs of matching records to both parties and satisfies three desiderata: (1) correctness in terms of perfect precision and high recall of matches, (2) provable end-to-end privacy guarantee, and (3) efficiency in terms of sub-quadratic communication and computational cost in $n$, where $n = \max(|D_A|, |D_B|)$. There are two sources of the cost incurred by PRL: (1) the number of cryptographic operations, and (2) the time taken for each cryptographic operation. Our protocols aim to reduce the number of cryptographic operations (i.e., the number of secure pairwise comparisons), the first source of cost, while using existing techniques to securely compare pairs of records.

Techniques that securely compare all pairs of records (APC) have a quadratic cost and hence fail to meet the efficiency requirement of our problem. On the other hand, techniques for efficient private set intersection (PSI) \cite{46, 101} satisfy all three desiderata for equality-like matching rules, but result in poor recall for general fuzzy matching rules. When records in $D_A$ and $D_B$ come from the same discrete domain, one could expand $D_A$ by adding all records that could potentially match with a record in $D_B$, and then find matches by running PSI on the expanded $D_A$ and $D_B$. However, this technique can be very inefficient: the expanded databases could be
much larger than the input databases for complex matching functions or when data are high dimensional. A long line of work [110, 68, 64, 65, 77, 20] has considered scaling APC by using blocking, which is a standard technique for scaling non-private record linkage with a small loss in recall of matching pairs. However, blocking can reveal sensitive properties of input records. We show that such hybrid protocols do not ensure an end-to-end privacy guarantee even in solutions where the blocking step satisfies a strong privacy notion, called differential privacy (DP) [39]. This negative result is in contrast to other success stories [120, 92, 99, 6, 100, 94, 53] on composing DP and secure computation. These settings either consider a client-server model where all data sits on the server or consider aggregated functions across partitioned data where the privacy goals of DP and secure computation do not conflict. In the case of scaling PRL, neither blocking nor DP blocking naturally composes with the strong privacy guarantee of S2PC. To our knowledge, this work presents the first solution to the above open problem, and makes the following contributions:

- We propose and formalize three desiderata for the PRL problem: (1) correctness, or perfect precision and high recall of matches, (2) provable end-to-end privacy, or insensitivity to the presence or absence of an individual record that is not a matching record, and (3) efficiency, or communication and computational costs that scale subquadratically in the input size. We show that all of the existing solutions for PRL violate at least one of these three desiderata. (Section 5.2)

- This motivates us to develop a novel privacy definition, which we call Output Constrained DP. Protocols satisfying this notion are allowed to truthfully return the output of a specific function, but must be insensitive to the presence or absence of individual records that do not affect the function output. (Section 5.3.1)

- We adapt the notion of Output Constrained DP to the context of PRL. Under this privacy notion, computationally bounded adversaries cannot distinguish
two different protocol executions when a single non-matching record is replaced by another non-matching record in one of the databases. This privacy notion, named \textit{DPRL}, allows protocols to truthfully release the set of matching records. (Section 5.3.2)

- We show that prior attempts \cite{65, 77, 20} to scale PRL using blocking do not satisfy our privacy definition DPRL (Theorem 82), and hence fail to achieve stronger privacy guarantees including differential privacy or S2PC. (Section 5.4)

- We develop novel protocols for private record linkage that leverage blocking strategies. Our protocols ensure end-to-end privacy (Theorems 80 and 85), provide at least as much recall as the non-private blocking strategy (Theorems 79 and 86), and achieve subquadratic scaling (Theorems 83 and 86).

- Using experiments on real and synthetic data, we investigate the 3-way trade-off between recall, privacy, and efficiency. Our key findings are: our protocols (1) are at least 2 orders of magnitude more efficient than S2PC baselines, (2) achieve a high recall and end-to-end privacy, and (3) achieve near linear scaling in the size of the input databases and the output set of matching pairs on real and synthetic datasets. (Section 5.5)

5.2 Problem Setting & Statement

In this section, we formulate our problem: finding pairs of records that are similar as per an input matching rule while ensuring three desiderata: \textit{correctness}, \textit{privacy}, and \textit{efficiency}. We then discuss prior attempts to solve this problem and how they do not satisfy one or more of the three aforementioned desiderata, thus motivating the need for a novel solution.
5.2.1 The Private Record Linkage Problem

Consider two parties Alice and Bob who have databases $D_A$ and $D_B$. Let records in $D_A$ come from some domain $T_A$ and let the records in $D_B$ come from domain $T_B$. Let $m : T_A \times T_B \rightarrow \{0, 1\}$ denote a matching rule, and let $D_A \bowtie_m D_B$ denote the set of matching pairs $\{(a, b)|a \in D_A, b \in D_B, m(a, b) = 1\}$. A matching rule can be distance-metric based: two records match if their distance is less than a threshold. For example, Euclidean distance is typically used for numeric attributes, whereas for string attributes, the distance metric is typically based on q-grams [31, 32, 111], phonetic encoding [67], or edit distance over strings [10, 106, 98]. A matching rule can also be conjunctions of predicates over different types of attributes. For instance, two records match if their names differ by at most 2 characters and their phone numbers differ by at most 1 digit. Alice and Bob would like to jointly compute $D_A \bowtie_m D_B$.

Our goal is to design a protocol $\Pi$ that Alice and Bob can follow to compute $D_A \bowtie_m D_B$, while satisfying the following three desiderata – correctness, privacy and efficiency.

- **Correctness**: Let $O_\Pi \subseteq D_A \times D_B$ denote the set of pairs output by the protocol $\Pi$ as the set of matching pairs. The protocol is correct if (a) the protocol returns to both Alice and Bob the same output $O_\Pi$, and (b) $O_\Pi = D_A \bowtie_m D_B$, and incorrect otherwise. Note that if Alice and Bob indeed receive the same output, $O_\Pi$ can only be incorrect in one way – some matching pairs $(a, b) \in D_A \bowtie_m D_B$ are not present in $O_\Pi$. This ensures perfect precision – no false positives. Hence, we quantify the correctness of a protocol $\Pi$ using a measure called recall, which

---

1 The standard record linkage problem involves learning a matching function in addition to computing the matches. Although the problem considered in this paper and in the private record linkage literature ignores this crucial aspect of record linkage, we have chosen to also use this term for continuity with existing literature on the topic.
is computed as:

$$r_{\Pi}(D_A, D_B) = \frac{|O_\Pi \cap (D_A \bowtie_m D_B)|}{|D_A \bowtie_m D_B|}.$$  \hspace{1cm} (5.1)

We require $\Pi$ to have a high recall (close to 1). This precludes trivial protocols that output an empty set.

- **Privacy:** We assume that the data in $D_A$ and $D_B$ are sensitive. As part of the protocol $\Pi$, Alice would like no one else (including Bob) to learn whether a specific non-matching record $a$ is in or out of $D_A$; and analogously for Bob. This precludes the trivial solution wherein Bob sends $D_B$ to $D_A$ in the clear so that Alice can compute $D_A \bowtie_m D_B$ using standard techniques in the record linkage literature [30]. It also precludes the trivial solution wherein Alice and Bob send their records to a trusted third party in the clear who can then compute $D_A \bowtie_m D_B$. Formally stating a privacy definition is challenging (as we will see later in the paper) and is a key contribution of this paper. We will assume throughout the paper that Alice and Bob are semi-honest, i.e., they follow the protocol honestly, but are curious about each others’ databases. We also assume that Alice and Bob are computationally bounded, i.e., they are probabilistic polynomially bounded turing machines.

- **Efficiency:** Jointly computing matching records would involve communication and computational cost. We assume that each record in the database has $O(1)$ length; i.e., it does not grow with $n = \max(|D_A|, |D_B|)$. The communication and computational costs are bounded below by the output size, i.e. $\Omega(M)$, where $M = |D_A \bowtie_m D_A|$. If $M$ is quadratic in $n$, then the costs have to be quadratic in $n$ to ensure high recall. Hence, we consider problems with sub-quadratic output size, and we say that the protocol is efficient if both the communication and computational costs are sub-quadratic in $n$, i.e., $o(n^2)$. 

127
We formalize our problem statement as follows.

**Problem 1 (PRL).** Let $D_A$ and $D_B$ be private databases held by two semi-honest parties, and let $m$ be a matching rule. Design a protocol $\Pi$ that outputs pairs of matching records to both parties such that (1) $\Pi$ ensures high recall close to 1, (2) $\Pi$ provably guarantees privacy, and (3) $\Pi$ has sub-quadratic communication and computational cost.

### 5.2.2 Prior Work

Before describing our solution, we outline five approaches for the PRL problem from prior work – APC, PSI, PSI+X, PRL+$B$ and PRL+$B_{DP}$. Table 5.1 summarizes their (in)ability to satisfy our three desiderata stated in Problem 1. Other related work on composing S2PC and DP is discussed in Section 5.6.

**All-Pairwise Comparisons (APC).** One approach to solve the PRL problem, which we call APC, works as follows: (1) design a secure 2-party algorithm that takes as input a record $a \in D_A$ and a record $b \in D_B$ and outputs to both parties the pair $(a, b)$ if the value of $m(a, b) = 1$ without leaking any additional information, and (2) run the secure comparison algorithm for *every* pair of records in $D_A \times D_B$. The secure comparison primitive can be implemented either using garbled circuits [130] or (partially) homomorphic encryption [97], depending on the matching rule. APC achieves a recall of 1, but requires a quadratic communication and computational cost for $|D_A| \times |D_B|$ secure pairwise comparisons.
APC provides a strong end-to-end privacy guarantee – it leaks no information other than the sizes of the databases and the set of matching records. This guarantee is formalized as follows.

**Definition 62 (IND-S2PC [51]).** A 2-party protocol $\Pi$ that computes function $f$ satisfies IND-S2PC if for any $D_A$, and for every pair of $D_B$ and $D'_B$ where $f(D_A, D_B) = f(D_A, D'_B)$, the view of Alice during the execution of $\Pi$ over $(D_A, D_B)$ is computationally indistinguishable from the view over $(D_A, D'_B)$, i.e. for any probabilistic polynomial adversary $T$,

$$Pr[T(\text{VIEW}^H_{A}(D_A, D_B)) = 1]$$

$$\leq Pr[T(\text{VIEW}^H_{A}(D_A, D'_B)) = 1] + \text{negl}(\kappa);$$

(5.2)

and the same holds for the view of Bob over $(D_A, D_B)$ and $(D'_A, D_B)$ for $f(D_A, D_B) = f(D'_A, D_B)$. $\text{negl}(\kappa)$ refers to any function that is $o(\kappa^{-c})$, for all constants $c$, and $\text{VIEW}^H_{A}(D_A, \cdot)$ (\text{VIEW}^H_{B}(\cdot, D_B) resp.) denotes the view of Alice (Bob resp.) during an execution of $\Pi$.

The IND-S2PC definition uses $\kappa$ as a “security” parameter to control various quantities. The size of the adversary is polynomial in $\kappa$, and the output of the protocol is at most polynomial in $\kappa$. The views of the protocol execution are also parameterized by $\kappa$.

In PRL, let $f_{\text{m}}$ be the function that takes as inputs $D_A$ and $D_B$, and outputs a triple $(|D_A|, |D_B|, D_A \bowtie_m D_B)$. The view of Alice, $\text{VIEW}^\Pi_{A}(D_A, \cdot)$, includes $(D_A, r, m_1, \ldots, m_t)$, where $r$ represents the outcome of Alice’s internal coin tosses, and $m_i$ represents the $i$-th message it has received. The output received by Alice after an execution of $\Pi$ on $(D_A, D_B)$, denoted $\omega^\Pi_A(D_A, D_B)$ is implicit in the party’s own view of the execution. The view of Bob can be similarly defined. In addition, the output size of VIEW will be (at most) polynomial in $\kappa$. Intuitively, IND-S2PC ensures
that the adversary Alice cannot distinguish any two databases $D_B$ and $D'_B$ from her view given the constraint $f(D_A, D_B) = f(D_A, D'_B)$, and the same applies to Bob. This IND-S2PC definition is a necessary condition for the standard simulation-based definition (Theorem 88 in Appendix A.1.1).

To summarize, APC guarantees end-to-end privacy and provides a recall of 1, but violates the efficiency requirement.

**Private Set Intersection (PSI).** We call the next class of approaches PSI, since they were originally designed for efficient private set intersection. Like APC, PSI also ensures IND-S2PC and the parties only learn the sizes of the databases and the set of matching records. The algorithms are efficient, but only ensure high recall for equality predicate like matching rules [46, 101].

The basic protocol works as follows: Alice defines a polynomial $p(x)$ whose roots are her set of elements $a \in D_A$. She sends the homomorphic encryptions of the coefficients to Bob. For each element $b \in D_B$, Bob computes the encrypted values $\tilde{b} = r \cdot p(b) + b$, where $r$ is a random value, and sends them back to Alice. These values are decrypted by Alice and then matched with $D_A$. If $b \notin D_A$, then the decrypted value of $\tilde{b}$ will be a random value not matching any records in $D_A$; otherwise, it will find a match from $D_A$. The basic protocol described thus far required $O(|D_A| + |D_B|)$ communications and $O(|D_A \times D_B|)$ operations on encrypted values. [46] further optimizes the computational cost with Horner’s rule and cryptographic hashing to replace a single high-degree polynomial with several low-degree polynomials. This reduces the computational cost to $O(|D_B| \cdot \ln \ln |D_A|)$, and hence is sub-quadratic in $n$, for $n = \max(|D_A|, |D_B|)$. State of the art PSI techniques [101] further improve efficiency.

PSI techniques are limited to equality like matching functions, and extensions [46, 132] allow for matching rules that require exact match on at least $t$ out of $T$.
features. However these techniques achieve poor recall for general matching rules. For example, they do not extend to matching rules that involve conjunctions and disjunctions of similarity functions evaluated on multiple attributes. They also do not extend to complex distance metrics, such as CosineSimilarity(First Name) > 0.9 OR CosineSimilarity(Last Name) > 0.9, which are typical in record linkage tasks [49].

**PSI with Expansion (PSI+X).** The PSI technique can be used to achieve high recall for general matching rules by using the idea of expansion. Suppose $D_A$ and $D_B$ have the same domains, i.e., $\Sigma_A = \Sigma_B = \Sigma$. For every record $a \in D_A$, one could add all records $a' \in \Sigma$ such that $m(a, a') = 1$ to get an expanded database $D_A^x$. An equi-join between $D_A^x$ and $D_B$ returns the required output $D_A \Join m D_B$, and satisfies IND-S2PC. However, the expanded dataset can be many orders of magnitude larger than the original dataset making this protocol, PSI+X, inefficient (in the size of the original datasets). Moreover, enumerating all matches per record is hard for a complex matching function. For instance, if the matching function $m$ can encode Boolean 3-CNF formulas, then finding values for $a$ such that $m(a, a') = 1$ could be an intractable problem. In such a case, any efficient expansion algorithm may need to enumerate a superset of matches, further increasing the computational cost. Lastly, even for relatively simple matching functions, we empirically illustrate low recall of PSI and inefficiency of PSI+X protocols respectively in Section 5.5.

**PRL with Blocking (PRL+B).** Blocking is commonly used to scale up non-private record linkage. Formally,

**Definition 63 (Blocking (B)).** Given $k$ bins $\{B_0, ..., B_{k-1}\}$, records in $D_A$ and $D_B$ are hashed by $B$ to a subset of the $k$ bins. The set of records in $D_A$ (respectively $D_B$) falling into the $i^{th}$ bin are represented by $B_i(D_A)$ (respectively $B_i(D_B)$). A blocking strategy $B^S \subseteq [0,k) \times [0,k)$ specifies pairs of bins of $D_A$ and $D_B$ that are compared,
i.e. records in $B_i(D_A)$ are compared with records in $B_j(D_B)$ if $(i,j) \in \mathcal{B}_S$.

We sometimes use $\mathcal{B}$ to refer to the entire blocking algorithm as well as the blocking functions used in the algorithm. We refer to the set of pairs of records that are compared by a blocking strategy as candidate matches. A blocking strategy $\mathcal{B}_S$ is sub-quadratic if the number of candidate matches

$$\text{cost}_{\mathcal{B}_S}(D_A, D_B) = \sum_{(i,j) \in \mathcal{B}_S} |B_i(D_A)||B_j(D_B)|$$

is $o(n^2)$, for $n = \max(|D_A|, |D_B|)$. Blocking techniques are useful as a pre-processing step [110, 68, 64] to achieve sub-quadratic efficiency and high recall. We can use blocking as a pre-processing step for APC – secure comparison is performed only for the candidate matches – resulting in an efficient protocol with high recall. However, the blocking strategy itself can leak information about the presence or absence of a record in the database. This was illustrated using an attack by Cao et al. [20]. This is because the number of candidate matches can vary significantly even if $D_B$ and $D'_B$ differ in only one record. We formally prove this negative result for a large class of blocking techniques which use locality sensitive hashing (LSH). A majority of the hash functions used by blocking algorithms like q-gram based hash signatures [5] or SparseMap [110] are instances of LSH.

**Definition 64** (Locality Sensitive Hashing (LSH)[50]). A family of functions $H$ is said to be $(d_1, d_2, p_1, p_2)$-sensitive, where $d_2 > d_1$ and $p_1 > p_2$, if for all $h \in H$,

1. if $\text{dist}(a, b) \leq d_1$, then $\Pr[h(a) = h(b)] \geq p_1$, and
2. if $\text{dist}(a, b) > d_2$, then $\Pr[h(a) = h(b)] \leq p_2$.

An LSH-based blocking considers a set of bins where each bin consists of records with the same hash values for all $h \in H$. A popular blocking strategy is to compare all the corresponding bins, and results in a set of candidate matches $\{(a, b) | h(a) = h(b) \}$. 


h(b)\forall h \in H, a \in D_A, b \in D_B\}$. In general, we can show that any LSH based blocking cannot satisfy IND-S2PC.

**Theorem 65.** An LSH based blocking with a family of $(d_1, d_2, p_1, p_2)$-sensitive hashing functions $H$ cannot satisfy IND-S2PC.

The proof can be found in Appendix A.2.1.

**PRL with DP Blocking (PRL+DP**. Differential privacy has arisen as a gold standard for privacy in situations where it is ok to reveal statistical properties of datasets but not reveal properties of individuals. An algorithm satisfies differential privacy if its output does not significantly change when adding/removing or changing a single record in its input. More formally,

**Definition 66** ($(\epsilon, \delta)$-Differential Privacy[39]). A randomized mechanism $M : \mathcal{D} \rightarrow \mathcal{O}$ satisfies $(\epsilon, \delta)$-differential privacy (DP) if

$$Pr[M(D) \in O] \leq \epsilon Pr[M(D') \in O] + \delta$$

(5.3)

for any set $O \subseteq \mathcal{O}$ and any pair of neighboring databases $D, D' \in \mathcal{D}$ such that $D$ and $D'$ differ by adding/removing a record.

A recent line of work has designed differentially private blocking algorithms as a preprocessing step to APC. DP hides the presence or absence of a single record, and hence the number of candidate matches stays roughly the same on $D_B$ and $D'_B$ that differ in a single record. While this approach seems like it should satisfy all three of our desiderata, we have found that none of the protocols presented in prior work (on DP Blocking) [65, 77, 20] provide an end-to-end privacy guarantee. In fact, each paper in this line of work finds privacy breaches in the prior work. We also show in the proof of Theorem 82 (Appendix A.2.1) that even the most recent of these protocols in [20] does not satisfy an end-to-end privacy guarantee. This is
because of a fundamental disconnect between the privacy guarantees in the two steps of these algorithms. DP does not allow learning any fact about the input datasets with certainty, while IND-S2PC (and PRL protocols that satisfy this definition) can reveal the output of the function $f$ truthfully. On the other hand, while DP can reveal aggregate properties of the input datasets with low error, protocols that satisfy IND-S2PC are not allowed to leak any information beyond the output of $f$. Hence, DP and IND-S2PC do not naturally compose.

To summarize, none of the prior approaches that attempt to solve Problem 1 satisfy all three of our desiderata. Approaches that satisfy a strong privacy guarantee (IND-S2PC) are either inefficient or have poor recall. Efficient PRL with blocking or DP blocking fail to provide true end-to-end privacy guarantees. A correct conceptualization of an end-to-end privacy guarantee is critical for achieving correctness, privacy and efficiency. Hence, in the following sections, we first define an end-to-end privacy guarantee for PRL to address this challenge (Section 5.3), and then present algorithms in this privacy framework to achieve sub-quadratic efficiency and high recall (Section 5.4).

5.3 Output Constrained DP

Designing efficient and correct algorithms for PRL is challenging and non-trivial because there is no existing formal privacy framework that enables the trade-off between correctness, privacy, and efficiency. In this section, we propose a novel privacy model to achieve this goal.

5.3.1 Output Constrained Differential Privacy

Both IND-S2PC (Def. 62) and DP (Def. 66) ensure the privacy goal of not revealing information about individual records in the dataset. However, there is a fundamental incompatibility between the two definitions. IND-S2PC reveals the output of a
function truthfully; whereas, nothing truthful can be revealed under differential privacy. On the other hand, DP reveals noisy yet accurate (to within an approximation factor) aggregate statistics about all the records in the dataset; but, nothing other than the output of a pre-specified function can be revealed under IND-S2PC.

The difference between these privacy definitions can be illustrated by rephrasing the privacy notions in terms of a distance metric imposed on the space of databases. Without loss of generality, assume Alice is the adversary. Let $G = (V, E)$ denote a graph, where $V$ is the set of all possible databases that Bob could have and $E$ is a set of edges that connect neighboring databases. The distance between any pair of databases is the shortest path distance in $G$. Intuitively, the adversary Alice’s ability to distinguish protocol executions on a pair of databases $D_B$ and $D'_B$ is larger if the shortest path between the databases is larger.

DP can be represented by the set of edges that connect neighboring databases that differ in the presence or absence of one record, $|D_B \setminus D'_B \cup D'_B \setminus D_B| = 1$. This means, any pair of databases $D_B$ and $D'_B$ are connected in this graph by a path of finite length that is equal to the size of their symmetric difference. While an adversary can distinguish protocol executions between some pair of “far away” databases, the adversary can never tell with certainty whether the input was a specific database. On the other hand, under IND-S2PC, every pair of databases that result in the same output for $f(D_A, \cdot)$ for a given $D_A$ are neighbors. However, there is neither an edge nor a path between databases that result in different outputs. Thus the output constraint divides the set of databases into disjoint complete subgraphs (in fact equivalence classes).

**Example 67.** Consider databases with domain $\{1, 2, 3, 4, 5, 6\}$. Given $D_A = \{1, 2\}$, the graph $G$ for the database instances for $D_B$ are shown in Figure 5.1. For the graph of differential privacy in Figure 5.1(a), every pair of database instances that differ
in one record is connected by an edge and form a neighboring pair. For instance, 
\(D_B = \{1\}\) and \(D'_B = \{1, 2\}\) are neighbors under DP. Figure 5.1(b) considers an 
output which consists of the size of \(D_B\) and the intersection between \(D_B\) and \(D_A\). Hence, all the instances in \(G_{IND-S2PC}\) have the same datasize and have the same 
intersection with \(D_A = \{1, 2\}\). For example, the fully connected 6 database instances 
all have 2 records, but have no intersection with \(D_A\). The instance \(\{1, 2\}\) has no 
neighboring databases, as it is same as the output, and hence none of the records in 
this database instance requires privacy protection.

Comparing these two graphs, we can see that all instances in \(G_{DP}\) are con-
ected, and hence an adversary can not distinguish protocol executions on any pair 
of databases with certainty, but is allowed to learn statistical properties (with some 
error). This is not true under \(G_{IND-S2PC}\), where some instances are disconnected. 
For instance, an adversary can distinguish between protocol executions on \(\{1, 2\}\) and 
\(\{1, 5\}\) since they give different outputs when matched with \(D_A\).

From Example 67, it is clear that the privacy guarantees given by DP and IND- 
S2PC are different. To ensure scalable record linkage with formal privacy guarantees, 
we need the best of both worlds: the ability to reveal records that appear in the 
match truthfully, the ability to reveal statistics about non-matching records, and yet 
not reveal the presence or absence of individual non-matching records in the dataset. 
Hence, we propose a weaker, but end-to-end, privacy definition for the two party 
setting.

**Definition 68 (f-Neighbors).** Given function \(f : \mathcal{D}_A \times \mathcal{D}_B \rightarrow \mathcal{O}\) and \(D_A \in \mathcal{D}_A\). 
For any pairs of datasets \(D_B, D'_B\), let \(\Delta(D_B, D'_B) = D_B \setminus D'_B \cup D'_B \setminus D_B\). This is 
the symmetric difference between \(D_B\) and \(D'_B\), and is the set of records that must 
be deleted and added to \(D_B\) to get \(D'_B\). \(D_B\) and \(D'_B\) are neighbors w.r.t to \(f(D_A, \cdot)\), 
denoted by \(N(f(D_A, \cdot))\) if

136
Figure 5.1: Neighboring databases for (a) DP, and (b) IND-S2PC for Example 67.

(1) $f(D_A, D_B) = f(D_A, D_B')$,
(2) $\triangle(D_B, D_B') \neq \emptyset$, and
(3) there is no database $D''_B \in D_B$, where $f(D_A, D_B) = f(D_A, D''_B)$, such that $\triangle(D_B, D''_B) \subseteq \triangle(D_B, D_B')$.

$N(f(\cdot, D_B))$ is similarly defined.

The third condition ensures that $D_B$ and $D'_B$ are minimally different in terms of record changes.

**Definition 69** (Output Constrained DP). A 2-party PRL protocol $\Pi$ for computing function $f : D_A \times D_B \to \mathcal{O}$ is $(\epsilon_A, \epsilon_B, \delta_A, \delta_B, f)$-constrained differential privacy (DP) if for any $(D_B, D'_B) \in N(f(D_A, \cdot))$, the views of Alice during the execution of $\Pi$ to any probabilistic polynomial-time adversary $T$ satisfies

$$Pr[T(VIEW^\Pi_A(D_A, D_B)) = 1] \leq e^{\epsilon_B} Pr[T(VIEW^\Pi_A(D_A, D'_B)) = 1] + \delta_B$$

(5.4)

and the same holds for the views of Bob with $\epsilon_A$ and $\delta_A$.

If $\epsilon_A = \epsilon_B = \epsilon$, $\delta_A = \delta_B = \delta$, we simply denote it as $(\epsilon, \delta, f)$-constrained DP. Similar to DP, Output Constrained DP satisfies composition properties that are useful for
Theorem 70 (Sequential Composition). Given $\Pi_1$ is $(\epsilon_1, \delta_1, f)$-constrained DP, and $\Pi_2$ is $(\epsilon_2, \delta_2, f)$-constrained DP, then applying these two protocols sequentially, i.e. $\Pi_2(D_A, D_B, \Pi_1(D_A, D_B))$ satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2, f)$-constrained DP.

Theorem 71 (Post-processing). Given $\Pi$ is $(\epsilon, \delta, f)$-constrained DP, and let $\omega^\Pi(D_A, D_B)$ be the output after the execution of $\Pi$, then any probabilistic polynomial (in $\kappa$) function $g(\omega^\Pi(D_A, D_B))$ satisfies $(\epsilon, \delta, f)$-constrained DP.

See Appendix A.2.2 and A.2.2 for the proofs of Theorem 70 and Theorem 71 respectively. Output constrained DP inherits other desirable properties from DP, for instance, its robustness to attacks [112, 59]. We omit details due to space constraints.

5.3.2 Differential Privacy for Record Linkage

PRL can be a direct application of Output Constrained Differential Privacy by considering $f_{m'}$. We have the following theorem to define the neighboring databases for PRL.

Theorem 72 (Neighbors for PRL). Given the function $f_{m'}$ in PRL, if $(D_B, D'_B) \in N(f_{m'}(D_A, *))$ for a given $D_A \in D$, then $|D_B| = |D'_B|$, $D_B$ and $D'_B$ must differ in only one pair of non-matching records with respect to the given $D_A$, i.e. $D'_B = D_B - b + b'$ and $b \neq b'$, where $m(b, a) = 0$ and $m(b', a) = 0$ for all $a \in D_A$.

Proof. The output constraint $f_{m'}(D_A, D_B) = f_{m'}(D_A, D'_B)$ implies that $|D_B| = |D'_B|$ and $D_A \bowtie_m D_B = D_A \bowtie_m D'_B$. If $D_B$ and $D'_B$ differ in a matching record, then their matching outputs with a given $D_A$ are different. Hence $D_B$ and $D'_B$ must differ in one or more non-matching records. In addition, to ensure $|D_B| = |D'_B|$, the number of non-matching records added to $D_B$ to get $D'_B$ must be the same as the number of non-matching records removed from $D_B$. If $\triangle(D_B, D'_B)$ contains more
than one pair of record additions and deletions, a subset of $\Delta(D_B, D'_B)$ can give a valid $D''_B$ such that $f_{m_m}(D_A, D_B) = f_{m_m}(D_A, D'_B)$. Hence, a neighboring pair $D_B, D'_B$ differ by exactly one pair of non-matching records.

Next we define the privacy guarantee that allows us to design efficient PRL protocols with provable privacy guarantees.

**Definition 73 (DPRL).** A 2-party PRL protocol $\Pi$ for computing function $f_{m_m} : D_A \times D_B \rightarrow O$ is $(\epsilon_A, \epsilon_B, \delta_A, \delta_B)$-DPRL if $\Pi$ satisfies $(\epsilon_A, \epsilon_B, \delta_A, \delta_B, f_{m_m})$-constrained DP.

### 5.3.3 Related Privacy Definitions

In this section we discuss related privacy definitions and their connections with DPRL. First, both DPRL and IND-S2PC assume a computationally bounded model. We show that DPRL is a weaker guarantee than IND-S2PC.

**Theorem 74.** All IND-S2PC protocols for record linkage satisfy $(0, \text{negl}(\kappa))$-DPRL.

*Proof.* IND-S2PC for record linkage is equivalent to DPRL with $\epsilon = 0$ and $\delta = \text{negl}(\kappa)$. The $\delta$ in DPRL is always greater than $\text{negl}(\kappa)$ but smaller than $o(1/n)$. □

Hence, APC, PSI, and PSI+X techniques that satisfy IND-S2PC, guarantee $(0, \text{negl}(\kappa))$-DPRL as well.

Indistinguishable computationally differential privacy (IND-CDP-2PC) [91] is another privacy notion under a computationally bounded model, and is a direct extension of DP to the two party setting where both parties are computationally bounded. DPRL is weaker than IND-CDP-2PC. Formally

**Theorem 75.** If a protocol for record linkage satisfies $\epsilon/2$-IND-CDP-2PC, then it satisfies $(\epsilon, \delta)$-DPRL.
The factor 2 arises since neighboring databases protected by DPRL have a symmetric difference of 2, while neighboring databases under IND-CDP-2PC have a symmetric difference of 1. The detailed proof can be found in Appendix A.1.2.

Blowfish Privacy [59] generalizes differential privacy to problems where constraints on the input database must hold (e.g., when certain query answers have been released by the database exactly). Output Constrained DP, including DPRL, is an extension of Blowfish in two ways: (1) from a computationally unbounded model to a computationally bounded model; (2) from a single-party setting to a two-party setting. Note that with the output constraint \( f_{\text{rel}}(D_A, D_B) = f_{\text{rel}}(D_A, D'_B) \) for record linkage, the number of different records between neighboring databases \( D_B \) and \( D'_B \) is only two. This is not necessarily true for other applications of Output Constrained DP, or Blowfish Privacy. This property is desirable for DP based algorithms since larger distances between neighboring databases typically require larger perturbation to hide the difference between neighbors resulting in poorer utility.

Another instantiation of Blowfish privacy, called Protected DP [69], aims to ensure the privacy of a protected subpopulation. In contrast, an unprotected “targeted” subpopulation receives no privacy guarantees. In DPRL, one could think of the non-matching records as the protected subpopulation, and the matching records as the targeted subpopulation. However, unlike in Protected DP, in DPRL the set of protected records are learned as an output of the DPRL protocol, and hence are not available as an input to the protocol like the targeted subpopulation in the Protected DP algorithms.

5.4 Protocols for DPRL

In this section, we introduce protocols that satisfy DPRL and permit a 3-way trade-off between correctness, privacy and efficiency. We first present a class of protocols that achieves \((\epsilon, \text{negl}(\kappa))\)-DPRL by using a blocking strategy that satisfies local dif-
ferential privacy (DP). Though these protocols achieve high recall with a sufficiently small privacy parameter, they only achieve a constant factor speedup in efficiency. Next, we present the Laplace Protocol (LP) that achieves all three desiderata of high recall, privacy and subquadratic efficiency. This protocol hides non-matching records by adding Laplace noise to the blocking strategy. We also show that attempts from prior work to use Laplace noise in blocking fail to satisfy DPRL (Theorem 82). Moreover, we design a Sort & Prune (SP) heuristic that is used in conjunction with LP (as well as the local DP based protocols) and helps additionally tradeoff efficiency and recall. Finally, we present the Greedy Match & Clean heuristic optimization (GMC), that can further improve efficiency. All the protocols presented in this section are proven to satisfy DPRL.

5.4.1 Local DP Protocol

Let $\mathcal{B}$ be a blocking that randomly hashes records into a pre-specified set of $k$ bins, such that for all $i \in [1 \ldots k],$

$$\Pr[\mathcal{B}(b) = i] \leq e^\epsilon \Pr[\mathcal{B}(b') = i].$$

(5.5)

Such a blocking $\mathcal{B}$ satisfies $\epsilon$-local DP (as defined in Appendix A.1.3), since each record is perturbed locally independent of the other records. We show that protocols that combine a local differentially private blocking with IND-S2PC protocols for record linkage can achieve $(\epsilon, \operatorname{negl}(\kappa))$-DPRL.

**Theorem 76.** All IND-S2PC protocols for record linkage with $\epsilon$-local differentially private blocking satisfies $(\epsilon, \operatorname{negl}(\kappa))$-DPRL.

The proof can be found in Appendix A.3.2. Such local differentially private protocols can be constructed from well known local differentially private algorithms based on randomized response (RR) [42] or the Johnson-Lindenstrauss (JL) transformation [14], where each record is hashed independent of others. We refer the reader
Algorithm 3: Laplace Protocol (LP)

Input: \(D_A, D_B, \epsilon_A, \epsilon_B, \delta_A, \delta_B, B\) (including \(B^S\))

Output: \(O\)

1 // Alice performs the following:
2 \(\tilde{B}(D_A) \leftarrow \text{LapNoise}(D_A, B, \epsilon_A, \delta_A)\);
3 // Bob performs the following:
4 \(\tilde{B}(D_B) \leftarrow \text{LapNoise}(D_B, B, \epsilon_B, \delta_B)\);
5 // Alice and Bob perform the following:
6 \(O = \emptyset\);
7 // Sort & prune \(B^S\) (S 5.4.3)
8 for \((i, j) \in B^S\) do
9     for \(a \in \tilde{B}_i(D_A)\) and \(b \in \tilde{B}_j(D_B)\) do
10        Add SMC\((a, b)\) to \(O\);
11 end
12 // Greedy match & clean (S 5.4.4)
13 end
14 return \(O\);

to Appendix A.3.2 for a concrete blocking algorithm based on RR. We show that while this algorithm permits high recall and privacy, it does not improve efficiency by more than a constant factor (a function of \(\epsilon\)) (Theorem 93). Whether any local DP based blocking algorithms can achieve subquadratic efficiency is an interesting open question.

5.4.2 Laplace Protocol (LP)

Algorithm Description. In this protocol, Alice and Bob agree on a blocking function \(B\) with \(k\) bins and strategy \(B^S\), which we take as input to the protocol. The Laplace Protocol (LP, as shown in Algorithm 3) works by inserting a carefully chosen number of dummy records into each bin of the blocking strategy such that the bin sizes are differentially private. While candidate matches may contain dummy records, they do not contribute to the output set of matches, because the dummy records do not match any record. These candidate matches are then securely matched using an IND-S2PC algorithm.
**Algorithm 4: Add Laplace Noise**

1. function LapNoise \((D, B, \epsilon, \delta)\);
2. for \(B_i \in B\) do
3. \(\eta_i \sim \text{Lap}(\epsilon, \delta, \Delta B)\);
4. \(\tilde{B}_i(D) \leftarrow \text{add } \eta_i^+ = \max(\eta_i, 0) \text{ dummy records to } B_i(D)\);
5. end
6. return \(\tilde{B}(D)\);

In the first step (Lines 1-4) of the protocol shown in Algorithm 3, Alice and Bob take their inputs \(D_A\) and \(D_B\), the agreed blocking protocol \(B\), and privacy parameters \(\epsilon_A\), \(\epsilon_B\), \(\delta_A\), and \(\delta_B\) as input, and compute noisy bins \(\tilde{B}(D_A)\) and \(\tilde{B}(D_B)\) respectively. The noisy bins are constructed as follows (Algorithm 4). Records in \(D\) are first hashed into bins according to the blocking protocol \(B\), and \(B(D)\) denotes the set of bins of records from \(D\). Then the counts of the bins are perturbed using noise drawn from a truncated and discretized Laplace distribution, such that the noisy counts satisfy \((\epsilon, \delta)\)-DPRL. The Laplace noise depends on not only the privacy parameters \(\epsilon\) and \(\delta\), but also the sensitivity of the given blocking protocol \(B\).

**Definition 77 (Sensitivity of \(B\)).** The sensitivity of the blocking strategy \(B\) for Bob, denoted by \(\Delta B_B\) is

\[
\max_{D_A \in \mathcal{D}} \max_{(D_B, D_B') \in \mathcal{N}(f_{\text{sum}}(D_A, \cdot))} \sum_{i=0}^{k} ||B_i(D_B)| - |B_i(D_B')||,
\]

the maximum bin count difference between \(D_B\) and \(D_B'\) for any \((D_B, D_B') \in \mathcal{N}(f_{\text{sum}}(D_A, \cdot))\) for all \(D_A \in \mathcal{D}\). \(\Delta B_A\) for Alice is similarly defined.

If the hashing of \(B\) is the same for Alice and Bob, then \(\Delta B_A = \Delta B_B = \Delta B\). We assume this in our paper. If \(B\) hashes each record to at most \(k'\) bins, then \(\Delta B = 2k'\).

**Definition 78 (Lap(\(\epsilon, \delta, \Delta B\))).** A random variable follows the Lap(\(\epsilon, \delta, \Delta B\)) distribution if it has a probability density function

\[
\Pr[\eta = x] = p \cdot e^{-(\epsilon/\Delta B)|x - \eta^0|}, \forall x \in \mathbb{Z},
\]
where \( p = \frac{e^{\epsilon/\Delta B} - 1}{e^{\epsilon/\Delta B} + 1} \), and \( \eta^0 = -\frac{\Delta B \ln(\frac{e^{\epsilon/\Delta B} + 1}{1 - (1-\delta)^{1/\Delta B}})}{\epsilon} \).

This distribution has a mean of \( \eta_0 \) and takes both positive and negative values. LP draws a noise value \( \eta \) from this distribution, and truncates it to 0 if \( \eta \) is negative. Then, \( \eta \) dummy records are added to the bin. These dummy records lie in an expanded domain, such that they do not match with any records in the true domain.

After Alice and Bob perturb their binned records, they will initiate secure matching steps to compare candidate matches, i.e. records in \( \tilde{B}_i(D_A) \times \tilde{B}_j(D_B) \) if \( (i, j) \in B^S \).

For each candidate match \( (a, b) \), Alice and Bob participate in a two party secure matching protocol \( SMC(a, b) \) that outputs the pair \( (a, b) \) to both Alice and Bob if \( m(a, b) = 1 \) (true matching pair) and null otherwise. Secure matching can be implemented either using garbled circuits [130] or (partially) homomorphic encryption [97], depending on the matching rule (see Appendix A.3.1 for an example).

**Correctness Analysis.** Compared to the original non-private blocking protocol \( B \), no records are deleted, and dummy records do not match any real record. Hence,

**Theorem 79.** Algorithm 3 gives the same recall as the non-private blocking protocol \( B \) it takes as input.

**Privacy Analysis.** Next, we show that LP satisfies DPRL.

**Theorem 80.** Algorithm 3 satisfies \( (\epsilon_A, \epsilon_B, \delta_A, \delta_B) \)-DPRL.

**Proof.** We prove privacy for Bob (the proof for Alice is analogous). In this protocol, Alice with input data \( D_A \) has a view consisting of (1) the number of candidate matching pairs arising in each \( (i, j) \in B^S \), (2) the output for each candidate matching pair. Algorithm 3 is the composition of two steps: (a) add dummy records to bins, and (b) secure comparison of records within bins.

Consider a neighboring pair \( (D_B, D_B') \in N(f_{m_a}(D_A, \cdot)) \) for a given \( D_A \). By Theorem 72, \( D_B \) and \( D_B' \) differ in only one non-matching record with respect to \( D_A \).
i.e. $D'_B = D_B - b_s + b'_s$ and $b_s \neq b'_s$, where $m(b_s, a) = 0$ and $m(b'_s, a) = 0$ for all $a \in D_A$. $D_B$ and $D'_B$ can differ by at most $\Delta B$ in their bin counts. We show in Lemma 92 (Appendix) that Algorithm 4 adds a sufficient number of dummy records to hide this difference: with probability $1 - \delta_B$, the probabilities of generating the same noisy bin counts for Bob, and hence the same number of candidate matching pairs consisting in each $(i, j) \in B^S$ from $D_B$ and $D'_B$ are bounded by $e^{\epsilon_B}$. Thus, Step (a) ensures $(\epsilon_B, \delta_B)$-DPRL for Bob. Given a fixed view from Step (a) which consists of the noisy bin counts and encrypted records from $\tilde{B}(D_B)$, Alice’s view regarding the output for each candidate matching pair $(a, b)$ is the same. The encrypted records for a given noisy bin counts can only differ in $b_s$ and $b'_s$, but both of them lead to the same output for each candidate matching, because they do not match any records in $D_A$. Each secure pairwise comparison satisfies $(0, \text{negl}(\kappa))$-DPRL, and since there are at most $n^2$ comparisons (recall $\kappa > n = \max(|D_A|, |D_B|)$). Thus Step (b) satisfies $(0, \text{negl}(\kappa))$-DPRL.

Therefore, using similar arguments for Alice and sequential composition, we get that Algorithm 3 satisfies DPRL.

\textbf{Theorem 81.} If Algorithm 3 (LP) takes $\eta_0 = \ln^2 n \cdot \Delta B / \epsilon$ for Eqn. (5.6), then LP satisfies $(\epsilon_A, \epsilon_B, o(1/n^k), o(1/n^k))$-DPRL, for any $k > 0$, where $n = \max(|D_A|, |D_B|)$.

\textbf{Proof.} (sketch) Taking $\eta_0 = \ln^2 n \cdot \Delta B / \epsilon$, the failing probability $\delta = 1 - (1 - \frac{1}{\ln n (\epsilon / \Delta B + 1)})^{\Delta B} \leq \frac{c}{n^k}$ for some constant $c$ (in terms of $\epsilon, \Delta B$). Hence $\delta = o(1/n^k)$ for all $k > 0$. 

LP only adds non-negative noise to the bin counts. One could instead add noise that could take positive and negative values, and suppress records if the noise is negative. We call this protocol LP-2. This is indeed the protocol proposed by prior work [65, 77, 20] that combined APC with DP blocking. However, we show that this minor change in LP results in the protocol violating DPRL (even though the noise
addition seems to satisfy DP)! Hence, LP-2 also does not satisfy IND-CDP-2PC (by Theorem 75).

**Theorem 82.** For every non-negative $\epsilon, \delta < \frac{\epsilon^2}{2\epsilon^2}$, there exists a pair of neighboring databases for which LP-2 does not ensure $(\epsilon, \delta)$-DPRL, where $p = \frac{\epsilon/n\delta - 1}{\epsilon/n\delta + 1}$.

**Proof.** (sketch) The output of the record suppression step is dependent on the ratio between the matching and non-matching records in the bin. This introduces a correlation between the matching and non-matching records. Consider a neighboring pair $D_B$ and $D'_B$ that differ by a non-matching pair $(b_*, b'_*)$ for a given $D_A$. If $b_*$ is in a bin full of non-matching records with $D_A$, and $b'_*$ is in a bin full of matching records with $D_A$ (except $b'_*$). $D_B$ is more likely to output all matching pairs than $D'_B$ if some record is suppressed. The detailed proof can be found in Appendix A.2.1. □

**Efficiency Analysis.** Last, we present our result on the efficiency of LP. Note that the communication and computational costs for LP are the same as $O(\text{cost}_{BS})$, where $\text{cost}_{BS}$ is the number of candidate matches, if you consider the communication and computational costs associated with a single secure comparison as a constant. Hence, we analyze efficiency in terms of the number of candidate pairs $\text{cost}_{BS}$ in LP.

**Theorem 83.** Given a blocking protocol $B$ with $k$ bins and blocking strategy $BS$, such that the number of candidate matches for $D_A$ and $D_B$, $\text{cost}_{BS}(D_A,D_B)$, is sub-quadratic in $n$, i.e. $o(n^2)$, where $n = \max(|D_A|,|D_B|)$. If (1) the number of bins $k$ is $o(n^c)$ for $c < 2$, and (2) each bin of a party is compared with $O(1)$ number of bins from the opposite party, then the expected number of candidate matches in Algorithm 3 is sub-quadratic in $n$.

**Proof.** Given $\epsilon$ and $\delta$, the expected number of dummy records added per bin $\mathbb{E}(\eta^+)\epsilon$ is a constant denoted by $c_\eta$ (Def. 78). Each bin of a party is compared with at most
$c_b$ bins from the opposite party, where $c_b$ is a constant. The number of candidate matches in LP is a random variable, denoted by $COST$, with expected value

$$\mathbb{E}(COST) = \sum_{(i,j) \in B^S} \mathbb{E}(|\tilde{B}_i(D_A)||\tilde{B}_j(D_B)|)$$

$$= \sum_{(i,j) \in B^S} |B_i(D_A)||B_j(D_B)| + \sum_{(i,j) \in B^S} \mathbb{E}(\eta_i^+)\mathbb{E}(\eta_j^+)$$

$$+ \sum_{(i,j) \in B^S} (\mathbb{E}(\eta_i^+)|B_j(D_B)| + \mathbb{E}(\eta_j^+)|B_i(D_A)|)$$

$$< cost_{BS}(D_A, D_B) + c_b^2c_bk + 2c_bk.$$  

Since $cost_{BS}(D_A, D_B)$ and $k$ are sub-quadratic in $n$, $\mathbb{E}(COST)$ is also sub-quadratic in $n$. When $\delta$ is a negligible term as defined in Theorem 81, the noise per bin is $O(\ln^2 n)$. As $k$ is $o(n^c)$ for $c < 2$, the expected value of $COST$ is still sub-quadratic in $n$.  

Conditions (1) and (2) in the above theorem are satisfied by, for instance, sorted neighborhood, and distance based blocking [30] (we use the latter in our experiments). While the asymptotic complexity of LP is sub-quadratic, it performs at least a constant number of secure comparisons for each pair $(i, j) \in B^S$ even if there are no real records in $B_i(D_A)$ and $B_j(D_B)$. We can reduce this computational overhead with a slight loss in recall (with no loss in privacy) using a heuristic we describe in the next section.

5.4.3 Sort & Prune $B^S$ (SP)

Algorithm 3 draws noise from the same distribution for each bin, and hence the expected number of dummy records is the same for every bin. The bins with higher noisy counts will then have a higher ratio of true to dummy records. This motivates us to match candidate pairs in bins with high noisy counts first. Instead of comparing bin pairs in $B^S$ in a random or index order, we would like to sort them based on
the noisy counts of $\tilde{B}(D_A)$ and $\tilde{B}(D_B)$. Given a list of descending thresholds $\bar{t} = [t_1, t_2, t_3, \ldots]$, the pairs of bins from the matching strategy $B^S$ can be sorted into groups denoted by $B^{S,t_l}$ for $l = 1, 2, \ldots$, where

$$B^{S,t_l} = \{ |\tilde{B}_i(D_A)| > t_l \land |\tilde{B}_j(D_B)| > t_l | (i, j) \in B^S \}.$$  

Each group consists of bin pairs from $B^S$ with both noisy counts greater than the threshold.

We let the thresholds $\bar{t}$ be the deciles of the sorted noisy bin sizes of $\tilde{B}(D_A)$ and $\tilde{B}(D_B)$. As the threshold decreases, the likelihood of matching true records instead of dummy records drops for bins. Alice and Bob can stop this matching process before reaching the smallest threshold in $\bar{t}$. If the protocol stops at a larger threshold, the recall is smaller. In the evaluation, if the protocol stops at 10% percentile of the noisy bin counts, the recall can reach more than 0.95. This allows a trade-off between recall and efficiency for a given privacy guarantee. We show that this step also ensures DPRL.

**Corollary 84.** Algorithm 3 with sort & prune step (SP) satisfies $(\epsilon_A, \epsilon_B, \delta_A, \delta_B)$-DPRL.

**Proof.** Similar to the proof in Theorem 80, Alice with input data $D_A$ has a view consisting of (1) the number of candidate matching pairs arising in each $(i, j) \in B^S$, and (2) the output for each candidate matching pair. As SP is a post-processing step based on the noisy bin counts, which is part of Alice’s original view, the overall protocol still satisfies the same DPRL guarantee by Theorem 71 (post-processing).

We next present an optimization that also uses a form of post-processing to significantly reduce the number of secure pairwise comparisons in practice, but whose privacy analysis is more involved than that of SP.
**Algorithm 5:** Greedy match and clean

**Input:** $O, \tilde{B}(D_A), \tilde{B}(D_B)$

1. repeat

2. // Alice performs the following:
   3. $O_A \leftarrow \pi_A O, O_B \leftarrow \pi_B O$;
   4. $\tilde{B}(D_A) \leftarrow \tilde{B}(D_A) - O_A$;
   5. $O' \leftarrow \text{PlainMatch}(O_B, \tilde{B}(D_A))$;
   6. Add $O'$ to $O$ and send $O$ to Bob;

7. // Bob performs the following:
   8. $O_A \leftarrow \pi_A O, O_B \leftarrow \pi_B O$;
   9. $\tilde{B}(D_B) \leftarrow \tilde{B}(D_B) - O_B$;
   10. $O' \leftarrow \text{PlainMatch}(O_A, \tilde{B}(D_B))$;
   11. Add $O'$ to $O$ and send $O$ to Alice;

12. until $O$ received by Alice has no updates;

5.4.4 Greedy Match & Clean (GMC)

LP executes a sequence of secure comparison protocols, one per candidate pair. After every comparison (or a block of comparisons), Alice and Bob learn a subset of the matches $O$. Based on the current output $O$, Alice and Bob can greedily search matching pairs in the clear from their respective databases (Lines 5, 10 in Algorithm 5), and add the new matching pairs to the output set $O$ until no new matching pairs can be found. In addition, Alice and Bob can remove records in the output from the bins $\tilde{B}(D_A)$ and $\tilde{B}(D_B)$ to further reduce the number of secure pairwise comparisons (Lines 4, 9). We can see that this optimization step is not simply post-processing, because it makes use of the true record in plain text for matching. In traditional differential privacy, when the true data is used for computation, the privacy guarantee decays. However, we show that this is not true for the GMC step in the setting of DPRL.

**Theorem 85.** Algorithm 3 with the greedy match & clean step (GMC) in Algorithm 5 satisfies $(\epsilon_A, \epsilon_B, \delta_A, \delta_B)$-DPRL.

**Proof.** First consider the privacy for Bob. Alice with input data $D_A$, has a view
consisting of (1) the number of candidate matching pairs arising in each \((i,j) \in B^S\), (2) the output for each candidate matching pair, (3) the output from plaintext comparisons with output records.

Consider a neighboring pair \((D_B, D'_B) \in N(f_{m_0}(D_A, \cdot))\) for a given \(D_A\). By Theorem 72, \(D_B\) and \(D'_B\) differ in only one non-matching record with respect to \(D_A\), i.e. \(D'_B = D_B - b_s + b'_s\) and \(b_s \neq b'_s\), where \(m(b_s, a) = 0\) and \(m(b'_s, a) = 0\) for all \(a \in D_A\). \(D_B\) and \(D'_B\) can differ by at most \(\Delta B\) in their bin counts. Similar to the proof for Theorem 80, the first step of the protocol adds dummy records to bins, and satisfies \((\epsilon_B, \delta_B)\)-DPRL.

In the second step, given a fixed view \(\text{VIEW}^*\) from the first step which consists of the noisy bin counts and encrypted records from \(\tilde{B}(D_B)\), Alice’s view regarding the output for each candidate matching pair \((a, b)\) is the same regardless \((a, b)\) are compared securely or in plaintext. Alice’s view regarding the output from plaintext comparisons with the records in the output set is also the same for a fixed \(\text{VIEW}^*\) from the first step. The encrypted records for a given noisy bin counts can only differ in \(b_s\) and \(b'_s\), and they will never be pruned away. Both of them also lead to the same output for secure pairwise comparisons or plaintext comparisons, because they do not match any records in \(D_A\). Thus Step (b) satisfies \((0, \text{negl}(\kappa))\)-DPRL.

Therefore, using similar arguments for Alice and sequential composition, we get that Algorithm 3 satisfies DPRL.

With the same privacy guarantee, LP with the GMC step can even improve the efficiency of LP without sacrificing recall.

**Theorem 86.** LP with the greedy match & clean step (GMC) performs no more secure pairwise comparisons than LP, and outputs at least as many matching pairs as LP.

We refer the reader to Appendix A.2.3 for the proof. Both SP and GMC are also
applicable on the local DP based protocols for the similar reasoning. Hence, we will only show how each optimization helps improve the efficiency of the basic LP in the evaluation.

5.5 Evaluation

We empirically evaluate the correctness, privacy, and efficiency of the protocols proposed in Section 5.4. Our experiments demonstrate the following results:

- The Laplace Protocol (LP, which includes all the optimizations) proposed in Section 5.4 is over 2 orders of magnitude more efficient than the baseline approaches while still achieving a high recall and end-to-end privacy.

- At any given level of privacy, LP incurs a computational cost that is near linear in the input database size.

- Greedy match & clean and Sort & prune optimization help reduce communication and computation costs. The former results in 50% lower cost than unoptimized LP in some cases.

- We explore the 3-way trade-offs between correctness, privacy, and efficiency of LP.

5.5.1 Evaluation Setup

Datasets and Matching Rules.

*Taxi dataset (Taxi):* To simulate linkage in the location domain, we extract location distribution information from the TLC Trip Record Data [118]. Each record includes a pickup location in latitude-longitude coordinates (truncated to 6 decimal places) and the date and hour of the pickup time. Taking the original dataset as $D_A$, we create $D_B$ by perturbing the latitude-longitude coordinates of each record in $D_A$ with random values uniformly drawn from $[-\theta, +\theta]^2$, where $\theta = 0.001$. Each day
has approximately 300,000 pickups. The data size can be scaled up by increasing
the number of days, $T$. We experiment with $T = 1, 2, 4, 8, 16$, with $T = 1$ being the
default. Any pair of records $a, b \in \mathcal{T}$ are called a match if they have the same day
and hour, and their Euclidean distance in location is no larger than $\theta$. The location
domain is within the bounding box (40.711720N, 73.929670W) and (40.786770N,
74.006600W). We project the locations into a uniform grid of $16 \times 16$ cells with size
0.005 $\times$ 0.005. A blocking strategy $\mathcal{B}^S$ based on the pickup time and grid is applied
to both datasets, resulting in $(16 \times 16 \times 24T)$ bins. $\mathcal{B}^S$ compares pairs of bins that
are associated with the same hour, and corresponding/neighboring grid cells. Thus,
each bin in $\mathcal{B}(D_A)$ is compared with 9 bins in $\mathcal{B}(D_B)$.

\textit{Abt and Buy product dataset (AB):} These datasets are synthesized from the
online retailers Abt.com and Buy.com [76] who would like to collaboratively study
the common products they sell as a function of time. Each record in either dataset
consists of a product name, brand and the day the product was sold. The product
names are tokenized into trigrams, and hashed into a bit vector with a bloom filter
having domain $\mathcal{T} = \{0, 1\}^{50}$. We consider 16 brands, and sample 5,000 records per
day from the original datasets for Abt and Buy each. The data size can be scaled
up with $T$ for $T = 1, 2, 4, 8, 16$, with 1 being the default for $T$. Any pair of records
$a, b \in \mathcal{T}$ are called a match if (a) they are sold on the same day, (b) they are of
the same brand, and (c) the hamming distance between their vectorized names is no
more than $\theta = 5$. A blocking strategy hashes records having the same value for day
and brand into the same bin, resulting in $16T$ bins, and compares records falling in
the corresponding bins.

\textbf{Protocols.} We evaluate four DPRL protocols: (1) Laplace protocol (LP), (2) all-
pairwise comparisons (APC), (3) private set intersection (PSI), and (4) PSI with
expansion (PSI+X). The default LP consists of the basic protocol described in Al-
Algorithm 3 along with optimization steps (SP and GMC) in S 5.4.3 and 5.4.4.

**Metrics.** There are three dimensions in the trade-off space: correctness, privacy and efficiency. The correctness of a protocol is measured by the recall, which is the fraction of the matching pairs output by the algorithm, as defined in Eqn. (5.1), with larger values close to 1 being better. The privacy metric is specified in advance for each algorithm using parameters $\epsilon, \delta$. For AP, PSI, and PSI+X, $\epsilon = 0$ and $\delta = \text{negl}(\kappa)$ by Theorem 74. We consider $\epsilon_A = \epsilon_B = \epsilon$ and $\delta_A = \delta_B$ for $\epsilon \in \{0.1, 0.4, 1.6\}$ and $\delta \in \{10^{-9}, 10^{-7}, 10^{-5}\}$ for LP. The default value for $\epsilon$ and $\delta$ is 1.6 and $10^{-5}$, respectively. Finally, we define efficiency of APC and LP protocols for a given dataset as the number of secure pairwise comparisons, and denote this by cost. The cost of PSI and PSI+X can be estimated as $\gamma n \ln \ln(n)$, where $\gamma$ is the expansion factor, or the ratio of sizes of the expanded and true databases. This represents the number of operations on encrypted values. For PSI, $\gamma$ is 1. We use the number of secure comparison/operations on encrypted values rather than the wallclock times as a measure of efficiency, since these operations dominate the total time. We discuss wallclock times in more detail in S 5.5.2.

**Figure 5.2:** The average log (cost) of LP, APC, PSI+X and non-private matching (np) for the Taxi and AB datasets vs log (data size). LP give lower costs than the baselines PSI+X and APC for all values of $\epsilon = 0.1, 0.4, 1.6$ and $\delta = 10^{-5}$, and scales near linearly.
5.5.2 Results and Discussions

Efficiency and scalability. In this section, we empirically investigate how LP scales as the data size increases \( T \in \{1, 2, 4, 8, 16\} \) in comparison to baselines APC and PSI+X, when all the algorithms achieve 100% recall. We do not include PSI as its recall is close to 10%. LP is evaluated at privacy parameter \( \epsilon \in \{1.6, 0.4, 0.1\} \) and fixed \( \delta = 10^{-5} \). At each \( \epsilon \), we report the average number of candidate pairs for LP over 10 runs for each value of \( T \). To achieve 100% recall, PSI+X expands each record \( b \) in \( D_B \) to every other record \( b' \) within a \( \theta \)-ball around \( b \). We add 2,369,936 records per record in the AB dataset, and 1000\( 2\pi \) records per record in the Taxi dataset.

In Figure 5.2, we report the log(base 10) value of the average cost, \( \log(cost) \), with respect to the log value of data size \( \log(n) \) for PSI+X, APC, and LP with varying \( \epsilon \) and the non-private setting (np) when they achieve a recall of 1.0. Results for Taxi are shown on the left, and AB are shown on the right. For both datasets, the baseline methods, PSI+X and APC, have data points and line segments above LP for the plotted data size range. When the Taxi dataset has a size of \( 10^{5.5} \), LP at \( \epsilon = 0.1 \) costs an order of magnitude less than APC, as shown by the leftmost brown point.
(APC) and blue point (LP, \(\varepsilon=0.1\)) in Figure 5.2(left). As the data size increases, the gap between APC and LP gets larger. When data size increases by 16 times (the right most points in the plots), LP at \(\varepsilon = 0.1\) costs over 2 orders of magnitude less than APC. When \(\varepsilon\) increases, the cost of LP shifts downward towards the non-private setting (np). When \(\varepsilon = 1.6\), LP has 3 orders of magnitude lower cost than APC for the given range of data sizes. The line for np is the lower bound for LP, where no dummy records are added to the bins. Similar observations are found in Figure 5.2(right) for the AB dataset, where LP improves APC by up to 2 orders for the plotted data size range.

PSI+X has a much larger cost than both APC and LP, mainly due to the fact that the expansion factor is far larger than the data size. We also observe that the lines that pass through the points of APC for both Taxi and AB datasets have a slope of 2, which corresponds to the quadratic communication and computational cost of APC. LP and PSI+X have slopes of values slightly larger than 1, and thus are linear time. Thus, for sufficiently large data sizes, PSI+X can beat APC. However, we do not expect PSI+X to beat LP due to the large expansion factor. Similar results are observed when the protocol stops before achieving full recall (Figure A.1 in Appendix A.4).

**Optimization steps.** We next study the effectiveness of the optimization steps for LP. We study 5 protocols as shown below:

- ‘basic’: the basic LP Algorithm 3 with no heuristic optimizations;
- ‘basic+SP’: the basic LP with the sort & prune step (SP). SP stops the protocol when the threshold reaches the 10% percentile of the noisy bin counts of \(\tilde{B}(D_A)\) and \(\tilde{B}(D_B)\). Together with the sorting step, bins pairs with insufficient counts can be pruned away, resulting in a recall slightly smaller than the highest possible recall;
• ‘basic+GMC’: the greedy match & clean step (GMC) in Algorithm 5 is applied to the basic LP;
• ‘basic+GMC+S’: in addition to the previous protocol, bins are sorted in order of size. Pruning is omitted so that the highest possible recall is achieved;
• ‘basic+GMC+SP’: the same protocol as ‘basic+GMC+S’, except it prunes the bins with counts in the bottom 10% percentile.

Hence, the default LP can be also denoted by ‘basic+GMC+S’ if recall is 1.0 and ‘basic+GMC+SP’ if recall is less than 1.0.

In Figure 5.3, we report the average cost with the standard deviation across 10 runs of the above mentioned protocols at $\epsilon = 0.1, 0.4, 1.6$ and $\delta = 10^{-5}$ for the Taxi and AB datasets when $T = 1$ and $T = 16$. Several interesting observations arise from this plot.

First, the most significant drop in cost is due to GMC. The protocols with the greedy step have smaller cost than other protocols for all $\epsilon$ and datasets. For the Taxi datasets at $T = 1$ or $T = 16$, ‘basic+GMC’ saves the cost of ‘basic’ by over 50% when $\epsilon = 1.6$. As $\epsilon$ decreases, these relative savings reduce because more dummy records are added and cannot be matched or removed by this greedy step. For the AB datasets, ‘basic+GMC’ reduces the cost of ‘basic’ by up to 16% at $\epsilon = 1.6$ and 11% at $\epsilon = 0.1$.

Next, adding the sorting step to GMC (GMC +S) improves upon GMC when the data sizes are large ($T=16$). For instance, when $\epsilon = 0.1$ and $T = 16$, ‘basic+GMC+S’ can further bring the cost down by approximately $8.0 \times 10^6$ candidate pairs for the AB datasets, and by $2.0 \times 10^9$ for the Taxi datasets.

Third, the cost of ‘basic+GMC+SP’ is reported at a recall reaching above 0.95. The reduction with respect to ‘basic+GMC+S’ is relatively small, but the absolute reduction in cost is significant in some setting. For instance, the number of candidate
Figure 5.4: LP with varying privacy settings plotted over the default Taxi datasets and AB datasets. Each trade-off line between recall and the efficiency (cost(LP)/cost(APC)) corresponds to the default LP at a privacy setting \((\epsilon, \delta)\). Figure 5.4(a) varies \(\epsilon\) and Figure 5.4(b) varies \(\delta\).

Last, for the AB dataset at \(T = 16\), ‘basic+SP’ has a smaller variance in cost than ‘basic’ at \(\epsilon = 0.1\). Similarly, ‘basic+GMC+SP’ has a smaller variance in cost than ‘basic+GMC’. This implies the sort & prune step can help prune away bins, and hence reduce the variance introduced by dummy records.

**Three-way trade-offs.** All the DPRL baseline methods including APC, PSI and PSI+X, have a fixed and strong privacy guarantee where \(\epsilon = 0\) and \(\delta = \text{negl}(\kappa)\). Hence, each baseline has a single point in a plot between recall and efficiency for a
given data size, where APC and PSI+X have a point with full recall and high cost, and PSI has a point with low recall and low cost. Here, we will show that LP allows a trade-off between recall and efficiency for a given privacy guarantee. The efficiency metric used here is the ratio of the cost(LP) to the cost(APC).

Figure 5.4(a) illustrates the case when both Alice and Bob require \((\epsilon, \delta)\)-DPRL protection where \(\epsilon = \{0.1, 0.4, 1.6\}\) and fixed \(\delta = 10^{-5}\). In Figure 5.4(b), we vary the values of \(\delta\) for \(\delta \in \{10^{-9}, 10^{-7}, 10^{-5}\}\) with fixed \(\epsilon = 1.6\). Each data point in the plot corresponds to the average cost(LP)/cost(APC) and average recall of the default LP for a given \((\epsilon, \delta)\) and the default data size with \(T = 1\). The default LP allows the sort & prune step as described in \(S\) 5.4.3 with a list of thresholds that are the 90\%, 80\%, ..., 0\% percentiles of the sorted bin sizes of \(\tilde{B}(D_A)\) and \(\tilde{B}(D_B)\). We report the average recall and cost(LP)/cost(APC) for each percentile. This gives a trade-off line for each \(\epsilon\) and \(\delta\) value.

We observe that all the trade-off lines obtain a high recall at very small values of cost(LP)/cost(APC). Even at \(\epsilon = 0.1\), LP incurs 100 times smaller cost than APC. LP has a slightly larger cost for AB dataset. In Figure 5.4(a), the trade-off lines between recall and efficiency shift rightwards as the privacy parameter \(\epsilon\) gets smaller. In other words, the cost is higher for a stronger privacy guarantee in order to output the same recall. Similar observations are found in Figure 5.4(b). However, the trade-off lines are more sensitive to \(\epsilon\) than \(\delta\). The red lines in Figure 5.4(a) and the red lines in Figure 5.4(b) correspond to the same privacy setting. As \(\delta\) reduces by 10000 times from \(10^{-5}\) to \(10^{-9}\), the trade-off line of LP for the Taxi datasets shifts the ratio of costs by at most 0.001 as shown in Figure 5.4(b) (left) while the trade-off line increases the ratio of costs to 0.07 as \(\epsilon\) reduces from 1.6 to 0.1 (Figure 5.4(a)).

As the Taxi and AB dataset have different data distributions over bins, the shapes of the trade-off lines are different. AB datasets are more skewed and have some bins with large counts. These bins also have many matching pairs, and hence we see a
steep rise for the first part of the trade-off lines for the AB datasets. When the data size increases, if the distribution of matching pairs remains similar, the trade-off lines between the efficiency and recall tends to stay the same. These trade-off lines can be useful when choosing the recall, privacy and efficiency for larger datasets.

**Wall clock times.** We implemented APC and LP in python, and implemented operations on encrypted records using the Paillier homomorphic cryptosystem using the python-paillier library [119]. As all algorithms require a one-time encryption of records we exclude this cost and only measure the cost of operations on the encrypted records. On a 3.1 GHz Intel Core i7 machine with 16 GB RAM, we found that computing the Hamming distance of two encrypted records with dimension $d = 50$ takes an average of $t_s = 77$ ms. That is, for datasets of size $n = 5000$, APC would take over 22 days to complete! Additionally, for the same dataset with $\epsilon = 1.6$, LP would only take 80 hours to achieve a recall of 1. In comparison, the wall clock time of LP ignoring the time spent in comparisons of encrypted records was only 120 seconds. We believe that this order of magnitude difference in time for secure operations and normal operations is true independent of the library or protocol used for secure comparisons. Thus, the computational cost of LP is dominated by the cost of secure comparison. How to improve the unit cost of each secure pairwise comparison is an important research topic, and is orthogonal to our research. Hence, in this evaluation, we focused only on the number of secure comparisons/operations on encrypted values to measure efficiency.

5.6 Related Work

In addition to the prior work [65, 77, 20] that attempted to combine DP and secure computation techniques in order to scale-up the PRL problem, there are other efforts that take similar approaches, but focus on solving different problems. Wagh et al.
formalized the notion of differentially private oblivious RAM (DP ORAM) and their corresponding protocols significantly improved the bandwidth overheads with a relaxed privacy guarantee. This privacy notion considers a client-server model where all data sit on a single server, while DPRL considers two party computation. Moreover, the protocols for DP ORAM only consider the trade-off between privacy and efficiency while DPRL considers an additional trade-off dimension: correctness. Several efforts [92, 99, 6, 100, 94, 53] also integrated DP with SMC in a distributed setting where data is vertically or horizontally partitioned between parties. The difference is that these papers focus on aggregate functions over the partitioned data, such as join size, marginal counts and sum, while PRL requires matching individual record pairs. This matching of individual record pairs does not naturally compose with DP, and hence motivated DPRL, a new privacy model for efficient PRL.

5.7 Summary

In this work, we propose a novel privacy model, called output constrained differential privacy, that shares the strong privacy protection of differential privacy, but allows for the truthful release of the output of a certain function on the data. We showed that this new privacy model can be applied to record linkage to define differential privacy for record linkage (DPRL). Under this framework, we proposed novel protocols for efficient PRL that satisfy three desiderata: correctness, privacy and efficiency. This is an important advance, since none of the prior techniques achieves all three desiderata. Despite this advance, further investigation into the practicality of DPRL protocols is a direction for future research. This includes investigation into their wall clock times in a specific operational environment and over datasets with more complex matching functions. Additional directions for future research include identifying DPRL protocols that further reduce the computational complexity of record linkage, such as applying a data-dependent blocking strategy, extending two-party DPRL to
a multi-party setting, and generalizing the notion of output constrained differential privacy to other applications beyond private record linkage.
This dissertation presents a novel policy-driven approach to design provable privacy guarantees for complex settings. This policy-driven approach results in a useful class of provable privacy definitions, named as Blowfish privacy, (a) generalize differential privacy to handle complex privacy preferences and constraints, (b) unify several variants of differential privacy that are used in practice, and (c) allow the creation of new well founded privacy definitions that allow flexible trade-offs between privacy, accuracy, and performance, based on the application’s requirements. The usefulness of this approach are shown in two use cases of data sharing: (1) analyzing location data which involves complex data types and privacy preferences, and (2) scaling private record linkage which involves secure computations between multiple parties.

6.1 Summary of Contributions

We provide the state-of-the-art definitions for provable privacy guarantees in Chapter 2 and show the changes and the need of customizing privacy guarantees for different applications in Chapter 3. We propose a new class of privacy definitions, called Blowfish privacy, with the goal of seeking better trade-off between privacy
and utility. The key feature of Blowfish is a policy, where users can specify sensitive information that needs to be protected and knowledge about their databases which has been released to potential adversaries. Such a rich set of “tuning knobs” in the policy enable users to improve the utility by customizing sensitive information and to limit attacks from adversaries with auxiliary knowledge. Using typical data analysis tasks like answering range queries on data and k-means clustering, we show how to tune utility using reasonable policies with weaker specifications of privacy. For the latter, we develop strategies that are more accurate than any differentially private mechanism. Moreover, we study how to calibrate noise for Blowfish policies with count constraints when publishing histograms, and the general result we obtain can be applied in several practical scenarios.

In Chapter 4, we described variants of differential privacy and algorithms that satisfy these variants for the tasks of answering queries over a single-time snapshot of location data, continuous queries over location streams and releasing synthetic location trajectory databases. We reason about the privacy semantics underlying the variants of differential privacy under the Pufferfish framework and Blowfish privacy. We also presented the first system, DPT, to release realistic but private trajectory data under user-level DP, the strongest privacy guarantees. Our analysis and empirical study demonstrate that this is an effective way to reveal information, while protecting privacy both in theory and in practice.

In Chapter 5, we proposed a novel privacy model, called output constrained differential privacy, that shares the strong privacy protection of differential privacy, but allows for the truthful release of the output of a certain function on the data. We showed that this new privacy model can be applied to record linkage to define differential privacy for record linkage (DPRL). Under this framework, we proposed novel protocols for efficient PRL that satisfy three desiderata: correctness, privacy and efficiency. This is an important advance, since none of the prior techniques
achieves all three desiderata. Despite this advance, further investigation into the practicality of DPRL protocols is a direction for future research.

6.2 Future Directions

In the course of my research, I realize that organizations that analyze sensitive user data to provide data driven solutions are unable or unwilling to afford privacy-preserving solutions in their products due to the lack of standards and resources. Those privacy-preserving solutions developed by Google and the U.S. Census require a group of privacy experts, and they are not directly transferable to other domains. As the market for diverse smart-devices and systems has expeditiously emerged, the need for privacy-preserving solutions within a wide range of domains and applications have become ever-pressing. However, ad hoc approaches lead to a drastic and inevitable cost to companies and individuals.

To address these challenges, my long-term goal is to design and build a toolbox called Cost-Aware Privacy Engine (CAPE) that (i) can be seamlessly integrated into existing systems and environments and that (ii) can ensure provable and customizable privacy based on the cost requirements (e.g., utility cost or computation cost) of different applications. With CAPE, data curators can deliver their privacy promises with little overhead while minimizing the cost of privacy. Below I outline a selected set of research questions that motivate me and that would provide set of concepts and systems to realize this goal.

Privacy Policy Generator for General Databases. In short term, I will continue my study on the customization of provable privacy guarantees using privacy policies. We have shown [59, 60, 82] that customizing privacy notions using policies achieves better utility or greater efficiency. Although the privacy policies created so far are different across applications, they focused on homogeneous and well-defined sensitive
information, such as the presence or absence of individuals, or a set of given attributes associated to individuals, which might not be true for general databases. Consider the following scenarios: (1) a relational database consisting of multiple tables, such as PATIENTS, DOCTORS, DIAGNOSES, and MEDICATIONS, how will the privacy of the records in PATIENTS or DOCTORS be affected if answering a query is only based on the MEDICATIONS or DIAGNOSES table; (2) a database for an Internet of Things (IoT) application that stores readings of sensors, such as the lighting and thermostats of a room, how will the privacy of a person in the building be affected by querying on these readings even none of the readings are about a person? There is no straightforward answer or solution to these questions. I envision my research in understanding, formalizing, and protecting this non-trivial sensitive information via utilizing privacy policies.

Using policies in privacy definitions may have its own challenges, as learned from the literature of access control where policies are commonly used [12, 129]. For example, how to verify whether the policies correctly capture what intends to protect and how to update policies? Hence, in the long run, I would like to build privacy policy generators for CAPE to automatically output privacy policies (1) that can match the given schema and constraints in a relational database using a rule-based approach, and (2) that can model the underlying sensitive subjects in the given data using a learning-based approach. Moreover, we need to characterize necessary and sufficient conditions for a privacy policy or a class of privacy policies to be useful and competitive, such as composability. Most composition theorems have been considered for the same type of privacy or security notions, but the composition between different privacy and security notions remains a challenge. This can happen in scenarios like integrating CAPE’s privacy policy with an existing security system or integrating policies across multiple CAPE systems. Learning from our prior work [60], composing differential privacy and secure multi-party computation is not so trivial. Our research
goal is to understand and encode these conditions into the policy generator so that the output privacy policies are guaranteed useful and strong.

**Trade-off Exploration Platform with Mechanism Generator.** In the process of designing privacy and building privacy-preserving systems, I realize that the mathematical formulation of provable privacy is difficult to grasp for general data curators. It can be even harder for them to decide about the privacy parameters of a privacy notion, such as the value of the privacy budget $\epsilon$. Our work [61] showed that visual exploration enhances people’s understanding of the trade-off between the privacy guarantees and the potential cost of the given guarantees. This inspires me to build a trade-off exploration platform in CAPE such that curators can make an informed decision on privacy parameters in the policies. Whenever a decision is made via this platform, a mechanism can be automatically generated and applied to the data. Building such a platform efficiently needs both theoretical understandings on the limit of the cost of privacy and algorithmic research on mechanisms that achieve the lowest possible cost given the same privacy guarantee.

To conclude this thesis, I would like to quote Mark Zuckerberg’s letter to his daughter, “for your generation to live in a better world, there is so much more our generation can do”. There will still be many challenges along the journey of defining and realizing privacy. Tackling these challenges requires collaborative efforts from different disciplines and together, we might make a small but positive change by giving our next generation an informed privacy choice in the future world of data-driven technologies.
Appendix A

Supporting Materials for PRL

A.1 Related Privacy Definitions

A.1.1 Simulation-Based S2PC

The standard simulation-based definition for SMC is defined below.

**Definition 87 (SIM-S2PC).** [51] For a functionality $f$, a 2-party protocol $\Pi$ which computes $f$ provides simulation-based secure 2-party computation (SIM-S2PC) if for all data sets $D_A, D_B$ of polynomial sizes (in $\kappa$), there exist probabilistic polynomial-time algorithms (simulators), denoted by $S_A$ and $S_B$ such that the distribution of $S_A$ (resp., $S_B$) is computationally indistinguishable from $\text{VIEW}_A^\Pi$ (resp., $\text{VIEW}_B^\Pi$), i.e. for any probabilistic polynomial-time (in $\kappa$) adversary $T$,

\begin{align*}
\Pr[T(S_A(D_A, f_A(D_A, D_B), f(D_A, D_B))) = 1] & \leq \Pr[T(\text{VIEW}_A^\Pi(D_A, D_B), \omega_A^\Pi(D_A, D_B)) = 1] + \text{negl}(\kappa) \\
\Pr[T(S_B(D_A, f_B(D_A, D_B), f(D_A, D_B))) = 1] & \leq \Pr[T(\text{VIEW}_B^\Pi(D_A, D_B), \omega_B^\Pi(D_A, D_B)) = 1] + \text{negl}(\kappa).
\end{align*}

If $f$ is deterministic, Alice gains no additional knowledge other than its respective
input \((D_A)\) and output \((f_A(D_A, D_B))\); similarly for Bob. When randomized functionalities are concerned, augmenting the view of the semi-honest party by the output of the other party is essential. In this case, for any protocol \(\Pi\) that computes the randomized functionality \(f\), it does not necessarily hold that \(\omega^\Pi(D_A, D_B) = f(D_A, D_B)\). Rather, these two random variables must be identically distributed. In order to study the possibility of composing DP and S2PC, we choose the indistinguishability-based definition for PRL, which is implied from SIM-S2PC.

**Theorem 88.** SIM-S2PC implies IND-S2PC.

**Proof.** Given the protocol \(\Pi\), for all possible inputs \((D_A, D_B)\), there exists a global simulator \(S_A\) such that the distribution of \(S_A\) is computationally indistinguishable from the view of Alice. As \(f(D_A, D_B) = f(D_A, D_B')\), \(S_A\) takes the same input and hence will have the same distribution for \(D_B\) and \(D_B'\). Hence, the views over \((D_A, D_B)\) or \((D_A, D_B')\) are indistinguishable. \(\Box\)

Any algorithm that satisfies SIM-CDP also satisfies IND-CDP [91], but it is unknown if the converse holds.

### A.1.2 Computationally Differential Privacy

Mironov et al. [91] defines a privacy notion, known as indistinguishable computationally differential privacy (IND-CDP-2PC). This notion is a direct extension of DP in two party setting where both parties are computationally bounded. Formally, we have

**Definition 89 (IND-CDP-2PC).** A 2-party protocol \(\Pi\) for computing function \(f\) satisfies \((\epsilon_A(\kappa), \epsilon_B(\kappa))\)-indistinguishable computationally differential privacy (IND-CDP-2PC) if \(\text{VIEW}_A^\Pi(D_A, \cdot)\) satisfies \(\epsilon_B(\kappa)\)-IND-CDP, i.e. for any probabilistic polynomial-time \((\kappa)\) adversary \(T\), for any neighboring databases \((D_B, D_B')\) dif-
Pr[\text{T}(\text{VIEW}_A^H(D_A, D_B)) = 1] 
\leq \epsilon_{\text{B}} \Pr[\text{T}(\text{VIEW}_A^H(D_A, D_B')) = 1] + \text{negl}(\kappa). \quad (A.3)

The same holds for Bob's view for any neighbors \((D_A, D_A')\) and \(\epsilon_A\).

A.1.3 Local Differential Privacy

The local model is usually considered in the model where individuals do not trust the curator with their data. The local version of differential privacy is defined as follows.

**Definition 90** (\(\epsilon\)-Local Differential Privacy). [42] A randomized mechanism \(M : \mathcal{T} \rightarrow \mathcal{O}\) satisfies \(\epsilon\)-local differential privacy if

\[
\Pr[M(r) = O] \leq \epsilon \Pr[M(r') = O] \quad (A.4)
\]

for any set \(O \subseteq \mathcal{O}\), and any records \(r, r' \in \mathcal{T}\) and \(\epsilon > 0\).

A.2 Theorems & Proofs

A.2.1 Privacy Leakage in Prior Work

**Theorem 65** (Limitations of PRL with Blocking)

Given \((d_1, d_2, p_1, p_2)\)-sensitive \(H = \{h_0, \ldots, h_{|H|-1}\}\), we use \(H(\cdot)\) for a record to denote the list of hashing values \([h_0(\cdot), \ldots, h_{|H|-1}(\cdot)]\). An LSH-based blocking considers a set of bins where records associated with the same value for \(H(\cdot)\) are hashed to the same bin. A popular blocking strategy is to compare all the corresponding bins, and results in a set of candidate matches \(\{(a, b) | h(a) = h(b) \forall h \in H, a \in D_A, b \in D_B\}\), i.e. \(\{(a, b) | H(a) = H(b), \forall a \in D_A, b \in D_B\}\). We can show that any LSH based blocking cannot satisfy IND-S2PC as stated in Theorem 65. Here is the proof.
Proof. Take a pair of databases \((D_B, D'_B)\) where \(f_m(D_A, D_B) = f_m(D_A, D'_B)\). Let the symmetric difference between \(D_B\) and \(D'_B\) be \((b, b')\) and \(\text{dist}(b, b') > d_2\). Hence, with high probability \(1 - p_2^{\lceil H \rceil}\), we have \(H(b) \neq H(b')\), and \(|\mathcal{B}_{H(b)}(D_B)| - |\mathcal{B}_{H(b)}(D'_B)| = 1\) and \(|\mathcal{B}_{H(b')}(D'_B)| - |\mathcal{B}_{H(b')}(D_B)| = 1\) as the rest of records are the same in \(D_B\) and \(D'_B\). Alice as a semi-honest adversary can set her dataset such that \(|\mathcal{B}_{H(b)}(D_A)| \neq |\mathcal{B}_{H(b')}(D_A)|\). Then, with high probability, the following inequality holds

\[
\text{cost}_{BS}(D_A, D_B) - \text{cost}_{BS}(D_A, D'_B) = \left(|\mathcal{B}_{H(b)}(D_B)| - |\mathcal{B}_{H(b)}(D'_B)||\mathcal{B}_{H(b)}(D_A)| + |\mathcal{B}_{H(b')}(D_B)| - |\mathcal{B}_{H(b')}(D'_B)||\mathcal{B}_{H(b')}(D_A)|\right) = |\mathcal{B}_{H(b)}(D_A)| - |\mathcal{B}_{H(b')}(D_A)| \neq 0.
\]

Hence, Alice can distinguish \(D_B\) and \(D'_B\) by \(\text{cost}_{BS}(D_A, D_B) \neq \text{cost}_{BS}(D_A, D'_B)\) with high probability \(1 - p_2^{\lceil H \rceil}\). Other blocking strategies can be similarly shown. Therefore, this LSH-based PRL does not satisfy IND-S2PC.

\[\square\]

**Theorem 82** *(Limitations of PRL with DP Blocking of Prior Approaches/LP-2)*

Several prior works \([65, 77, 20]\) combine PRL techniques with differentially private blocking (PRL + \(\mathcal{B}_{DP}\)). These approaches can be summarized in three steps: (1) DP blocking, (2) records addition and suppression, (3) secure pair-wise comparisons based on blocking strategy \(\mathcal{B}^S\). In the first step, Alice and Bob process their data independently. Each party generates an \(\epsilon\)-differentially private partition of the data, where each partition is associated with a noisy count \(\tilde{o}_i = |\mathcal{B}_i(D_B)| + \eta_i\), where \(\Pr[\eta_i = x] = pe^{-\epsilon/\Delta\mathcal{B}|x|}\), for \(x \in \mathbb{Z}\) and \(p = e^{\epsilon/\Delta\mathcal{B}} - 1\) is the normalized factor \(^1\). \(\Delta\mathcal{B}\) is the sensitivity of the blocking strategy (Def 77).

Next, for each partition, if the noise \(\eta_i\) is positive, dummy records are added; otherwise, records in that partition are suppressed randomly to obtain the published

\(^1\) We use discrete version of Laplace distribution to avoid rounding.
count. This results in new bins, denoted by \( \tilde{B}_i(D_A) \) and \( \tilde{B}_j(D_B) \). In the last step, Alice and Bob jointly compare record pairs \((a, b)\), where \( a \in \tilde{B}_i(D_A) \) and \( b \in \tilde{B}_j(D_B) \) for all \((i, j)\) in \( \mathcal{B}^S \) as in APC. They only exchange the true records \((a, b)\) if they match.

[20] considers a third party for identifying candidate pairs for Alice and Bob, so that Alice and Bob has no direct access to the noisy bins of the opposite party, but has access to the number of secure comparisons. However, this hybrid protocol above does not satisfy \((\varepsilon, \delta)\)-DPRL as stated in Theorem 82. The failure to satisfy DPRL is mainly caused by the record suppression step for the negative noise drawn from a zero-mean Laplace distribution, as shown in the following proof.

**Proof.** Without loss of generality, we consider Alice as the adversary. For any arbitrary \( \varepsilon \) and small \( \delta < \frac{\varepsilon^2}{2e^\varepsilon} \), there exists a counter example fails \((\varepsilon, \delta)\)-DPRL. For simplicity, we illustrate how to construct counterexamples using a blocking strategy \( \mathcal{B} \) with sensitivity \( \Delta B = 2 \), where Alice and Bob use the same hashing and each record is hashed to at most 1 bin. For other blocking strategies, counterexamples can be similarly constructed.

Fix a \( D_A \), consider \( D_B \) such that \( \mathcal{B}_0(D_B) = \{b_*\} \) and \( \mathcal{B}_1(D_B) = \{b_1, \ldots, b_{n_1}\} \), where \( 1 \leq n_1 < \frac{\varepsilon^2}{e^\varepsilon - 1} \). (Note that \( \frac{\varepsilon^2}{e^\varepsilon - 1} > 2 \) because \( \delta < \frac{\varepsilon^2}{2e^\varepsilon} \).) In addition, all records in \( \mathcal{B}_1(D_B) \) can find some matching ones from \( D_A \), but \( b_* \) does not match any record in \( D_A \). A neighboring database \( D'_B \) can be constructed from \( D_B \) by removing \( b_* \) from \( \mathcal{B}_0 \), and adding another \( b'_* \) that can be hashed to \( \mathcal{B}_1 \). It is easy to see that \((D_B, D'_B) \in \mathcal{N}(f_{\text{sm}}(D_A, \cdot)) \).

Without a third party [65, 77], Alice and Bob has access to the number of secure comparisons and the noisy bin counts (in addition to the input data sizes and the matching output). Consider a set of views of Alice \( \text{VIEW}^* \) with output that contains all matching pairs from \( \mathcal{B}_1(D_A) \bowtie_m \mathcal{B}_1(D_B) \) and noisy counts for bin \( \mathcal{B}_0 \) and \( \mathcal{B}_1 \) for Bob being 0 and \( n_1 \) respectively. Let \( \tilde{\mathcal{B}}(D_A) \) be the noisy bins that Alice uses for
the final secure pairwise comparisons. The probabilities to generate these views from $D_B$ and $D'_B$ are respectively:

$$\Pr[\text{VIEW}^*|\tilde{B}(D_A), D_B] = \Pr[\eta_0 = 0] \Pr[\eta_1 = 0] = p^2, \quad (A.6)$$

$$\Pr[\text{VIEW}^*|\tilde{B}(D_A), D'_B] = \Pr[\eta_0 = 1] \Pr[\eta_1 = -1 \& \text{ suppress } b'_s]$$

$$= p^2/(\epsilon^2(n_1 + 1)) > \delta \quad (A.7)$$

The inequality above is due to $n_1 < \frac{p^2}{\epsilon^2 \delta} - 1$. Hence, we have

$$\Pr[\text{VIEW}^*|\tilde{B}(D_A), D_B] = (\epsilon^2 + n_1 \epsilon^2) \Pr[\text{VIEW}^*|\tilde{B}(D_A), D'_B]$$

$$> \epsilon^2 \Pr[\text{VIEW}^*|\tilde{B}(D_A), D'_B] + \delta. \quad (A.8)$$

Hence, $(\epsilon, \delta)$-DPRL is violated.

With a third party[20], Alice and Bob has access to the final output, and the total number of secure pairwise comparisons, but not the noisy bin counts. We can construct examples where knowing the number of secure comparisons leaks the noisy bin counts. After which the previous arguments (for the case with no third party) can show that this protocol does not ensure DPRL for all epsilon and delta. For instance, consider Alice has only 1 record in $\tilde{B}_0(D_A)$, and more than 1 records in other bins, if the output $O = D_A \bowtie_m D_B$, and the total number of secure pairwise comparisons is $|O| + 1$. This secure pairwise matching that returns false can only happen between a record of Bob from $\tilde{B}_0(D_B)$ with the record from $\tilde{B}_0(D_A)$. Hence, Alice can infer the noisy counts of $\tilde{B}(D_B)$. Then the argument for the case with no third party can be used.

In addition, by Theorem 74 and Theorem 75, DPRL is weaker than IND-S2PC and IND-CDP-2PC, we have the following result.

**Corollary 91.** LP-2 satisfies neither IND-CDP-2PC nor IND-S2PC.
A.2.2 Properties of Output Constrained DP

Theorem 70 (Sequential Composition)

Proof. Consider Alice as a probabilistic polynomial-time (in $\kappa$) adversary $T$, with input $D_A$. $(D_B, D_B')$ are neighbors w.r.t. $f(D_A, \cdot)$. We have the probabilities of distinguishing $D_B$ and $D_B'$ bounded by

$$\Pr[T(\text{VIEW}^\Pi_2^\Pi_1(D_A, D_B)) = 1]$$

$$\leq \int x \Pr[T(\text{VIEW}^\Pi_2^\Pi_1(D_A, D_B, x)) = 1] \cdot \Pr[x = \text{VIEW}^\Pi_1(D_A, D_B)]dx$$

$$\leq \int (e^{\varepsilon_2} \Pr[T(\text{VIEW}^\Pi_2^\Pi_1(D_A, D_B', x)) = 1] + \delta_2) \cdot \Pr[x = \text{VIEW}^\Pi_1(D_A, D_B)]dx$$

$$\leq \int (e^{\varepsilon_2} \Pr[T(\text{VIEW}^\Pi_2^\Pi_1(D_A, D_B', x)) = 1]) \cdot (e^{\varepsilon_1} \Pr[x = \text{VIEW}^\Pi_1(D_A, D_B')] + \delta_1) dx + \delta_2$$

$$\leq e^{\varepsilon_1 + \varepsilon_2} \Pr[T(\text{VIEW}^\Pi_2^\Pi_1(D_A, D_B')) = 1] + \delta_1 + \delta_2$$

$\square$

Theorem 71 (Post-processing)

Proof. Since $g$ is efficient and in composition with $T$ can be used as adversary itself. If $g(\omega^\Pi(D_A, D_B))$ does not satisfy $(\epsilon, \delta)$-IND-DPRL, then $\Pi$ does not satisfy $(\epsilon, \delta)$-IND-DPRL. $\square$

Theorem 75 (Relation with IND-CDP-2PC)

We show that DPRL is weaker than IND-CDP-2PC.

Proof. $\epsilon/2$-IND-CDP-2PC is equivalent to $\epsilon$-IND-DP-2PC, where neighboring databases have a symmetric difference of 2. The set of neighboring databases for DPRL is a
A.2.3 Properties for DPRL Protocols

Theorem 80 (Privacy of Laplace Protocol)

Lemma 92. With probability $1 - \delta$, the probability for Alice having the same view from neighboring databases $(D_B, D'_B) \in \mathcal{N}(f_m(D_A, \cdot))$ is bounded by $e^\epsilon$.

Proof. Given $(D_B, D'_B) \in \mathcal{N}(f_m(D_A, \cdot))$ and $B$, the maximum difference in the bin counts of $D_B$ and $D'_B$ is $\Delta B$. Let $B_\Delta$ be the set of bins that $D_B$ and $D'_B$ have different counts, and $\sum_{i \in B_\Delta} |B_i(D_B) - B_i(D'_B)| \leq \Delta B$. If all the noise for these bins are non-negative, then the probability to output the same noisy counts $(c_0, \ldots, c_{k-1})$ from $D_B$ and $D'_B$ is bounded by

$$
\ln\left(\frac{\Pr[(c_0, \ldots, c_{k-1}) \mid D_B]}{\Pr[(c_0, \ldots, c_{k-1}) \mid D'_B]}\right) \\
= \ln\left(\frac{\prod_{i=0}^{k-1} \Pr[\eta_i = c_i - |B_i(D_B)|]}{\prod_{i=0}^{k-1} \Pr[\eta_i = c_i - |B_i(D'_B)|]}\right) \\
= \sum_{i \in B_\Delta} \ln(\Pr[\eta_i = c_i - |B_i(D_B)|]) - \ln(\Pr[\eta_i = c_i - |B_i(D'_B)|]) \\
\leq \frac{\epsilon}{\Delta B} \cdot \left(\sum_{i \in B_\Delta} |B_i(D_B) - B_i(D'_B)|\right) \leq \epsilon
$$

The probability to draw a negative noise $\eta$ from $\text{Lap}(\epsilon, \delta, \Delta B)$ is

$$
\Pr[\eta < 0] = \sum_{i=-\infty}^{-1} p \cdot e^{-(\epsilon/\Delta B)(x-\eta)} = \frac{e^{-(\eta_0 \epsilon/\Delta B)}}{e^{\epsilon/\Delta B} + 1} \quad (A.9)
$$

Given $\eta_0 = -\frac{\Delta B \ln((\epsilon/\Delta B + 1)(1-(1-\delta)^{1/\Delta B}))}{\epsilon}$, we have $\Pr[\eta < 0] = 1 - (1 - \delta)^{1/\Delta B}$. For each neighboring pair, at most $\Delta B$ bins differ and fail to have $\Pr[\eta \geq 0]$. Hence, the overall failing probability is $1 - (1 - \Pr[\eta < 0])^{\Delta B} = \delta$. With $1 - \delta$, the probability of having the same view from PRL neighboring databases is bounded by $e^\epsilon$. 

\[\square\]
**Theorem 86 (Correctness & Efficiency of GMC)**

**Proof.** First, we will show that the efficiency of LP with the greedy match & clean step (GMC) is better than LP alone. The first part of the protocol that adds dummy records is the same. The second part of the protocol without GMC compares all the candidate matches using the secure matching protocol $SMC(a, b)$. On the other hand, with GMC, if a record pair $(a, b)$ is compared securely, then $(a, b)$ must be one of the candidate matches. Hence, the number of the secure pairwise comparisons with GMC will be no more than the protocol without GMC.

Next, we will show the correctness of LP with GMC. Let $O_{LP}, O_{LP+GMC}$ be the final output of LP protocol without GMC and with GMC. We would like to show that if $(a, b) \in O_{LP}$, then $(a, b) \in O_{LP+GMC}$. Suppose this is not true, then there exists a matching pair $(a, b) \in O_{LP}$, but $(a, b) \notin O_{LP+GMC}$. If so, then one of the records in $(a, b)$ must be removed from the bins before its turn of secure pairwise comparison $SMC(a, b)$. Without loss of generality, let’s say $a$ is cleaned from Alice’s bins before $SMC(a, b)$. The condition to remove $a$ is that $a$ has already been in the current output. Hence, Bob is able to compare $a$ with all his records in plain text and identify this matching pair $(a, b)$. This leads to a contradiction. Hence, $O_{LP} \subseteq O_{LP+GMC}$. Moreover, if a matching pair $(a, b)$ is not a candidate match based on the blocking strategy $B^5$, and if $a$ has been already found matching with another record of Bob, then GMC can add $(a, b)$ into $O_{LP+GMC}$. Hence, it is possible that LP with GMC gains even more matching pairs than LP alone. \(\square\)

### A.3 Additional Protocols

#### A.3.1 Example for Secure Pairwise Match

Here we give an example for the function $SMC(a, b)$ that outputs $(a, b)$ if they match; null otherwise. The matching rule is that Euclidean distance of $a$ and $b$ is less than 175.
\( \theta \). First, Party Alice creates a homomorphic public/private key pair \((pk, pr)\), and sends the public key \(pk\) to party Bob. Let \(E_{pk}(\cdot)\) denote the encryption function with public key \(pk\) and \(D_{pr}(\cdot)\) the decryption function with private key \(pr\). Paillier’s cryptosystem supports the following operations on the encrypted plaintexts \(m_1\) and \(m_2\) without the knowledge of the private key:

- **Addition:** \(E_{pk}(m_1 + m_2) = E_{pk}(m_1) + h E_{pk}(m_2)\);

- **Multiplication with constant** \(c\): \(E_{pk}(cm_1) = c \times_h E_{m_1}\)

These two operations allow secure computation of Euclidean distances, i.e. \(\text{dist}(a, b) = \sum_i (a[i] - b[i])^2 = \sum_i (a[i]^2 - 2a[i]b[i] + (b[i])^2)\), and also hamming distances for bit vectors, i.e. \(\text{dist}(a, b) = \sum_i |a[i] - b[i]| = \sum_i a[i] + b[i] - 2a[i]b[i]\).

As summarized in Algorithm 6, given bit vectors \(a\) and \(b\), Alice will send to Bob the encrypted values \((a_{id}, \{E_{pk}(a[i])\})\) where \(a_{id}\) is a randomly generated record identifier for record \(a\). Next, party Bob computes for each of its records \(b_{id}\) the value \(E_{pk}(a[i]) + h E_{pk}(a[i]) \times_h (-2b[i]) + h E_{pk}(b[i])\) which is equal to \(E_{pk}(|a[i] - b[i]|)\) for all \(i\), and computes the encrypted \(E_{pk}(\sum_i |a[i] - b[i]|)\). A random number \(r\) is generated and added to the encrypted distance, such that the true distance is hidden from Alice if \((a, b)\) is not a matching pair. Party Bob creates the message \((b_{id}, E_{pk}(\sum_i |a[i] - b[i]| + r)\) for each record pair comparison. Alice can then decrypt the message with her private key and obtain the relative distance \(d = \sum_i |a[i] - b[i]| + r\). Since Bob knows \(\theta + r\), a secure comparison protocol, such as Yao’s garbled circuit [130], can be used to evaluate if \(d \leq \theta + r\). If this algorithm outputs “True”, Alice and Bob will exchange their true record values.
**Algorithm 6: Secure Match a and b**

1. function SecureMatch \((a, b, \theta)_{pk,pr}\);
2. \textbf{Input :} \(a, b \in \{0, 1\}^d\), hamming distance threshold \(\theta\), public/private key pair \((pk, pr)\)
3. \textbf{Output :} \((a, b)\) or \(\emptyset\)
4. Alice: randomly generates an id \(a_{id}\) and sends to Bob;
5. Bob: randomly generates an id \(b_{id}\) and an integer \(r\);
6. Bob: initiates \(s\) with \(E_{pk}(r)\);
7. for \(i \in [0, \ldots, d - 1]\) do
   8. Alice: sends to Bob \(E_{pk}(a[i])\);
   9. Bob: updates \(s = s + h \cdot E_{pk}(a[i]) + h \cdot (E_{pk}(a[i]) \times h (-2b[i])) + h \cdot E_{pk}(b[i])\);
10. end
11. Bob: sends \((b_{id}, s)\) to Alice;
12. Alice: decrypts \(s = D_{pr}(s)\);
13. \textbf{if} \(s \leq \theta + r\) \textbf{(secure integer comparison)} \textbf{then}
14. return \((a, b)\);
15. \textbf{else}
16. return \(\emptyset\);
17. end

A.3.2 Local DP Protocol

**Theorem 76**

Let \(B\) be a blocking that randomly hashes records into a pre-specified set of \(k\) bin, such that \(\Pr_{B}[^{B}(b)=i] \leq e^{\epsilon}\). Such a blocking \(B\) satisfies \(\epsilon\)-local DP (Appendix A.1.3). Protocols that combine a local differentially private blocking with IND-S2PC protocols for record linkage to achieve \((\epsilon, \text{negl}(\kappa))-\text{DPRL}\).

**Proof.** (sketch) We prove privacy for Bob (the proof for Alice is analogous). In this protocol, Alice with input Data \(D_A\) has a view consisting of (1) the number of candidate matching pairs arising in each \((i, j) \in B^2\), (2) the output for each candidate matching pair. Consider a neighboring pair \((D_B, D'_B) \in N(f_{m_a}(D_A, \cdot))\) for a given \(D_A\). By Theorem 72, \(D_B\) and \(D'_B\) differ in only one non-matching record with respect to \(D_A\), i.e. \(D'_B = D_B - b_a + b'_a\) and \(b_a \neq b'_a\), where \(m(b_a, a) = 0\) and \(m(b'_a, a) = 0\) for all \(a \in D_A\). Given both \(b_a, b'_a\) can be hashed into the same bin with probability

\(\Pr_{B}[^{B}(b_a)=i] \leq e^{\epsilon}\). Such a blocking \(B\) satisfies \(\epsilon\)-local DP (Appendix A.1.3). Protocols that combine a local differentially private blocking with IND-S2PC protocols for record linkage to achieve \((\epsilon, \text{negl}(\kappa))-\text{DPRL}\).
ratio bounded by $e^\epsilon$, the probabilities of generating the same number of candidate matching pairs from $D_B$ and $D'_B$ are also bounded by the same ratio. The encrypted records only differ in $b_s$ and $b'_s$, and both of them lead to the same output for each candidate matching, because they do not match any records in $D_A$.

In this work, we use randomized response (RR) [42] as an example to achieve DPRL. Other local DP algorithms, such as Johnson-Lindenstrauss (JL) transform [14] can be similarly applied.

**RR based Blocking**

Given a fixed hash function $h : T \rightarrow [0, k - 1]$, records in $D_B$ are hashed into $k$ bins, $B_0, ..., B_{k-1}$ respectively. Let us define a RR based on this fixed hashing function with privacy budget $\epsilon_B$ for Bob. Each record $b \in D_B$ is randomly hashed into $B_{h(b)}$ with probability $p_B = \frac{e^{\epsilon_B}}{k-1+e^{\epsilon_B}}$ and the other $(k-1)$ bins with probability $q_B = \frac{1}{k-1+e^{\epsilon_B}}$. We denote the resulted bins by $\tilde{B}(D_B)$, and the resulted bin for each record $b \in D_B$ by $\tilde{B}(b)$. Similarly, using the same fixed hashing function and randomized response, Alice’s records $D_A$ are randomly hashed into the $k$ bins, $\tilde{B}(D_A)$ with corresponding $p_A, q_A$ based on $\epsilon_A$. This randomized response with probabilities $(p_A, q_A)$ and $(p_B, q_B)$ ensures $\epsilon_A$-local DP and $\epsilon_B$-local DP respectively.

Consider a basic blocking strategy $\mathcal{B}^S = \{(i, i)|i \in [0, k)\}$, all corresponding bins are compared. If the hash function $h$ is a LSH, then matching records are likely fall into the same bin as Alice and Bob use the same hash function. The probability that such records $(a, b)$ appear in the same bin after randomization $\Pr[\tilde{B}(a) = \tilde{B}(b)|h(a) = h(b)]$ is $p_Ap_B + (k-1)q_Aq_B$. This probability increases with the privacy budget $\epsilon_A, \epsilon_B$, and hence recall will improves.

A further trade-off between correctness, privacy and efficiency is allowed by considering a general blocking strategy $\mathcal{B}^S = \{(i, (i+j)\%k)|i \in [0, k), j \in [0, k')]\}$, each
record in $\tilde{B}_i(D_A)$ is securely compared with $k'$ neighboring bins of Bob. The basic blocking strategy corresponds to the case where $k' = 1$. As $k'$ increases, more candidate matching pairs are securely compared, resulting potentially higher recall and more communication and computation cost. When $k' = k$, the resulted protocol is equivalent to the all pairwise comparisons baseline.

**Optimal RR Probability**

If the window size $k'$ is given in advance as a parameter for the efficiency, the expected recall of randomized response can be further optimized. Let $p_i^B$ be the probability for a record $b \in D_B$ to be randomly hashed into $B_{(h(b)+i)\%k}$. To ensure $\epsilon_B$-DPRL, the probabilities to hash any pairs of bins should be bounded by $e^{\epsilon_B}$. The hashing probability for Alice records, $p_i^A$, is similarly defined and constrained. As each record of Alice’s can fall into any bin, and each bin of Alice’s is compared with $k'$ neighboring bins of Bob’s, the probability that a matching pair $(a, b)$ is compared after randomization is

$$P(a, b) = \sum_{i=0}^{k-1} \sum_{j=0}^{k'-1} p_i^A p_{i+j \%k}^B. \quad (A.10)$$

The expected recall can be improved by maximizing $p_{(a,b)}$ (Eqn. (A.10)) with the constraints on

(a) Ratio: $\frac{p_i^B}{p_i^A} \leq e^{\epsilon_B}, \forall i \neq i'; \frac{p_i^A}{p_i^B} \leq e^{\epsilon_A}, \forall i \neq i';$

(b) Sum: $\sum_{i=0}^{k-1} p_i^A = 1; \sum_{i=0}^{k-1} p_i^B = 1.$

The solution in Section A.3.2 where we assign a high probability to a single bin, and a low probability to the rest of the bins is a valid solution to the maximization problem, but it is not always the optimal solution. In general, this optimization can be solved by existing tools for quadratic objectives with linear constraints, such as quadratic programming.
Here, we present an explicit solution for the special case when $\epsilon_B = \epsilon \geq 0$ and $\epsilon_A \to \infty$. This case corresponds to the situation where Alice’s data is public and only Bob’s data requires $\epsilon$-DPRL protection. Then the expected recall can be maximized by solving the following linear optimization:

$$\max_{p_0^B, \ldots, p_{k-1}^B} \sum_{i=0}^{k'-1} p_i^B \quad \text{s.t.} \quad \sum_{i=0}^{k-1} p_i^B = 1, \text{ and } \frac{p_i^B}{p_{i'}^B} \leq e^\epsilon \quad \forall i \neq i'$$

The expected recall is maximized with value $p_{(a,b)} = \frac{k'e^\epsilon}{k-k' + ke^\epsilon}$ at

$$p_i^B = \begin{cases} \frac{e^\epsilon}{k-k' + ke^\epsilon} & \text{for } i = 0, \ldots, k' - 1 \\ \frac{1}{k-k' + ke^\epsilon} & \text{for } i = k', \ldots, k - 1 \end{cases}$$

If Bob’s records are uniformly distributed over the bins, then the compression ratio w.r.t all pairwise comparisons (APC) is $\rho = \frac{k'}{k}$. The maximized expected recall can be written as

$$p_{(a,b)} = \frac{\rho e^\epsilon}{1 - \rho + \rho e^\epsilon}.$$  \hspace{1cm} (A.11)

This equation explicitly form the relationship between correctness ($p_{(a,b)}$), privacy ($\epsilon$) and efficiency ($\rho$) of this protocol.

**Theorem 93.** *The basic RR mechanism achieves a constant factor speedup in efficiency given $\epsilon_B = \epsilon \geq 0$, $\epsilon_A \to \infty$, $\delta_A = \delta_B = \text{negl}(\kappa)$ and recall $r$.***

**Proof.** Given a recall $r = p_{(a,b)}$, we have $\rho = 1 - \frac{e^\epsilon(1-r)}{r + e^\epsilon(1-r)}$ based on Eqn. (A.11). The improvement in efficiency $\rho$ is a constant factor in terms of $r$ and $\epsilon$, independent of $n$. \qed

Next, we compute the optimal amongst a restricted class of strategies for the more general case where $\epsilon_A = \epsilon_B = \epsilon$, though the explicit form for this case is unknown yet. The strategies we consider are those where Alice and Bob (a) use
symmetric probabilities to assign a bin to each record, and (b) they both assign a high probability $p_{\top}$ to place a record from bin $i$ to bins $i$ through $(i + x - 1)\%k$ (for some $1 \leq x \leq k'$), and a low probability $p_{\bot}$ to assign a record from bin $i$ to the rest of the bins. Note that, when only one of Alice or Bob is randomizing their records, $x = k'$ results in the RR probabilities that optimize the expected recall.

In order to satisfy the constraints in the above maximization problem, we need

$$p_{\top} = \frac{e^x}{k-x+x\cdot e^x} \quad \text{and} \quad p_{\bot} = \frac{1}{k-x+x\cdot e^x}.$$  

The expected recall can be derived as follows. Without loss of generality consider a matching pair $(a, b)$ that fall into bin 0. There are 3 ways $(a, b)$ are matched after randomization:

(i): Both Alice and Bob randomize their records to a $\top$ bin (i.e., some bin $0 \leq j \leq x - 1$). Since $x \leq k'$, these records are definitely compared. This occurs with probability $\frac{x(x+1)}{2}p_{\top}^2$

(ii): Only one of Alice and Bob randomize their records to a $\top$ bin (i.e., some bin $0 \leq j \leq x - 1$). There $2k'x - x(x + 1)$ ways in which exactly one of $a$ or $b$ is randomized to a $\top$ bin, but still end up getting compared by the algorithm. This occurs with probability $(2k'x - x(x + 1))p_{\top} \cdot p_{\bot}$

(ii): Both Alice and Bob randomize their records to a $\bot$ bin. This occurs with probability $(kk' - (2k'x - \frac{x(x+1)}{2}))p_{\bot}^2$ In total, the probability that $(a, b)$ are compared is expressed in terms of $x$ as

$$p_{(a,b)}(x) = \frac{x(x+1)}{2}p_{\top}^2 + (2k'x - x(x + 1))p_{\top} \cdot p_{\bot} + (kk' - (2k'x - \frac{x(x+1)}{2}))p_{\bot}^2$$

The derivative of $p_{(a,b)}(x)$ w.r.t $x$ is

$$p'_{(a,b)}(x) = C_1 \cdot [(e^x - 1)C_2x + k(2k' + e^x - 1)],$$

181
where $C_1 = \frac{e^\epsilon - 1}{2(k - x + xe^\epsilon)^2}$ and $C_2 = (e^\epsilon - 3 + 2k - 4k')$. When $C_2 > 0$, the derivative is always positive, the expected recall is maximized when $x = k'$, as $0 < x \leq k'$. We will leave the complete analysis to the future work.

A.4 Additional Plot

Figure A.1 shows the log(base 10) value of the average cost with respect to the log value of data size for PSI+X, APC, and LP with $\epsilon \in \{0.1, 0.4, 1.6\}$ and $\delta = 10^{-5}$ and the non-private setting (np) when they achieve a recall $> 0.95$. Similar to Figure 5.2, LP gives lower costs than the baselines, and scales near linearly.
Bibliography


Biography

Xi He was born in Jingzhou, a city located in the south west of China. Xi received a double degree in Applied Mathematics and Computer Science from the University of Singapore in 2012. She joined Duke University as a Ph.D. student in the Department of Computer Science and has been working with Prof. Machanavajjhala on privacy. She has published in SIGMOD, VLDB, and CCS, and has given tutorials on privacy at VLDB 2016 and SIGMOD 2017. She received best demo award on differential privacy at VLDB 2016 and was awarded a 2017 Google Ph.D. Fellowship in Privacy and Security.