Chapter 5
Analyzing Your Location Data with Provable Privacy Guarantees

Ashwin Machanavajjhala and Xi He

Abstract The ubiquity of smartphones and wearable devices coupled with the ability to sense locations through these devices has brought location privacy into the forefront of public debate. Location information is actively collected to help improve ad targeting, provide useful services to users (e.g., traffic prediction), or study human mobility/activity patterns and correlate them to the health of individuals. In this chapter, we highlight the privacy concerns in large-scale collections of location data from user-centric mobile devices and explain how simple cloaking based techniques might be ineffective. This motivates the need for algorithms that collect and analyze location data with formal provable privacy guarantees. We discuss the state of the art in specifying formal privacy guarantees for location data, as well as algorithms that achieve these formal privacy guarantees. We conclude with open research directions in this area.

5.1 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 [54] and encourage vehicle pooling [45]. Location data has also been found predictive of human purchasing behavior [3], emergency behavior following large-scale disaster [47], and epidemiological patterns [50], and hence improves existing prediction models in

A. Machanavajjhala · X. He (✉)
Duke University, Durham, NC, USA
e-mail: ashwin@cs.duke.edu; hexi88@cs.duke.edu

A. Gkoulalas-Divanis, C. Bettini (eds.), *Handbook of Mobile Data Privacy*,
https://doi.org/10.1007/978-3-319-98161-1_5

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. [14] showed using mobility data of 1.5 million individuals over a period of 15 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. The second property of location data is that individual's location patterns exhibit high predictability. Song et al. [46] 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 [1, 28, 40, 55], or perturbed [4]. For instance, sensitive locations such as home and work addresses can be discerned easily based on the frequency with which locations are visited [5, 21], and perturbed locations can be reconstructed based on temporal correlations within the sequence of locations [53]. 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. [35] 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 [38]. 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 [51], 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.

The rest of this chapter is organized as follows. Section 5.2 describes an important provable privacy notion that satisfies all these desiderata, known as *differential privacy*, and presents several variants of differential privacy for location data and corresponding algorithms for these variants. Section 5.3 introduces a more general privacy framework, *Pufferfish privacy*, which can capture all the variants of differential privacy for location data discussed in Sect. 5.2, explain the privacy semantics underlying these notions, and allow new and rigorous privacy definitions to be created based on the needs of different applications. A general algorithm to ensure Pufferfish privacy is also presented. However, not all the privacy definitions instantiated under Pufferfish privacy can guarantee privacy under composition. Hence, we present a special class of privacy notions instantiated under this framework, called *Blowfish privacy*, in Sect. 5.4. This privacy class guarantees privacy under composition, and allows users to tune privacy-utility tradeoffs by specifying privacy policies. We conclude in Sect. 5.5 with a discussion of challenges and some open research directions.

5.2 Differential Privacy

Differential privacy was first introduced in 2006 [15] 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. In this chapter, we define differential privacy (Sect. 5.2.1), discuss variants of this definition in the context of location data (Sect. 5.2.2), and survey algorithms for differentially private release of location data (Sect. 5.2.3).

5.2.1 Definition and Properties of Differential Privacy

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

Table 5.1 Table of notation

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

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) $\mathfrak{M}$ that satisfies that privacy definition. Then the data curator will apply $\mathfrak{M}$ to the data to obtain a sanitized output $\omega \equiv \mathfrak{M}(\mathfrak{Data})$, where $\mathfrak{Data}$ is the random variable representing the true database instance owned by the data curator which is unknown to the adversary. We use *records*($\mathfrak{Data}$) to denote the set of records in $\mathfrak{Data}$ and $t(\mathfrak{Data})$ to denote the record values (tuples). These notations and the key notations from the rest of this chapter is summarized in Table 5.1.

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 5.1 (Differential Privacy [15, 16]) Given a privacy parameter $\epsilon > 0$, a randomized algorithm $\mathfrak{M}$ satisfies ϵ-differential privacy if for any outputs $\omega \in$ *range*($\mathfrak{M}$) 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:

$$\Pr[\mathfrak{M}(D) = \omega] \leq \exp(\epsilon) \Pr[\mathfrak{M}(D') = \omega], \tag{5.1}$$

where the probability only depends on the randomness in $\mathfrak{M}$.

In this definition, the number of individuals in the database is 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 ϵ is usually known as the privacy budget. When ϵ is small, the output distributions of $\mathfrak{M}$ 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.

Laplace mechanism is an important building block for designing differentially private algorithms.

Definition 5.2 (The Laplace Mechanism) Given any function $f : \mathcal{I} \to \mathbb{R}^k$, the Laplace mechanism is defined as:

$$\mathfrak{M}_L(D, f(\cdot), \epsilon) = f(D) + (\eta_1, \ldots, \eta_k), \tag{5.2}$$

where η_i are i.i.d random variables drawn from $Lap(\Delta f/\epsilon)$, and Δf 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 5.3 (l_1-Sensitivity) The l_1-sensitivity of a function $f : \mathcal{I} \to \mathbb{R}^k$ is

$$\Delta f = \max_{D, D' \in \mathcal{I}, (D, D') \in N_{dp} (\text{ or } N_{dp}^n)} ||f(D) - f(D')||_1, \tag{5.3}$$

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 [18] for more details.

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

Theorem 5.1 (Sequential Composition [15, 16]) *Let $D \in \mathcal{I}$ be an input database. Let $\mathfrak{M}_1(\cdot)$ and $\mathfrak{M}_2(\cdot, \cdot)$ be algorithms with independent sources of randomness that satisfy ϵ_1- and ϵ_2-differential privacy, resp. Then an algorithm that outputs both $\mathfrak{M}_1(D) = \omega_1$ and $\mathfrak{M}_2(\omega_1, D) = \omega_2$ satisfies $(\epsilon_1 + \epsilon_2)$-differential privacy.*

If the second algorithm $\mathfrak{M}_2$ 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 $\mathfrak{M}_2$ is unchanging. Formally,

Theorem 5.2 (Post-processing [15, 16]) *Let $D \in \mathcal{I}$ be an input database. Let $\mathfrak{M}_1(\cdot)$ be an algorithm that satisfies ϵ-differential privacy. Then if an algorithm $\mathfrak{M}_2$ is applied to the output of $\mathfrak{M}_1(\cdot)$, then the overall mechanism $\mathfrak{M}_2 \circ \mathfrak{M}_1(\cdot)$ also satisfies ϵ-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 5.3 (Parallel Composition [15, 16]) *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 $\mathfrak{M}_i$ be mechanisms that each ensure ϵ_i-differential privacy. Then the sequence of $\mathfrak{M}_i(D \cap \mathcal{H}_i)$ 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 [51] 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 [18]. When the queries are not adaptively chosen, tighter bounds on the privacy loss are known [18, 23, 32].

There are other privacy axioms which differential privacy and other good provable privacy notions can satisfy [36]. These properties make differential privacy an appealing choice for many privacy-aware applications and research.

5.2.2 *Variants of Differential Privacy 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 Σ 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 $D \in \mathcal{I}$ 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 jth event of individual h_i which takes a location value from the spatial domain Σ 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.

We will focus on regular trajectories and bounded differential privacy in this section. Neighboring databases for bounded differential privacy N_{np}^{n} 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 5.4 (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

$$|t| = |t'|, \textit{ and if } t[j^*] \neq t'[j^*], \textit{ then } \forall j \neq j^*,\ t[j] = t'[j] \tag{5.4}$$

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 5.5 (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

  $$t \neq t' \tag{5.5}$$

  This protection is applicable to scenarios where the entire location sequences are released [10, 26, 37, 39, 56].
- ***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 5.6 (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

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

  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 [29]. 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 ϵ. If an algorithm ensures ϵ-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.

5.2.3 *Differentially Private Algorithms for Location Data*

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 ϵ, 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 Σ 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 $Lap(1/\epsilon)$ satisfies ϵ-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 ϵ-user-DP, the noise added to the query is drawn from $Lap(T/\epsilon)$, and hence the answer has an expected error of $\sqrt{2}T/\epsilon$. Similarly, to ensure w-event privacy, adding noise from $Lap(w/\epsilon)$ to $f(D, l)$ is sufficient. This noise results in an expected error $\sqrt{2}w/\epsilon$, 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.

5.2.3.1 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 of differential privacy described in Sect. 5.2.2 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

[1] The domain is assumed to be discrete, otherwise it can be discretized.

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 [13, 25, 42, 43, 48, 56] 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 $dom(v)$ 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 $dom(v)$ intersects with R. If $dom(v)$ is fully contained in R, the answer is incremented by $\tilde{c}(v)$. If $dom(v)$ 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 $dom(v)$ partially intersects q and v is a leaf node, then we inspect the data points in $dom(v)$, 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 $Lap(2(h+1)/\epsilon)$ to the count of each node in T satisfies ϵ-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 [13], correlations between noisy counts [13, 25] and pruning tree nodes [42, 43, 48] where the maximum tree height h is given, or by designing algorithms independent of h [56].

- **Optimizations with a given maximum tree height h.** Most of the prior work proposed algorithms [13, 25, 42, 43, 48] 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 [13] 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 $Lap(2(h+1)/\epsilon)$ to all nodes is equivalent to uniform budgeting by split ϵ uniformly across each level. Cormode et al. [13] 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 ϵ_i is $Var(Lap(\epsilon_i)) = 2/\epsilon_i^2$. Since the noise is independently generated in each node, the total variance is $Err(q) = \sum_{i=0}^{h} 2n_i/\epsilon_i^2$, where n_i is the number

of nodes at height i contributed to the query q and ϵ_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 [13, 25]. The last optimization [42, 43, 48] 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 [56] 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 [56].

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 [13].

5.2.3.2 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 $(f(D[j], l))_{j=1,2,\ldots}$ 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 (ϵ, w)-event privacy, but this approach is not optimal.

Kellaris et al. [29] proposed a sliding window methodology. The overall mechanism denoted by $\mathfrak{M}$ which takes an input stream with prefix $D[1 : j]$ can be decomposed into j sub-mechanisms $\mathfrak{M}_1, \ldots, \mathfrak{M}_j$. Let each $\mathfrak{M}_k$ for $k \in [1 : j]$ generate independent randomness to achieve ϵ_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 $\mathfrak{M}$ satisfies w-event privacy. This means that if the sum of privacy budgets per sliding w-window is no more than ϵ, the overall mechanism ensures (ϵ, w)-event privacy.

With this sliding window methodology, two baselines were given by [29]: (1) uniformly allocating ϵ/w budget to each event so that the sum of the budget per sliding window is always ϵ; (2) publishing a single event with privacy budget ϵ 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 [29] proposed to skip publications of counts that are similar to previously released ones.

In the new solutions, each sub-mechanism $\mathfrak{M}_j$ has two parts $\mathfrak{M}_{j,1}$ and $\mathfrak{M}_{j,2}$. The first part $\mathfrak{M}_{j,1}$ 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 $\mathfrak{M}_{j,2}$ 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 ϵ_1/w to $\mathfrak{M}_{j,1}$ for computing distance differentially privately at time stamp j, and use the remaining budget ϵ_2 for publishing counts, where $\epsilon_2 = \epsilon - \epsilon_1$. We will see how the publication budget ϵ_2 is spent within each sliding window.

- **Budget Distribution**: The publication budget ϵ_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 ϵ_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 [29] 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 [6–8, 11] developed for streaming setting. Among them, only PeGaSus [11] 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 ϵ-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 [20], but may result in poorer data quality.

5.2.3.3 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. Under this setting, we learn a model first from the original ground truth and then generate a synthetic database from the model. Depending on the privacy definition, the sensitivities of the queries used to compute the sufficient statistics of the model will change. We will focus on user-DP here for databases of location sequences, but the techniques presented can be extended for event/window-DP. Counting queries for sub-sequence of locations are common queries used for building the model. If the entire sequence per user is short and fixed, e.g. home location and work location, the l_1-sensitivity for the sub-sequence counting queries is small, and hence the privacy budget can be split over different sets of queries. Related work can be found in [37, 39, 56]. On the other hand, if sequences are long, a Markov process is commonly considered [10, 26, 56] to model the correlation between the events. Formally,

Definition 5.7 (Markov Process) A sequence of locations $(l_1 l_2 \cdots l_n) \in \Sigma^n$ is said to follow an order ℓ Markov process if for every $\ell \leq j < n, l \in \Sigma$

$$\Pr[l_{i+1} = l | l_1 \cdots l_i] = \Pr[l_{i+1} = l | l_{i-\ell+1} \cdots l_i]. \tag{5.6}$$

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

$$\Pr[l_{i+1} = l | l_{i-\ell+1} \cdots l_i] = \frac{f(D, xl)}{f(D, x)}, \tag{5.7}$$

where $f(D, x)$ denotes the number of occurrences of x in the database D. Starting symbols ($\top$) and stopping symbols ($\bot$) are prepended and appended (respectively)

to the original trajectories to capture the starting and stopping probabilities in the Markov process. The synthesis of a trajectory begins with a starting symbol ($\top$). Based on the transition probabilities from the Markov process, a next location is sampled continuously till reaching the stopping symbol ($\perp$). This model requires to maintain all ℓ-gram counts for $1 \leq \ell \leq h$, where $h-1$ is the maximum order of Markov process considered. A *prefix tree* T of heights h is used to store these counts, where nodes in T are $\Sigma^1 \cup \ldots \cup \Sigma^h$, and edges connected each ℓ-gram x to $\ell+1$-gram xl for all $l \in \Sigma$.

To ensure user-DP, prior approaches add noises drawn from a Laplace distribution to parts of the prefix tree T [10, 56]. 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, He et al. [26] proposed an end-to-end system, named Differentially Private Trajectories (*DPT*) to address these challenges. The schematic overview of this system is shown in Fig. 5.1.

DPT discretizes the spatial domain at multiple resolutions to capture different step sizes (see Step 1 in Fig. 5.1). Every resolution has a prefix tree (Step 2). Within each resolution, only movements from each grid cell to neighboring cells in one step are allowed. Though there is a *larger* number of prefix trees, each prefix tree has a much *smaller* branching factor, thus resulting in a big reduction in the number of counts maintained by the model. DPT uses a novel model selection algorithm (Step 3) to set the tree heights and to prune unrealistic resolutions in a differentially private manner. The following steps add noises drawn from the Laplace distribution to the chosen prefix trees (Step 4), and prune adaptively these noisy trees (Step 5) to further improve utility. In the last sampling step (Step 6), a novel postprocessing strategy is applied by DPT to restore the directionality of synthetic trajectories which could be lost due to the noise added to the private model. Based on these optimizations, this end-to-end system can synthesize trajectories spanning large geographical areas with significantly more utility than the prior work [10] and is orders of magnitude faster. These synthetic trajectories have been shown mirroring the original trajectories on three utility metrics—distribution of diameter (i.e., distance traveled), conditional distributions of destinations given starting regions, and frequent patterns. However, synthetic trajectories cannot join with other datasets due to the absence of join keys. Prior work [16] has shown that non-interactive setting can have more error than an interactive setting, main due to the difficulty of supplying utility that has not yet been specified at the time the data synthesis is carried out. Moreover, additional efforts have to be applied to synthetic trajectories such that they are realistic and satisfying real-world constraints.

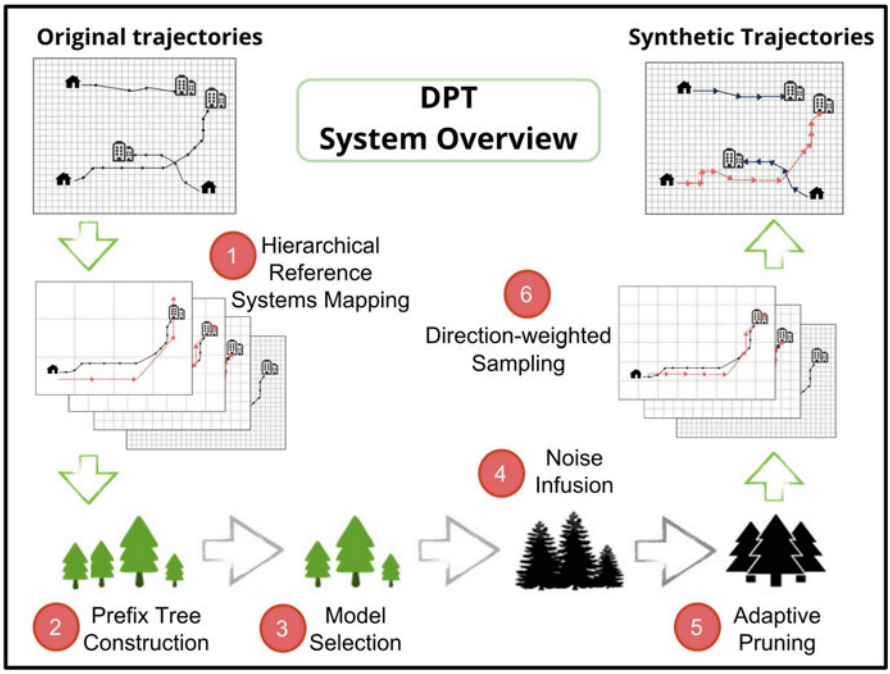

Fig. 5.1 DPT framework overview

In summary, from these prior work [4, 10, 13, 19, 26, 29, 37, 39, 49, 53, 56], we see that differential privacy has been well explored for location data. Applications considered various forms of neighboring databases and hence different algorithms. However, it is not clear that (1) what information is protected via the different specification of neighboring databases; and (2) which algorithms can be used to protect against adversaries with prior background knowledge. Moreover, the algorithms presented so far are designed mainly for counting queries. For location-based services that require location information at a particular time stamp, even event-DP, the weakest privacy notion seen so far, is too strong to provide good accuracy. Hence, we will first see how to quantify the privacy guarantees of algorithms through the lens of secrets and adversaries using a more general framework called *Pufferfish* in Sect. 5.3. Then, we will show how to design algorithms for location-based services that can achieve both accuracy and provable privacy guarantees named as *Blowfish Privacy* under this framework in Sect. 5.4.2.

5.3 Pufferfish Privacy

In this section, we summarize the Pufferfish privacy framework [31] 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 Sect. 5.3.1, we introduce how the Pufferfish framework defines privacy in terms of secrets and (the prior knowledge available to) adversaries, rather than neighboring databases. In Sect. 5.3.2, we show that variants of differential privacy are instantiations of the Pufferfish framework, and explain under what assumptions about secrets and adversaries each of these variants ensure *semantic* privacy guarantees. Finally, in Sect. 5.3.3, we describe algorithms for general Pufferfish privacy definitions.

5.3.1 Definition of Pufferfish Framework

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

- **The set of potential secrets** $\mathbb{S}$ represents the information that data curator would like to protect. A secret can be specified as a statement such as "*Bob is at location* $l \in \mathcal{T}$", "*Bob is not at location* $l \in \mathcal{T}$". In general, a domain expert should add a statement s to the potential secrets $\mathbb{S}$ if either the claim that s is true or the claim that s is false can be harmful. The resulted $\mathbb{S}$ forms a domain for the *discriminative pairs*, a subset of $\mathbb{S} \times \mathbb{S}$.
- **The set of discriminative pairs** $\mathbb{S}_{pairs}$, is a subset of $\mathbb{S} \times \mathbb{S}$. The role of $\mathbb{S}_{pairs}$ is to tell how to protect the potential secrets $\mathbb{S}$. For any discriminative pair $(s_i, s_j) \in \mathbb{S}_{pairs}$, 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 $l_1, l_2 \in \mathcal{T}$. The set of changes for neighboring databases shown in Sect. 5.2.2 are examples for the set of discriminative pairs.

  This specification allows highly customizable privacy guarantees. For instance, many location-based applications such as OpenPaths [41] and Airbnb [2] 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 [34], this privacy

preference can be expressed via a set of discriminative pairs $\mathbb{S}_{pairs}$ 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** $\mathbb{D}$ 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 [17, 30]. 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 [44]. Formally, $\mathbb{D}$ is represented as a set of probability distributions over $\mathcal{I}$ (the possible database instances). Each probability distribution $\theta \in \mathbb{D}$ 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 $\Pr(\mathfrak{Data} = D|\theta)$ to represent the probability, under θ, that the true database is D. Below we give some examples of possible choices of $\mathbb{D}$ and their interpretations.

  Example 5.1 (No Assumptions) $\mathbb{D}$ can consist of all possible probability distributions over database instances (i.e. including those with arbitrary correlations between records). This corresponds to making no assumptions.

  Example 5.2 (Independent Individuals but Markov Model-Based Events) Several work [49, 53] 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, $\mathbb{D}$ consists of all θ for which

$$\Pr[\mathfrak{Data} = \{r_1, \ldots, r_n|\theta\}] = f_1(r_1) \times f_2(r_2) \times \ldots \times f_n(r_n) \tag{5.8}$$

  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_θ for $\mathbb{D}$, 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. $\Pr(r_i[j] = l_2|r_i[j-1] = l_1)$.

  Readers may refer to [31] 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 5.8 (Pufferfish Privacy [31]) Given a set of potential secrets $\mathbb{S}$, a set of discriminative pairs $\mathbb{S}_{pairs}$, a set of data evolution scenarios $\mathbb{D}$, and a privacy parameter $\epsilon > 0$, a potentially randomized algorithm $\mathfrak{M}$ satisfies ϵ-Pufferfish

$(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$ privacy if (i) for all possible outputs $\omega \in range(\mathfrak{M})$, (ii) for all pairs $(s_i, s_j) \in \mathbb{S}_{pairs}$ of potential secrets, (iii) for all distributions $\theta \in \mathbb{D}$ for which $\Pr(s_i|\theta) \neq 0$ and $\Pr(s_j|\theta) \neq 0$, the following holds:

$$\Pr[\mathfrak{M}(\mathfrak{Data}) = \omega|s_i, \theta] \leq e^{\epsilon} \Pr[\mathfrak{M}(\mathfrak{Data}) = \omega|s_j, \theta] \tag{5.9}$$

$$\Pr[\mathfrak{M}(\mathfrak{Data}) = \omega|s_j, \theta] \leq e^{\epsilon} \Pr[\mathfrak{M}(\mathfrak{Data}) = \omega|s_i, \theta], \tag{5.10}$$

where $\mathfrak{Data}$ is a random variable representing the true dataset (which is unknown to the adversaries).

5.3.2 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 σ_i be the statement $r_i \in records(\mathfrak{Data})$ (i.e. "records r_i belonging to individual h_i is in the data", and let $\sigma_{(i,t)}$ be the statement $r_i \in records(\mathfrak{Data}) \wedge r_i = t$(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

$$\mathbb{S} = \{\sigma_{i,t} : h_i \in \mathcal{H}, t \in \mathcal{T}\} \cup \{\neg\sigma_i : h_i \in \mathcal{H}\} \tag{5.11}$$

$$\mathbb{S}_{pairs} = \{(\sigma_{i,t}, \neg\sigma_i) : h_i \in \mathcal{H}, t \in \mathcal{T}\} \tag{5.12}$$

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 $\mathbb{D}$ 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

$$\Pr[\mathfrak{Data}|\theta\}] = \prod_{r_i \in records(\mathfrak{Data})} f_i(r_i) \Pr[\sigma_i] \prod_{r_i \notin records(\mathfrak{Data})} (1 - Pr[\sigma_i]), \tag{5.13}$$

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 5.4 ([31]) *With the choices of* $\mathbb{S}$ *and* $\mathbb{S}_{pairs}$ *defined in Eqs.* (5.11) *and* (5.12)*, and the set of data evolution scenarios* $\mathbb{D}$ *as specified in Eq.* (5.13)*, unbounded* ϵ*-differential privacy is equivalent to* ϵ*-Pufferfish*$(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$*.*

The variants of differential privacy for location data described in Sect. 5.2.2 can all be described under this framework. For instance, event-DP can be shown to be equivalent to a Pufferfish instantiation where: (1) the set of secrets are properties of an individual's location at a single time point; and (2) 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: (1) the set of secrets are properties of the entire trajectory; and (2) 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 5.2. Let's name this privacy *Event-MarkovAdversary-Privacy*. There are algorithms like the Markov Quilt Mechanism described next, that can ensure more privacy than event-DP, and more accuracy than user-DP, helping us better tradeoff privacy and utility.

5.3.3 Algorithms for Pufferfish Privacy

We first present a special algorithm for location data which consider adversaries with assumptions shown in Example 5.2, that individuals are independent but sequences of events are correlated and are modeled by Markov model. Then, we will present a general algorithm for Pufferfish privacy.

5.3.3.1 Markov Quilt Mechanism

Markov Quilt Mechanism was proposed by Wang et al. [49]. This mechanism applies Event-MarkovAdversary-Privacy and considers counting queries (with l_1-sensitivity of 1) over a location database over a period of time. Based on the definition of the adversary in Event-MarkovAdversary-Privacy, the sequence of

locations in the location trajectory can be modeled as a Bayes net, a chain $X_1 \rightarrow X_2 \rightarrow \ldots X_T$, where each event X_i only depends on its previous event X_{i-1}. Based on this correlation, the impact of X_i on X_{i+1} is more significant than the impact of X_i on X_{i+k} when k is large. Thus it is sufficient to add noise proportional to the number of events that are highly correlated with the event at each time point. The notion of Markov Quilt is based on the set of highly correlated events for a given event. The size of the Markov Quilt depends on the strength of the correlation, and not on the total size of the trajectory. Hence, unlike user-DP which would add noise proportional to the length of the trajectory, Markov Quilt mechanism will add noise proportional to the size of the Markov Quilt which could be much smaller, thus protecting against adversaries who know correlations as well as ensuring low error. The details of this mechanism can be referred to [49].

5.3.3.2 General Algorithm 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 Eq. (5.3), 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 $(s_i, s_j) \in \mathbb{S}_{pairs}$. Hence, Wang et al. [49] consider the two distributions given a secret pair $(s_i, s_j) \in \mathbb{S}_{pairs}$, i.e.

$$\mu_{i,\theta} = \Pr(f(\mathfrak{Data}) = \cdot | s_i, \theta)$$

$$\mu_{j,\theta} = \Pr(f(\mathfrak{Data}) = \cdot | s_j, \theta)$$

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 5.9 (∞-Wasserstein Distance [49]) Let $(\mathcal{X}, d)$ be a Radon space, and μ, ν be two probability distribution on $\mathcal{X}$ with finite p-th moment. The ∞-Wasserstein distance between μ, ν with $d(x, y) = |x - y|$:

$$W_\infty(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu,\nu)} \max_{(x,y)\in A} |x - y|, \tag{5.14}$$

where $A = \{(x, y) | \gamma(x, y) \neq 0\}$ is the support of γ, and $\Gamma(\mu, \nu)$ is the set of all couplings γ over μ and ν.

Intuitively, $\gamma \in \Gamma(\mu, \nu)$ is a way to shift probability mass between μ and ν, and $W_\infty(\mu, \nu)$ can be interpreted as the maximum "distance" that any probability mass moves while transforming μ to ν in the most optimal way. Wang et al. [49] proposed a general mechanism for Pufferfish framework Pufferfish framework $(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$ with privacy budget ϵ and query f. This mechanism first computes the generalized sensitivity, defined as

$$\Delta W_\infty(f, \mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D}) = \max_{(s_i,s_j)\in\mathbb{S}_{pairs}} \max_{\theta\in\mathbb{D} \mid \Pr(s_i|\theta)\neq 0, \Pr(s_j|\theta)\neq 0} W_\infty(\mu_{i,\theta}, \mu_{j,\theta}) \tag{5.15}$$

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

Definition 5.10 (Wasserstein Mechanism) Given any function $f : \mathcal{I} \to \mathbb{R}^k$, the Wasserstein mechanism is defined as :

$$\mathfrak{M}_W(D, f(\cdot), \mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D}, \epsilon) = f(D) + (\eta_1, \ldots, \eta_k), \tag{5.16}$$

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

Wang et al. [49] showed that Wasserstein mechanism provides ϵ-Pufferfish privacy in the framework $(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$. 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 [49], and hence can result in higher accuracy for query f.

5.4 Blowfish Privacy

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 [31]. Hence, we introduce a simple but useful class of privacy definitions named Blowfish privacy [22, 27] that addresses limitations of the general framework of Pufferfish. The definition and properties of Blowfish privacy are presented in Sect. 5.4.1. We also show in Sect. 5.4.2 algorithms for special instances of Blowfish privacy and also a general algorithm for Blowfish privacy.

5.4.1 Definition and Properties of Blowfish Privacy

The key building block of an instantiation of Blowfish privacy is named *policy graph*. A policy graph is a graph representation of $\mathbb{S}_{pairs}$, pairs of domain values in $\mathcal{T}$ that an adversary must not be able to distinguish.

Definition 5.11 (Policy Graph [22, 27]) A policy graph is a graph $G = (V, E)$ with $V \subseteq \mathcal{T} \cup \{\perp\}$, where $\perp$ is the name of a special vertex, and $E \subseteq (\mathcal{T} \cup \{\perp\}) \times (\mathcal{T} \cup \{\perp\})$.

An edge $(u, v) \in E$ corresponds to a pair of domain values that an adversary should not be able to distinguish between. $\perp$ is a dummy value not in $\mathcal{T}$, and an edge $(u, \perp) \in E$ means that an adversary should not be able to distinguish between the presence of a tuple with value u, or the absence of the tuple from the database. If a policy graph does not include $\perp$, we can focus on databases with fixed known size. Based on this policy graph G, we re-define the concept of neighboring databases of differential privacy in the following way.

Definition 5.12 (Neighbors [22, 27]) Consider a policy graph $G = (V, E)$. Let D_i and D_j be two databases in $\mathcal{I}$. D_i and D_j are neighbors, denoted by $(D_i, D_j) \in N(P)$, iff exactly one of the following is true:

- D_i and D_j differ in the value of exactly one entry such that $(u, v) \in E$, where u is the value of the entry in D_i and v is the value of the entry in D_j;
- D_i differs from D_j in the presence or absence of exactly one entry, with value u, such that $(u, \perp) \in E$.

Example 5.3 (Event-DP) Recall event-DP in Sect. 5.2.2 considers neighboring databases differ in a single event. The set of discriminative pairs for event-level neighbors can be specified as

$$\mathbb{S}^{event}_{pairs} = \{(r_i = t, r_i = t') | h_i \in \mathcal{H};\ t, t' \in \Sigma^*,\ |t| = |t'|, \\ \forall j^*, \textit{if } t[j^*] \neq t'[j^*], \textit{ then } \forall j \neq j^*,\ t[j] = t'[j]\}. \tag{5.17}$$

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 event-DP.

Example 5.4 (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,

$$\mathbb{S}^{event,\theta}_{pairs} = \{(r_i = t, r_i = t') | h_i \in \mathcal{H},\ t, t' \in \Sigma^*,\ |t| = |t'|, \\ \forall j^*, \textit{if } d(t[j^*], t'[j^*]) \leq \theta, \textit{ then } \forall j \neq j^*,\ t[j] = t'[j]\} \tag{5.18}$$

This captures variants of event DP proposed in prior work like Geo-indistinguishability [4] and (θ, ϵ)-location privacy [19], 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 θ. Compared to event-level discriminative pairs $\mathbb{S}^{event}_{pairs}$, $\mathbb{S}^{event,\theta}_{pairs}$ 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 $\mathbb{S}^{event,\theta}_{pairs}$ is a subgraph of the policy graph for $\mathbb{S}^{event}_{pairs}$. 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 [27] includes $\mathcal{I}_Q$ 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 $\mathcal{I}_Q$. 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 5.5 (δ-Location Set Based Differential Privacy) The privacy definition proposed by Xiao et al [53] 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 $\mathbb{D}$ is represented by a hidden Markov model (HMM) which consists of a single transition matrix P_θ and an emission probability P_θ^e. The prior distribution for an individual h_i being at location l at timestamp j given the previous $(j-1)$ events $\Pr(r_i[j] = l | l_{j-1} \dots l_1)$ can be derived from P_θ and P_θ^e, and can eliminate unlikely secrets from $\mathbb{S}$. The remaining possible locations are specified by a new term called δ-location set. Formally, for any $j \in [1, 2, \dots]$, the δ-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.

$$\Delta X_j = \min\{l | \sum_l \Pr(r_i[j] = l | l_{j-1} \dots l_1) \geq 1 - \delta\}. \tag{5.19}$$

At any time point j, a randomized mechanism $\mathfrak{M}$ satisfies ϵ-differential privacy on δ-location set ΔX_j, if for any output ω_j and any two locations l_1 and l_2 in ΔX_j, the following holds: $\Pr(\mathfrak{M}(l_1) = \omega) \leq e^\epsilon \Pr(\mathfrak{M}(l_2) = \omega)$.

This privacy definition is also known as *δ-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 δ-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.

Readers may refer to [27] for the general version of Blowfish neighbors with constraints. For more general policy graph with constraints, we can define Blowfish privacy as follows.

Definition 5.13 (Blowfish Privacy [22, 27]) Let $\epsilon > 0$ be a real number and policy $P = (\mathcal{T}, G, \mathcal{I}_Q)$ be a policy. A randomized mechanism M satisfies (ϵ, P)-Blowfish privacy if for every pair of neighboring databases $(D_i, D_j) \in N(P)$, and every set of outputs $S \subseteq range(M)$, we have

$$Pr[\mathfrak{M}(D_i) \in S] \leq e^\epsilon Pr[\mathfrak{M}(D_j) \in S] \tag{5.20}$$

If we consider policy without constraints, we simplify our notation for Blowfish privacy as (ϵ, G)-Blowfish privacy.

Now the privacy guarantee is not only controlled by the privacy parameter ϵ, but also by the policy graph G. Consider two databases $D_1 = D \cup \{u\}$ and $D_2 = D \cup \{v\}$ that differ in one tuple. Given a mechanism $\mathfrak{M}$ that satisfies (ϵ, G)-Blowfish privacy, we have

$$Pr[\mathfrak{M}(D_i) \in S] \leq e^{\epsilon \cdot d_G(u,v)} Pr[\mathfrak{M}(D_j) \in S] \tag{5.21}$$

where $d_G(u, v)$ is the shortest distance between u, v in G. This implies that an adversary may better distinguish pairs of nodes farther apart in the graph than those that are closer. Similarly, an adversary can distinguish between u, v with probability 1, when u and v appear in disjoint components of G, where $d_G(u, v) \to \infty$. Note that when G is a complete graph K, then (ϵ, K)-Blowfish privacy is equivalent to ϵ-differential privacy.

Theorem 5.5 (Sequential Composition [27]) *Let $P = (\mathcal{T}, G, \mathcal{I}_Q)$ be a policy and $D \in \mathcal{I}_Q$ be an input database. Let $\mathfrak{M}_1(\cdot)$ and $\mathfrak{M}_2(\cdot, \cdot)$ be algorithms with independent sources of randomness that satisfy (ϵ_1, P) and (ϵ_2, P)-Blowfish privacy, resp. Then an algorithm that outputs both $\mathfrak{M}_1(D) = \omega_1$ and $\mathfrak{M}_2(\omega_1, D) = \omega_2$ satisfies $(\epsilon_1 + \epsilon_2, P)$-Blowfish privacy.*

Algorithms that satisfy Blowfish also satisfy the postprocessing theorem (like Theorem 5.2) and a restricted form of parallel composition. We refer the reader to [27] for details.

5.4.2 Mechanisms for Blowfish Privacy

Blowfish privacy generalized (bounded and unbounded) differential privacy. In fact, we can show that any algorithm that satisfies ϵ-bounded DP (or $\epsilon/2$-unbounded DP) also satisfies (ϵ, G)-Blowfish privacy for any policy graph G (when the definition has no constraints Q). Thus, in the absence of constraints, differentially private algorithms can be used to satisfy Blowfish privacy definitions. However, leveraging the policy can lead to algorithms that provide more accuracy than DP algorithms as we will see in the rest of this section.

5.4.2.1 Releasing Perturbed Locations

In many Location-Based Services (LBSs), an actual location needs to be shared. While one can design event-DP algorithms using variants of randomized response to release perturbed locations, they would have poor utility. Due to the large domain size of locations, the probability that one would report a point close to

the true location would be vanishingly small. On the other hand, we can develop methods to release perturbed locations with high accuracy under Blowfish policies corresponding to Geo-indistinguishability [4] and δ-location set differential privacy [53]. The details of the algorithms are described below.

- **Geo-indistinguishable mechanism** was designed by Andres et al. [4] to ensure ϵ-geo-indistinguishability. The location domain Σ is modeled as the Euclidean plane equipped with the standard notion of Euclidean distance. For the ideal case of the continuous plane, where $\Sigma = \mathbb{R}^2$. Given a true location $r_i[j] = l$, where $l \in \mathbb{R}^2$ and privacy parameter ϵ, the mechanism would like to draw a location $l \in \mathbb{R}^2$ with probability function

$$D_\epsilon(l_0)(l) = \frac{\epsilon^2}{2\pi} e^{-\epsilon d(l_0,l)}. \tag{5.22}$$

  It is easy to show that this mechanism satisfies ϵ-geo-indistinguishability. The actual sampling process takes place in a system of polar coordinates that centered at l. Equation 5.22 can be transformed into PDF of the polar laplacian centered in the origin l_0, where $D_{\epsilon,R}(r) = e^2 r e^{-\epsilon r}$, $D_{\epsilon,\Theta}(\theta) = \frac{1}{2\pi}$. Based on the PDF, the angle θ can be drawn uniformly $[0, 2\pi]$. For radius r, we first draw z uniformly in $[0, 1)$ and set $r = C_\epsilon^{-1}(z)$, where $C_\epsilon(r) = 1 - (1 + \epsilon z)e^{\epsilon z}$ is the cumulative function for $D_{\epsilon,R}(r)$. Readers may refer to [4] for the adjusted mechanism for discrete coordinates.

  However, this mechanism is susceptible to attacks when the adversary knows correlations across time points in a trajectory[9, 53]. Hence, this temporal correlation-based attacks motivates the following mechanism.
- **Planar Isotropic Mechanism** was proposed by Xiao et al. [53] to ensure δ-location set based differential privacy (Example 5.5), where the sequence of locations are correlated by a Hidden Markov model. As l_1-sensitivity fails to capture the geometric sensitivity in multidimensional space, Xiao et al. [53] proposed a new notion, *sensitivity hull* to bound the change in the query output caused by the modification of an event. The sensitivity hull of a query f is defined as the convex hull of Δf where Δf is the set of $f(l_1) - f(l_2)$ for any pair l_1 and l_2 in δ-location set ΔX. This sensitivity hull was further transformed into isotropic position to ensure optimal solution of K-norm Mechanism [24] for 2-dimensional space. K-norm Mechanism is defined as below.

  Definition 5.14 (K-norm Mechanism) Given a linear function $F : R^N \rightarrow R^d$ and its sensitivity hull K, a mechanism is K-norm mechanism if for any output z, the following holds:

$$Pr(z) = \frac{1}{\Gamma(d+1)VOL(K/\epsilon)} exp(-\epsilon||z - Fx^*||_K), \tag{5.23}$$

  where Fx^* is the true answer, $||\cdot||_K$ is the (Minkowski) norm of K, $\Gamma()$ is Gamma function and $VOL()$ indicates volume.

The releasing of location at timestamp j can be summarized as four steps: (1) the sensitivity hull K is computed from the given δ-location set ΔX at timestamp j, (2) the sensitivity hull K is then transformed to its isotropic position K_I to ensure the optimal solution for K-norm mechanism, (3) a sample is picked from K_i and perturbed to z' by K-norm mechanism; (4) finally this perturbed sample z' is transformed to z in the original space and published. This mechanism has been shown in [53] with error $O\left(\frac{1}{\epsilon}\sqrt{AREA(K)}\right)$ at most, and this is the lower bound of any mechanism that satisfies δ-location set based differential privacy.

5.4.2.2 Aggregate Perturbation for Count Queries (General Algorithms for Blowfish Privacy)

In [22, 27], Blowfish private mechanisms were designed to answer aggregate queries under different policy graphs. Each policy graph can instantiate a new notion of neighboring databases. Rather than re-designing a new algorithm for each notion of neighboring databases, [22] showed a transformational equivalence between a large class of Blowfish private algorithms and standard differentially private 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 $\mathfrak{M}$ is an (ϵ, G)-Blowfish private mechanism for answering workload W on input x if and only if $\mathfrak{M}$ is also an ϵ-differentially private mechanism for answering W_G on x_G. This result does not hold in general, but [22] showed that under a class of mechanism called *matrix mechanism*, transformational equivalence holds for any policy graph.

Equivalence for Matrix Mechanism. Matrix mechanism framework was designed for optimally answering a workload of linear queries [33]. 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 Δ_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 pseudoinverse, such that $WAA^+ = W$. The matrix mechanism is given by the following:

$$\mathfrak{M}_A(W, x) = Wx + WA^+ Lap(\Delta_A/\epsilon)^p \tag{5.24}$$

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

Definition 5.15 (Policy Specific l_1 Sensitivity) The l_1 policy specific sensitivity of a query matrix W with respect to policy graph G is

$$\Delta_w(G) = \max_{x,x' \in N(G)} ||Wx - Wx'||_1 \tag{5.25}$$

Let P_G be a matrix that satisfies the following properties.

- P_G has $|V| - 1$ rows and $|E|$ columns.
- Let $W_G = W P_G$. 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 5.6 ([22]) *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 Eq. (5.24) is both a (ϵ, G)-Blowfish private mechanism for answering W on x and an ϵ-differentially private algorithm for answering W_G on x_G. Since $Wx = W_G x_G$, the mechanism has the same error in both instances.*

We illustrate the strategies proposed in [22] with the example of answering range counting queries over two dimensional location for *distance-threshold* policy graphs. These graphs are based on similar secret specification as Geo-indistinguishability, by considering the set of discriminative secrets

$$\mathbb{S}^{\theta}_{pairs} = \{(r_i[j] = l, r_i[j] = l') \mid d(l, l') \leq \theta,\ l, l' \in \Sigma, j = 1, 2, \ldots\}. \tag{5.26}$$

Particularly for a grid-based location domain of size $k \times k$, this class of policy graph $G^{\theta}_{k^2}$ is defined based on the l_1-distance in the domain $[k]^2$, where $[k]$ denotes the set of integers between 1 and k (inclusive). The vertices in G are the grid cells. There is an edge u, v in E if and only if $|u - v|_1 \leq \theta$.

Consider rectangle range counting query $q([x, y], [x', y'])$. When $\theta = 1$, the transformed query $q_{G^1_{k^2}}$ is the sum of four disjoint range counting queries in the transformed domain. Hence, the strategy for answering the transformed query workload would be to answer $2(k - 1)$ one-dimensional range count queries under differential privacy. The error per query under $(\epsilon, G^1_{k^2})$-Blowfish privacy for all rectangle range counting queries is $O(\log^3 k/\epsilon^2)$ while the best known data independent strategy answering the same workload with ϵ-differential privacy is the Privelet strategy [52] with a much larger asymptotic error of $O(\log^6 k/\epsilon^2)$ per query.

When $\theta > 1$, the policy graph is more complex. The algorithm proposed leverages subgraph approximation and can achieve an error of $O(\log^3 k \log^3 \theta/\epsilon^2)$ per query under $(\epsilon, G^{\theta}_{k^2})$, which is still better than using Privelet ($O(\log^6 k/\epsilon^2)$ per query) when $\log \theta$ is small compared to $\log k$.

5.5 Conclusions and Open Challenges

In this chapter, we identified desiderata that algorithms for privacy-preserving release and analysis of location data must ensure. These include ensuring provable privacy guarantees of an individual's properties even when adversaries have strong prior knowledge and satisfying composition properties that allow for a graceful degradation of privacy even under multiple releases of the data. 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 also presented Pufferfish, a framework for defining privacy that allows us to reason about the privacy semantics underlying the variants of differential privacy. We concluded by describing instantiations of Pufferfish that allow for ensuring privacy when adversaries may know correlations within the location stream, and described a subclass of Pufferfish, called Blowfish, that satisfies composition theorems.

While a wide range of provable privacy definitions are available today, it is still unclear which of these definitions are applicable to a given scenario, and whether the algorithms satisfying these definitions allow realistic analysis of location data with acceptable errors. 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.

Location trajectories have inherent correlations, both within a single trajectory and across trajectories. We have seen examples of the former, and some solutions to handle these correlations when they take a specific form. However, there is little work that acknowledges and handles correlations across individuals. 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 [12]. Whether techniques like the Markov Quilt mechanism for handling correlations will be applied to such cases is an interesting open question.

All of the work presented assumes that (a) users require the same level of privacy, and (b) users are able to specify privacy levels in terms of the privacy parameter ϵ. The former is clearly not true in the real world. There is work that suggests that users have different privacy seeking behaviors depending on their demographic attributes as well as based on their context. Moreover, it is not clear whether or not users will be able to express their privacy preferences in terms of an ϵ privacy parameter. These challenges will motivate new interesting privacy research to further advance the state-of-the-art techniques, and ensure their adoption in real systems.

References

1. O. Abul, F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7–12, 2008, Cancún, México*, pages 376–385, 2008.
2. Airbnb. Airbnb privacy policy, 2017. https://www.airbnb.com/terms/privacy_policy.
3. S. Altman, N. Sivo, E. Tana, and B. Knapp. Location-based advertising message serving for mobile communication devices, 2008. US Patent App. 11/931,113.
4. M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 901–914, New York, NY, USA, 2013. ACM.
5. Atockar. Riding with the stars: Passenger privacy in the nyc taxicab dataset, 2014. https://research.neustar.biz/author/atockar/.
6. J. Bolot, N. Fawaz, S. Muthukrishnan, A. Nikolov, and N. Taft. Private decayed predicate sums on streams. In *Proceedings of the 16th International Conference on Database Theory*, ICDT '13, pages 284–295, New York, NY, USA, 2013. ACM.
7. J. Cao, Q. Xiao, G. Ghinita, N. Li, E. Bertino, and K.-L. Tan. Efficient and accurate strategies for differentially-private sliding window queries. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, pages 191–202, New York, NY, USA, 2013. ACM.
8. T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24, Nov. 2011.
9. K. Chatzikokolakis, C. Palamidessi, and M. Stronati. *A Predictive Differentially-Private Mechanism for Mobility Traces*, pages 21–41. Springer International Publishing, Cham, 2014.
10. R. Chen, G. Acs, and C. Castelluccia. Differentially private sequential data publication via variable-length n-grams. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 638–649, New York, NY, USA, 2012. ACM.
11. Y. Chen, A. Machanavajjhala, M. Hay, and G. Miklau. Pegasus: Data-adaptive differentially private stream processing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1375–1388, 2017.
12. E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *ACM SIGKDD Conference*, KDD '11, 2011.
13. G. Cormode, C. M. Procopiuc, D. Srivastava, E. Shen, and T. Yu. Differentially private spatial decompositions. In *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1–5 April, 2012*, pages 20–31, 2012.
14. Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports*, 3, 2013.
15. C. Dwork. Differential privacy. In *IN ICALP*, pages 1–12. Springer, 2006.
16. C. Dwork, F. Mcsherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *In Proceedings of the 3rd Theory of Cryptography Conference*, pages 265–284. Springer, 2006.
17. C. Dwork and M. Naor. On the difficulties of discloure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality*, 2010.
18. C. Dwork and A. Roth. The algorithmic foundations of differential privacy. Technical report, Theoretical Computer Science, 2013.
19. E. ElSalamouny and S. Gambs. Differential privacy models for location-based services. *Transactions on Data Privacy*, 9(1):15–48, 2016.
20. L. Fan and L. Xiong. Real-time aggregate monitoring with differential privacy. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2169–2173. ACM, 2012.

21. S. Gambs, M.-O. Killijian, and M. N. n. del Prado Cortez. Show me how you move and i will tell you who you are. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '10, pages 34–41, New York, NY, USA, 2010. ACM.
22. S. Haney, A. Machanavajjhala, and B. Ding. Design of policy-aware differentially private algorithms. *Proc. VLDB Endow.*, 9(4):264–275, Dec. 2015.
23. M. Hardt and K. Talwar. On the geometry of differential privacy. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, 2010.
24. M. Hardt and K. Talwar. On the geometry of differential privacy. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5–8 June 2010*, pages 705–714, 2010.
25. M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1):1021–1032, 2010.
26. X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, and D. Srivastava. Dpt: Differentially private trajectory synthesis using hierarchical reference systems. *Proc. VLDB Endow.*, 8(11):1154–1165, July 2015.
27. X. He, A. Machanavajjhala, and B. Ding. Blowfish privacy: Tuning privacy-utility trade-offs using policies. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 1447–1458, New York, NY, USA, 2014. ACM.
28. H. Hu, J. Xu, S. T. On, J. Du, and J. K.-Y. Ng. Privacy-aware location data publishing. *ACM Trans. Database Syst.*, 35(3):18:1–18:42, July 2010.
29. G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. Differentially private event sequences over infinite streams. *Proc. VLDB Endow.*, 7(12):1155–1166, Aug. 2014.
30. D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 193–204, New York, NY, USA, 2011. ACM.
31. D. Kifer and A. Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. *ACM Trans. Database Syst.*, 39(1):3:1–3:36, Jan. 2014.
32. C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '10, 2010.
33. C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6–11, 2010, Indianapolis, Indiana, USA*, pages 123–134, 2010.
34. J. Lin, M. Benisch, N. Sadeh, J. Niu, J. Hong, B. Lu, and S. Guo. A comparative study of location-sharing privacy preferences in the united states and china. *Personal Ubiquitous Comput.*, 17(4):697–711, Apr. 2013.
35. C. Y. T. Ma, D. K. Y. Yau, N. K. Yip, and N. S. V. Rao. Privacy vulnerability of published anonymous mobility traces. *IEEE/ACM Trans. Netw.*, 21(3):720–733, June 2013.
36. A. Machanavajjhala and D. Kifer. Designing statistical privacy for your data. *Commun. ACM*, 58(3):58–67, Feb. 2015.
37. A. Machanavajjhala, D. Kifer, J. M. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7–12, 2008, Cancún, México*, pages 277–286, 2008.
38. A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), Mar. 2007.
39. D. J. Mir, S. Isaacman, R. Cáceres, M. Martonosi, and R. N. Wright. DP-WHERE: differentially private modeling of human mobility. In *Proceedings of the 2013 IEEE International Conference on Big Data, 6–9 October 2013, Santa Clara, CA, USA*, pages 580–588, 2013.
40. A. Monreale, G. L. Andrienko, N. V. Andrienko, F. Giannotti, D. Pedreschi, S. Rinzivillo, and S. Wrobel. Movement data anonymity through generalization. *Trans. Data Privacy*, 3(2):91–121, 2010.
41. OpenPaths. Openpaths privacy policy, 2012. https://openpaths.cc/privacy.

42. W. H. Qardaji, W. Yang, and N. Li. Differentially private grids for geospatial data. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8–12, 2013*, pages 757–768, 2013.
43. W. H. Qardaji, W. Yang, and N. Li. Understanding hierarchical methods for differentially private histograms. *PVLDB*, 6(14):1954–1965, 2013.
44. S. Sankararaman, G. Obozinski, M. I. Jordan, and E. Halperin. Genomic privacy and limits of individual detection in a pool. *Nature Genetics*, 2009.
45. P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti. Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences*, 111(37):13290–13294, 2014.
46. C. Song, Z. Qu, N. Blumm, and A.-L. Barabási. Limits of predictability in human mobility. *Science*, 2010.
47. X. Song, Q. Zhang, Y. Sekimoto, and R. Shibasaki. Prediction of human emergency behavior and their mobility following large-scale disaster. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 5–14, 2014.
48. D. Su, J. Cao, N. Li, E. Bertino, and H. Jin. Differentially private k-means clustering. *CoRR*, abs/1504.05998, 2015.
49. Y. Wang, S. Song, and K. Chaudhuri. Privacy-preserving analysis of correlated data. *CoRR*, abs/1603.03977, 2016.
50. A. Wesolowski, C. J. E. Metcalf, N. Eagle, J. Kombich, B. T. Grenfell, O. N. Bjørnstad, J. Lessler, A. J. Tatem, and C. O. Buckee. Quantifying seasonal population fluxes driving rubella transmission dynamics using mobile phone data. *Proceedings of the National Academy of Sciences*, 112(35):11114–11119, 2015.
51. R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei. Minimality attack in privacy preserving data publishing. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 543–554. VLDB Endowment, 2007.
52. X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1–6, 2010, Long Beach, California, USA*, pages 225–236, 2010.
53. Y. Xiao and L. Xiong. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12–6, 2015*, pages 1298–1309, 2015.
54. M. Xu, J. Wu, Y. Du, H. Wang, G. Qi, K. Hu, and Y. Xiao. Discovery of important crossroads in road network using massive taxi trajectories. *CoRR*, abs/1407.2506, 2014.
55. R. Yarovoy, F. Bonchi, L. V. S. Lakshmanan, and W. H. Wang. Anonymizing moving objects: how to hide a MOB in a crowd? In *EDBT 2009, 12th International Conference on Extending Database Technology, Saint Petersburg, Russia, March 24–26, 2009, Proceedings*, pages 72–83, 2009.
56. J. Zhang, X. Xiao, and X. Xie. Privtree: A differentially private algorithm for hierarchical decompositions. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 155–170, New York, NY, USA, 2016. ACM.