

## Article

# UDPLDP-Tree: Range Queries Under User-Distinguished Personalized Local Differential Privacy

Dongli Deng <sup>1</sup> , Sen Zhao <sup>1</sup> and Meixia Miao <sup>2,\*</sup>
<sup>1</sup> State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an 710126, China; dldeng@stu.xidian.edu.cn (D.D.); senzhao@stu.xidian.edu.cn (S.Z.)

<sup>2</sup> School of Cyberspace Security, Xi'an University of Posts and Telecommunications, Xi'an 710121, China

\* Correspondence: miaomiao1916@xupt.edu.cn

## Abstract

Local Differential Privacy (LDP) and its personalized variants (PLDP) have been widely used for privacy-preserving data analytics. However, existing schemes often enforce a uniform indistinguishability level among users, failing to accommodate the nuanced privacy needs of diverse individuals. To address this, we propose User-Distinguished Local Differential Privacy (UDPLDP), a novel framework that formalizes user-level distinguishability to support more flexible, non-uniform privacy budgets. Under this framework, we tackle the fundamental task of frequency range queries, namely UDPLDP-Tree, which overcomes the challenge due to limited user-level distinguishability, insufficient robustness in estimation under complex data distributions, and the assumption of uniform privacy requirements across different attributes in existing multi-dimensional schemes. To demonstrate the effectiveness, we conduct extensive experiments and the results show that UDPLDP-Tree reduces the mean squared error (MSE) by about 30–50% compared with a recent state-of-the-art baseline.

**Keywords:** local differential privacy; range query; user-distinguished protection; personalized privacy

## 1. Introduction

With the growing use of big data analytics, data collectors are increasingly interested in user data to improve their services. However, such practices pose potential risks of information leakage [1–4]. Even when explicit identifiers are removed, auxiliary data and linkage attacks can still re-identify individuals, as shown by the de-anonymization of the Netflix Prize dataset [5], the uniqueness of human mobility traces [6], and simple demographic linkage [7]. To address these risks, Local Differential Privacy (LDP) [8,9] has been introduced as a method to preserve privacy while still allowing for meaningful data analysis. Under LDP, users locally perturb their data before sending it to data collectors, ensuring privacy while maintaining the usability of the data. This approach has been adopted in several real-world applications, including Google Chrome browser [10], Apple [11] and Microsoft [12].

Despite its effectiveness, standard LDP typically employs a single, uniform privacy budget for all users, which may not accommodate diverse privacy preferences in practice [13]. To overcome the limitations of standard LDP, personalized Local Differential Privacy (PLDP) [14] enables each user to select a personal privacy budget, providing more flexibility without compromising local privacy guarantees. However, in PLDP, once a



Academic Editors: Jose de Vasconcelos, Carla Cordeiro and Hugo Barbosa

Received: 30 December 2025

Revised: 29 January 2026

Accepted: 5 February 2026

Published: 10 February 2026

**Copyright:** © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

user's privacy budget is individualized, the indistinguishability level between that user's input and the inputs of all other users remains the same. This diverges from real-world application scenarios, as shown in the following example.

**Example (Pairwise Distinguishability among Patients).** Consider three patients diagnosed with cancer, stomachache, and the common cold, respectively. In general, cancer patients require much stronger privacy protection than common-cold patients, whereas patients with stomachache have intermediate privacy needs. Ideally, the indistinguishability between stomachache and cancer patients should be stronger than that between stomachache and common-cold patients. However, existing PLDP schemes assign a single fixed privacy level to each user, which forces the indistinguishability between a stomachache patient and all other users to be the same. If the privacy level is set high enough to protect cancer patients, the stomachache patient also becomes difficult to distinguish from common-cold patients. This introduces excessive noise into the released data and reduces the accuracy of statistical analysis. Conversely, lowering the privacy level to improve accuracy and better distinguish stomachache from common-cold patients weakens the protection between stomachache and cancer patients, thereby potentially compromising the privacy of cancer patients.

To address these issues, we propose User-Distinguished Local Differential Privacy (UDPLDP), a novel framework that allows different users to have distinct levels of indistinguishability, reflecting their varying privacy requirements. The framework is also designed to support a wide range of data analysis tasks under UDPLDP. Among the various privacy-preserving data analysis tasks under UDPLDP, range queries are a fundamental type and have been extensively studied in recent years [15–21]. However, practical range queries face a series of challenges; therefore, applying the UDPLDP framework to practical range queries is not straightforward.

**Challenges.** First, existing work focuses on applying the same budgets over the entire range domain. However, different users may need different privacy budgets for a specific range in real-world applications [15–21]. Second, current methods often assume data is uniformly [15–20] or linearly distributed [21] within a range. However, this fails to capture complex non-linear patterns and peaks in real data, leading to large estimation errors. Third, existing multi-dimensional schemes usually assign the same privacy budget to all attributes [16–21]. In practice, different attributes (like age vs. medical condition) have different sensitivities, and a uniform budget either risks privacy or hurts data accuracy.

In this paper, we address the aforementioned challenges under the UDPLDP framework and develop a more practical range query scheme. Our main contributions are summarized as follows:

- We propose the UDPLDP framework, which formalizes user-level distinguishability to provide a flexible theoretical foundation for privacy-preserving data analysis under non-uniform privacy budgets.
- We develop a practical range query scheme that addresses the aforementioned challenges, specifically by handling heterogeneous privacy requirements, capturing complex data distributions, and managing attribute-specific sensitivities.
- We conduct extensive experiments on eight datasets (four synthetic and four real-world) and compare with representative baselines, demonstrating that our proposed scheme achieves consistently lower MSE under the same privacy budgets.

The remainder of this paper is organized as follows: Section 2 provides the preliminaries. Section 3 defines the UDPLDP framework. In Section 4, we address the challenges of one-dimensional range queries by proposing the UDPLDP-Tree. Section 5 extends these methods to support multi-dimensional cases. Section 6 presents the experimental results, while Section 7 reviews related work. Finally, Section 8 concludes the paper.

## 2. Preliminaries

In this section, we present the necessary preliminaries of our scheme.

### 2.1. Local Differential Privacy

**Definition 1** ( $\epsilon$ -Local Differential Privacy [8,9]). A perturbation mechanism  $\mathcal{M}$  satisfies  $\epsilon$ -LDP if and only if, for any two input data  $v, v' \in D$  and any output  $y \in \text{Range}(\mathcal{M})$ , we have

$$\Pr[\mathcal{M}(v) = y] \leq e^\epsilon \Pr[\mathcal{M}(v') = y]. \quad (1)$$

In Equation (1), the parameter  $\epsilon$  represents the privacy budget, which bounds the ratio of the probability distributions of the mechanism's output for any two adjacent inputs  $v, v' \in D$ . Here,  $\text{Range}(\mathcal{M})$  denotes the set of all possible outcomes, and the probability is evaluated on the internal randomness of  $\mathcal{M}$ . The magnitude of  $\epsilon$  inherently characterizes the trade-off between privacy and utility: a smaller  $\epsilon$  signifies a tighter privacy bound by necessitating higher levels of perturbation, which consequently degrades the utility of the data. In contrast, a larger  $\epsilon$  relaxes the privacy constraint, facilitating a higher accuracy of the estimation at the expense of reduced privacy protection.

### 2.2. Frequency Oracle

Frequency oracles (FOs) [22] are fundamental primitives in LDP-based data analysis to estimate categorical frequency distributions. Among various FO protocols, Optimized Unary Encoding (OUE) [23] is widely utilized for its unbiased estimation and its variance that remains independent of the domain size. The OUE protocol protects these values through the following three strategies.

- Encoding. Given a domain  $D$ , the user  $i$  represents their private value  $v_i$  as a one-hot vector  $x_i \in \{0, 1\}^{|D|}$ , where only the position corresponding to  $v_i$  is 1.
- Perturbation. Each bit of  $x_i$  is perturbed independently with probabilities  $p = \frac{1}{2}$  and  $q = \frac{1}{e^\epsilon + 1}$ . A 1-bit is kept with probability  $p$  and flipped otherwise, while a 0-bit is flipped to 1 with probability  $q$  and kept otherwise. The user then uploads the perturbed vector  $x'_i$ .
- Aggregation and Estimation. After receiving the perturbed vectors  $\{x'_i\}_{i=1}^N$ , the server counts the number of reported 1's for each index  $j$ , i.e.,  $\text{count}[j] = \sum_{i=1}^N x'_i[j]$ . The unbiased frequency estimator is given by  $\hat{f}[j] = \frac{\text{count}[j] - Nq}{N(p-q)}$ , where  $p = \frac{1}{2}$  and  $q = \frac{1}{e^\epsilon + 1}$ . The variance of OUE is

$$\text{Var}_{\text{OUE}}(\epsilon) = \frac{4e^\epsilon}{N(e^\epsilon - 1)^2}, \quad (2)$$

where  $\text{Var}$  is independent of the domain size  $|D|$ .

### 2.3. B-Spline Basis Functions

A B-spline curve [24] is defined as a parametric piecewise polynomial curve. Given a non-decreasing knot vector  $T = \{t_0, t_1, \dots, t_m\}$  defined in the parameter domain  $[a, b]$ , and a set of control points  $\{\mathbf{P}_i\}_{i=0}^n$ , the B-spline curve  $\mathbf{C}(t)$  is expressed as

$$\mathbf{C}(t) = \sum_{i=0}^n N_{i,p}(t) \mathbf{P}_i, \quad a \leq t \leq b \quad (3)$$

where  $N_{i,p}(t)$  denotes the  $i$ -th B-spline basis function of degree  $p$ . The relationship between the number of knots ( $m$ ), the degree ( $p$ ), and the number of control points ( $n$ ) is given

by  $m = n + p + 1$ . The basis functions are defined recursively using the Cox–de Boor formula [24]. For degree 0, we have

$$N_{i,0}(t) = \begin{cases} 1, & t_i \leq t < t_{i+1} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

and for any degree  $p \geq 1$ ,

$$N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t), \quad (5)$$

where terms with a zero denominator are defined to be zero.

### 3. UDPLDP: User-Distinguished Local Differential Privacy

In this section, we introduce UDPLDP, a privacy definition that generalizes personalized Local Differential Privacy by incorporating flexible pairwise indistinguishability constraints.

#### 3.1. Personalized Local Differential Privacy

We revisit PLDP and its inherent limitation, which motivates UDPLDP. PLDP generalizes standard Local Differential Privacy by allowing users to adopt different privacy budgets.

Personalization is motivated by heterogeneous privacy sensitivities: only a small fraction of users may need very strong protection (heavy noise), while many others accept milder privacy for better utility. In practice, a uniform worst-case privacy level often yields worst-case accuracy and unstable outputs, so personalization avoids forcing the entire system to operate at the utility level dictated by a small high-sensitivity subgroup.

From the perspective of its design objective, PLDP extends standard Local Differential Privacy by allowing different users to adopt different privacy budgets, thereby providing a more flexible privacy model.

**Definition 2** (Personalized Local Differential Privacy ( $\mathcal{E}$ -PLDP) [14]). *Given a set of users  $\mathcal{U}$ , where each user  $u \in \mathcal{U}$  is associated with a privacy budget  $\varepsilon_u \in \mathcal{E}$ , a randomized mechanism  $M$  satisfies  $\mathcal{E}$ -PLDP if and only if, for any user  $u$ , any two possible input values  $x, x' \in D$ , and any output  $y \in \text{Range}(M)$ , the following condition holds:*

$$\frac{\Pr[M(x) = y]}{\Pr[M(x') = y]} \leq e^{\varepsilon_u}. \quad (6)$$

Under PLDP, the privacy budget  $\varepsilon_u$  bounds the distinguishability between any two possible inputs of user  $u$ , regardless of the identities or privacy requirements of other users. This design enables user-level personalization while retaining the simplicity of a single-user privacy constraint.

#### 3.2. Limitation of PLDP: Uniform Pairwise Indistinguishability

Despite enabling individualized budgets, PLDP is structurally limited because a single  $\varepsilon_u$  imposes a uniform privacy semantics for user  $u$ : it controls indistinguishability involving  $u$  with the same strength across all counterparts, and thus cannot express pair-dependent requirements such as making  $u$  much harder to distinguish from users in one group than from users in another. In short, PLDP cannot specify different indistinguishability strengths across user pairs  $(u, v)$ .

This limitation becomes salient in practice when users' privacy levels are public or inferable and aggregates are computed from mixed-accuracy reports. For example, consider an anonymous workplace survey with a binary question such as whether an employee took mental-health leave this month. Alice selects high privacy with a small  $\epsilon_A$  and heavy noise, while Bob selects low privacy with a large  $\epsilon_B$  and light noise. Once a team-level count is released, Bob's near-accurate report can act as an anchor: an adversary can largely account for Bob's contribution and thereby infer Alice's status with much higher confidence than what assigning  $\epsilon_A$  alone intends to capture. This leakage does not require any correlation between users' data; it stems from public privacy levels and mixed-noise aggregation.

These observations motivate UDPLDP: instead of only assigning a single  $\epsilon_u$  per user, UDPLDP explicitly models user-distinguished (individual-to-individual) indistinguishability so that protection can be specified and optimized at the granularity of user pairs.

### 3.3. User-Distinguished Local Differential Privacy

To address the above limitation, we propose UDPLDP, which generalizes PLDP by introducing user-dependent pairwise indistinguishability through a privacy budget function.

Let the data domain be  $D$ , and each user  $u$  is associated with a privacy budget  $\epsilon_u \in \mathcal{E}$ . In the local model, the private input is the user's value  $x \in D$ , while the budget  $\epsilon_u$  is treated as public metadata (i.e., we do not aim to hide which privacy level a user chooses). Accordingly, we view a mechanism invocation as parameterized by the budget and denote it by  $M(x; \epsilon)$ . We say two invocations  $(x; \epsilon_u)$  and  $(x'; \epsilon_{u'})$  are adjacent if they correspond to two possible private values  $x, x' \in D$  (potentially under different public budgets). This adjacency specifies the neighboring inputs whose distinguishability we aim to bound.

**Definition 3** (User-Distinguished Local Differential Privacy ( $\mathcal{E}$ -UDPLDP)). *Given a set of users  $\mathcal{U}$ , where each user  $u \in \mathcal{U}$  is assigned a privacy budget  $\epsilon_u \in \mathcal{E}$ , a randomized mechanism  $M$  satisfies  $\mathcal{E}$ -UDPLDP if and only if, for any two users  $u, u' \in \mathcal{U}$ , any two input values  $x, x' \in D$ , and any output  $y \in \text{Range}(M)$ , the following condition holds:*

$$\frac{\Pr[M(x; \epsilon_u) = y]}{\Pr[M(x'; \epsilon_{u'}) = y]} \leq \exp(\min\{\epsilon_u, \epsilon_{u'}\}). \quad (7)$$

*That is, the pairwise indistinguishability is governed by the stricter privacy requirement. When  $u = u'$ , the definition reduces to the standard PLDP/LDP-style guarantee  $\Pr[M(x; \epsilon_u) = y] / \Pr[M(x'; \epsilon_u) = y] \leq e^{\epsilon_u}$ .*

Unlike PLDP, which is inherently user-wise (the privacy loss is not parameterized by which other user it is compared against), UDPLDP explicitly constrains cross-user distinguishability under heterogeneous public budgets via the pairwise bound  $\min\{\epsilon_u, \epsilon_{u'}\}$ . From a generalized-DP viewpoint, UDPLDP can also be interpreted as a lightweight policy/relation-based instantiation in the local model, where the policy strength between a user pair  $(u, u')$  is induced by  $\min\{\epsilon_u, \epsilon_{u'}\}$ .

## 4. UDPLDP-Tree: Range Query Processing over UDPLDP Framework

In this section, we introduce UDPLDP-Tree, a private tree structure built upon the  $\mathcal{E}$ -UDPLDP mechanism presented in Section 3. By integrating piecewise curve-fitting techniques, UDPLDP-Tree is specifically tailored to support range-frequency queries while strictly adhering to user-level personalized privacy budgets.

#### 4.1. Problem Definition

We consider  $N$  users, where each user  $u_i$  holds a  $d$ -dimensional private record  $\mathbf{v}_i = (v_i^{(1)}, v_i^{(2)}, \dots, v_i^{(d)})$ . Here,  $v_i^{(j)} \in D_j$  denotes the value of attribute  $A_j$  for user  $u_i$ , and  $D_j$  is the public domain of  $A_j$ .

Let  $\Phi \subseteq \{A_j \mid 1 \leq j \leq d\}$  be a set of queried attributes with  $\lambda = |\Phi|$ . For each  $A_j \in \Phi$ , the analyst specifies a one-dimensional range  $[l_j, r_j] \subseteq D_j$ , and we write  $\ell = (l_j)_{A_j \in \Phi}$  and  $\mathbf{r} = (r_j)_{A_j \in \Phi}$ , where  $\ell$  and  $\mathbf{r}$  collect the lower and upper bounds for all attributes in  $\Phi$ . Denote by  $\mathbf{v}_i^\Phi$  the projection of  $\mathbf{v}_i$  onto the attribute set  $\Phi$ . The corresponding  $\lambda$ -dimensional hyper-rectangle is  $[\ell, \mathbf{r}] = \{\mathbf{x} \mid l_j \leq x^{(j)} \leq r_j, \forall A_j \in \Phi\}$ . The range query asks for the fraction of user records that fall into this multi-dimensional interval:

$$Q([\ell, \mathbf{r}]) = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(\mathbf{v}_i^\Phi \in [\ell, \mathbf{r}]), \quad (8)$$

where  $\mathbb{I}(\cdot)$  is the indicator function that returns 1 if its argument is true and 0 otherwise.

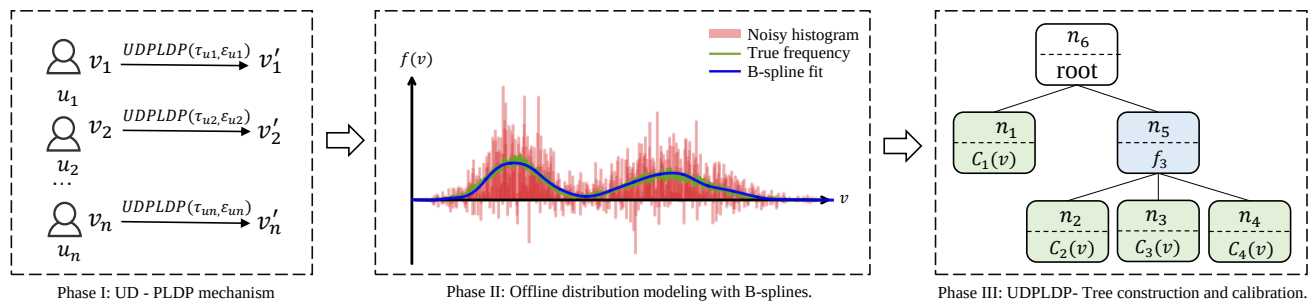
#### 4.2. Technical Overview

UDPLDP-Tree enables accurate range-frequency queries under heterogeneous local privacy, where (i) users may have different privacy sensitivities over sub-ranges and (ii) real distributions are often skewed and non-linear. These factors make existing LDP range query mechanisms inefficient in budget usage and inaccurate in estimation. We summarize our key ideas as follows.

- **Range-wise heterogeneous budgets.** We extend UDPLDP to range queries by letting each user specify a privacy range  $\tau_u \subseteq D$ , so privacy is enforced only on overlapping sensitive ranges. Here,  $\tau_u$  is treated as public metadata specified by the user together with  $\varepsilon_u$  before perturbation; we do not aim to hide  $\tau_u$  itself. Intuitively,  $\tau_u$  captures the finest exposure granularity the user is willing to accept, such as allowing disclosure at the district level but not at the street level. On top of this model, we design a multi-level unary oracle (Section 4.4.2) following OUE with  $a_p = \frac{1}{2}$ , and calibrate  $\{b_p\}$  by minimizing the MSE upper bound subject to UDPLDP feasibility constraints (Equations (15)–(19)), yielding an unbiased and variance-optimized frequency estimator.
- **Skewed and non-linear distributions.** Pure tree-based LDP methods suffer on small intervals: noise can dominate low counts, and leaf nodes often drive range query error. Merging nodes reduces noise but introduces bias under uniform/linear assumptions and misses sharp peaks or heavy tails. We therefore use a hybrid strategy: (i) fit a cubic B-spline from a dedicated user subset (Section 4.5), and (ii) build a UDPLDP-Tree whose internal nodes store UDPLDP-protected aggregates while boundary intervals are answered by integrating the stored cubics. This combines low-variance aggregation with expressive boundary modeling.

#### 4.3. Overall Workflow

We now present the overall workflow of our framework in the one-dimensional setting. Our objective is to answer range-frequency queries over a discrete domain  $D$  under heterogeneous, user-specific local privacy requirements, while maintaining high accuracy for non-linear data distributions. As illustrated in Figure 1, the framework consists of three consecutive phases, followed by a query answering procedure.



**Figure 1.** Workflow of our method.

- Phase I: UDPLDP-based local frequency reporting.** In this phase, we instantiate the UDPLDP framework to support local frequency estimation under range-wise heterogeneous privacy budgets. Each user specifies a sensitive range  $\tau_u \subseteq D$  and a privacy level  $\epsilon_u$ . Both  $(\tau_u, \epsilon_u)$  are user-declared public personalization metadata used to parameterize the mechanism, rather than private values to be protected. Based on these specifications, we design a multi-level unary frequency oracle under the  $\mathcal{E}$ -UDPLDP model. The oracle is built upon OUE and is calibrated via constrained optimization to achieve unbiased estimation with minimized variance. Details are given in Section 4.4.
- Phase II: Offline distribution modeling with B-splines.** This phase fits a smooth approximation of the global frequency distribution over  $D$  to capture non-linear patterns. At the start of each collection round, we perform a one-time random user split independent of private values: each user joins  $U_{\text{model}}$  with probability  $\rho$  and  $U_{\text{tree}}$  otherwise, where  $\rho$  is fixed. This disjoint split ensures each user participates in at most one interactive stage, so the modeling step and subsequent tree calibration do not incur sequential composition. Using only reports from  $U_{\text{model}}$ , we build a private histogram and fit a cubic B-spline. The fitted spline is treated as an offline shared prior and is reused only via post-processing in later steps, without any adaptive tuning based on outputs from  $U_{\text{tree}}$ . Details are provided in Section 4.5.
- Phase III: UDPLDP-Tree construction and calibration.** In this phase, we build a hierarchical index on top of the fitted B-spline model to support efficient range queries. Each spline segment is treated as a leaf interval, and a balanced binary tree is constructed bottom-up. The tree is then pruned to minimize the overall estimation error. Using the reserved user set  $U_{\text{tree}}$ , we apply the UDPLDP mechanism to estimate frequencies for selected internal nodes. The final structure is a pruned UDPLDP-Tree whose internal nodes store noisy frequency estimates and whose leaves store local cubic polynomials. Details are given in Section 4.6.

Given a range query  $Q([l, r])$ , the UDPLDP-Tree decomposes the query range into a set of disjoint node intervals. Fully covered internal nodes contribute their calibrated frequency estimates, while at most two boundary leaf nodes are evaluated by integrating their local cubic polynomials over the corresponding sub-intervals. The query answer is obtained by summing these contributions, achieving accurate range-frequency estimation under heterogeneous local privacy constraints.

#### 4.4. Local Frequency Reporting Under UDPLDP

In the overall workflow, local frequency estimation serves as the basic primitive for answering range-frequency queries. This subsection focuses on designing such a primitive under range-wise heterogeneous privacy constraints. We now adapt the UDPLDP framework to the range query setting, where users may only require privacy protection over localized intervals of the domain. This subsection first formalizes a range-aware UDPLDP

model and then presents a unary frequency oracle that satisfies this model. In what follows, we first specify how users declare their sensitive ranges and how indistinguishability is enforced within overlaps, and then instantiate a concrete oracle whose parameters will be optimized for accuracy under these constraints.

#### 4.4.1. UDPLDP with User-Specified Privacy Ranges

We instantiate the UDPLDP framework introduced in Section 3 to the range query setting by restricting where pairwise indistinguishability must be enforced. Since range queries only depend on values within queried intervals, users may require privacy protection only over localized regions of the domain. Accordingly, each user  $u$  specifies a pair  $(\tau_u, \varepsilon_u)$ , where  $\tau_u \subseteq D$  denotes the privacy range in which the user's value is considered sensitive, and  $\varepsilon_u$  is the corresponding privacy budget selected from a finite, ordered set  $\mathcal{E} = \{\varepsilon_1, \dots, \varepsilon_m\}$ , with smaller values indicating stronger protection; values outside  $\tau_u$  are treated as non-sensitive for user  $u$ . In our personalization setting,  $(\tau_u, \varepsilon_u)$  are explicitly declared and treated as public mechanism parameters (similar to a user's chosen privacy level), while the private value is still  $v_u \in D$ . Operationally, a user chooses  $\tau_u$  as the coarsest region she is willing to reveal about her value (e.g., "within this district"), so reporting outside  $\tau_u$  being exact only discloses that coarse membership. This range-aware specification aligns the privacy requirement with the semantics of range queries, while still allowing heterogeneous budgets across users.

As in the general UDPLDP model, privacy guarantees between users with heterogeneous budgets are governed by a symmetric, monotone non-decreasing pairwise budget function  $r : \mathcal{E} \times \mathcal{E} \rightarrow \mathbb{R}_{>0}$ . Under range query semantics, UDPLDP is enforced only when both inputs fall within the users' sensitive regions: for any users  $u, u'$ , any  $x \in \tau_u$  and  $x' \in \tau_{u'}$  with  $x, x' \in \tau_u \cap \tau_{u'}$ , and any output  $y \in \text{Range}(\mathcal{M})$ , the perturbation mechanism  $\mathcal{M}$  satisfies

$$\frac{\Pr[\mathcal{M}(x) = y]}{\Pr[\mathcal{M}(x') = y]} \leq \exp(\min\{\varepsilon_u, \varepsilon_{u'}\}). \quad (9)$$

When all users specify the full domain ( $\tau_u = D$ ) and share a common privacy budget, this formulation reduces to standard  $\varepsilon$ -LDP. With the range-aware UDPLDP notion in place, we next construct a concrete frequency oracle that (i) perturbs only sensitive coordinates, and (ii) satisfies the above pairwise indistinguishability over overlapping privacy ranges.

#### 4.4.2. Frequency Estimation Under UDPLDP

Given the above range-aware UDPLDP model, we now construct a frequency oracle that supports multiple privacy levels under the  $\mathcal{E}$ -UDPLDP model, and formulate an optimization problem for choosing its parameters. At a high level, the oracle follows the OUE-style unary encoding and bitwise randomized response, but applies perturbation only within each user's declared privacy range. The overall procedure of the proposed frequency oracle, including user-side perturbation and server-side aggregation, is described in Algorithm 1.

**Algorithm 1:** Frequency estimation under  $\mathcal{E}$ -UDPLDP.

---

**Input:** User set  $\mathcal{U}$ , domain  $D$ , and user specifications  $\{(v_u, \tau_u, \varepsilon_u)\}_{u \in \mathcal{U}}$ .  
**Output:** Estimated frequencies  $\{\hat{f}_v\}_{v \in D}$ .

- 1 **Parameter setup.**
- 2 Set  $a_p \leftarrow \frac{1}{2}$  for all privacy levels  $\varepsilon_p \in \mathcal{E}$ .
- 3 Compute the level-wise coverage statistics (e.g.,  $\{n_p\}$  in Equation (19)) from the public metadata  $\{(\tau_u, \varepsilon_u)\}$  before perturbation.
- 4 Compute  $\{b_p\}_{p=1}^m$  by solving the optimization problem in (19) under the constraints in (15).
- 5 **User-side perturbation.**
- 6 **foreach** user  $u \in \mathcal{U}$  *in parallel* **do**
- 7     Let  $p$  be such that  $\varepsilon_u = \varepsilon_p$ .
- 8     Encode  $v_u$  into a one-hot vector  $x_u \in \{0, 1\}^{|D|}$ .
- 9     **foreach**  $v \in D$  **do**
- 10         **if**  $v \in \tau_u$  **then**
- 11             **if**  $v = v_u$  **then**
- 12                 sample  $y_u[v] \sim \text{Bernoulli}(a_p)$
- 13             **else**
- 14                 sample  $y_u[v] \sim \text{Bernoulli}(b_p)$
- 15         **else**
- 16             set  $y_u[v] \leftarrow x_u[v]$
- 17     User  $u$  sends  $(p, y_u)$  to the collector.
- 18 **Collector-side aggregation.**
- 19 Initialize  $\text{count}_p[v] \leftarrow 0$  and  $n_p[v] \leftarrow 0$  for all  $p$  and  $v \in D$ .
- 20 **foreach** received report  $(p, y_u)$  **do**
- 21     **foreach**  $v \in D$  **do**
- 22         **if**  $v \in \tau_u$  **then**
- 23              $\text{count}_p[v] \leftarrow \text{count}_p[v] + y_u[v]$
- 24              $n_p[v] \leftarrow n_p[v] + 1$
- 25 **foreach**  $v \in D$  **do**
- 26     compute  $\hat{f}_v$  using the estimator in (16).

---

To make the design and its analysis explicit, we next elaborate the four components underlying Algorithm 1: encoding, range-aware perturbation with heterogeneous budgets, aggregation, and parameter selection.

- **Encoding.** Each user  $u$  with private value  $v_u \in D$  encodes it as a one-hot vector  $x_u \in \{0, 1\}^{|D|}$ . The coordinate corresponding to  $v_u$  is set to 1, and all other coordinates are set to 0. This encoding is identical to that used in OUE and is shared by all privacy levels. This common representation allows us to isolate the effect of heterogeneous privacy budgets to the perturbation parameters alone.
- **Perturbation under  $\mathcal{E}$ -UDPLDP.** Building on the above encoding, we now describe how user reports are privatized to satisfy the range-aware  $\mathcal{E}$ -UDPLDP constraints. Users are partitioned into  $m$  privacy levels  $\mathcal{E} = \{\varepsilon_1, \dots, \varepsilon_m\}$  according to their privacy budgets. For a level- $p$  user  $u$ , only coordinates corresponding to values in the privacy range  $\tau_u$  are treated as sensitive and will be perturbed; all other coordinates are left unchanged and are regarded as non-sensitive in our model. For each sensitive

position  $k$  of a level- $p$  user, the mechanism perturbs the bit  $x[k] \in \{0, 1\}$  using a bitwise randomized response with parameters  $(a_p, b_p)$ :

$$\begin{aligned}\Pr[y[k] = 1 \mid x[k], \varepsilon_p] &= a_p x[k] + b_p (1 - x[k]), \\ \Pr[y[k] = 0 \mid x[k], \varepsilon_p] &= (1 - a_p) x[k] + (1 - b_p) (1 - x[k]).\end{aligned}\quad (10)$$

Thus, if  $x[k] = 1$ , then  $y[k] = 1$  (resp. 0) with probability  $a_p$  (resp.  $1 - a_p$ ), and if  $x[k] = 0$ , then  $y[k] = 1$  (resp. 0) with probability  $b_p$  (resp.  $1 - b_p$ ). Throughout this paper, we adopt an OUE-style parametrization and fix  $a_p = \frac{1}{2}$  and  $0 < b_p < \frac{1}{2}$  for all  $p \in \{1, \dots, m\}$ . We next verify that this bitwise mechanism indeed satisfies  $\mathcal{E}$ -UDPLDP when comparing users from potentially different privacy levels.

Let users  $u$  and  $u'$  belong to privacy levels  $p$  and  $q$ , respectively. Since each bit of the unary vector is perturbed independently, for any output vector  $y$ , we have

$$\frac{\Pr[y \mid x]}{\Pr[y \mid x']} = \frac{\prod_{k=1}^{|D|} \Pr(y[k] \mid x[k], \varepsilon_p)}{\prod_{k=1}^{|D|} \Pr(y[k] \mid x'[k], \varepsilon_q)}.\quad (11)$$

Because  $x$  and  $x'$  differ only on two coordinates, say positions  $r$  and  $s$ , all remaining factors in the numerator and denominator are identical and cancel out. When writing  $x[r] = 1$ ,  $x[s] = 0$  and  $x'[r] = 0$ ,  $x'[s] = 1$ , we obtain

$$\frac{\Pr[y \mid x]}{\Pr[y \mid x']} = \frac{\Pr(y[r] \mid x[r] = 1, \varepsilon_p) \Pr(y[s] \mid x[s] = 0, \varepsilon_p)}{\Pr(y[r] \mid x'[r] = 0, \varepsilon_q) \Pr(y[s] \mid x'[s] = 1, \varepsilon_q)}.\quad (12)$$

Let  $y = (y[r], y[s]) \in \{0, 1\}^2$  be the corresponding output. By (10) and independence across coordinates, enumerating the four possible outputs  $(y[r], y[s])$  and substituting  $a_p = a_q = \frac{1}{2}$  yields

$$\frac{\Pr[y \mid x]}{\Pr[y \mid x']} = \begin{cases} \frac{1 - b_p}{1 - b_q}, & y = (0, 0), \\ \frac{b_p}{b_q}, & y = (1, 1), \\ \frac{b_p}{1 - b_q}, & y = (0, 1), \\ \frac{1 - b_p}{b_q}, & y = (1, 0). \end{cases}\quad (13)$$

Since  $0 < b_p < \frac{1}{2}$  implies  $b_p < 1 - b_p$ , we have

$$\frac{b_p}{1 - b_q} < \frac{1 - b_p}{1 - b_q} < \frac{1 - b_p}{b_q}, \quad \frac{b_p}{b_q} < \frac{1 - b_p}{b_q},\quad (14)$$

so the maximum ratio is attained at  $y = (1, 0)$  and equals  $(1 - b_p)/b_q$ . According to the definition of  $\mathcal{E}$ -UDPLDP, this ratio must satisfy

$$\frac{1 - b_p}{b_q} \leq \exp(\min(\varepsilon_p, \varepsilon_q)), \quad \forall p, q \in \{1, \dots, m\}.\quad (15)$$

This family of inequalities characterizes the feasible region of the level-wise false-positive probabilities  $\{b_p\}$  under the  $\mathcal{E}$ -UDPLDP model and will serve as the privacy constraint in our subsequent parameter optimization. Having established the privacy constraint, we now turn to how the collector aggregates these heterogeneous, range-aware reports into a global frequency estimate.

- **Aggregation.** Given the perturbed unary reports, the collector performs level-wise aggregation to account for heterogeneous privacy parameters. On the collector side, users are grouped by their privacy level. Fix a value  $v \in D$ . For each level  $p$ , let  $n_p[v]$  denote the number of level- $p$  users whose privacy range  $\tau_u$  contains  $v$ , i.e., the users that treat the coordinate corresponding to  $v$  as sensitive and apply randomized response on this bit. Equivalently,  $n_p[v] = |\{u : \varepsilon_u = \varepsilon_p \wedge v \in \tau_u\}|$ , which depends only on the public metadata  $\{(\tau_u, \varepsilon_u)\}$  and can be computed before collecting perturbed values. We also define  $N[v] = \sum_{p=1}^m n_p[v]$  as the total number of users who randomize the bit of  $v$ . For each level  $p$ , the collector records  $\text{count}_p[v]$ , the number of level- $p$  users whose report has bit 1 at the position corresponding to  $v$ . Given  $(a_p, b_p)$ , we decode these statistics using an OUE-style linear estimator:

$$\hat{f}_v = \sum_{p=1}^m \frac{\text{count}_p[v] - n_p[v] b_p}{N[v] (a_p - b_p)}. \quad (16)$$

Since only users with  $v \in \tau_u$  randomize this bit, we have

$$\mathbb{E}[\text{count}_p[v]] = n_p[v] (f_v^p a_p + (1 - f_v^p) b_p), \quad (17)$$

where  $f_v^p$  is the true frequency of value  $v$  among level- $p$  users who treat  $v$  as sensitive. Substituting into (16) gives  $\mathbb{E}[\hat{f}_v] = f_v$ , so the estimator is unbiased for the underlying global frequency  $f_v$ . To guide parameter selection, we next quantify the resulting estimation error under heterogeneous levels.

- **Estimation error.** We analyze the MSE of the above unbiased estimator as a function of the level-wise parameters, which will later serve as the optimization objective. Because user groups at different levels are disjoint,  $\text{count}_p[v]$  are independent across  $p$ , and the MSE of  $\hat{f}_v$  satisfies

$$\text{MSE}(\hat{f}_v) = \sum_{p=1}^m \frac{\text{Var}(\text{count}_p[v])}{N[v]^2 (a_p - b_p)^2} \leq \sum_{p=1}^m \frac{n_p[v] b_p (1 - b_p)}{N[v]^2 (a_p - b_p)^2}, \quad (18)$$

where the inequality follows from the standard variance bound of OUE under  $a_p = 1/2$ . Finally, we instantiate the oracle by choosing  $\{b_p\}$  to minimize this bound subject to the UDPLDP feasibility constraints derived above.

- **Parameter optimization.** We minimize the MSE upper bound in (18) over  $\mathbf{b} = (b_1, \dots, b_m)$  under the feasibility constraints in (15):

$$\min_{\mathbf{b}} \sum_{p=1}^m \frac{n_p b_p (1 - b_p)}{N^2 (a_p - b_p)^2} \quad \text{s.t.} \quad \frac{1 - b_p}{b_q} \leq \exp(\min(\varepsilon_p, \varepsilon_q)), \quad \forall p, q. \quad (19)$$

The collector can obtain all quantities needed above: since each user declares  $(\tau_u, \varepsilon_u)$  as public metadata, it can precompute  $n_p[v] = |\{u : \varepsilon_u = \varepsilon_p \wedge v \in \tau_u\}|$  before reading any privatized bits. For a single parameter set shared by all  $v \in D$ , we aggregate  $n_p = \frac{1}{|D|} \sum_{v \in D} n_p[v]$  and  $N = \sum_{p=1}^m n_p$ , i.e., we minimize the average MSE bound over  $v \in D$ .

Equation (19) is convex in  $\{b_p\}$  under our OUE-style parametrization. When  $a_p = \frac{1}{2}$ , each separable term  $g_p(b_p) = \frac{n_p b_p (1 - b_p)}{N^2 (a_p - b_p)^2}$  is strictly convex on  $0 < b_p < \frac{1}{2}$  since  $\frac{d^2}{db_p^2} \left( \frac{b_p (1 - b_p)}{(\frac{1}{2} - b_p)^2} \right) = \frac{2}{(\frac{1}{2} - b_p)^4} > 0$ . The UDPLDP constraints rewrite to linear inequalities  $1 - b_p \leq \exp(\min(\varepsilon_p, \varepsilon_q)) b_q$  for all  $p, q$ ; together with  $0 < b_p < \frac{1}{2}$ , the feasible set is convex. Hence, a unique global optimum exists. We solve it with SLSQP (SciPy), which attains the global optimum due to convexity.

We solve Equation (19) once per collection round; if  $\{(\tau_u, \varepsilon_u)\}$  changes, we recompute  $\{n_p\}$  and re-solve in the next round. Privacy depends only on the feasibility constraints in Equation (15), so errors in  $\{n_p\}$  affect utility (the bound) but not privacy. When metadata statistics are approximate (e.g., subsampling), we use conservative/smoothed  $\{n_p\}$  to avoid over-optimistic calibration.

The solution provides level-wise probabilities that satisfy heterogeneous budgets and minimize the unary MSE bound under  $\mathcal{E}$ -UDPLDP; we then plug them into Algorithm 1 to finalize the range-aware frequency primitive.

In summary, this subsection provides a range-aware, unbiased unary frequency oracle that supports heterogeneous local privacy budgets. This oracle serves as the fundamental building block for both the spline-based distribution modeling and the UDPLDP-Tree construction in the subsequent sections. In particular, the next sections will invoke this oracle as a modular component, while focusing on how to exploit its estimates to model the distribution and to answer range queries efficiently.

#### 4.5. Distribution Modeling via B-Splines

This subsection builds on the range-aware unary frequency oracle in Section 4.4.2 and uses its multi-level estimates to model the underlying one-dimensional distribution. The  $\mathcal{E}$ -UDPLDP mechanism yields, for each domain value  $v \in D$ , an unbiased but noisy frequency estimate  $\hat{f}_v$ , which collectively forms a noisy histogram of the true distribution over  $D$ . However, directly answering range queries on this noisy histogram would inherit its local perturbation noise and lead to unstable estimates, especially near query boundaries. To mitigate this issue, we smooth these noisy point-wise estimates into a continuous representation that suppresses local fluctuations while supporting accurate integration over arbitrary query ranges.

##### 4.5.1. Histogram Representation and Spline Fitting Objective

Before spline fitting, we show unary modeling incurs no extra privacy loss by splitting users into disjoint subsets. We split once per collection round, independent of private values: each user joins  $U_{\text{model}}$  with probability  $\rho$  (otherwise  $U_{\text{tree}}$ ), where  $\rho$  is fixed. Users in  $U_{\text{model}}$  run the unary  $\mathcal{E}$ -UDPLDP mechanism once to fit a global model, while  $U_{\text{tree}}$  is reserved for UDPLDP-Tree construction (Section 4.6). The spline is fitted only from perturbed reports in  $U_{\text{model}}$  and never uses information from  $U_{\text{tree}}$ , keeping modeling and calibration independent. Since the sets are disjoint, each user participates in only one phase, so there is no sequential composition across phases for any user. The modeling step is pure post-processing of locally perturbed reports, hence adds no privacy loss.

With  $U_{\text{model}}$  fixed, we now turn the unary outputs from  $U_{\text{model}}$  into a set of data points and formulate the spline fitting objective. Specifically, running the unary  $\mathcal{E}$ -UDPLDP estimator on users in  $U_{\text{model}}$  produces a noisy frequency estimate  $\hat{f}_{v_k}$  for each domain value  $v_k \in D$ . We assemble these estimates into planar points

$$Q_k = (v_k, \hat{f}_{v_k})^T, \quad k = 0, \dots, |D| - 1, \quad (20)$$

which can be viewed as a noisy histogram of the underlying distribution over  $D$ .

Our goal is to approximate these points with a smooth curve that captures the global distribution trend while suppressing local noise. To this end, we model the distribution as a degree- $s$  planar B-spline curve

$$C(t) = \sum_{i=0}^{n_c} N_{i,s}(t) P_i, \quad t \in [0, 1], \quad (21)$$

where  $P_i \in \mathbb{R}^2$  are control points and  $N_{i,s}(\cdot)$  are B-spline basis functions of degree  $s$ . The curve is clamped at both endpoints so that it interpolates the first and last histogram points:

$$C(0) = Q_0 = P_0, \quad C(1) = Q_{|D|-1} = P_{n_c}. \quad (22)$$

The remaining points  $Q_1, \dots, Q_{|D|-2}$  are approximated in the least-squares sense. Having specified the curve model and endpoint constraints, we next derive the corresponding least-squares system and show how its solution yields the unknown interior control points.

#### 4.5.2. Least-Squares Formulation and Solution

Let  $\{\bar{t}_k\}_{k=0}^{|D|-1}$  denote parameter values associated with the data points. The fitting objective is

$$\min_{P_1, \dots, P_{n_c-1}} \sum_{k=1}^{|D|-2} \|Q_k - C(\bar{t}_k)\|^2. \quad (23)$$

Substituting (21) and using the endpoint constraints (22), we can rewrite

$$C(\bar{t}_k) = N_{0,s}(\bar{t}_k)Q_0 + \sum_{i=1}^{n_c-1} N_{i,s}(\bar{t}_k)P_i + N_{n_c,s}(\bar{t}_k)Q_{|D|-1}. \quad (24)$$

Defining

$$R_k = Q_k - N_{0,s}(\bar{t}_k)Q_0 - N_{n_c,s}(\bar{t}_k)Q_{|D|-1}, \quad k = 1, \dots, |D| - 2, \quad (25)$$

the problem reduces to a standard linear least-squares form:

$$\min_{P_1, \dots, P_{n_c-1}} \sum_{k=1}^{|D|-2} \left\| R_k - \sum_{i=1}^{n_c-1} N_{i,s}(\bar{t}_k)P_i \right\|^2. \quad (26)$$

To express (26) in matrix form, we stack the unknown interior control points and the residual terms as

$$P = (P_1, \dots, P_{n_c-1})^T, \quad R = (R_1, \dots, R_{|D|-2})^T. \quad (27)$$

We further define the basis matrix

$$B = \begin{bmatrix} N_{1,s}(\bar{t}_1) & \cdots & N_{n_c-1,s}(\bar{t}_1) \\ \vdots & \ddots & \vdots \\ N_{1,s}(\bar{t}_{|D|-2}) & \cdots & N_{n_c-1,s}(\bar{t}_{|D|-2}) \end{bmatrix} \in \mathbb{R}^{(|D|-2) \times (n_c-1)}. \quad (28)$$

With these definitions, the objective in (26) can be compactly written as

$$\min_P \|BP - R\|^2. \quad (29)$$

Taking derivatives of (29) with respect to  $P$  and setting them to zero yields the normal equations

$$B^T B P = B^T R. \quad (30)$$

In our application, the basis matrix  $B$  and the vector  $R$  are fully determined by the data points  $\{Q_k\}$ , the parameter values  $\{\bar{t}_k\}$ , and the fixed endpoint control points  $P_0$  and  $P_{n_c}$ . The unknowns are the interior control points  $P_1, \dots, P_{n_c-1}$  collected in  $P$ . Since the  $x$ -coordinates of the control points follow the domain axis and are fixed by  $\{v_k\}$ , (30) is solved only for the frequency (vertical) coordinate. To complete the specification of this system,

we next describe how the parameter values  $\{\bar{t}_k\}$  and the knot vector are constructed, and then explain how the fitted spline is converted into a representation that can be directly integrated during query answering.

#### 4.5.3. A Bias–Variance Bound for Spline Modeling

We further provide an explicit upper bound that relates the spline modeling error to the knot spacing and the variance of the UDPLDP histogram noise. Let  $f(\cdot)$  denote the (unknown) underlying frequency function over the value axis, and let the fitted spline model  $\hat{f}_h(\cdot)$  be obtained by the least-squares procedure in Equation (29) using noisy histogram points  $Q_k = (v_k, \hat{f}_{v_k})^\top$ . Let  $h = \max_j(\zeta_{j+1} - \zeta_j)$  be the maximum knot spacing (equivalently, the maximum leaf-interval length), and let  $S_h$  be the degree- $s$  spline space induced by the knot vector  $\{\zeta_j\}$ .

Define the best-approximation spline  $f_h^* = \arg \min_{g \in S_h} \|f - g\|_2$ , and denote by  $K$  the number of free spline coefficients (i.e., the number of unknown interior control points). Let  $\xi_k = \hat{f}_{v_k} - f(v_k)$  be the histogram noise. Since our unary estimator is unbiased,  $\mathbb{E}[\xi_k] = 0$ . Let  $\sigma^2 = \max_k \text{Var}(\xi_k) = \max_v \text{Var}(\hat{f}_v)$  denote the worst-case histogram noise variance under the modeling subset  $U_{\text{model}}$  (which can be upper bounded by Equation (18) with  $N$  replaced by  $|U_{\text{model}}|$ ).

**Proposition 1** (Spline modeling error bound). *Assume  $f \in C^{s+1}$  and the knots are quasi-uniform. There exist constants  $C_1, C_2 > 0$  depending only on  $s$  and knot regularity such that*

$$\mathbb{E}[\|\hat{f}_h - f\|_2^2] \leq C_1 h^{2(s+1)} \|f^{(s+1)}\|_2^2 + C_2 \sigma^2 \frac{K}{|D|}. \quad (31)$$

Moreover, since  $K = O(1/h)$  for fixed degree  $s$ , the variance term scales as  $O\left(\frac{\sigma^2}{|D|h}\right)$ , while the squared bias term scales as  $O(h^{2(s+1)})$ .

Equation (31) makes explicit the bias–variance trade-off in spline modeling: denser knots (smaller  $h$ ) reduce approximation bias but increase the degrees of freedom and thus amplify the effect of histogram noise, whereas larger  $|U_{\text{model}}|$  reduces  $\sigma^2$  and shrinks the variance term.

#### 4.5.4. Parameterization and Polynomial Conversion

To fully specify the least-squares system in Equation (30), we next define the data parameterization and the associated knot vector. We adopt a standard B-spline approximation setup: the two endpoints are interpolated, while interior points are fitted in the least-squares sense. The spline degree  $s$  is a tunable modeling hyper-parameter. In our implementation we use the cubic case ( $s = 3$ ), a common default that provides  $C^2$  continuity and a stable bias–variance trade-off for fitting an LDP-noisy histogram: lower degrees often underfit and produce non-smooth piecewise trends, while higher degrees can be more noise-sensitive and less stable. Notably, model expressiveness is mainly governed by the knot/control-point density, and the privacy guarantee is unchanged because spline fitting is pure post-processing on locally perturbed reports.

- (i) **Parameterization.** We adopt a cumulative chord-length parameterization for the data points  $\{Q_k\}$ . (Alternative parameterizations such as equally-spaced and centripetal can also be used; we choose chord-length since it adapts the parameter spacing to local variations of the noisy histogram.) Let  $L_{\text{chord}} = \sum_{k=1}^{|D|-1} \|Q_k - Q_{k-1}\|$  denote the total chord length of the polyline through  $Q_0, \dots, Q_{|D|-1}$ . We set  $\bar{t}_0 = 0$ ,  $\bar{t}_{|D|-1} = 1$ , and for  $k = 1, \dots, |D| - 2$  define  $\bar{t}_k = \bar{t}_{k-1} + \frac{\|Q_k - Q_{k-1}\|}{L_{\text{chord}}}$ . This parameterization allo-

cates larger parameter increments to regions where the noisy histogram varies more rapidly, which improves the local fitting quality of the spline.

- (ii) Clamped knot vector construction. Given the parameter values  $\{\bar{t}_k\}$ , we construct a clamped knot vector  $U = \{u_0, \dots, u_{n_c+s+1}\}$  using the De Boor averaging method. First, we enforce endpoint interpolation by setting the boundary knots with multiplicity  $s + 1$ :  $u_0 = \dots = u_s = \bar{t}_0, u_{n_c+1} = \dots = u_{n_c+s+1} = \bar{t}_{|D|-1}$ . Then, for internal knots, let  $n = |D| - 1$  and  $m = n_c$  (so the number of control points is  $m + 1 = n_c + 1$ ). Define  $d = \frac{n+1}{m-s+1}$ . For  $j = 1, \dots, m - s$ , we compute  $i = \lfloor jd \rfloor, \alpha = jd - i, u_{j+s} = (1 - \alpha)\bar{t}_{i-1} + \alpha\bar{t}_i$ . This construction places internal knots according to the distribution of  $\{\bar{t}_k\}$ , which empirically yields well-conditioned normal equations in our implementation.
- (iii) Number of control points. The number of control points  $n_c + 1$  controls the bias-variance trade-off: too few control points underfit the global trend, while too many may follow local perturbation noise. To avoid dataset-specific manual tuning, we optionally use a simple residual-driven refinement rule: starting from an initial  $n_c + 1$ , we increase the number of control points until the spline approximation error on  $\{Q_k\}$  falls below a fixed threshold or a preset maximum is reached.

After solving the normal equations and obtaining the control points, we convert the fitted B-spline into a piecewise cubic polynomial form to support efficient query answering in UDPLDP-Tree, where repeated evaluation and integration over sub-intervals are needed.

We partition the domain into intervals  $I_j = [v_j, v_{j+1})$  and represent the curve on each interval as

$$C(x) = \alpha_j x^3 + \beta_j x^2 + \gamma_j x + \delta_j, \quad x \in I_j. \quad (32)$$

The coefficients  $(\alpha_j, \beta_j, \gamma_j, \delta_j)$  are computed from the fitted spline and stored.

This representation enables closed-form integration on partially covered leaf intervals, avoiding extra discretization or simplifying assumptions. As a result, UDPLDP-Tree can refine the (at most two) boundary leaves via analytic integrals while using low-variance aggregates from fully covered internal nodes; the next subsection details this construction and query procedure.

#### 4.6. UDPLDP-Tree Construction and Calibration

Based on the fitted B-spline model in Section 4.5, we build a hierarchical index, UDPLDP-TREE, for one-dimensional range-frequency queries under user-distinguished local privacy. The design is hybrid: leaf nodes store the cubic polynomials from the private spline fit, while internal nodes store UDPLDP-protected frequencies over unions of consecutive leaf intervals. This separates boundary refinement from interior aggregation: during query answering, fully covered internal nodes provide low-variance aggregates, and only the (at most two) boundary leaf intervals are handled by integrating their cubics. Construction has three steps: (i) define the tree over spline knot spans, (ii) calibrate internal nodes via privacy-compatible user allocation, and (iii) prune the tree to minimize expected range query error.

##### 4.6.1. Tree Structure and Node Semantics

We begin by specifying the node granularity and what is stored at each node. The fitted spline induces  $L$  knot spans on the normalized domain,  $I_j = [\zeta_j, \zeta_{j+1})$  for  $j = 1, \dots, L$ . We align each  $I_j$  with one leaf and store its cubic coefficients  $(\alpha_j, \beta_j, \gamma_j, \delta_j)$  from Equation (32). We then build a balanced binary tree bottom-up over these leaves. Each internal node  $n_k$  corresponds to a contiguous block of leaves, denoted by an index interval  $I_k = [\ell_k, r_k]$  (in leaf order), and stores a frequency estimate  $\hat{f}_k$  for the mass within its covered domain range.

With these semantics fixed, the remaining question is how to estimate  $\hat{f}_k$  under UDPLDP without repeatedly querying the same user along a path.

#### 4.6.2. User Allocation and Node-Wise Estimation

Given a potentially unbalanced UDPLDP-Tree  $\mathcal{T}$ , we adopt path-wise user allocation [17,21] rather than layer-wise allocation [15,16], since nodes on a root-to-leaf path are nested and overlap.

We calibrate internal nodes using the held-out set  $U_{\text{tree}}$  with  $N' = |U_{\text{tree}}|$ . Path-wise allocation avoids repeatedly querying the same users on overlapping intervals, which would inflate variance and complicate privacy accounting. When processing a non-root internal node  $n_k$ , let  $U_k$  be the users passed from its parent and sample  $U'_k \subseteq U_k$  with fraction  $1/h_k$ , where  $h_k$  is the number of remaining internal levels in the subtree of  $n_k$ . The  $1/h_k$  rule spreads users over the remaining levels without dataset-specific per-layer tuning. We pass only the remaining users to children, i.e., for each child  $n_c$ , set  $U_c := U_k \setminus U'_k$ , and denote  $\eta_k = |U'_k|/N'$ .

Since selected users are removed from the entire subtree, along any root-to-leaf path, the allocated sets are pairwise disjoint; thus, each user contributes to at most one node on that path and privacy composes in parallel (not sequential) across nodes on the same path.

Given  $U'_k$ , each user  $u \in U'_k$  reports depending on whether membership in  $I_k$  is sensitive: if  $\tau_u \subseteq I_k$ , the contribution is counted deterministically; otherwise,  $u$  reports the perturbed unary bit  $\mathbb{1}[x_u \in I_k]$  using the level-specific  $(a_p, b_p)$  from Section 4.4.2. The collector aggregates reports by privacy level and applies the same multi-level unary estimator to obtain an unbiased  $\hat{f}_k$ . Because  $\eta_k$  controls the node sample size, it also controls  $\text{Var}(\hat{f}_k)$ ; we exploit this in pruning to trade off noise reduction and query flexibility. We also include an empirical comparison with a representative non-path allocation baseline to validate the benefit of path-wise allocation.

#### 4.6.3. Pruning by Expected Range Query Error

Starting from the balanced tree, following the workload-agnostic pruning strategies used in PrivNUD and PriPL-Tree [17,21], we perform a top-down pruning to reduce noise while preserving query flexibility.

Concretely, we compare alternative local structures by their expected query error under uniformly random one-dimensional ranges, which captures how often each internal estimate is used and how noisy it is under the allocation. We adopt the “uniformly random ranges” model as a workload-agnostic surrogate: when no prior query workload is available, treating all intervals  $[l, r]$  (with  $l \leq r$ ) as equally likely avoids tailoring the structure to any specific dataset or query pattern, while leading to a closed-form and efficiently computable node-usage weight. The pruning criterion is the expected query error under uniformly random one-dimensional ranges, measured by

$$\text{Err} = \sum_{n_k \in \mathcal{N}} w_k \cdot \text{Var}(\hat{f}_k) \propto \sum_{n_k \in \mathcal{N}} \frac{w_k}{\eta_k}, \quad (33)$$

where  $\mathcal{N}$  is the set of internal nodes,  $w_k$  is the probability that  $n_k$  participates in answering a random range query, and the proportionality uses the variance bound of the unary estimator together with  $\eta_k$ . Under this model, the sample space contains exactly  $M(M+1)/2$  intervals over a domain of size  $M = |D|$ . Here,  $w_k$  is the fraction of intervals for which node  $n_k$  is selected in the canonical tree decomposition (i.e.,  $I_k \subseteq [l, r]$  but the parent interval

is not fully contained), so  $w_k$  directly quantifies the expected “usage frequency” of  $n_k$ . For a non-root node  $n_k$  with leaf-index interval  $[\ell_k, r_k]$  and parent  $n_p$  with  $[\ell_p, r_p]$ , we compute

$$w_k = \frac{\ell_k(M - r_k + 1) - \ell_p(M - r_p + 1)}{(M + 1)M/2}. \quad (34)$$

In practice, removing a node only affects allocations and weights on the ancestor–descendant paths around that node, so we evaluate Equation (33) locally on the affected subtree and keep the structure that yields a smaller error. We also include empirical validation by reporting results with and without pruning under the same user-allocation rule.

At last, given a pruned UDPLDP-Tree  $\mathcal{T}$  and a range  $[l, r]$ , we compute the estimated answer by the procedure in Algorithm 2. The algorithm performs a stack-based traversal to aggregate *fully covered* nodes: whenever a node interval  $I_k \subseteq [l, r]$  but its parent is not fully contained, we add its estimated mass  $\hat{f}_k$  to the accumulator, which ensures that each covered region is counted exactly once (Lines 2–11). For the remaining boundary parts, the algorithm refines the estimate at the leaf level by integrating the stored piecewise cubic polynomial on each partially overlapped leaf interval  $I_j$  over  $[l, r] \cap I_j$  (Lines 12–15), yielding the final estimate  $Q([l, r])$ .

---

**Algorithm 2:** Range query answering on UDPLDP-Tree.

---

**Input:** Pruned UDPLDP-Tree  $\mathcal{T}$  and range  $[l, r]$ .

**Output:** Estimated frequency  $Q([l, r])$ .

---

```

1   $A \leftarrow 0;$  // accumulator
2  initialize a stack  $S$  with the root of  $\mathcal{T}$ ;
3  Internal-node aggregation.
4  while  $S$  is not empty do
5      pop a node  $n_k$  with interval  $I_k$  from  $S$ ;
6      if  $I_k \cap [l, r] = \emptyset$  then
7          continue
8      else if  $I_k \subseteq [l, r]$  and the parent of  $n_k$  is not fully contained in  $[l, r]$  then
9           $A \leftarrow A + \hat{f}_k;$  // fully covered internal node
10     else if  $n_k$  is internal then
11         push all children of  $n_k$  onto  $S$ ;

12 Boundary-leaf refinement.
13 foreach leaf  $I_j$  that partially overlaps  $[l, r]$  do
14      $[l_{\text{sub}}, r_{\text{sub}}] \leftarrow [l, r] \cap I_j;$ 
15     add the contribution of the cubic polynomial on  $I_j$  over  $[l_{\text{sub}}, r_{\text{sub}}]$  to  $A$ ;
16 return  $A$ 

```

---

In practice, removing a node only changes allocations and weights along nearby ancestor–descendant paths, so we evaluate Equation (33) locally on the affected subtree and keep the structure with smaller error. While Equation (34) derives  $w_k$  under uniformly random ranges for a closed form, the pruning objective is workload-agnostic. For a workload  $\pi$ , one can set  $w_k$  to the probability (under  $\pi$ ) that  $n_k$  is used to answer a random query, and apply the same top-down pruning: frequently used nodes get larger  $w_k$  and are retained, while rarely used nodes are pruned. A systematic empirical study of workload-adaptive pruning is left for future work. We also report results with and without pruning under the same allocation rule to empirically validate the pruning criterion.

#### 4.7. Privacy and Error Analysis

This subsection completes the description of UDPLDP-Tree by analyzing (i) why the overall mechanism satisfies the intended  $\mathcal{E}$ -UDPLDP guarantee and (ii) how different design stages contribute to the final range query error. We first establish privacy by tracing all user interactions and invoking the post-processing property, and then decompose the estimation error into interpretable components aligned with the modeling and tree-calibration pipeline.

##### 4.7.1. Privacy Guarantee

UDPLDP-Tree can answer range query while satisfying rigorous UDPLDP guarantees as shown in the following theorem.

**Theorem 1.** *UDPLDP-Tree satisfies  $\mathcal{E}$ -UDPLDP.*

**Proof.** UDPLDP-Tree consists of four phases: (i) frequency-oracle reporting, (ii) B-spline fitting, (iii) tree calibration, and (iv) query answering. We analyze privacy phase by phase.

- (i) Frequency oracle (users in  $U_{\text{model}}$ ). The split of users into  $U_{\text{model}}$  and  $U_{\text{tree}}$  is independent of users' private values, hence it incurs no privacy loss. Each user  $u \in U_{\text{model}}$  runs the unary  $\mathcal{E}$ -UDPLDP mechanism once to report a perturbed unary vector for her value. By the privacy theorem of the unary mechanism (see Section 4.4.2), this interaction satisfies  $\mathcal{E}$ -UDPLDP.
- (ii) B-spline fitting. The aggregator computes the noisy histogram and fits the B-spline model using only the perturbed reports from Phase (i). This phase does not access raw inputs and is therefore pure post-processing. So, it consumes no additional privacy budget.
- (iii) Tree calibration (users in  $U_{\text{tree}}$ ). Given the tree structure obtained from Phase (ii), each user  $u \in U_{\text{tree}}$  is assigned a set of internal nodes  $S_u$  whose corresponding intervals are pairwise disjoint (not on the same root-to-leaf path). User  $u$  forms a membership indicator vector  $\mathbf{v}_u \in \{0, 1\}^{|S_u|}$  over  $S_u$ , where exactly one coordinate equals 1 (the unique interval containing  $x$ ) and all others are 0; thus,  $\mathbf{v}_u$  is one-hot. She then perturbs each coordinate independently using the same bit randomizer as in the unary mechanism (parameterized by  $\epsilon_u$ ), producing an output vector  $\mathbf{O}$ . Hence, the entire tree-calibration interaction satisfies  $\mathcal{E}$ -UDPLDP for each user in  $U_{\text{tree}}$ .
- (iv) Query answering. Tree construction/pruning, node-frequency decoding, and answering range queries are computed solely from the perturbed reports collected in Phases (i) and (iii). Therefore, they are post-processing and incur no additional privacy loss.

Finally, since  $U_{\text{model}}$  and  $U_{\text{tree}}$  are disjoint and each user participates in at most one interactive phase, the overall algorithm satisfies  $\mathcal{E}$ -UDPLDP.

□

##### 4.7.2. Error Analysis

With privacy established, we now analyze the error sources of UDPLDP-Tree for a range query  $Q([l, r])$ . Because UDPLDP-Tree answers queries by combining (i) a smoothed spline model at the leaves and (ii) calibrated internal-node aggregates, its error naturally decomposes along the same pipeline. Specifically, the error has three components: (i) noise and sampling error of the unary estimation on  $U_{\text{model}}$ ; (ii) approximation error of the cubic B-spline fitted on the noisy histogram; and (iii) noise and sampling error of internal-node estimation on  $U_{\text{tree}}$ .

- (i) Unary estimation error on  $U_{\text{model}}$ . We first quantify the noise level of the histogram that serves as the input to B-spline fitting. For each domain point (or bin)  $x \in D$ , the collector obtains an unbiased estimate  $\hat{f}_x$  using Equation (16) on reports from  $U_{\text{model}}$ . Its mean squared error is bounded by Equation (18), where  $N$  and  $\{n_p\}$  are interpreted on  $U_{\text{model}}$  (i.e.,  $n_p$  counts level- $p$  users whose privacy range contains  $x$ ).
- (ii) Spline modeling error. Let  $C_s(\cdot)$  be the fitted cubic B-spline (least squares) from the noisy histogram  $\{\hat{f}_x\}_{x \in D}$ . For any  $x \in [\zeta_j, \zeta_{j+1})$ ,

$$C_s(x) - f_x = (C_s(x) - \hat{f}_x) + (\hat{f}_x - f_x), \quad (35)$$

so by  $(a + b)^2 \leq 2a^2 + 2b^2$ ,

$$(C_s(x) - f_x)^2 \leq 2(C_s(x) - \hat{f}_x)^2 + 2(\hat{f}_x - f_x)^2. \quad (36)$$

The first term is the spline-fitting residual to  $\hat{f}$ , and the second is the unary histogram noise from (i). Hence, the modeling error decomposes into a data-dependent fitting residual plus unary-estimation noise; it decreases for smoother distributions and/or finer knots.

Spline integration can introduce boundary bias in addition to the variance of locally perturbed internal-node estimates. Under smoothness and maximum knot spacing  $h$ , the degree- $s$  spline approximation error is  $O(h^s)$  in the interior; since a range  $Q$  has at most two partial boundary segments, their contribution scales as  $O(h^{s+1}/|Q|)$ . In contrast, internal-node estimators are unbiased with variance  $O(1/(N\varepsilon^2))$  under our unary randomizer. Thus, for our regimes (moderate  $\varepsilon$ , realistic  $N$ , and ranges spanning many cells), internal-node noise is dominant and boundary bias is lower-order. Boundary bias dominates mainly when LDP noise is negligible (very large  $N$  and high  $\varepsilon$ ) and the workload focuses on very short boundary ranges or highly non-smooth tails, consistent with DP-model-bias interactions reported in security-sensitive settings [25].

- (iii) Internal-node estimation error on  $U_{\text{tree}}$ . Finally, we quantify the uncertainty contributed by internal-node aggregates that are used whenever a query fully covers a node interval. Users in  $U_{\text{tree}}$  are allocated to non-root internal nodes as described in Section 4.6. For an internal node  $n_k$ , let  $U'_k$  be its assigned user set and  $N_k = |U'_k|$ . Applying the same multi-level unary estimator to  $U'_k$  yields an unbiased node frequency estimate  $\hat{f}_k$ , whose variance admits the same functional bound as Equation (18) with  $N$  replaced by  $N_k$ .

We now connect the above point-wise and node-wise errors to the final range query estimator. Given a query interval  $[l, r]$ , UDPLDP-Tree decomposes  $[l, r]$  into (a) a set of disjoint fully covered internal-node intervals and (b) at most two boundary sub-ranges within leaf intervals. The query answer is estimated by summing the selected internal-node estimates  $\{\hat{f}_k\}$  and adding the spline integrals over the boundary sub-ranges. Since  $U_{\text{model}}$  and  $U_{\text{tree}}$  are disjoint, and the path-wise allocation ensures that the user sets  $\{U'_k\}$  for different internal nodes are disjoint, the internal-node estimation errors contribute additively in variance. Therefore, the MSE of the range estimator is given by (a) the sum of variances of the internal nodes used in the decomposition and (b) the spline-modeling error on the boundary parts, whose point-wise error is controlled by the decomposition in (ii) together with the histogram MSE in (i).

#### 4.7.3. Computational and Communication Costs

We summarize the computational and communication costs of UDPLDP-Tree in the one-dimensional setting. Let  $M = |D|$  be the domain size,  $m = |\mathcal{E}|$  be the number of

privacy levels,  $N_{\text{model}} = |U_{\text{model}}|$ , and  $N_{\text{tree}} = |U_{\text{tree}}|$ . Let  $L$  be the number of spline knot spans (thus  $L$  leaf intervals in the tree) and  $|\mathcal{T}| = O(L)$  be the number of nodes after pruning.

- (i) Local frequency reporting. Each user perturbs only the coordinates within her declared range  $\tau_u$ ; hence, the user-side time is  $O(|\tau_u|)$ . In terms of communication, each user sends the privacy-level index and the randomized bits for  $\tau_u$ , which is  $O(|\tau_u|)$  bits (and degenerates to  $O(M)$  in the worst case when  $\tau_u = D$ ). On the collector side, level-wise aggregation maintains counters per privacy level, yielding time  $O(\sum_{u \in U_{\text{model}}} |\tau_u|)$  and memory  $O(mM)$  for the sufficient statistics.
- (ii) B-spline fitting. The spline is fitted once from the noisy histogram of  $M$  points. Because B-spline bases have local support ( $s + 1$  non-zero bases per parameter value), forming the normal equations is linear in  $M$  (up to a constant depending on  $s$ ), and solving for the  $n_c - 1$  unknown interior control-point ordinates is efficient due to the banded structure. The subsequent conversion to the piecewise cubic form stores one polynomial per leaf interval, requiring  $O(L)$  time and  $O(L)$  memory.
- (iii) Tree calibration and pruning. Tree construction and pruning are offline and take  $O(|\mathcal{T}|)$  time and  $O(|\mathcal{T}|)$  space. For calibration, each user in  $U_{\text{tree}}$  reports membership indicators for the nodes she is allocated to; if a user is assigned  $|\mathcal{S}_u|$  nodes, her computation and communication are  $O(|\mathcal{S}_u|)$ . The collector-side aggregation scales as  $O(\sum_{u \in U_{\text{tree}}} |\mathcal{S}_u|)$  with an additional linear factor in  $m$  for maintaining level-wise sufficient statistics, and the memory overhead is  $O(m|\mathcal{T}|)$ .
- (iv) Query answering. After offline construction, answering a query requires traversing the pruned tree to collect the canonical decomposition plus integrating at most two boundary cubics; thus the per-query time is  $O(\log |\mathcal{T}|)$  in typical balanced cases, with  $O(1)$  extra arithmetic for boundary refinement. Overall, the dominant online cost is logarithmic in the tree size, while the costs that scale with  $M$  and  $m$  occur only once per data-collection round.

## 5. Multi-Dimension Extension with Attribute-Weighted Privacy

This section extends UDPLDP-Tree from one-dimension to multi-dimension range queries. In multi-attribute data, sensitivities differ across dimensions, so a uniform budget can under-protect sensitive attributes or over-protect less sensitive ones. We address this by introducing attribute weights to scale user-level budgets, and combining 1-D UDPLDP-Trees with grid-based decompositions [17,20] for multi-dimension range answering. Importantly, we apply the same attribute-weighted budget rule to all reporting tasks in our multi-dimension framework, including one-dimension marginal reporting and two-dimension grid reporting, so the privacy semantics remains consistent when moving from single-attribute to attribute-pair estimation.

### 5.1. Motivation: Attribute-Heterogeneous Privacy Requirements

We start by motivating why a dimension-aware privacy allocation is needed in the multi-dimension setting. Most existing multi-dimension LDP-based range-frequency mechanisms implicitly assume that all attributes share the same privacy budget, and that users have homogeneous privacy requirements across attributes. This assumption is often unrealistic. In practice, attributes such as disease type, income, and precise location are substantially more sensitive than, e.g., age group or coarse demographic information. Enforcing a single budget for all attributes has two adverse effects: highly sensitive attributes may be under-protected, while less sensitive attributes are over-perturbed, leading to unnecessary noise and degraded accuracy for multi-dimension range-frequency queries.

Motivated by this gap, we introduce an attribute-weighted personalization scheme: the system first pre-assigns attribute-wise sensitivity weights for all dimensions, and each user then specifies a base privacy budget and a multi-dimension privacy range; the attribute-specific effective budgets are obtained by scaling the base budget with the pre-assigned weights. The key idea is to keep the user-chosen base budget as the protection level for the most sensitive attributes, while allowing less sensitive attributes (or reports involving them) to use larger effective budgets and thus lower noise. The next subsection formalizes this mechanism and clarifies how it preserves UDPLDP personalization while differentiating protection across attributes.

## 5.2. Attribute-Weighted UDPLDP Mechanism

We define an attribute-weighted UDPLDP mechanism by scaling each user's base budget with public attribute weights. For a  $d$ -dimension record  $\mathbf{v}_u = (v_u^{(1)}, \dots, v_u^{(d)})$  with  $v_u^{(j)} \in D_j$ , each attribute  $A_j$  has a public weight  $\omega^{(j)} \geq 1$ , where  $\omega^{(j)} = 1$  is the highest sensitivity. Here,  $\omega^{(j)}$  should be interpreted as an inverse-sensitivity (utility) weight:  $\omega^{(j)} = 1$  denotes the most sensitive attributes, while a larger  $\omega^{(j)}$  means the attribute is less sensitive and can therefore tolerate weaker protection (i.e., less noise) for better utility. Recall that in (local) differential privacy, a larger  $\epsilon$  implies weaker privacy and thus less injected noise. Each user declares a base budget  $\epsilon_u \in \mathcal{E}$  (for  $\omega^{(j)} = 1$ ) and an attribute-wise privacy range  $\tau_u = \prod_{j=1}^d \tau_u^{(j)}$ .

Effective budgets are then assigned per reporting task according to which attributes the task involves. For a 1-D task on  $A_j$ , we set  $\epsilon_u^{(j)} = \omega^{(j)} \epsilon_u$ . More generally, when a reporting task involves an attribute subset  $S \subseteq \{1, \dots, d\}$ , we assign the effective task budget as  $\epsilon_u^{(S)} = \left(\prod_{j \in S} \omega^{(j)}\right) \epsilon_u$ . We use a multiplicative aggregation so that the task budget is amplified only when all involved attributes are relatively less sensitive; in particular, if any attribute is maximally sensitive ( $\omega^{(j)} = 1$ ), the task budget will not be overly inflated. This unified rule will be used by our framework below: Phase I corresponds to  $|S| = 1$  (marginals), while Phase II corresponds to  $|S| = 2$  (pairwise grids). For a task on  $S$ , we define the induced task domain  $D_S = \prod_{j \in S} D_j$  and induced privacy range  $\tau_u^{(S)} = \prod_{j \in S} \tau_u^{(j)}$ . Since  $\omega^{(j)}$  is public and does not depend on users' private values, it only changes the intended protection level per attribute and does not introduce additional leakage beyond the UDPLDP model.

We instantiate the task-level randomizer on  $D_S$  by reusing the 1-D UDPLDP building block (unary OUE-style construction and the same pairwise budget operator in Section 4.4.2). For  $S = \{j\}$ , this is a UDPLDP mechanism on  $D_j$  using  $\{\epsilon_u^{(j)}\}_u$  and  $\{\tau_u^{(j)}\}_u$ . For an attribute-pair task on  $\langle A_i, A_j \rangle$ , we instantiate the same unary UDPLDP mechanism on the grid domain  $D_i \times D_j$  using the task budgets  $\{\epsilon_u^{(i,j)}\}_u$ , where  $\epsilon_u^{(i,j)} = \omega^{(i)} \omega^{(j)} \epsilon_u$  is exactly the  $|S| = 2$  case of the unified rule above.

Crucially, the UDPLDP constraint is enforced at the level of each task: for any user pair  $(u, u')$ , indistinguishability is controlled by applying the same pairwise operator to the effective task budgets  $\min\{\epsilon_u^{(S)}, \epsilon_{u'}^{(S)}\}$ . Accordingly,  $M^{(S)}$  satisfies user-level UD-PLDP if, for all  $u, u' \in \mathcal{U}$ ,  $x \in \tau_u^{(S)}$ ,  $x' \in \tau_{u'}^{(S)}$ , and  $y \in \text{Range}(M^{(S)})$ ,

$$\frac{\Pr[M^{(S)}(x) = y]}{\Pr[M^{(S)}(x') = y]} \leq \exp\left(\min\{\epsilon_u^{(S)}, \epsilon_{u'}^{(S)}\}\right). \quad (37)$$

where  $M^{(S)}$  denotes the task-level local randomizer instantiated on  $D_S$  (e.g.,  $D_j$  for  $S = \{j\}$  and  $D_i \times D_j$  for  $S = \{i, j\}$ ).

If a user participates in multiple reporting tasks, the overall privacy loss follows standard sequential composition across those interactions. In our framework, we avoid

unintended budget accumulation by allocating users to disjoint tasks, so that each user contributes to at most one task (either one marginal or one grid). We next build a query answering framework that scales to  $\lambda$ -dimension ranges.

### 5.3. Multi-Dimension Framework

Given the attribute-weighted UDPLDP mechanism above, we adopt a standard decomposition framework that reduces  $\lambda$ -dimension ( $\lambda \geq 2$ ) range queries to one-dimension marginals and two-dimension grids, similar in spirit to prior work [20]. We emphasize that our contribution lies in the privacy allocation scheme; the decomposition serves as a compatible and efficient wrapper for turning the task-level UDPLDP mechanisms into multi-dimension query answers. The apparent “1-D then 2-D” structure is purely algorithmic: both are instances of the same task-level UDPLDP mechanism with different task domains and effective task budgets. The framework proceeds in three phases that estimate 1-D marginals, 2-D pairwise joint counts, and then reconcile them for consistency.

- **Phase I: One-dimension marginals via UDPLDP-Trees.** We first allocate a subset of users to estimate one-dimension marginal distributions, since these marginals provide both standalone 1-D query answers and constraints for refining higher-dimensional estimates. For each attribute  $A_j$ , we build an independent UDPLDP-Tree on domain  $D_j$  using only the attribute-specific budgets  $\{\epsilon_u^{(j)}\}_u$  and privacy ranges  $\{\tau_u^{(j)}\}_u$ . This yields a set of 1-D UDPLDP-Trees that provide smoothed, piecewise-cubic approximations of the marginals and can answer one-dimension range-frequency queries along each attribute.
- **Phase II: Two-dimension grids for attribute pairs.** While marginals alone are insufficient for  $\lambda \geq 2$  range queries, a lightweight way to capture cross-attribute dependence is to estimate pairwise joint distributions. For each attribute pair  $\langle A_i, A_j \rangle$  with  $i \neq j$ , we construct a regular grid over  $D_i \times D_j$  using the same granularity as in [20] and assign a disjoint subset of users to this pair. Each assigned user  $u$  contributes exactly one privatized report to the grid task, using the effective task budget  $\epsilon_u^{(i,j)} = \omega^{(i)} \omega^{(j)} \epsilon_u$  defined in the mechanism above. For any user  $u$  assigned to  $\langle A_i, A_j \rangle$ , we form a single attribute-pair privacy budget  $\epsilon_u^{(i,j)} = \omega^{(i)} \omega^{(j)} \epsilon_u$  and apply the UDPLDP unary mechanism (Section 4.4.2) to her grid cell in  $D_i \times D_j$  under this budget. Aggregating the privatized reports over all users yields noisy but unbiased estimates of the cell frequencies in each two-dimension grid. The resulting indistinguishability for any user pair  $(u, u')$  in this task is bounded by  $\min\{\epsilon_u^{(i,j)}, \epsilon_{u'}^{(i,j)}\}$ , consistent with the UDPLDP semantics under heterogeneous effective budgets.
- **Phase III: Response matrices and consistency refinement.** The final step converts the separate 1-D and 2-D estimates into a coherent set of statistics that can be used for multi-dimension query answering. For each attribute pair  $\langle A_i, A_j \rangle$ , we combine the one-dimension marginals from the two corresponding UDPLDP-Trees with the two-dimension grid counts to construct a response matrix that approximates the joint distribution on  $D_i \times D_j$  [17,20]. A lightweight consistency refinement (e.g., a weighted least-squares adjustment) can be applied so that the row and column marginals of each response matrix match the 1-D UDPLDP-Tree estimates while remaining close to the noisy grid counts. This step does not consume additional privacy budget.

## 6. Evaluation

In this section, we evaluate the performance of UDPLDP-Tree and its extension for both one-dimension and multi-dimension range queries.

### 6.1. Experimental Setup

- **Environment.** All experiments are conducted on a desktop with an 11th Gen Intel Core i5-11500 CPU @ 2.70 GHz, 16 GB RAM, and Intel UHD Graphics 750.
- **Datasets.** Following the dataset configuration in prior work, we use four synthetic datasets (Gaussian, MixGaussian, Cauchy, Zipf) and four real-world datasets (Adult [26], Loan [27], Salary [28], Financial [29]). Each dataset contains five attributes. For the synthetic data, every dimension is independently drawn from a target distribution: Gaussian from  $\mathcal{N}(0, 1)$ , Cauchy from Cauchy(0, 1), and Zipf from Zipf(1.1). The MixGaussian dataset is generated from a mixture of two Gaussians,  $\mathcal{N}(0, 0.5)$  and  $\mathcal{N}(3, 0.8)$ . For the real-world datasets, we select five attributes per dataset and discretize them into finite domains for range-frequency evaluation. Table 1 summarizes the dataset sizes and basic statistics after discretization.

**Table 1.** Summary of datasets.

| Dataset     | U.#    | L.# | Mean   | Var.    | Dataset   | U.#       | L.# | Mean  | Var.    |
|-------------|--------|-----|--------|---------|-----------|-----------|-----|-------|---------|
| Gaussian    | $10^6$ | 0   | 155.0  | 775.34  | Adult     | 32,561    | 4   | 30.36 | 336.88  |
| MixGaussian | $10^6$ | 0   | 135.12 | 2516.62 | Loan      | 148,045   | 3   | 48.67 | 1620.19 |
| Cauchy      | $10^6$ | 5   | 159.37 | 609.50  | Salary    | 2,013,799 | 2   | 33.59 | 515.88  |
| Zipf        | $10^6$ | 5   | 24.40  | 2517.47 | Financial | 6,362,620 | 5   | 0.23  | 2.65    |

U.#: The number of users (i.e., samples). L.#: The number of leptokurtic attributes with a kurtosis exceeding 3.

- **Metrics.** We use the MSE as the main measure of utility. For a workload  $\mathcal{Q}$  of range-frequency queries, let  $f_Q$  and  $\hat{f}_Q$  denote the true and estimated answers for  $Q \in \mathcal{Q}$ . The error is computed as  $\text{MSE}(\mathcal{Q}) = \frac{1}{|\mathcal{Q}|} \sum_{Q \in \mathcal{Q}} (f_Q - \hat{f}_Q)^2$ . Unless otherwise stated, each configuration is evaluated on 1000 randomly generated range queries with a given range-length distribution, and the final MSE is obtained by averaging over 20 independent runs of the experiment. In each run, we use an independent random seed that controls (i) the sampling of users/records, (ii) the randomness of the local perturbation mechanism, and (iii) the random query workload generation. For readability, the figures plot the mean MSE over the 20 independent runs.
- **Competitors.** For one-dimension range-frequency tasks, we compare UDPLDP-Tree with AHEAD [16], PrivNUD [17], and PriPL-Tree [21]. For multi-dimension experiments, we additionally include HDG [20] as a strong hierarchical baseline. These baselines cover both hierarchical schemes (AHEAD, HDG, PrivNUD) and recent model-assisted schemes (PriPL-Tree) under local privacy. All implementations follow the original papers with default/recommended parameters. All methods use the same datasets, privacy budgets, query workloads, and user population size, so MSE differences reflect mechanism design rather than mismatched settings.
- **Default Settings.** Unless otherwise specified, we split users evenly between the modeling and tree sets, i.e.,  $|U_{\text{model}}| : |U_{\text{tree}}| = 1:1$ . The average privacy budget is fixed to  $\bar{\epsilon} = 0.4$  for all methods. We set the query volume to  $\text{vol}(\mathcal{Q}) = 0.5$ , i.e., the range-length/domain-size ratio on each queried attribute. We use the default 1:1 split as in PriPL-Tree [21]; fixing this split across all methods/datasets ensures like-for-like comparison, since the ratio trades off modeling vs. tree-estimation variance. While [21] derives an optimal split for its LP-based model, extending split optimization to our spline-based model is left for future work. To instantiate heterogeneous privacy ranges, each user samples a granularity level  $g$  from a fixed categorical distribution and sets  $\tau_u$  as a contiguous interval (fixed length ratio) containing her true value, clipped at boundaries; we use the same  $\{(\tau_u, \epsilon_u)\}$  generation for all methods. Unless

stated otherwise, we use the default B-spline setting in Section 4.5 and the default tree calibration/pruning rules in Section 4.6.

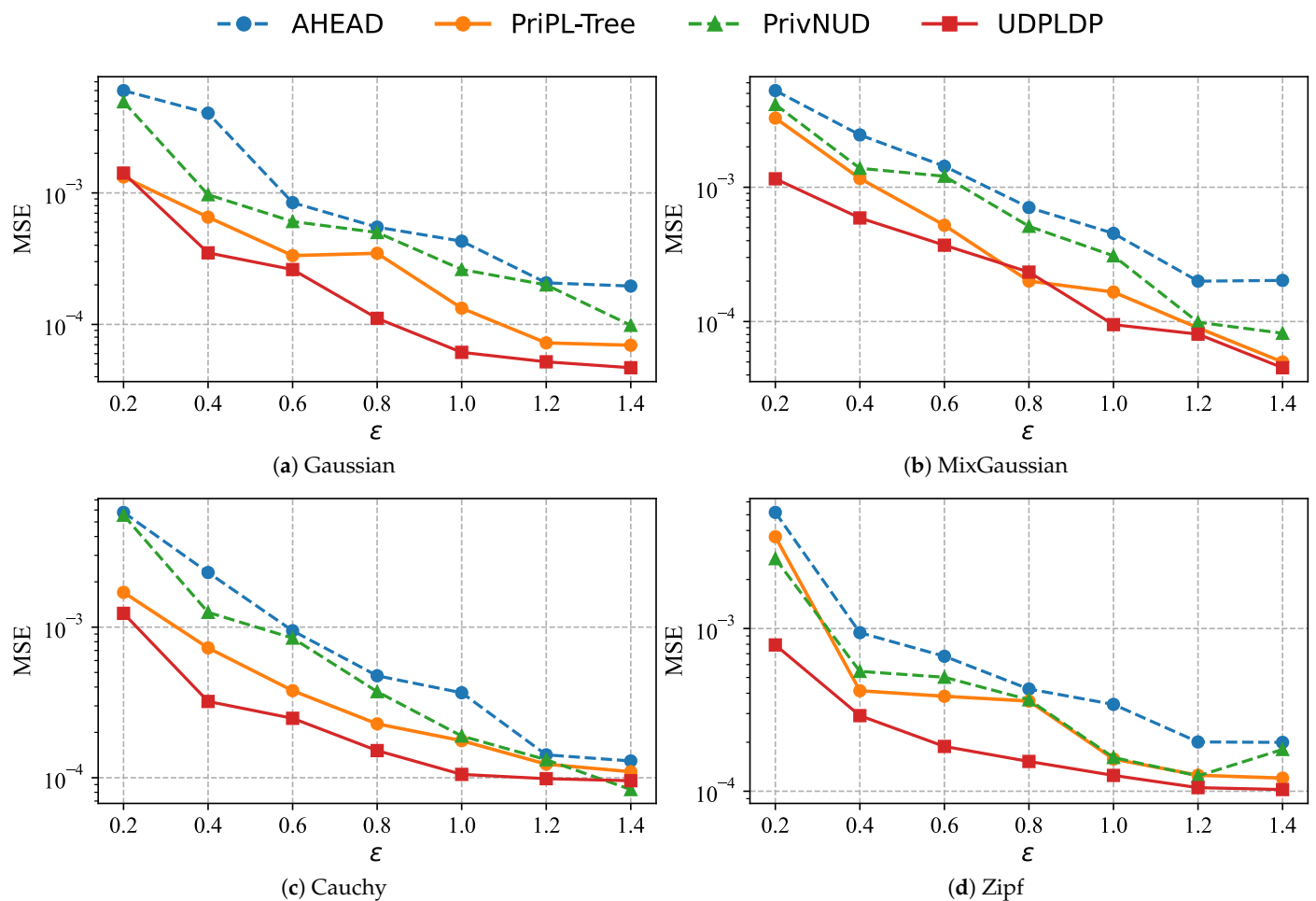
## 6.2. Evaluation for One-Dimension Range Queries

In this subsection, we evaluate the performance of UDPLDP-Tree on one-dimension range queries. We first compare UDPLDP-Tree with AHEAD, PrivNUD, and PriPL-Tree on synthetic and real-world datasets. Note that the one-dimension comparison is not limited to a single baseline: we include both a representative hierarchical method (AHEAD) and two recent model-assisted methods (PrivNUD and PriPL-Tree), which enables a like-for-like evaluation across different design families under the same settings.

### 6.2.1. Results for One-Dimension Range Queries on Synthetic Datasets

We begin with four synthetic datasets (Gaussian, MixGaussian, Cauchy, and Zipf). For each dataset, we report the MSE of different methods under the default settings described in Section 5. The curves show how UDPLDP-Tree behaves under different distribution shapes, including light-tailed, heavy-tailed, and multi-modal cases, and provide a clean comparison with AHEAD, PrivNUD, and PriPL-Tree.

Figure 2 reports the MSE of one-dimension range queries on the four synthetic datasets. Across all privacy budgets  $\epsilon$  and all distributions, UDPLDP-Tree consistently achieves the lowest error. For example, when  $\epsilon \leq 0.6$ , UDPLDP-Tree typically reduces the MSE by around 30~50% compared with PriPL-Tree, and by about 2~5 $\times$  compared with AHEAD, while also outperforming PrivNUD. As  $\epsilon$  increases, the MSE of all methods decreases, but UDPLDP-Tree maintains a stable margin, indicating that our user-distinguished privacy allocation and distribution modeling bring benefits over the whole privacy regime. The gains are particularly evident on the heavy-tailed Cauchy and Zipf datasets, where the B-spline-based modeling and adaptive tree partition enable UDPLDP-Tree to better capture skewed densities and avoid over-allocating noise to low-density regions. Overall, these results confirm that combining UDPLDP with piecewise curve fitting and hierarchical aggregation yields a more accurate one-dimension range query mechanism than existing LDP approaches on synthetic data.



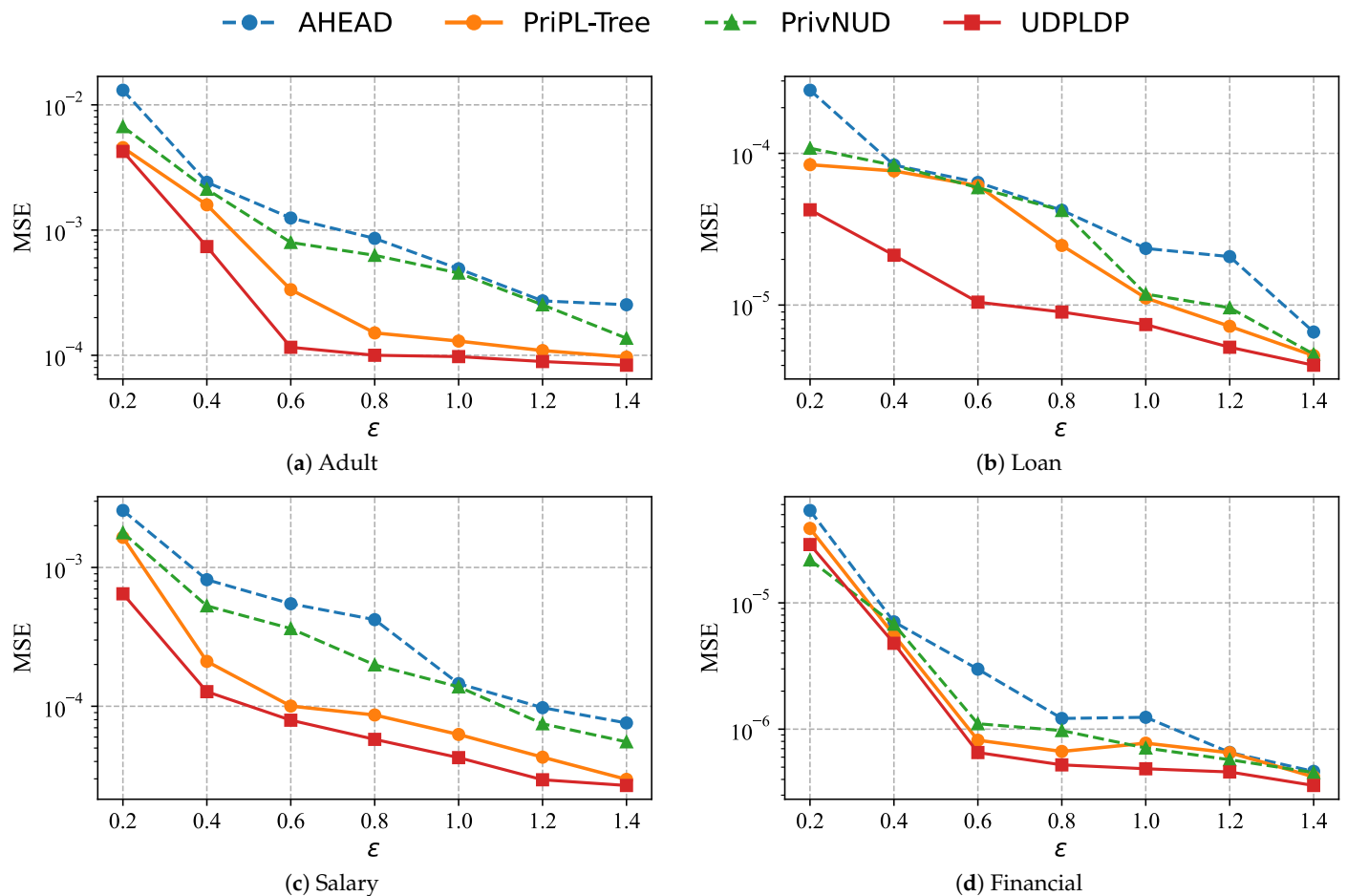
**Figure 2.** One-dimension range query MSE comparison under four synthetic datasets.

### 6.2.2. Results for One-Dimension Range Queries on Real-World Datasets

We next evaluate one-dimension range queries on the four real-world datasets (Adult, Loan, Salary, and Financial). Under the same default configuration, we plot the MSE of all methods and observe how they perform on practical, heterogeneous data. These results demonstrate the robustness of UDPLDP-Tree when facing skewed frequencies and attribute discretization in real applications.

Figure 3 reports the MSE of one-dimension range queries on the four real-world datasets. Across all privacy budgets  $\epsilon$ , UDPLDP-Tree consistently attains the lowest error. On Adult, when  $\epsilon = 0.6$ , UDPLDP-Tree reduces the MSE to around  $10^{-4}$ , which is about three times smaller than PriPL-Tree and nearly an order of magnitude lower than AHEAD and PrivNUD. On Loan and Salary, for  $\epsilon$  in the range  $[0.4, 1.0]$ , UDPLDP-Tree typically achieves about  $2\text{--}4\times$  lower MSE than PriPL-Tree and PrivNUD, and more than one order of magnitude improvement over AHEAD. Even on the relatively easy Financial dataset, where all methods already obtain very small errors, UDPLDP-Tree still delivers the best accuracy for most privacy budgets.

These gains are consistent with our design. Real-world datasets often exhibit skewed and clustered distributions, together with heterogeneous user sensitivities. The UDPLDP mechanism allows users with different privacy ranges and budgets to contribute with appropriate noise levels, while the B-spline modeling and UDPLDP-Tree construction adapt the partition to dense regions and avoid spending privacy budget on sparse intervals. As a result, UDPLDP-Tree captures the underlying frequency patterns more accurately, leading to substantially lower MSE than existing LDP schemes on all four real datasets.



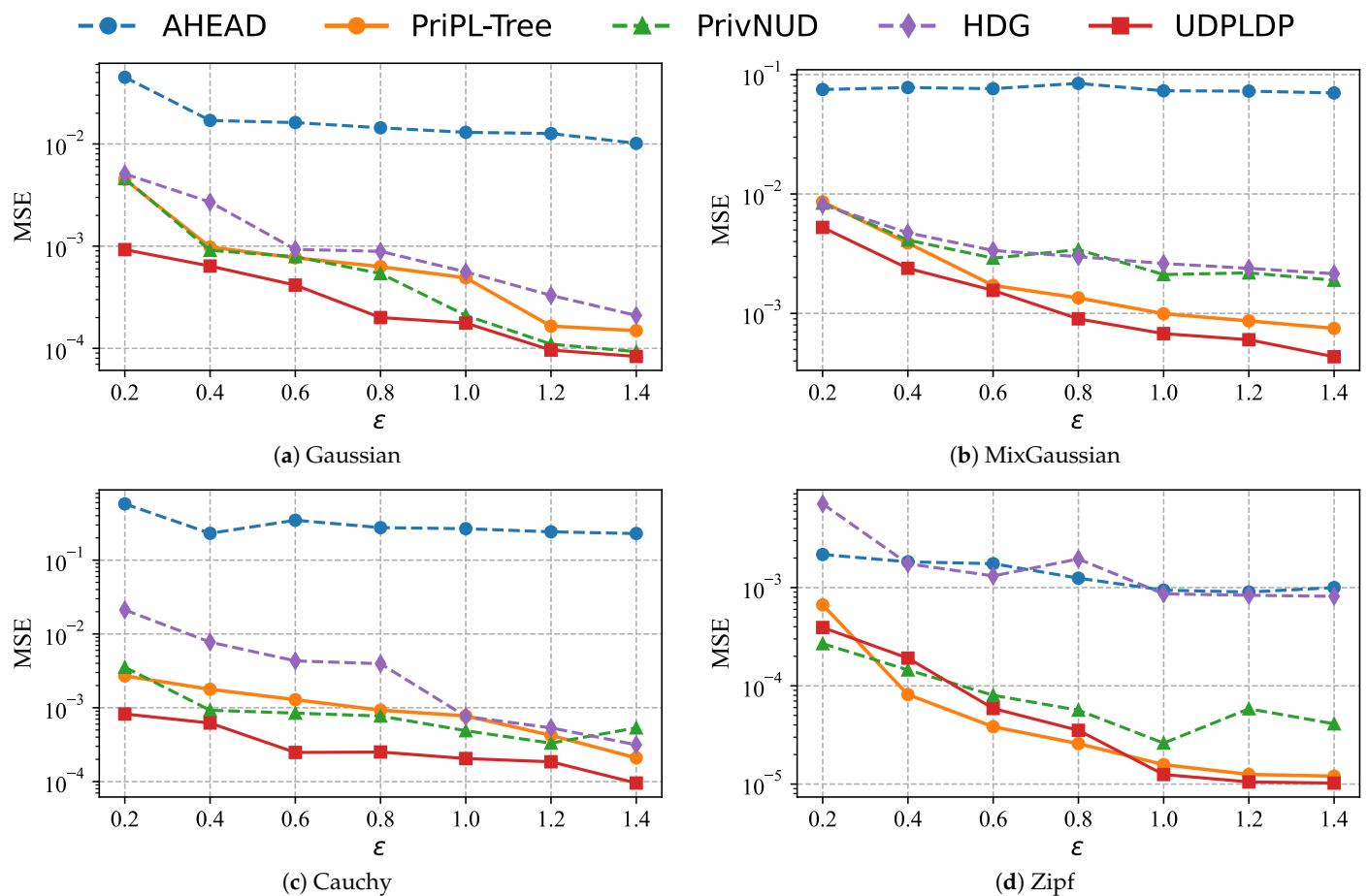
**Figure 3.** One-dimension range query MSE comparison under four real datasets.

### 6.3. Evaluation for Multi-Dimension Range Queries

We next evaluate the multi-dimension extension of UDPLDP-Tree under attribute-weighted privacy. In this setting, each user holds a  $d$ -dimensional record, while each query involves a subset of attributes and asks for the fraction of records falling into a multi-dimension hyper-rectangle. Following Section 5, we focus on three-dimension range queries and vary the total privacy budget  $\epsilon$  while keeping the attribute-weight configuration fixed. For multi-dimension evaluation, we additionally include HDG as a strong hierarchical baseline; together with AHEAD, PrivNUD, and PriPL-Tree, the competitor set covers multiple state-of-the-art design families under the same privacy budgets and query workloads. For clarity, we denote our proposed multi-dimension mechanism as UDPLDP in the plots. The performance metric is the MSE between the estimated and true query answers.

#### 6.3.1. Results for Multi-Dimension Range Queries on Synthetic Datasets

Figure 4 reports the MSE of multi-dimensional range queries on the four synthetic datasets. Across all distributions and privacy budgets  $\epsilon$ , UDPLDP-Tree achieves the best accuracy and shows a clear and stable advantage over all baselines. On light-tailed Gaussian data, UDPLDP-Tree yields uniformly lower MSE than PriPL-Tree, PrivNUD, and HDG, and improves over AHEAD by one to two orders of magnitude. Similar trends appear on the multi-modal MixGaussian distribution, where UDPLDP-Tree maintains a noticeable margin for all  $\epsilon$ , typically reducing the MSE by about 30~60% compared with PriPL-Tree and by several times compared with PrivNUD and HDG.



**Figure 4.** Mutil-dimension range query MSE comparison under four synthetic datasets.

The advantage of UDPLDP-Tree becomes even more pronounced on the heavy-tailed Cauchy and Zipf datasets. In these skewed settings, the MSE of AHEAD, PrivNUD, and HDG decreases slowly and remains relatively high even at large  $\epsilon$ , while UDPLDP-Tree quickly converges to the lowest error across the whole privacy regime. This behavior confirms that combining user-distinguished privacy allocation with multi-dimensional B-spline modeling and hierarchical partitioning helps UDPLDP-Tree concentrate the privacy budget on dense regions and reduce unnecessary noise in sparse areas, thereby mitigating the “curse of dimensionality” that affects existing LDP schemes in multi-dimensional range queries.

### 6.3.2. Results for Multi-Dimension Range Queries on Real-World Datasets

Figure 5 presents the MSE of multi-dimensional range queries on the four real-world datasets. The overall pattern is consistent with the synthetic results: UDPLDP-Tree consistently achieves the lowest error for all tested  $\epsilon$ . On the Adult and Loan datasets, UDPLDP-Tree significantly outperforms AHEAD, HDG, and PrivNUD, and provides a substantial improvement over PriPL-Tree. For example, in the medium-privacy regime ( $\epsilon$  between 0.4 and 1.0), UDPLDP-Tree typically reduces the MSE by roughly 2–4 $\times$  compared with PriPL-Tree, while the gap to AHEAD can reach one order of magnitude.

On the Salary and Financial datasets, where attribute correlations and frequency skew are more evident, UDPLDP-Tree remains the most accurate method across all privacy budgets. The competing schemes either suffer from large errors at small  $\epsilon$  or exhibit a much slower decrease in MSE as  $\epsilon$  grows. By contrast, UDPLDP-Tree leverages the UDPLDP mechanism and attribute-weighted budget allocation to tailor noise to users’ heterogeneous

privacy ranges, and uses the multi-dimensional UDPLDP-Tree structure to align partitions with high-density regions. These design choices enable UDPLDP-Tree to capture real-world joint distributions more precisely, resulting in robust and consistently lower MSE than existing LDP-based methods on all four datasets.

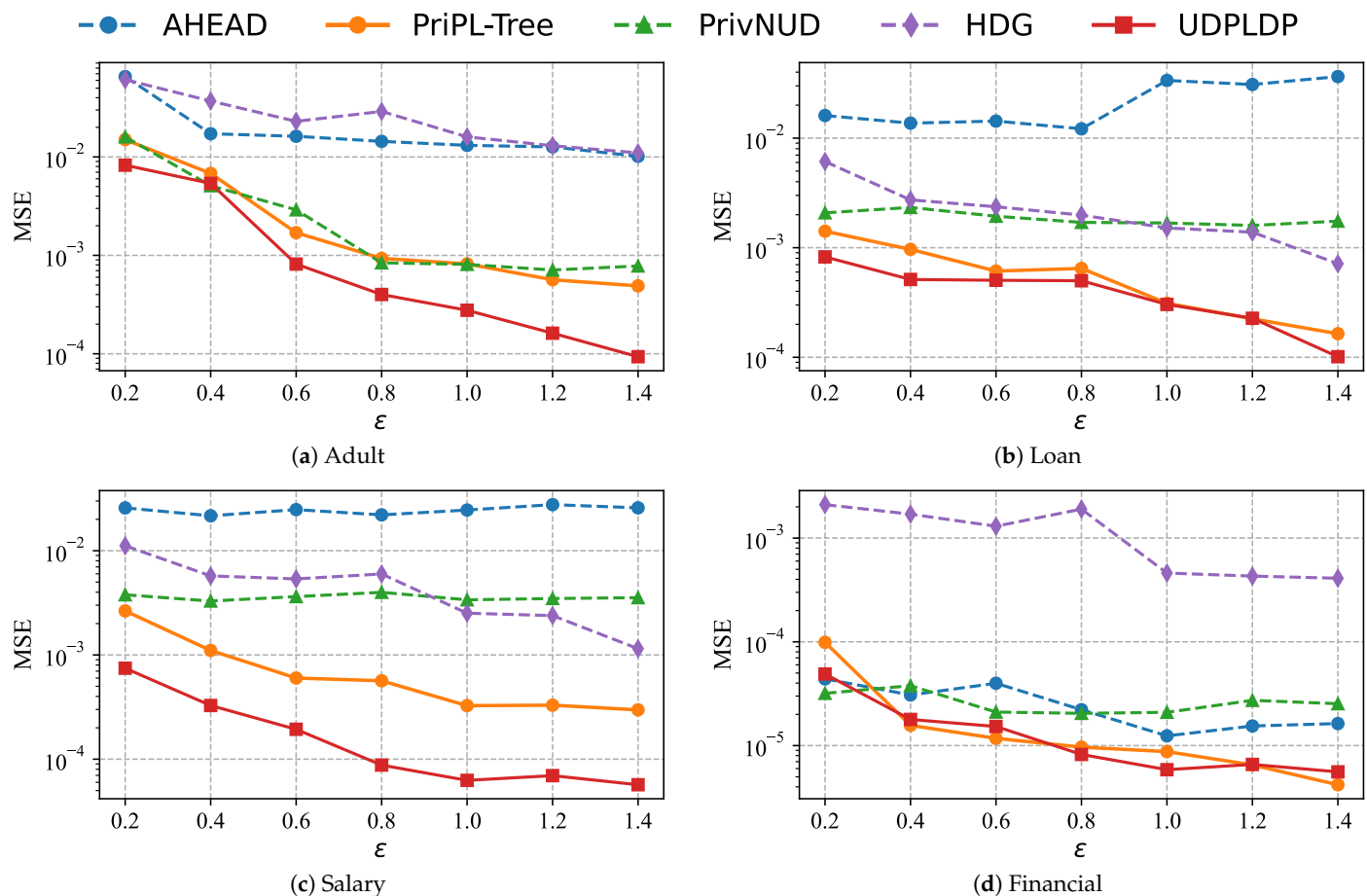


Figure 5. Mutil-dimension range query MSE comparison under four real datasets.

## 7. Related Work

This section reviews prior studies on personalized privacy protection and range query answering under both centralized DP and local DP.

### 7.1. Personalized Privacy Protection

Personalized privacy was first explored in location-centric settings, e.g., through personalized anonymization models [30] and later mechanisms designed to resist long-term observation attacks in location privacy publishing [31]. To obtain formal guarantees, personalized differential privacy (PDP) further studies heterogeneous protection through mechanism design or dataset partitioning, including adaptive personalized DP [32], heterogeneous DP via stretching transformations [33], and partitioning-based personalized DP mechanisms [34]. However, most PDP-style solutions assume the curator can access users' raw data before perturbation, which is undesirable in high-sensitivity applications.

To avoid collecting raw data, subsequent work shifts personalization to the local model and develops personalized LDP mechanisms for basic statistics. Representative examples include locally private spatial data aggregation [14], personalized histogram/frequency estimation with customized privacy levels [35,36], and personalized local aggregation mechanisms for numeric queries [37]. In parallel, value-dependent sensitivity has been

leveraged to improve distribution estimation [38], and input-discriminative protection further strengthens such value awareness under LDP [13].

### 7.2. Range Query Under DP

Range query answering under centralized DP is often built on hierarchical summaries. Early work uses tree-based decompositions together with constrained inference to enforce consistency among noisy node estimates [39,40]. Subsequent studies reduce tree noise through improved representations (e.g., wavelet transforms) [41] and more careful privacy-budget allocation and partitioning strategies tailored to data distributions and query workloads [42,43]. For multi-dimension settings, variants such as optimized quad-trees and balanced box-decomposition trees further improve geometric range counting [44,45]. Beyond trees, grid-based methods with optimized granularity are commonly used for high-dimensional queries [40], and more recent model-driven designs learn from noisy low-dimensional range answers to predict results without complex indexes [46].

### 7.3. Range Query Under LDP

Existing range query mechanisms under LDP can be broadly grouped into tree-based and grid-based approaches. Tree-based methods build hierarchical summaries over the domain and typically tune the fan-out to balance noise and bias, as in HH [15] and HIO [18]. Later improvements further reduce error by adapting the hierarchy to the data, e.g., merging low-frequency intervals (AHEAD [16]), customizing node-wise fan-out (PrivNUD [17]), or enhancing the tree with Haar wavelet transforms (DHT [15] following [41]). Grid-based methods instead discretize the domain and answer queries via cell aggregation; representative designs include hybrid dimensional grids (HDG [20]) and prefix-sum cubes that replace simple grids (PRISM [19]). Beyond these two families, some works study more general query-answering primitives in LDP, such as matrix mechanisms for linear queries that can specialize to ranges [47]. Furthermore, PriPL-Tree models non-uniform partitions using a tree with piecewise-linear fits [21].

## 8. Conclusions

In this paper, we first propose the UDPLDP framework, which supports personalized indistinguishability levels for different users. Subsequently, we leverage this framework to facilitate range queries, introducing the UDPLDP-Tree to overcome the challenges of data utility and privacy in both one-dimensional and multi-dimensional spaces. Finally, experimental evaluations validate the efficiency and effectiveness of our approach. In future work, we plan to extend the UDPLDP framework to support a wider range of complex query types to improve the data utility.

**Author Contributions:** Conceptualization, D.D., S.Z. and M.M.; methodology, D.D. and S.Z.; software, D.D.; validation, D.D., S.Z. and M.M.; formal analysis, D.D. and S.Z.; investigation, D.D.; resources, M.M.; data curation, D.D.; writing—original draft preparation, D.D.; writing—review and editing, S.Z. and M.M.; visualization, D.D.; supervision, M.M.; project administration, M.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Key Research and Development Program of Shaanxi (Grant No. 2024GX-ZDCYL-01-08) and the National Natural Science Foundation of China (Grant No. 62272386).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The datasets used in this study are openly available in the UCI Machine Learning Repository and Kaggle at the URLs/DOI provided in References [22–25]. Synthetic datasets were generated according to the distributions described in the paper.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Appendix A

We summarize the key notations used throughout this paper in Table A1.

**Table A1.** Key notations.

| Symbol                         | Description                                                   |
|--------------------------------|---------------------------------------------------------------|
| $\mathcal{D}$                  | One-dimensional discrete data domain.                         |
| $\mathcal{U}$                  | Set of users.                                                 |
| $\tau_u$                       | Privacy range of user $u$ .                                   |
| $\varepsilon_u$                | Privacy budget of user $u$ .                                  |
| $\mathcal{E}$                  | Ordered set of privacy levels.                                |
| $\mathcal{M}$                  | Local perturbation mechanism.                                 |
| $(a_p, b_p)$                   | Perturbation parameters at privacy level $p$ .                |
| $N$                            | Total number of users.                                        |
| $n_p$                          | Level- $p$ users whose privacy range covers value $v$ .       |
| $f_v, \hat{f}_v$               | True/estimated global frequency of value $v$ .                |
| $\mathcal{U}_{\text{model}}$   | Users used for B-spline-based distribution modeling.          |
| $\mathcal{U}_{\text{tree}}$    | Users reserved for UDPLDP-Tree.                               |
| $x_k, \hat{f}_k, Q_k$          | Center, noisy frequency and data point $(x_k, \hat{f}_k)^T$ . |
| $s$                            | Degree of the B-spline curve.                                 |
| $I_j = [\zeta_j, \zeta_{j+1})$ | $j$ -th domain interval (leaf of UDPLDP-Tree).                |
| $\eta_k$                       | Fraction of users assigned to node $n_k$ .                    |
| $w_k$                          | Query-usage weight of node $n_k$ .                            |

## References

1. Dwork, C.; McSherry, F.; Nissim, K.; Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 265–284.
2. Ji, S.; Li, W.; Srivatsa, M.; Beyah, R. Structural data de-anonymization: Quantification, practice, and implications. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, 3–7 November 2014*; Association for Computing Machinery: New York, NY, USA, 2014; pp. 1040–1053.
3. Papernot, N.; Abadi, M.; Erlingsson, U.; Goodfellow, I.; Talwar, K. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv* **2016**, arXiv:1610.05755.
4. Ye, Q.; Hu, H.; Huang, K.; Au, M.H.; Xue, Q. Stateful switch: Optimized time series release with local differential privacy. In *Proceedings of the IEEE INFOCOM 2023—IEEE Conference on Computer Communications, New York, NY, USA, 17–20 May 2023*; IEEE: New York, NY, USA, 2023; pp. 1–10.
5. Narayanan, A.; Shmatikov, V. Robust De-anonymization of Large Datasets (How to Break Anonymity of the Netflix Prize Dataset). *arXiv* **2008**, arXiv:cs/0610105.
6. de Montjoye, Y.A.; Hidalgo, C.A.; Verleysen, M.; Blondel, V.D. Unique in the Crowd: The privacy bounds of human mobility. *Sci. Rep.* **2013**, *3*, 1376.
7. Sweeney, L. Simple Demographics Often Identify People Uniquely. *Health (San Fr.)* **2000**; pp. 1–35.
8. Raskhodnikova, S.; Smith, A.; Lee, H.K.; Nissim, K.; Kasiviswanathan, S.P. What can we learn privately. In *Proceedings of the 54th Annual Symposium on Foundations of Computer Science, Washington, DC, USA, 25–28 October 2008*; IEEE: New York, NY, USA, 2008; pp. 531–540.
9. Duchi, J.C.; Jordan, M.I.; Wainwright, M.J. Local privacy and statistical minimax rates. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, Berkeley, CA, USA, 26–29 October 2013*; IEEE: New York, NY, USA, 2013; pp. 429–438.
10. Erlingsson, Ú.; Pihur, V.; Korolova, A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 3–7 November 2014, Scottsdale, AZ, USA*; Association for Computing Machinery: New York, NY, USA, 2014; pp. 1054–1067.

11. Fanti, G.; Pihur, V.; Erlingsson, Ú. Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries. *arXiv* **2015**, arXiv:1503.01214.
12. Ding, B.; Kulkarni, J.; Yekhanin, S. Collecting telemetry data privately. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017*; Curran Associates Inc.: Red Hook, NY, USA, 2017; Volume 30.
13. Gu, X.; Li, M.; Xiong, L.; Cao, Y. Providing input-discriminative protection for local differential privacy. In *Proceedings of the 2020 IEEE 36th International Conference on Data Engineering (ICDE), Dallas, TX, USA, 20–24 April 2020*; IEEE: New York, NY, USA, 2020; pp. 505–516.
14. Chen, R.; Li, H.; Qin, A.K.; Kasiviswanathan, S.P.; Jin, H. Private spatial data aggregation in the local setting. In *Proceedings of the 2016 IEEE 32nd International Conference on Data Engineering (ICDE), Helsinki, Finland, 16–20 May 2016*; IEEE: New York, NY, USA, 2016; pp. 289–300.
15. Kulkarni, T. Answering range queries under local differential privacy. In *Proceedings of the 2019 International Conference on Management of Data, New York, NY, USA, 30 June–5 July 2019*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 1832–1834.
16. Du, L.; Zhang, Z.; Bai, S.; Liu, C.; Ji, S.; Cheng, P.; Chen, J. AHEAD: Adaptive hierarchical decomposition for range query under local differential privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual, 15–19 November 2021*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 1266–1288.
17. Wang, N.; Wang, Y.; Wang, Z.; Nie, J.; Wei, Z.; Tang, P.; Gu, Y.; Yu, G. Privnud: Effective range query processing under local differential privacy. In *Proceedings of the 2023 IEEE 39th International Conference on Data Engineering (ICDE), Anaheim, CA, USA, 3–7 April 2023*; IEEE: New York, NY, USA, 2023; pp. 2660–2672.
18. Wang, T.; Ding, B.; Zhou, J.; Hong, C.; Huang, Z.; Li, N.; Jha, S. Answering multi-dimensional analytical queries under local differential privacy. In *Proceedings of the 2019 International Conference on Management of Data*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 159–176.
19. Wang, Y.; Cheng, X. Prism: Prefix-sum based range queries processing method under local differential privacy. In *Proceedings of the 2022 IEEE 38th International Conference on Data Engineering (ICDE), Virtual, 30 June–5 July 2019*; IEEE: New York, NY, USA, 2022; pp. 433–445.
20. Yang, J.; Wang, T.; Li, N.; Cheng, X.; Su, S. Answering multi-dimensional range queries under local differential privacy. *arXiv* **2020**, arXiv:2009.06538.
21. Wang, L.; Ye, Q.; Hu, H.; Meng, X. PriPL-Tree: Accurate Range Query for Arbitrary Distribution under Local Differential Privacy. *arXiv* **2024**, arXiv:2407.13532.
22. Wang, T.; Lopuhaä-Zwakenberg, M.; Li, Z.; Skoric, B.; Li, N. Locally differentially private frequency estimation with consistency. *arXiv* **2019**, arXiv:1905.08320.
23. Wang, T.; Blocki, J.; Li, N.; Jha, S. Locally differentially private protocols for frequency estimation. In *Proceedings of the 26th USENIX Security Symposium (USENIX Security 17), Vancouver, BC, Canada, 16–18 August 2017*; USENIX Association: Berkeley, CA, USA, 2017; pp. 729–745.
24. De Boor, C.; De Boor, C. *A Practical Guide to Splines*; Springer: New York, NY, USA, 1978; Volume 27.
25. Taheri, R.; Shojafar, M.; Arabikhan, F.; Gegov, A. Unveiling vulnerabilities in deep learning-based malware detection: Differential privacy driven adversarial attacks. *Comput. Secur.* **2024**, *146*, 104035.
26. Becker, B.; Kohavi, R. Adult. 1996. Available online: <https://archive.ics.uci.edu/dataset/2/adult> (accessed on 4 February 2026).
27. Kaggle. All Lending Club Loan Data. 2007. <https://www.kaggle.com/datasets/wordsforthewise/lending-club> (accessed on 29 December 2025).
28. Kaggle. SF Salaries. 2017. Available online: <https://www.kaggle.com/datasets/kaggle/sf-salaries> (accessed on 29 December 2025).
29. Kaggle. Synthetic Financial Datasets for Fraud Detection. 2017. Available online: <https://www.kaggle.com/datasets/ealaxi/paysim1> (accessed on 29 December 2025).
30. Gedik, B.; Liu, L. Location privacy in mobile systems: A personalized anonymization model. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05), 6–10 June 2005, Columbus, OH, USA*; IEEE: New York, NY, USA, 2005; pp. 620–629.
31. Niu, B.; Chen, Y.; Wang, Z.; Li, F.; Wang, B.; Li, H. Eclipse: Preserving differential location privacy against long-term observation attacks. *IEEE Trans. Mob. Comput.* **2020**, *21*, 125–138.
32. Niu, B.; Chen, Y.; Wang, B.; Wang, Z.; Li, F.; Cao, J. AdaPDP: Adaptive personalized differential privacy. In *Proceedings of the IEEE INFOCOM 2021—IEEE Conference on Computer Communications, Vancouver, BC, Canada, 10–13 May 2021*; IEEE: New York, NY, USA, 2021; pp. 1–10.
33. Alaggan, M.; Gambs, S.; Kermarrec, A.M. Heterogeneous differential privacy. *arXiv* **2015**, arXiv:1504.06998.

34. Li, H.; Xiong, L.; Ji, Z.; Jiang, X. Partitioning-based mechanisms under personalized differential privacy. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Jeju, Republic of Korea, 23–26 May 2017*; Springer: Cham, Switzerland, 2017; pp. 615–627.
35. Wang, S.; Huang, L.; Tian, M.; Yang, W.; Xu, H.; Guo, H. Personalized privacy-preserving data aggregation for histogram estimation. In *Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, USA, 9–10 December 2015*; IEEE: New York, NY, USA, 2015; pp. 1–6.
36. Yiwen, N.; Yang, W.; Huang, L.; Xie, X.; Zhao, Z.; Wang, S. A utility-optimized framework for personalized private histogram estimation. *IEEE Trans. Knowl. Data Eng.* **2018**, *31*, 655–669.
37. Akter, M.; Hashem, T. Computing aggregates over numeric data with personalized local differential privacy. In *Proceedings of the Australasian Conference on Information Security and Privacy, Auckland, New Zealand, 23–26 May 2017*; Springer: Cham, Switzerland, 2017; pp. 249–260.
38. Murakami, T.; Kawamoto, Y. {Utility-optimized} local differential privacy mechanisms for distribution estimation. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security 19), Santa Clara, CA, USA, 14–16 August 2019*; USENIX Association: Berkeley, CA, USA, 2019; pp. 1877–1894.
39. Hay, M.; Rastogi, V.; Miklau, G.; Suciu, D. Boosting the accuracy of differentially-private histograms through consistency. *arXiv* **2009**, arXiv:0904.0942.
40. Qardaji, W.; Yang, W.; Li, N. Differentially private grids for geospatial data. In *Proceedings of the 2013 IEEE 29th International Conference on Data Engineering (ICDE), Brisbane, Australia, 8–13 April 2013*; IEEE: New York, NY, USA, 2013; pp. 757–768.
41. Xiao, X.; Wang, G.; Gehrke, J. Differential privacy via wavelet transforms. *IEEE Trans. Knowl. Data Eng.* **2010**, *23*, 1200–1214.
42. Cormode, G.; Procopiuc, C.; Srivastava, D.; Shen, E.; Yu, T. Differentially private spatial decompositions. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, Arlington, VA, USA, 1–5 April 2012*; IEEE: New York, NY, USA, 2012; pp. 20–31.
43. Li, C.; Hay, M.; Miklau, G.; Wang, Y. A data-and workload-aware algorithm for range queries under differential privacy. *arXiv* **2014**, arXiv:1410.0265.
44. Zhang, J.; Xiao, X.; Xie, X. Privtree: A differentially private algorithm for hierarchical decompositions. In *Proceedings of the 2016 International Conference on Management of Data, San Francisco, CA, USA, 26 June–1 July 2016*; Association for Computing Machinery: New York, NY, USA, 2016; pp. 155–170.
45. Huang, Z.; Yi, K. Approximate range counting under differential privacy. In *Proceedings of the 37th International Symposium on Computational Geometry (SoCG 2021), Virtual, 7–11 June 2021*; Schloss Dagstuhl–Leibniz-Zentrum für Informatik: Wadern, Germany, 2021; pp. 45:1–45:14.
46. Zeighami, S.; Ahuja, R.; Ghinita, G.; Shahabi, C. A neural database for differentially private spatial range queries. *arXiv* **2021**, arXiv:2108.01496.
47. McKenna, R.; Maity, R.K.; Mazumdar, A.; Miklau, G. A workload-adaptive mechanism for linear queries under local differential privacy. *arXiv* **2020**, arXiv:2002.01582.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.