

On the Utility Gain of Iterative Bayesian Update for Locally Differentially Private Mechanisms^{*}

Héber H. Arcolezi, Selene Cerna, and Catuscia Palamidessi

Inria and École Polytechnique (IPP), Palaiseau, France

{heber.hwang-arcolezi,selene-leya.cerna-nahuis,catuscia.palamidessi}@inria.fr

Abstract. This paper investigates the utility gain of using Iterative Bayesian Update (IBU) for private discrete distribution estimation using data obfuscated with Locally Differentially Private (LDP) mechanisms. We compare the performance of IBU to Matrix Inversion (MI), a standard estimation technique, for seven LDP mechanisms designed for one-time data collection and for other seven LDP mechanisms designed for multiple data collections (*e.g.*, RAPPOR). To broaden the scope of our study, we also varied the utility metric, the number of users n , the domain size k , and the privacy parameter ϵ , using both synthetic and real-world data. Our results suggest that IBU can be a useful post-processing tool for improving the utility of LDP mechanisms in different scenarios without any additional privacy cost. For instance, our experiments show that IBU can provide better utility than MI, especially in high privacy regimes (*i.e.*, when ϵ is small). Our paper provides insights for practitioners to use IBU in conjunction with existing LDP mechanisms for more accurate and privacy-preserving data analysis. Finally, we implemented IBU for all fourteen LDP mechanisms into the state-of-the-art `multi-freq-ldpy` Python package (<https://pypi.org/project/multi-freq-ldpy/>) and open-sourced all our code used for the experiments as tutorials.

Keywords: Expectation-Maximization · Iterative Bayesian Update · Local Differential Privacy · Distribution Estimation.

1 Introduction

The widespread availability of Big Data has led to the development of new methods for extracting valuable insights from large datasets. However, the increased capacity to collect and analyze data also raises significant concerns about privacy, particularly in cases where sensitive information about individuals is involved. Thus, the direct collection and storage of users' raw data on a centralized server should be avoided as these data are subject to illegal access [21] or internal fraud [28], for example. To address this issue, recent works have proposed several mechanisms satisfying Differential Privacy (DP) [12, 13], in the distributed setting, referred to as Local DP (LDP) [19].

^{*} Version of Record (DBSec'23): https://doi.org/10.1007/978-3-031-37586-6_11.

One of the strengths of LDP is its simple trust model: since each user sanitizes their data locally, user privacy is protected even if the server is malicious. However, the increased privacy of the local DP model comes at the cost of reduced utility. That is, as each user’s data is obfuscated, several lower bounds exist on the error of these LDP mechanisms. This strongly differentiates the local DP model from the central DP model. [8]. To address this issue, one line of research has largely explored the issue of improving the utility of LDP distribution estimation mechanisms [7, 16, 18, 24, 25, 30], in which the data collector estimates the number of users for each possible value based on the sanitized data of the users. Distribution estimation is a primary objective of LDP [29], as it is a building block for more complex tasks (*e.g.*, heavy hitter estimation [7], marginal estimation [9, 17, 20, 23], and distribution monitoring [3, 6, 16]).

Two commonly used estimators for distribution estimation under LDP are [29]: Matrix Inversion (MI), which is based on simple linear algebra techniques, and Iterative Bayesian Update (IBU) [1, 2], which is based on the well-known Expectation-Maximization [10] algorithm. Thus, another line of research has explored how to improve the estimation at the server side through post-processing techniques [14, 15, 22, 26]. For instance, to improve the utility of the MI estimator, in [18], the authors evaluate two post-processing techniques, namely, normalization (also adopted in this paper) and projection on the simplex. More recently, Wang *et al.* [26] revisited and introduced, in total, ten post-processing techniques for the MI estimator. Regarding IBU, ElSalamouny & Palamidessi [14, 15] investigated a generalization of IBU for personalized LDP and, in [22], the authors improved IBU considering small number of users.

Main contributions. While the aforementioned works have explored how to improve the estimation of LDP mechanisms on the server side through post-processing techniques, no study has compared the performance of MI and IBU estimators for more than three LDP mechanisms. This paper aims to fill this research gap by conducting an in-depth empirical analysis of the utility gain by using IBU instead of MI for *fourteen state-of-the-art LDP mechanisms*. More precisely, we have experimentally assessed seven state-of-the-art LDP distribution estimation mechanisms, namely, Generalized Randomized Response (GRR) [18], Binary Local Hashing (BLH) [7], Optimal Local Hashing (OLH) [25], Symmetric Unary Encoding (SUE) [16], Optimal Unary Encoding (OUE) [25], Subset Selection (SS) [24, 30], and Thresholding with Histogram Encoding (THE) [25]. In addition, we also extended the analysis to seven state-of-the-art memoization-based LDP mechanisms, namely, Longitudinal GRR (L-GRR) [3], four Longitudinal Unary Encoding (L-UE) [3, 16] mechanisms, and two Longitudinal Local Hashing (L-LH) [6] mechanisms. To further broaden the scope of our study, we have evaluated two popular utility metrics by varying the number of users n , the privacy guarantee ϵ , and the domain size k , with both synthetic and real-world datasets. Finally, this paper open-sources the implementation of IBU for all aforementioned LDP mechanisms into the `multi-freq-ldpy` Python package [4], as well as the code used for our experiments in the form of tutorials.

In summary, the three main contributions of this paper are:

- We present an in-depth analysis of the utility gain of IBU over MI for seven state-of-the-art LDP mechanisms designed for one-time data collection;
- To the authors’ knowledge, we are the first to extend IBU and conduct a detailed analysis of its utility gain over MI for seven state-of-the-art longitudinal LDP mechanisms designed for multiple data collections;
- We open-sourced the IBU implementation for all aforementioned LDP mechanisms into `multi-freq-ldpy` [4], facilitating future research in the area and making it easier for researchers to replicate and build upon our findings.

Outline. The rest of this paper is organized as follows. Section 2 introduces the notation, the problem, and the metrics, and briefly reviews LDP and the two estimators, *i.e.*, MI and IBU. In Section 3, we present all LDP distribution estimation mechanisms analyzed in this paper. Next, Section 4 details the experimental setting and main results before discussing related work in Section 5. Finally, we conclude this work indicating future perspectives in Section 6.

2 Preliminaries

In this section, we present the notation, the problem, the utility metrics, the LDP privacy model, and both MI and IBU estimators, used in this paper.

2.1 Notations & Problem Statement

The main notation used throughout this paper is summarized in Table 1. This paper considers a distributed setting with n users and one untrusted server. Each user holds a value $v \in V$, where V ranges on a finite domain \mathcal{D} of size k . The distribution of V is represented by $\mathbf{f} = \{f(v)\}_{v \in \mathcal{D}} \in [0, 1]^k$, *i.e.*, $\mathbf{f}(v)$ is the probability of $v \in \mathcal{D}$. To satisfy LDP, each user i , for $i \in [1..n]$, will apply an obfuscation mechanism $\mathcal{M}_{(\epsilon)}$ to their value v^i to obtain an output $y^i = \mathcal{M}_{(\epsilon)}(v^i)$ from a finite set Y ranging on \mathcal{D} . The obfuscation mechanism $\mathcal{M}_{(\epsilon)}$ maps $v \in \mathcal{D}$ to $y \in \mathcal{D}$ through a channel matrix A_{vy} , *i.e.*, the probability that $\mathcal{M}_{(\epsilon)}$ yields y from v . Upon collecting the obfuscated data of all users, the server computes the observed distribution of y^i denoted as $\tilde{\mathbf{f}} = \{\tilde{f}(v)\}_{v \in \mathcal{D}} \in [0, 1]^k$, where $\tilde{\mathbf{f}}(v)$ is the probability of $v \in \mathcal{D}$ and is calculated by counting the number of times that v is observed in Y and divided by n . The server’s goal is to estimate \mathbf{f} by calculating $\tilde{\mathbf{f}}$ and with the knowledge of the obfuscation mechanism $\mathcal{M}_{(\epsilon)}$. We denote the estimated distribution of \mathbf{f} by $\hat{\mathbf{f}} = \{\hat{f}(v)\}_{v \in \mathcal{D}}$.

As utility metrics, we use the Mean Squared Error (MSE) and the Mean Absolute Error (MAE) to measure the difference between the original distribution \mathbf{f} and estimated distribution $\hat{\mathbf{f}}$, which are formally defined as:

- **MSE.** Given \mathbf{f} and $\hat{\mathbf{f}}$, the MSE is an expectation of the squared error:

$$\text{MSE} = \frac{1}{k} \sum_{v \in \mathcal{D}} (f(v) - \hat{f}(v))^2.$$

Symbol	Description
\mathcal{D}	Data domain.
k	Domain size $k = \mathcal{D} $.
n	Number of users.
V	Discrete set of original data $\{v^1, v^2, v^3, \dots, v^n\}$, ranging on \mathcal{D} .
Y	Discrete set of obfuscated data $\{y^1, y^2, y^3, \dots, y^n\}$, ranging on \mathcal{D} .
\mathbf{f}	Distribution of original data, where $\mathbf{f} = \{f(v)\}_{v \in \mathcal{D}} \in [0, 1]^k$.
$\tilde{\mathbf{f}}$	Distribution of obfuscated data, where $\tilde{\mathbf{f}} = \{\tilde{f}(v)\}_{v \in \mathcal{D}} \in [0, 1]^k$.
$\hat{\mathbf{f}}$	Estimated distribution of \mathbf{f} , where $\hat{\mathbf{f}} = \{\hat{f}(v)\}_{v \in \mathcal{D}}$.
$\mathcal{M}_{(\epsilon)}$	ϵ -LDP obfuscation mechanism, $\mathcal{M}_{(\epsilon)} : V \rightarrow Y$.
A_{vy}	Channel matrix, probability that $\mathcal{M}_{(\epsilon)}$ yields $y \in \mathcal{D}$ from $v \in \mathcal{D}$.
Γ	Utility gain of IBU over MI defined in Eq. (1).
$[a, b]$	Set of all real numbers $\geq a$ and $\leq b$.
$[1..a]$	Set of integers $\{1, 2, 3, \dots, a\}$.
\mathbf{a}_i	i -th coordinate of vector \mathbf{a} .

Table 1: Notations

– **MAE.** Given \mathbf{f} and $\hat{\mathbf{f}}$, the MAE is an expectation of the absolute error:

$$\text{MAE} = \frac{1}{k} \sum_{v \in \mathcal{D}} |f(v) - \hat{f}(v)|.$$

In this paper, we are interested in comparing the two main statistical estimators for $\hat{\mathbf{f}}$, namely, MI and IBU. Based on the two popular metrics MSE and MAE, we define the utility gain Γ of IBU over MI as follows:

$$\Gamma(\%) = 100 \cdot \max \left(\frac{\text{Metric}_{\text{MI}} - \text{Metric}_{\text{IBU}}}{\text{Metric}_{\text{MI}}}, 0 \right), \quad (1)$$

that is, the utility gain is clipped to zero to represent no gain.

2.2 Local Differential Privacy (LDP)

In this paper, we use LDP [19] as the privacy model considered, formalized as:

Definition 1 (ϵ -Local Differential Privacy). A randomized mechanism \mathcal{M} satisfies ϵ -LDP, where $\epsilon > 0$, if for any input $v, v' \in \mathcal{D}$, we have:

$$\forall y \in \mathcal{D} : \Pr[\mathcal{M}(v) = y] \leq e^\epsilon \cdot \Pr[\mathcal{M}(v') = y].$$

Proposition 1 (Post-Processing [13]). If \mathcal{M} is ϵ -LDP, then for any function g , the composition of \mathcal{M} and g , i.e., $g(\mathcal{M})$ satisfies ϵ -LDP.

The parameter ϵ controls the privacy-utility trade-off. Note that lower values of ϵ result in tighter privacy protection and vice versa. Moreover, an LDP mechanism is called “pure LDP” if it satisfies the following definition [25].

Definition 2 (Pure LDP [25]). An ϵ -LDP mechanism is pure if there are two probability parameters $0 < q^* < p^* < 1$ such that for all $v \in \mathcal{D}$,

$$\begin{aligned} \Pr[\mathcal{M}(v) \in \{y|v \in S(y)\}] &= p^*, \\ \forall v \neq v' \in \mathcal{D} \quad \Pr[\mathcal{M}(v') \in \{y|v \in S(y)\}] &= q^*, \end{aligned}$$

where $S(y)$ is the set of items that y supports. That is, $S(y)$ maps each possible output y to a set of input values that y “supports”.

2.3 Matrix Inversion (MI) Estimator

All obfuscation mechanisms we analyze in this paper are pure LDP (cf. Definition 2), which makes their analysis advantageous. For instance, for any pure LDP mechanism, the server estimates the distribution $\hat{\mathbf{f}} = \{\hat{f}(v)\}_{v \in \mathcal{D}} \in \mathbb{R}^k$ as [25]:

$$\forall v \in \mathcal{D}, \quad \hat{f}(v) = \frac{\sum_{i=1}^n \mathbb{1}_{S(y^i)}(v^i) - nq^*}{n(p^* - q^*)}, \quad (2)$$

in which y^i represents the obfuscated data provided by each user (for $i \in [1..n]$), and the function $\mathbb{1}_{S(y^i)}(v^i)$ equals 1 if y^i supports the value v^i , and 0 otherwise. However, when the number of users is small, many elements estimated with Eq. (2) can be negative. Therefore, we clip the negative estimates to 0 and re-normalize the estimated frequencies so that they sum up to 1, ensuring that they constitute a valid probability distribution. Another way to write Eq. (2) is: $\hat{\mathbf{f}} = \tilde{\mathbf{f}} A_{vy}^{-1}$, in which $\tilde{\mathbf{f}}$ is the observed distribution and A_{vy}^{-1} is the inverse of A_{vy} [14, 18, 22]. For any pure LDP mechanism, we observed that the channel matrix A_{vy} is a square matrix with values p^* in the diagonal and q^* otherwise.

2.4 Iterative Bayesian Update (IBU) Estimator

The IBU estimator [1, 2, 14] is based on the Expectation Maximization (EM) [10] method, which is a powerful method for obtaining parameter estimates when part of the data is missing. IBU estimates the distribution of \mathbf{f} as follows. Let $\tilde{\mathbf{f}}$ be the observed distribution on Y , A_{vy} be the channel matrix, which represents the probability of obtaining y from v , and $\hat{\mathbf{f}}$ be the estimated distribution on V . IBU starts with a full-support distribution $\hat{\mathbf{f}}^0$ on V (e.g., the uniform distribution). Next, it iteratively generates new distributions by updating Eq. (3), where $(*)$, (\cdot) , and $(/)$ represent the element-wise product, dot product, and element-wise division, respectively. Finally, it converges until a specified tolerance is reached.

$$\hat{\mathbf{f}}^{t+1} = \tilde{\mathbf{f}} \cdot \frac{\hat{\mathbf{f}}^t * A_{vy}}{\hat{\mathbf{f}}^t \cdot A_{vy}}. \quad (3)$$

3 LDP Distribution Estimation Mechanisms

In this section, we briefly review fourteen state-of-the-art LDP distribution estimation mechanisms, *i.e.*, seven for one-time collection (Section 3.1) and seven for multiple collections (Section 3.2).

3.1 LDP Mechanisms for One-Time Data Collection

Algorithm 1 exhibits the general procedure for one-time discrete distribution estimation under pure LDP. The five following subsections briefly present state-of-the-art pure LDP mechanisms.

Algorithm 1 General pure LDP procedure for distribution estimation.

Input : Original data of users, privacy parameter ϵ , mechanism $\mathcal{M}_{(\epsilon)}$.
Output : Estimated discrete distribution.

User-side

- 1: **for** each user $i \in [1..n]$ with input data $v^i \in V$ **do**
- 2: **Encode**(v^i) into a specific format (**if needed**);
- 3: **Obfuscate**(v^i) as $y^i = \mathcal{M}_{(\epsilon)}(v^i)$;
- 4: Transmit y^i to the aggregator.
- 5: **end for**

Server-side

- 6: Obtain the support set $S(y)$ and probabilities p^* and q^* for $\mathcal{M}_{(\epsilon)}$.
- 7: **Estimate** Aggregate the obfuscated data y^i ($i \in [1..n]$) to estimate $\{\hat{f}(v)\}_{v \in \mathcal{D}}$.
- 8: **return** : Estimated discrete distribution $\hat{\mathbf{f}}$ (*i.e.*, a k -bins histogram).

Generalized Randomized Response (GRR) The GRR [18] mechanism generalizes the Randomized Response (RR) surveying technique proposed by Warner [27] for $k \geq 2$ while satisfying ϵ -LDP.

Encode. GRR [18] uses no particular encoding, *i.e.*, $\text{Encode}_{\text{GRR}}(v) = v$.

Obfuscate. Given the user's personal data v , GRR outputs v with probability p , and any other randomly chosen value $v' \in \mathcal{D} \setminus \{v\}$ otherwise. More formally:

$$\forall y \in \mathcal{D} : \quad \Pr[\mathcal{M}_{\text{GRR}(\epsilon, k)}(v) = y] = \begin{cases} p = \frac{e^\epsilon}{e^\epsilon + k - 1}, & \text{if } y = v, \\ q = \frac{1}{e^\epsilon + k - 1}, & \text{otherwise,} \end{cases} \quad (4)$$

in which y is the obfuscated value sent to the server.

Estimate. For aggregation at the server side, it is important to obtain the support set $S(y)$ and the probabilities p^* and q^* . For GRR, an obfuscated value y only supports itself, *i.e.*, $S_{\text{GRR}}(y) = \{y\}$. Given the support set and with $p^* = p$ and $q^* = q$, the server can estimate item frequencies using Eq. (2) for MI and Eq. (3) for IBU.

Local Hashing (LH) LH mechanisms [7, 25] can handle a large domain size k by using hash functions to map an input data to a smaller domain of size g , for $2 \leq g \ll k$, and then applying GRR to the hashed value. Let \mathcal{H} be a universal hash function family such that each hash function $H \in \mathcal{H}$ hashes a value $v \in \mathcal{D}$ into $[1..g]$, *i.e.*, $H : \mathcal{D} \rightarrow [1..g]$. There are two variations of LH mechanisms:

- **Binary LH (BLH)** [7]. A simple case that just sets $g = 2$;
- **Optimal LH (OLH)** [25]. An optimized version that selects $g = \lfloor e^\epsilon + 1 \rfloor$.

Encode. $\text{Encode}_{\text{LH}}(v) = \langle H, H(v) \rangle$, in which $H \in \mathcal{H}$ is selected at random.

Obfuscate. LH mechanisms only obfuscate the hash value $H(v)$ with GRR and does not change H . In particular, the LH reporting mechanism is:

$$\mathcal{M}_{\text{LH}(\epsilon)}(v) := \langle H, \mathcal{M}_{\text{GRR}(\epsilon, g)}(H(v)) \rangle,$$

in which $\mathcal{M}_{\text{GRR}(\epsilon, g)}$ is given in Eq. (4), while operating on the new domain $[1..g]$. Each user reports, the hash function and obfuscated value $\langle H, y \rangle$ to the server.

Estimate. The support set for LH mechanisms is $S_{\text{LH}}(\langle H, y \rangle) = \{v | H(v) = y\}$, *i.e.*, the set of values that are hashed into the obfuscated data y . LH mechanisms are pure with probabilities $p^* = p$ and $q^* = \frac{1}{g}$ [25]. Thus, the server can estimate item frequencies using Eq. (2) for MI and Eq. (3) for IBU.

Unary Encoding (UE) UE mechanisms [16, 25] interpret the user's input data $v \in \mathcal{D}$, as a one-hot k -dimensional vector and obfuscate each bit independently.

Encode. $\text{Encode}_{\text{UE}}(v) = \mathbf{v} = [0, \dots, 0, 1, 0, \dots, 0]$ is a binary vector with only the bit at the position corresponding to v set to 1 and the other bits set to 0.

Obfuscate. The obfuscation function of UE mechanisms randomizes the bits from \mathbf{v} independently to generate \mathbf{y} as follows:

$$\forall i \in [1..k]: \quad \Pr[\mathbf{y}_i = 1] = \begin{cases} p, & \text{if } \mathbf{v}_i = 1, \\ q, & \text{if } \mathbf{v}_i = 0, \end{cases} \quad (5)$$

in which \mathbf{y} is sent to the server. There are two variations of UE mechanisms:

- **Symmetric UE (SUE)** [16]. Also known as Basic One-Time RAPPOR, SUE selects $p = \frac{e^{\epsilon/2}}{e^{\epsilon/2} + 1}$ and $q = \frac{1}{e^{\epsilon/2} + 1}$ in Eq. (5), such that $p + q = 1$;
- **Optimal UE (OUE)** [25]. OUE selects $p = \frac{1}{2}$ and $q = \frac{1}{e^\epsilon + 1}$ in Eq. (5).

Estimate. An obfuscated vector \mathbf{y} supports an item v if and only if the v -th bit of \mathbf{y} , denoted as \mathbf{y}_v , equals to 1. Formally, we have $S_{\text{UE}}(\mathbf{y}) = \{i | \mathbf{y}_i = 1\}$, for $i \in [1..k]$. UE mechanisms are pure with $p^* = p$ and $q^* = q$. Thus, the server can estimate item frequencies using Eq. (2) for MI and Eq. (3) for IBU.

Subset Selection (SS) The SS [24, 30] mechanism was proposed for the case where the obfuscation output is a subset of values Ω of the original domain V . The user's true value $v \in \mathcal{D}$ has higher probability of being included in the subset Ω , compared to the other values in $\mathcal{D} \setminus \{v\}$. The optimal subset size that minimizes the MSE is $\omega = \lfloor \frac{k}{e^\epsilon + 1} \rfloor$.

Encode. $\text{Encode}_{\text{SS}}(v) = \Omega = \emptyset$ is an empty subset.

Obfuscate. Given the empty subset Ω , the true value v is added to Ω with probability $p = \frac{\omega e^\epsilon}{\omega e^\epsilon + k - \omega}$. Finally, it adds values to Ω as follows:

- If $v \in \Omega$, then $\omega - 1$ values are sampled from $\mathcal{D} \setminus \{v\}$ uniformly at random (without replacement) and are added to Ω ;
- If $v \notin \Omega$, then ω values are sampled from $\mathcal{D} \setminus \{v\}$ uniformly at random (without replacement) and are added to Ω .

Afterward, the user sends the subset Ω to the server.

Estimate. An obfuscated subset Ω supports an item v if and only if $v \in \Omega$. Formally, we have $S_{SS}(\Omega) = \{v | v \in \Omega\}$. The SS mechanism is pure with $p^* = p$ and $q^* = \frac{\omega e^\epsilon (\omega - 1) + (k - \omega)\omega}{(k - 1)(\omega e^\epsilon + k - \omega)}$. Thus, the server can estimate item frequencies using Eq. (2) for MI and Eq. (3) for IBU.

Thresholding with Histogram Encoding (THE) In Histogram Encoding (HE) [25], an input data $v \in \mathcal{D}$ is encoded as a one-hot k -dimensional histogram and each bit is obfuscated independently with the Laplace mechanism [12]. For any two input data $v, v' \in \mathcal{D}$, the L1 distance between the two vectors is $\Delta = 2$. **Encode.** $\text{Encode}_{HE}(v) = \mathbf{v} = [0.0, \dots, 0.0, 1.0, 0.0, \dots, 0.0]$, where only the v -th component is 1.0 and the other bits are set to 0.0.

Obfuscate. HE outputs \mathbf{y} such that $\mathbf{y}_i = \mathbf{v}_i + \text{Lap}\left(\frac{2}{\epsilon}\right)$, in which $\text{Lap}(\beta)$ is the Laplace distribution where $\Pr[\text{Lap}(\beta) = x] = \frac{1}{2\beta} e^{-|x|/\beta}$.

Estimate. Since Laplace noise with zero mean is added to each bit independently, a simple aggregation method is to sum the noisy counts for each bit. However, this aggregation method does not provide a support function and is not pure (a.k.a. SHE in [25]). Wang *et al.* [25] proposed THE such that the user reports (or the server computes) the support set as $S_{THE}(\mathbf{y}) = \{v | \mathbf{y}_v > \theta\}$, *i.e.*, each noise count that is $> \theta$ supports the corresponding value. According to [25], the optimal threshold value θ that minimizes the MSE = $\frac{2e^{\epsilon\theta/2} - 1}{(1 + e^{\epsilon(\theta-1/2)} - 2e^{\epsilon\theta/2})^2}$ is within $(0.5, 1)$. THE is pure with $p^* = 1 - \frac{1}{2}e^{\frac{\epsilon}{2}(\theta-1)}$ and $q^* = \frac{1}{2}e^{-\frac{\epsilon}{2}\theta}$. Thus, the server can estimate item frequencies using Eq. (2) for MI and Eq. (3) for IBU.

3.2 LDP Mechanisms for Multiple Data Collections

The LDP mechanisms presented in Section 3.1 consider a one-time data collection. In this section, we consider the server is interested in multiple collections throughout time. More precisely, for longitudinal data (*i.e.*, data that is monitored over time $t \in [\tau]$), all major LDP mechanisms make use of an internal memoization mechanism [16]. Memoization was designed to enable longitudinal collections through memorizing an obfuscated version of the true value v as $y = \mathcal{M}_{\epsilon_\infty}(v)$, and consistently reusing y as the input to a second step of obfuscation in time $t \in [\tau]$. Algorithm 2 exhibits the general procedure for longitudinal discrete distribution estimation under LDP guarantees.

Without loss of generality, we consider $\tau = 1$ (one-time collection) in this paper, which will allow us to compare all LDP mechanisms under the same privacy guarantees: (i) ϵ_∞ -LDP, which is the privacy guarantees of the first obfuscation step and the upper bound when $\tau \rightarrow \infty$; (ii) ϵ_1 -LDP, which is the

privacy guarantees of the first report by chaining two LDP mechanisms (*i.e.*, the lower bound when $\tau = 1$). Naturally, $\epsilon_1 \leq \epsilon_\infty$ and, in practice, we want $\epsilon_1 \ll \epsilon_\infty$.

The three following subsections briefly present state-of-the-art longitudinal LDP mechanisms [3, 6, 16], which are based on GRR, LH, and UE mechanisms, *i.e.*, same encoding, perturbation, and aggregation methods. Therefore, we focus on presenting the composition of two LDP mechanisms (*i.e.*, $\mathcal{M}_1 \circ \mathcal{M}_2$) and both p^* and q^* parameters that are based on the privacy guarantees $\epsilon_\infty, \epsilon_1$.

Algorithm 2 Memoization-based procedure for longitudinal distribution estimation under LDP guarantees [3, 6, 16].

Input : Original data of users, privacy parameters $\epsilon_\infty, \epsilon_1$, mechanisms $\mathcal{M}_1, \mathcal{M}_2$.
Output : Estimated discrete distribution $\hat{\mathbf{f}}$ at each $t \in [\tau]$.

User-side

- 1: **for** each user $i \in [1..n]$ with input data $v^i \in V$ **do**
- 2: **Encode**(v^i) into a specific format (**if needed**);
- 3: **Obfuscate**(v^i) as $y^i = \mathcal{M}_{1(\epsilon_\infty)}(v^i)$; \triangleright First obfuscation step: p_1^* and q_1^*
- 4: **Memoize**(y^i) for v^i .
- 5: **for** each time $t \in [\tau]$ **do**:
- 6: **Obfuscate**(y^i) as $y_t^i = \mathcal{M}_{2(\epsilon_1)}(y^i)$; \triangleright Second obfuscation step: p_2^* and q_2^*
- 7: Transmit y_t^i to the aggregator.
- 8: **end for**
- 9: **end for**

Server-side

- 10: Obtain the support set $S(y)$ and probabilities p_1^*, q_1^*, p_2^* , and q_2^* for $\mathcal{M}_{1(\epsilon_\infty)}, \mathcal{M}_{2(\epsilon_1)}$.
- 11: **for** each time $t \in [\tau]$ **do**:
- 12: **Estimate** Aggregate the obfuscated data y_t^i ($i \in [1..n]$) to estimate $\{\hat{f}(v)\}_{v \in \mathcal{D}}$.
- 13: **end for**

Longitudinal GRR (L-GRR) The L-GRR [3] mechanism chains GRR in both obfuscation steps of Algorithm 2 (*i.e.*, $\text{GRR} \circ \text{GRR}$).

L-GRR Parameters:

- First obfuscation step: $p_1^* = \frac{e^{\epsilon_\infty}}{e^{\epsilon_\infty} + k - 1}$ and $q_1^* = \frac{1 - p_1}{k - 1}$;
- Second obfuscation step: $p_2^* = \frac{e^{\epsilon_\infty + \epsilon_1} - 1}{-ke^{\epsilon_1} + (k - 1)e^{\epsilon_\infty} + e^{\epsilon_1} + e^{\epsilon_\infty} - 1}$ and $q_2^* = \frac{1 - p_2}{k - 1}$.
- Aggregation: $p^* = p_1^*p_2^* + q_1^*q_2^*$ and $q^* = p_1^*q_2^* + q_1^*p_2^*$.

Longitudinal UE (L-UE) mechanisms Arcolezi *et al.* [3] analyzed all four combinations between OUE and SUE in both obfuscation steps, *i.e.*, L-SUE ($\text{SUE} \circ \text{SUE}$, a.k.a. Basic RAPPOR [16]), L-SOUE ($\text{SUE} \circ \text{OUE}$), L-OUE ($\text{OUE} \circ \text{OUE}$), and L-OSUE ($\text{OUE} \circ \text{SUE}$). Without loss of generality, we present the parameters of L-OSUE only and refer the readers to [3] and to multi-freq-ldpy [4] to access the parameters of the other L-UE mechanisms.

L-OSUE Parameters:

- First obfuscation step: $p_1^* = \frac{1}{2}$ and $q_1^* = \frac{1}{e^{\epsilon_\infty} + 1}$;
- Second obfuscation step: $p_2^* = \frac{e^{\epsilon_\infty} e^{\epsilon_1} - 1}{e^{\epsilon_\infty} - e^{\epsilon_1} + e^{\epsilon_\infty + \epsilon_1} - 1}$ and $q_2^* = 1 - p_2$.
- Aggregation: $p^* = p_1^* p_2^* + (1 - p_1^*) q_2^*$ and $q^* = q_1^* p_2^* + (1 - q_1^*) q_2^*$.

Longitudinal LH (L-LH) Arcolezi *et al.* [6] extended LH mechanisms for two obfuscation steps. More specifically, L-LH mechanisms use a hash function $H \in \mathcal{H}$ to map a value $v \in \mathcal{D} \rightarrow [1..g]$ and use L-GRR to obfuscate the hash value $H(v)$ in the new domain $[1..g]$. There are two variations of L-LH:

- L-BLH: A simple case that just sets $g = 2$;
- L-OLH: Let $a = e^{\epsilon_\infty}$ and $b = e^{\epsilon_1}$, this optimized version selects $g = 1 + \max \left(1, \left\lfloor \frac{1 - a^2 + \sqrt{a^4 - 14a^2 + 12ab(1 - ab) + 12a^3b + 1}}{6(a - b)} \right\rfloor \right)$.

L-LH Parameters:

- First obfuscation step: $p_1^* = \frac{e^{\epsilon_\infty}}{e^{\epsilon_\infty} + g - 1}$ and $q_1^* = \frac{1}{g}$;
- Second obfuscation step: $p_2^* = \frac{e^{\epsilon_\infty + \epsilon_1} - 1}{-ge^{\epsilon_1} + (g - 1)e^{\epsilon_\infty} + e^{\epsilon_1} + e^{\epsilon_1 + \epsilon_\infty} - 1}$ and $q_2^* = \frac{1 - p_2}{g - 1}$.
- Aggregation: $p^* = p_1^* p_2^* + q_1^* q_2^*$ and $q^* = p_1^* q_2^* + q_1^* p_2^*$.

4 Experimental Evaluation

In this section, we present the setting of our experiments and our main results.

4.1 Setup of Experiments

Environment. All algorithms are implemented in Python 3 and run on a local machine with 2.50GHz Intel Core i9 and 64GB RAM. The codes we develop for all experiments are available in the *tutorials* folder of the `multi-freq-ldpy` GitHub repository (<https://github.com/hharcolezi/multi-freq-ldpy>) [4].
IBU parameters. We fix the # iterations to 10000 and the tolerance to 10^{-12} .
Data distribution. For ease of reproducibility, we generate synthetic data following five well-known distributions and use one real-world dataset.

- **Gaussian.** We generate n samples following the Gaussian distribution with parameters $\mu = 1000$ and $\sigma^2 = 100$ and bucketize to a k -bins histogram;
- **Exponential.** We generate n samples following the Exponential distribution with parameters $\lambda = 1$ and bucketize to a k -bins histogram;
- **Uniform.** We generate n samples following the Uniform distribution in range $[100, 10000]$ and bucketize to a k -bins histogram;
- **Poisson.** We generate n samples following the Poisson distribution with parameters $\lambda = 5$ and bucketize to a k -bins histogram;
- **Triangular.** We generate n samples following the Triangular distribution with parameters $a = 100$ (right), $c = 4500$ (center), and $b = 10000$ (left) and bucketize to a k -bins histogram;

- **Real.** We query for n samples of the Income dataset, which is retrieved with the `folktables` [11] Python package. We only use the “PINCP” numerical attribute and set the parameters: `survey_year=‘2018’`, `horizon=‘5-Year’`, `survey=‘person’`. Afterward, we bucketize to a k -bins histogram.

Varying parameters. When generating our data, we considered different:

- **Domain size.** We varied the domain size k of attributes as $k \in \{2, 50, 100, 200\}$;
- **Number of users.** We varied the number of users n as $n \in \{20000, 100000\}$.

Methods evaluated. We assessed the following LDP mechanisms for:

- **One-time collection.** All seven pure LDP mechanisms from Section 3.1, *i.e.*, GRR, SUE, OUE, SS, THE, BLH, and OLH;
- **Multiple collections.** All seven longitudinal pure LDP mechanisms from Section 3.2, *i.e.*, L-GRR, L-SUE, L-SOUE, L-OUE, L-OSUE, L-BLH, and L-OLH.

Stability. As LDP protocols and the synthetic data generation procedure are randomized, we report average results over 20 runs.

Metrics. We evaluated the privacy-utility trade-off of all fourteen LDP mechanisms according to:

- **Utility metrics.** We measured the utility gain of IBU over MI following Eq. (1) for both MSE and MAE metrics;
- **Privacy guarantees.** We varied ϵ of LDP mechanisms for one-time collection as $\epsilon \in \{1, 2, 4\}$, representing high, medium, and low privacy regimes, respectively. Besides, we varied the privacy guarantees ϵ_∞ and ϵ_1 of longitudinal LDP mechanisms for multiple collections as $\epsilon_\infty \in \{2, 4, 8\}$ and $\epsilon_1 = \frac{\epsilon_\infty}{2}$. That is, the first report has the same privacy guarantees as one-time LDP mechanisms, and, thus, similar results are expected for LDP mechanisms with both one-time and longitudinal versions (*e.g.*, GRR and L-GRR).

4.2 Main Results

First, considering the real data distribution and the MSE metric, Fig. 1 (one-time LDP mechanisms) and Fig. 2 (longitudinal LDP mechanisms) illustrate the IBU utility gain calculated as in Eq. (1), by fixing $n = 20000$, $k = 100$, and varying $\epsilon \in \{1, 2, 4\}$ (resp. $\epsilon = \epsilon_1$). These two figures also illustrate the Average Gain (AvgGain) considering all LDP mechanisms within the same subplot. From Fig. 1, one can notice that IBU consistently and considerably outperformed MI for all LDP mechanisms in the medium ($\epsilon = 2$) and low privacy regimes ($\epsilon = 4$). For $\epsilon = 1$, IBU was only better for GRR, OUE, SS, and OLH, while achieving similar results for SUE and THE, and unsatisfactory results for BLH. Similar behavior can be noticed on the left-side plot of Fig. 2 in which IBU did not improve over MI for L-SUE and L-BLH. This is because both privacy guarantees

are equal for both one-time and longitudinal LDP mechanisms, *i.e.*, $\epsilon_1 = \epsilon$, and thus, similar results will occur, on expectation, when running more iterations for stability. Overall, for both Fig. 1 and Fig. 2, the higher AvgGain occurs in the medium privacy regime, followed by low and high privacy regimes, respectively.

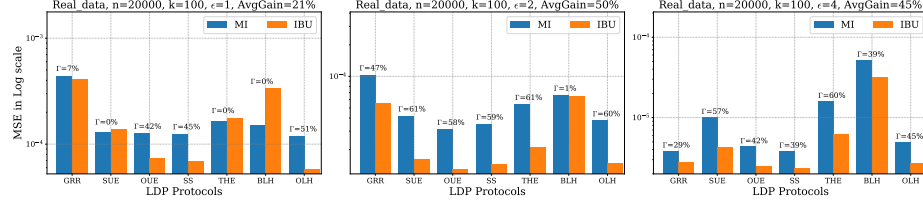


Fig. 1: IBU utility gain in % (*cf.* Eq. (1)) for each one-time pure LDP mechanism with the MSE metric, the real distribution, $n = 20000$, $k = 100$, and $\epsilon \in \{1, 2, 4\}$. AvgGain indicates the average gain considering all LDP mechanisms in the same subplot.

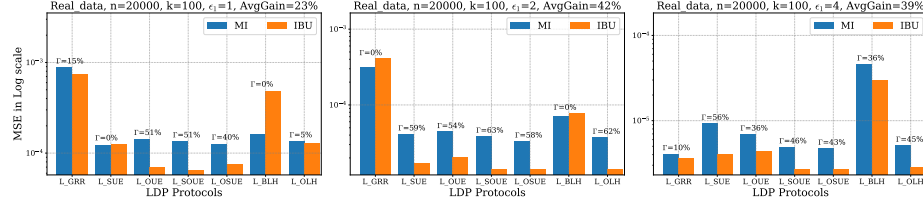


Fig. 2: IBU utility gain in % (*cf.* Eq. (1)) for each longitudinal pure LDP mechanism with the MSE metric, the real distribution, $n = 20000$, $k = 100$, $\epsilon_\infty \in \{2, 4, 8\}$, and $\epsilon_1 = \frac{\epsilon_\infty}{2}$. AvgGain indicates the average gain considering all LDP mechanisms in the same subplot.

Second, in Fig. 3 (one-time LDP mechanisms) and Fig. 4 (longitudinal LDP mechanisms), we compare the IBU utility gain in different types of distributions (including real data) by varying ϵ (left-side plots), n (centered plots), and k (right-side plots). In both figures, we observed that the uniform distribution presented the highest gain, while the Gaussian and Exponential distributions obtained the lowest gains. This is due to the fact that more values close or equal to zero are generated in Gaussian and Exponential distributions, which makes estimation difficult not only for IBU but also for MI. When analyzing the variation of ϵ in both figures, we noticed that the lower ϵ (high privacy regime), the higher the IBU gain, which is desirable in practice. Furthermore, when examining the variation of the number of users n , we noticed that the gains obtained considering all LDP mechanisms for all data distributions present a

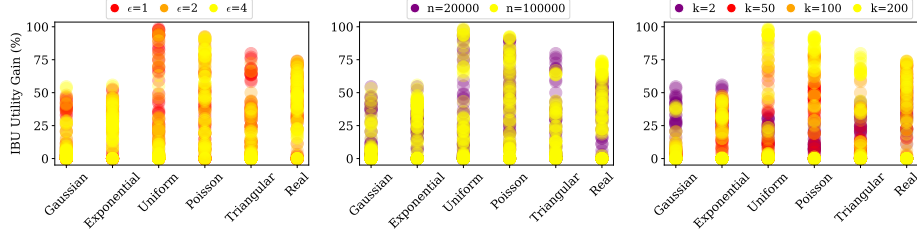


Fig. 3: IBU utility gain is in %. Each subplot shows all gains obtained by distribution type when the privacy level (ϵ), number of users (n), and domain size (k) are varied, respectively from left to right. We computed all gains with all one-time pure LDP mechanisms considered and the MSE metric.

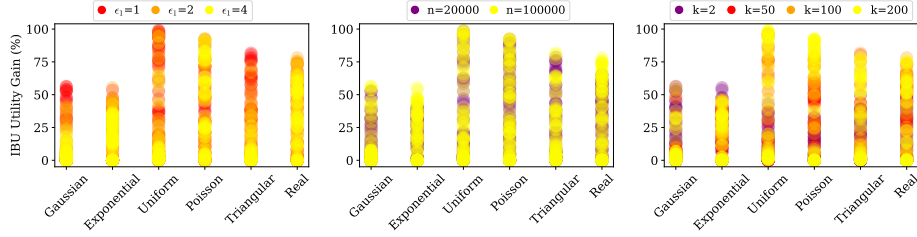


Fig. 4: IBU utility gain is in %. Each subplot shows all gains obtained by distribution type when the privacy level (ϵ), number of users (n), and domain size (k) are varied, respectively from left to right. We computed all gains with all longitudinal pure LDP mechanisms considered and the MSE metric.

better performance when n is higher, especially with longitudinal mechanisms. This is because longitudinal mechanisms have a higher variance and require more samples to reduce their estimation error [3, 6, 16]. Last, when analyzing the behavior of the domain size k , in both figures, it is shown that IBU outperforms MI, as the domain grows, *i.e.*, IBU supports data with high dimensionality, which better reflects real-world data collections.

Finally, Table 2 (one-time LDP mechanisms) and Table 3 (longitudinal LDP mechanisms) exhibit the averaged utility gain of IBU for all LDP mechanisms, all data distributions, and both MSE and MAE metrics by considering $k \in \{2, 50, 100, 200\}$, $n \in \{20000, 100000\}$, and $\epsilon \in \{1, 2, 4\}$ (resp. $\epsilon_1 = \epsilon$). From these tables, one can gather the averaged gain for each utility metric (MSE and MAE) by LDP mechanism considering all data distributions (*i.e.*, last row). For instance, in Table 2, while the THE mechanism is the one which presented the higher utility gain followed by the SUE mechanism, GRR presented the lowest utility gain. Similar results can be observed in Table 3, *i.e.*, the L-SUE mechanism presented the higher utility gain and L-GRR the lowest one. In [14], the authors also remarked that the IBU utility gain for GRR was not substantial. Indeed, while in Table 3 L-UE mechanisms showed higher utility gain than L-LH

mechanisms, in Table 2, both UE and LH present similar results. Additionally, from Table 2 and Table 3, one can also gather the averaged gain for each utility metric (MSE and MAE) by data distribution considering all LDP mechanisms (*i.e.*, last two columns). For example, for both tables, the higher utility gain was observed for the Poisson distribution followed by the real data distribution. The lower utility gain considering all LDP mechanisms was for the Gaussian distribution followed by the Triangular distribution.

Dist.	GRR		SUE		OUE		SS		THE		BLH		OLH		Avg.	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Gauss.	1	1	13	7	10	6	3	1	13	7	16	9	11	7	9	5
Exp.	16	11	26	15	27	16	19	11	26	15	16	10	27	16	22	13
Unif.	0	0	29	21	20	14	14	10	31	22	57	43	18	12	24	17
Poiss.	39	28	41	26	44	28	41	27	41	27	14	6	46	30	38	24
Triang.	0	0	21	13	15	9	10	6	23	14	36	21	15	9	17	10
Real	31	21	40	23	42	25	34	19	42	25	21	11	44	27	36	21
Avg.	14	10	28	17	26	16	20	12	29	18	26	16	26	16	24	15

Table 2: Averaged IBU utility gain in % (*cf.* Eq. (1)) for all one-time pure LDP mechanisms and all data distributions (Dist.), considering $k \in \{2, 50, 100, 200\}$, $n \in \{20000, 100000\}$, and $\epsilon \in \{1, 2, 4\}$. Results highlighted in **bold font** represent the two highest utility gains, on average.

Dist.	L-GRR		L-SUE		L-OUE		L-SOUE		L-OSUE		L-BLH		L-OLH		Avg.	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Gauss.	14	5	13	8	9	5	10	7	12	7	2	0	7	4	9	5
Exp.	4	1	27	16	26	15	27	16	27	16	4	2	20	12	19	11
Unif.	36	25	31	22	12	8	16	11	18	13	54	43	21	16	26	19
Poiss.	5	2	43	28	48	32	49	32	44	29	11	6	42	30	34	22
Triang.	28	17	24	15	11	7	13	9	16	10	26	14	14	9	18	11
Real.	4	1	43	25	43	27	44	27	43	25	9	4	34	22	31	18
Avg.	15	8	30	19	24	15	26	17	26	16	17	11	23	15	23	14

Table 3: Averaged utility gain in % of IBU over MI (*cf.* Eq. (1)) for all longitudinal pure LDP mechanisms and all data distributions (Dist.), considering $k \in \{2, 50, 100, 200\}$, $n \in \{20000, 100000\}$, $\epsilon_\infty \in \{2, 4, 8\}$, and $\epsilon_1 = \frac{\epsilon_\infty}{2}$. Results highlighted in **bold font** represent the two highest utility gains, on average.

Implementation details. The `multi-freq-ldpy` Python package [4] is function-based and simulates the LDP data collection pipeline of n users and one server. Thus, for each solution and/or mechanism, there is always a *client* and an *aggregator* function. As `multi-freq-ldpy` had only aggregator functions based on MI, our implementation of IBU contributes with another aggregator function for LDP mechanism in both `pure_frequency_oracle` and `long_freq_est`

```

# Multi-Freq-LDPy functions for GRR protocol
from multi_freq_ldpy.pure_frequency_oracles.GRR import GRR_Client,
    GRR_Aggregator_IBU

# NumPy library
import numpy as np

# Parameters for simulation
eps = 1 # privacy guarantee
n = int(1e6) # number of users
k = 5 # attribute's domain size

# Simulation dataset following Uniform distribution
dataset = np.random.randint(k, size=n)

# Simulation of client-side data obfuscation
rep = [GRR_Client(user_data, k, eps) for user_data in dataset]

# Simulation of server-side aggregation
GRR_Aggregator_IBU(rep, k, eps, nb_iter=10000, tol=1e-12, err_func="max_abs")
>>> array([0.199, 0.201, 0.199, 0.202, 0.199])

```

Listing 1.1: Code snippet for performing distribution estimation using IBU [1, 2] with data obfuscated through the GRR [18] mechanism.

modules. For example, the complete code to execute one-time distribution estimation using IBU [1, 2] with data obfuscated through the GRR [18] mechanism is illustrated in Listing 1.1 with the resulting estimated frequency for a given set of parameters and a randomly generated dataset. One can notice that the `GRR_Aggregator_IBU` function receives as input: the set of obfuscated data (`rep`), the domain size k , the privacy guarantee ϵ , the # iterations (`nb_iter`), and a small tolerance value (`tol`) that works along with an error function (`err_func`). The three last parameters ‘`nb_iter`’, ‘`tol`’, and ‘`err_func`’ are stopping criteria for IBU to terminate, and the values in Listing 1.1 are the default parameters.

5 Related Work

The literature on the local DP model [29] has largely explored the issue of minimizing the utility loss of LDP mechanisms. On the one hand, some works [7, 16, 18, 24, 25, 30] focused on designed new encoding and perturbation functions, often leading to new privacy-utility trade-offs as well as robustness to privacy attacks [5]. On the other hand, recent research works [14, 15, 18, 22, 26] focused on improving the estimation method on the server side through post-processing techniques for the MI estimator and using IBU. For instance, the authors in [18] investigated distribution estimation with GRR and SUE by using two post-processing approaches for MI, namely, a method that clips negative elements of $\hat{\mathbf{f}}$ to 0 and re-normalizes $\hat{\mathbf{f}}$ so that its sum is 1, and a method that projects $\hat{\mathbf{f}}$ onto the probability simplex. Wang *et al.* [26] studied and proposed, in total 10 post-processing approaches for MI ranging from methods that enforce only non-negativity of elements in $\hat{\mathbf{f}}$ or that $\hat{\mathbf{f}}$ sums to 1, and other methods that

enforce both. Experiments in [26] were performed with the OLH mechanism. Regarding IBU [1, 2], recent works have proposed and investigated its performance for discrete distribution estimation [14, 15, 22] and for joint distribution estimation [17, 23]. More precisely, ElSalamouny & Palamidessi [14, 15] proposed a generalization of IBU for personalized LDP, *i.e.*, considering different privacy guarantees ϵ_i ($i \in [1..n]$) and different LDP mechanisms (*i.e.*, GRR and SUE).

While the aforementioned research works [14, 15, 18, 22, 26] answer interesting questions experimenting only with the GRR [18], SUE [16], and OLH [25] mechanisms, we consider in this work fourteen ϵ -LDP mechanisms, *i.e.*, seven for one-time data collection and seven for multiple data collections. In our analysis we varied the utility metrics (*i.e.*, MSE and MAE), the data distribution (*i.e.*, synthetic data following standard distribution and one real-world data), the number of users n , the domain size k , and the privacy guarantees ϵ . Last, we have also open-sourced the IBU implementation into `multi-freq-ldpy` [4], thereby enabling researchers to easily use and expand upon our results.

6 Conclusion & Perspectives

In conclusion, this paper presents an in-depth investigation into the effectiveness of Iterative Bayesian Update (IBU) as a post-processing technique for improving the utility of LDP mechanisms used for private discrete distribution estimation. Based on our experiments on both synthetic and real-world data, we compared the performance of IBU to Matrix Inversion (MI), a standard estimation technique. We assessed the utility gain of IBU over MI for seven state-of-the-art LDP mechanisms designed for one-time collection [7, 16, 18, 24, 25, 30] and for seven state-of-the-art longitudinal LDP mechanisms [3, 6, 16] designed for multiple data collections. On average, both THE [25] and SUE (a.k.a. Basic One-Time RAP-POR) [16] mechanisms showed the highest IBU utility gain in our experiments, which involved varying n , k , ϵ , and the data distribution. Regarding longitudinal LDP mechanisms, L-UE mechanisms [3, 16] presented higher IBU utility gain, on average, than L-GRR [3] and L-LH [6] mechanisms. Overall, our results show that IBU can significantly improve the utility of LDP mechanisms for certain data distributions (*e.g.*, Poisson) and specific settings (*cf.* Table 2 and Table 3).

Based on the findings of this paper, there are several areas that could be explored for future work. Some potential avenues for further research include investigating the performance of IBU on non-pure LDP mechanisms as well as for high-dimensional data, *e.g.*, $k \gg 200$. Additionally, we plan to investigate different settings for the IBU initialization and stopping criteria, *i.e.*, considering non-uniform initial distributions and different tolerance calculation functions. Finally, we also aim to implement GIBU (Generalized IBU) [14] and the estimation methods proposed in [15] for personalized LDP, into `multi-freq-ldpy` [4].

Acknowledgements This work was supported by the European Research Council (ERC) project HYPATIA under the European Union’s Horizon 2020 research and innovation programme. Grant agreement n. 835294.

References

1. Agrawal, D., Aggarwal, C.C.: On the design and quantification of privacy preserving data mining algorithms. In: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM (May 2001). <https://doi.org/10.1145/375551.375602>
2. Agrawal, R., Srikant, R., Thomas, D.: Privacy preserving OLAP. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. ACM (Jun 2005). <https://doi.org/10.1145/1066157.1066187>
3. Arcolezi, H.H., Couchot, J.F., Bouna, B.A., Xiao, X.: Improving the utility of locally differentially private protocols for longitudinal and multidimensional frequency estimates. Digital Communications and Networks (2022). <https://doi.org/10.1016/j.dcan.2022.07.003>
4. Arcolezi, H.H., Couchot, J.F., Gambs, S., Palamidessi, C., Zolfaghari, M.: Multi-freq-ldpy: Multiple frequency estimation under local differential privacy in python. In: Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W. (eds.) Computer Security – ESORICS 2022. pp. 770–775. Springer Nature Switzerland, Cham (2022). https://doi.org/10.1007/978-3-031-17143-7_40
5. Arcolezi, H.H., Gambs, S., Couchot, J.F., Palamidessi, C.: On the risks of collecting multidimensional data under local differential privacy. Proc. VLDB Endow. **16**(5), 1126–1139 (jan 2023). <https://doi.org/10.14778/3579075.3579086>
6. Arcolezi, H.H., Pinzón, C.A., Palamidessi, C., Gambs, S.: Frequency estimation of evolving data under local differential privacy. In: Proceedings of the 26th International Conference on Extending Database Technology, EDBT 2023, Ioannina, Greece, March 28 - March 31, 2023. pp. 512–525. OpenProceedings.org (2023). <https://doi.org/10.48786/EDBT.2023.44>
7. Bassily, R., Smith, A.: Local, private, efficient protocols for succinct histograms. In: Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing. p. 127–135. STOC '15, Association for Computing Machinery, New York, NY, USA (2015). <https://doi.org/10.1145/2746539.2746632>
8. Cheu, A.: Differential privacy in the shuffle model: A survey of separations. arXiv preprint arXiv:2107.11839 (2021)
9. Costa Filho, J.S., Machado, J.C.: Felip: A local differentially private approach to frequency estimation on multidimensional datasets. In: Proceedings of the 26th International Conference on Extending Database Technology, EDBT 2023, Ioannina, Greece, March 28 - March 31, 2023. pp. 671–683. OpenProceedings.org (2023). <https://doi.org/10.48786/EDBT.2023.56>
10. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society: Series B (Methodological) **39**(1), 1–22 (Sep 1977). <https://doi.org/10.1111/j.2517-6161.1977.tb01600.x>
11. Ding, F., Hardt, M., Miller, J., Schmidt, L.: Retiring adult: New datasets for fair machine learning. Advances in Neural Information Processing Systems **34** (2021)
12. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Theory of Cryptography, pp. 265–284. Springer Berlin Heidelberg (2006). https://doi.org/10.1007/11681878_14
13. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Foundations and Trends® in Theoretical Computer Science **9**(3–4), 211–407 (2014)

14. ElSalamouny, E., Palamidessi, C.: Generalized iterative bayesian update and applications to mechanisms for privacy protection. In: 2020 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE (Sep 2020). <https://doi.org/10.1109/eurosp48549.2020.00038>
15. ElSalamouny, E., Palamidessi, C.: Reconstruction of the distribution of sensitive data under free-will privacy. arXiv preprint arXiv:2208.11268 (2022)
16. Erlingsson, U., Pihur, V., Korolova, A.: RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. pp. 1054–1067. ACM, New York, NY, USA (2014). <https://doi.org/10.1145/2660267.2660348>
17. Fanti, G., Pihur, V., Erlingsson, Ú.: Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries. Proceedings on Privacy Enhancing Technologies **2016**(3), 41–61 (May 2016). <https://doi.org/10.1515/popets-2016-0015>
18. Kairouz, P., Bonawitz, K., Ramage, D.: Discrete distribution estimation under local privacy. In: International Conference on Machine Learning. pp. 2436–2444. PMLR (2016)
19. Kasiviswanathan, S.P., Lee, H.K., Nissim, K., Raskhodnikova, S., Smith, A.: What can we learn privately? In: 2008 49th Annual IEEE Symposium on Foundations of Computer Science. pp. 531–540 (2008). <https://doi.org/10.1109/FOCS.2008.27>
20. Liu, G., Tang, P., Hu, C., Jin, C., Guo, S.: Multi-dimensional data publishing with local differential privacy. In: Proceedings of the 26th International Conference on Extending Database Technology, EDBT 2023, Ioannina, Greece, March 28 - March 31, 2023. pp. 183–194. OpenProceedings.org (2023). <https://doi.org/10.48786/edbt.2023.15>
21. McCandless, D., Evans, T., Quick, M., Hollowood, E., Miles, C., Hampson, D., Geere, D.: World’s biggest data breaches & hacks (jan 2021), available online: <https://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/> (accessed on 11 March 2023)
22. Murakami, T., Hino, H., Sakuma, J.: Toward distribution estimation under local differential privacy with small samples. Proceedings on Privacy Enhancing Technologies **2018**(3), 84–104 (Apr 2018). <https://doi.org/10.1515/popets-2018-0022>
23. Ren, X., Yu, C.M., Yu, W., Yang, S., Yang, X., McCann, J.A., Yu, P.S.: Lopub: High-dimensional crowdsourced data publication with local differential privacy. IEEE Transactions on Information Forensics and Security **13**(9), 2151–2166 (2018). <https://doi.org/10.1109/TIFS.2018.2812146>
24. Wang, S., Huang, L., Wang, P., Nie, Y., Xu, H., Yang, W., Li, X.Y., Qiao, C.: Mutual information optimally local private discrete distribution estimation. arXiv preprint arXiv:1607.08025 (2016)
25. Wang, T., Blocki, J., Li, N., Jha, S.: Locally differentially private protocols for frequency estimation. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 729–745. USENIX Association, Vancouver, BC (Aug 2017)
26. Wang, T., Lopuhaa-Zwakenberg, M., Li, Z., Skoric, B., Li, N.: Locally differentially private frequency estimation with consistency. In: Proceedings 2020 Network and Distributed System Security Symposium. Internet Society (2020). <https://doi.org/10.14722/ndss.2020.24157>
27. Warner, S.L.: Randomized response: A survey technique for eliminating evasive answer bias. Journal of the American Statistical Association **60**(309), 63–69 (Mar 1965). <https://doi.org/10.1080/01621459.1965.10480775>

28. Wong, J.C.: Facebook to be fined \$5bn for cambridge analytica privacy violations – reports (2019), available online: <https://www.theguardian.com/technology/2019/jul/12/facebook-fine-ftc-privacy-violations> (accessed on 11 March 2023)
29. Xiong, X., Liu, S., Li, D., Cai, Z., Niu, X.: A comprehensive survey on local differential privacy. *Security and Communication Networks* **2020**, 1–29 (Oct 2020). <https://doi.org/10.1155/2020/8829523>
30. Ye, M., Barg, A.: Optimal schemes for discrete distribution estimation under locally differential privacy. *IEEE Transactions on Information Theory* **64**(8), 5662–5676 (2018). <https://doi.org/10.1109/TIT.2018.2809790>