Hindawi
Security and Communication Networks
Volume 2022, Article ID 1330251, 20 pages
https://doi.org/10.1155/2022/1330251

Research Article

Key-Value Data Collection with Distribution Estimation under Local Differential Privacy

Xiaoguang Li,[1] Haonan Yan,[1] Gewei Zheng,[1] Hui Li,[1] and Fenghua Li[1,2]

[1]*State Key Laboratory of Integrated Services Networks, School of Cyber Engineering, Xidian University, Xi'an, China*
[2]*Institute of Information Engineering, Chinese Academic of Sciences, Beijing, China*

Correspondence should be addressed to Hui Li; lihui@mail.xidian.edu.cn

Received 20 October 2021; Revised 7 March 2022; Accepted 24 June 2022; Published 8 August 2022

Academic Editor: Khizar Hayat

Local differential privacy (LDP) is a promising privacy-preserving technology from users' perspective, as users perturb their private information locally before reporting to the aggregator. We study the problem of collecting heterogeneous data, that is, key-value pairs under LDP, which is widely involved in real-world applications. Although previous LDP work on key-value data collection achieves a good utility on frequency estimation of key and distribution estimation of value, they have three downfalls: (1) existing work perturbs numerical value in a discrete manner that does not exploit the ordinal nature of the numerical domain and lead to poor accuracy, (2) they do not lead to improved privacy budget composition and consume more privacy budget than necessary to achieve the given privacy level, and (3) the frequency estimation of the key is not the most accurate due to the lack of consistency requirement. In this paper, we propose a novel mechanism to collect key-value data under LDP leveraging the numerical nature of the domain and result in better utility. Due to our correlated perturbation, the mechanism consumes less privacy budget than previous work while keeping the privacy level. We also adopt consistency as the postprocessing, which is applied to the estimated key frequency to further improve the accuracy. Comprehensive experiments demonstrate that our approach consistently outperforms the state-of-the-art mechanisms under the same LDP guarantee.

1. Introduction

Differential privacy (DP) [1] is the state-of-the-art technology for private data release, which provides provable and measurable privacy protection regardless of the adversary's background knowledge. Different from DP in the centralized setting that protects data after data collection, local differential privacy (LDP) has been proposed to protect data during data collection. In LDP, the server is assumed to be untrusted. Each user locally obfuscates his/her personal data using the LDP mechanism before uploading. After receiving the perturbed data from all users, the server performs data analytics or answers queries. LDP technology enables collecting statistics of users under a privacy guarantee and has been widely deployed in practice. For example, Apple deploys the LDP mechanism in iOS to identify heavy hitters in emojis while keeping user privacy [2]; RAPPOR has been deployed in Google's browser Chrome to collect and analyze the web browsing behavior of users under LDP [3].

Early LDP work mainly focuses on simple statistical queries such as frequency estimation on categorical data [4] and mean estimation [5, 6] on numerical data. Nowadays, LDP is also applied for hybrid data types or queries [7–9], for example, key-value data, which has categorical data and numerical data simultaneously and is widely used in practice. The following examples show the potential applications of key-value data:

(i) Product rating analysis: online market platforms such as Amazon and eBay are collecting users' ratings for the products they bought and show the ratings online as a reference for other buyers. These rating data are usually in the form of key-value data where the key is the product and the value is the rating.

(ii) Software usage analysis: software developers and providers such as Microsoft need to collect usage time about each software to analyze users' preferences. This data is usually in the form of key-value pairs, where the key is the software identifier and the value is the usage time of this software.

Literature [7, 8] are the first to study the problem of collecting key-value data under LDP. They design mechanisms to support two estimation tasks: (1) the frequency of the key and (2) the mean estimation of the value. Recently, Ye et al. [9] expand the previous work and are the first to propose PrivKVM* to estimate the frequency of the key and the distribution of the value. It discretizes the domain into many bins and reports which bin contains the private value using categorical frequency oracle (e.g., GRR [10]) while considering the correlation between keys and values. However, it has three main limitations. First, it perturbs the numerical value in a discrete manner, which does not work well because it does not exploit the ordinal information of the numerical domain, that is, a perturbed report that is close to the true value also has useful information for the distribution estimation. Second, although the mechanism considers the correlation between keys and values, it does not lead to an improved budget composition. Third, it does not consider the consistency in the estimated frequency of the key. That is, the estimated frequency should be consistent with the property of frequency: (1) each frequency should be non-negative and (2) the sum of all estimated frequencies should be 1. Without enforcing the consistency requirement, the mechanism may not result in the most accurate answers to the key frequency [11].

Motivated by this, in this paper, we study the problem of collecting key-value data under LDP and propose a mechanism to solve the above three limitations: existing mechanisms (1) do not exploit the ordinal information about the numerical domain, (2) do not consider consistency to achieve the most accuracy, and (3) do not result in improved budget composition. Our mechanism aims to collect the two most fundamental statistics of key-value pairs: key frequency and value distribution. It contains three steps: (1) padding and sampling, (2) perturbation, and (3) aggregating and estimating key frequency and value distribution.

In step 1, each user pads his key-value data by dummy into the same length (make the sampling rate identical for all users) and samples one key-value pair. The reason for using the sampling protocol is that multiple key-value pairs are possessed by each user; if all the pairs are reported to the server, each pair will split the privacy budget, which results in large noise in each pair and bad utility.

In step 2, we solve the first and second limitations. Each user perturbs the sampled key-value pair in a correlated manner because there is an inherent correlation between key and value and reporting the value may also reveal information about the presence of the corresponding key [8]. Thus, we first perturb the key and then perturb the value according to the perturbation results of the key. If a possessed key is still possessed after perturbation, we then report the value via the LDP mechanism that utilizes the ordered nature of the domain and directly perturbs the value in the numerical domain, which solves the first limitation by leveraging the ordinal information about the numerical domain. However, there is a challenge when a non-possessed key is perturbed as possessed. By LDP definition, the perturbed values of the dummy keys and the perturbed values of the genuine keys are indistinguishable. Thus, the perturbed values of the dummy keys would affect the distribution estimation. To address this problem, we generate fake values for these keys that can satisfy LDP. Previous work selects values uniformly at random from the discrete output domain as fake values. However, such fake value generation does not work for our mechanism. In our mechanism, the output domain is a numerical continuous interval, and such fake value generation will violate the privacy guarantee since the probability of outputting the values that are not those discrete values is unbounded. Therefore, we adopt the existing fake value generation for our mechanism.

By an outgrowth of the fake value generation, we show that our correlated perturbation has a privacy amplification effect: it consumes less privacy budget overall than the summation of budget in key and value perturbation. This solves the third limitation by providing a tighter budget composition, which achieves a better privacy-utility trade-off than basic sequential composition (used in PrivKVM*).

In step 3, we solve the third limitation. The server collects the perturbed results from all users and estimates the key frequency and the value distribution. For the frequency estimation, the server enforces consistency requirements to improve the accuracy. Since the fake values generated in the perturbation step affect the distribution, removing the influence of the fake value is another challenge. The consistency is not designed for this challenge since it only requires the estimated frequencies are sum-up-to-1 and non-negative but cannot detect the fake value and remove them. To address this problem, we design a method to statistically remove the fake value in the distribution estimation.

Our main contributions are summarized as follows:

(1) Novel LDP mechanism for key-value data collection: we propose a mechanism that supports frequency estimation and distribution estimation over key-value data. It takes advantage of the numerical nature of the data domain and achieves better accuracy than existing solutions.

(2) Improved privacy budget composition: we show that the privacy budget composition of our correlated perturbation mechanism has a tighter bound than sequential composition, which provides a privacy amplification effect and achieves a better privacy-utility trade-off.

(3) Consistency as postprocessing to improve accuracy: we enforce consistency as postprocessing for key frequency estimation in our mechanism, which can further improve the accuracy than the existing LDP mechanism for key-value data collection.

(4) Comprehensive evaluation: we implement the mechanism and evaluate it on real-world data sets. The results show our mechanism outperforms existing LDP schemes. In particular, our mechanism

significantly improves the accuracy, that is, reducing the error of current mechanisms by about an order of magnitude in most cases, especially when ε is small (large noise).

2. Preliminary

2.1. Local Differential Privacy. In the centralized setting of differential privacy, a trusted server or data aggregator has all users' personal data, and it is responsible for responding to queries while using DP mechanisms to protect user privacy. However, the assumption of a trust server may not hold in practice. Local differential privacy addresses this problem. In the local setting, each user perturbs his personal data and then uploads the perturbed results to the server for data analysis. In this way, the server can be untrusted because it cannot access the original data.

Definition 1 (local differential privacy (LDP) [12]). A randomized algorithm $\mathscr{M}$ is ε-LDP if and only if for any input x_1 and x_2, the probability ratio of outputting the same result is bounded by e^{ε}. Formally,

$$\Pr(\mathscr{M}(x_1) = y) \le e^{\varepsilon} \Pr(\mathscr{M}(x_2) = y). \tag{1}$$

By Definition 1, given any output y, the adversary cannot infer the original input is x_1 or x_2 with high confidence. Here, the confidence is controlled by the parameter ε (called privacy budget). The smaller the ε, the closer the probability $\Pr(\mathscr{M}(x_1) = y)$ is to the probability $\Pr(\mathscr{M}(x_2) = y)$. That is, the mechanism provides stronger privacy protection since the adversary has lower confidence to distinguish whether the original input is x_1 or x_2.

When multiple LDP mechanisms are combined together to generate a new mechanism, the sequential composition theorem guarantees the total privacy of the new mechanism.

Theorem 1 (sequential composition theorem [12]). *If a sequence of mechanisms $\mathscr{M}_1, \mathscr{M}_2, \ldots, \mathscr{M}_n$ satisfies ε_1-LDP, ε_2-LDP, …, ε_n-LDP, then the sequential composition $\mathscr{M} = [\mathscr{M}_1, \mathscr{M}_2, \ldots, \mathscr{M}_n]$ satisfies $\sum_{i=1}^{n} \varepsilon_i$-LDP.*

By Theorem 1, given a total privacy budget ε and a computation task, we can split the task into multiple parts and allocate each part a portion of privacy budget ε to achieve ε-LDP.

Theorem 2 (postprocessing [12]). *If a mechanism $\mathscr{M}$ satisfies ε-LDP, given any function $\mathscr{F}$ that cannot access the original data and noise, then the $\mathscr{F} \circ \mathscr{M}$ also satisfies ε-LDP.*

By Theorem 2, the postprocessing does not violate the privacy guarantee of LDP mechanisms. In this paper, we use the postprocessing method to further improve the utility of our mechanism.

2.2. Basic LDP Mechanisms

2.2.1. Unary Encoding (UE). Unary encoding first encodes an input $x = i$ into a binary vector $x = [0, 0, 1, \ldots, 0]$ where the only i-th position is 1 and others are 0 [10]. Also, the length of the vector is equal to the domain size d of the input. Then it perturbs each bit as follows ($q \le 0.5 \le p$):

$$\Pr(\hat{x}[i] = 1) = \begin{cases} p, & \text{if } \mathbf{x}[i] = 1, \\ q, & \text{if } \mathbf{x}[i] = 0, \end{cases} \quad \forall i = 1, 2, \ldots, d. \tag{2}$$

As shown in [10], unary encoding provides ε-LDP, where $\varepsilon = \ln(p(1-q)/q(1-p))$.

After receiving the perturbed results from all n users, the server or aggregator can estimate the frequency of users who possess the i-th item, that is, the input value is $x = i$. Denote the estimated frequency of the i-th item by f_i; the aggregator can estimate the frequency f_i by the following unbiased estimator $\hat{f}_i$:

$$\hat{f}_i = \frac{\sum_{j=1}^{n} \mathbb{I}\left(\hat{x}^j[i] = i\right) - nq}{p - q}, \tag{3}$$

where $\mathbb{I}$ is the indicator function and $\hat{x}^j[i]$ indicates the i-th bit of the vector of the user j.

2.2.2. Square Wave Mechanism (SW Mechanism). SW mechanism is designed for numerical distribution estimation under LDP [13]. The intuition behind this mechanism is to increase the probability that a noisy reported value can carry useful information about the input. For the numerical domain, a noisy reported value that is different from but close to the true value also contains useful information for distribution estimation. Therefore, given an input x, the SW mechanism reports values closer to x with a higher probability than values farther away from x. Formally speaking, the SW mechanism $\mathscr{M}$ assumes the domain of the input is $D = [0, 1]$ (any bounded value can be linearly transformed into this domain) and the domain of the output is $\hat{D} = [-b, 1 + b]$, then it perturbs the input x with the following probabilities:

$$\Pr(\mathscr{M}(x) = \hat{x}) = \begin{cases} p, & \text{if } |x - \hat{x}| \le b, \\ q, & \text{if } |x - \hat{x}| \le b, \end{cases} \quad \forall x \in D, \forall \hat{x} \in \hat{D}. \tag{4}$$

As shown in [13], the value p and q are set to be

$$p = \frac{e^{\varepsilon}}{2be^{\varepsilon} + 1}, \qquad q = \frac{1}{2be^{\varepsilon} + 1}, \tag{5}$$

which are derived by maximizing the difference between p and q while satisfying the total probability is add-up-to 1. Also, the parameter b is set to be

$$b = \frac{\varepsilon e^{\varepsilon} - e^{\varepsilon} + 1}{2e^{\varepsilon}\left(e^{\varepsilon} - \varepsilon - 1\right)}, \tag{6}$$

which is obtained by maximizing the mutual information between the input and output of the SW mechanism.

Since the output domain and the input domain are different, the server/aggregator uses expectation-maximization

(EM) algorithm to reconstruct the distribution after receiving the noisy reported values.

3. Key-Value Data Collection under LDP

3.1. Problem Statement

3.1.1. System Model. There are n users and one server in our system model, where each user possesses one or multiple key-value pairs $\langle k, v\rangle$. The domain of the key is assumed to be $K = \{1, 2, \ldots, d\}$, while the domain of the value is assumed to be $V = [0, 1]$ (any bounded value can be linearly transformed into this domain). Besides, we assume the user i possesses the set of key-value pairs $\mathcal{S}_i$. The goal of the server is to collect the key-value data from all users and then estimate the (1) frequency and the (2) value distribution of a certain key. In other words, the server calculates the fraction of users who have a certain key and the value distribution of that key among those who have it.

3.1.2. Threat Model. We assume the server is untrusted, and a data breach might occur as a result of unauthorized data publishing or hacking. The adversary is considered to have access to all users' output and to be aware of the perturbation algorithm in the mechanism locally established on the user side. Furthermore, we assume that all users will honestly follow the perturbation mechanism.

3.2. PrivKVM.* To the best of our knowledge, PrivKVM* [9] is the state-of-the-art LDP framework for key-value data collection that can support frequency estimation of key and distribution estimation of value. We first briefly describe the mechanism and then summarize the main differences between our work and PrivKVM*.

3.2.1. Workflow of PrivKVM.* PrivKVM* collects the key-value data in two phases. In the first phase, each user first samples one key-value pair uniformly at random from the full domain of the key and then perturbs it in a correlated manner. Specifically, it first perturbs the key and then perturbs the value according to the perturbed result of the key. The following are the four cases of perturbation:

(1) The sampled key is possessed by the user, and the key is perturbed as possessed. The user perturbs the value using the technology called GVPP, which is a categorical frequency oracle with a boundary. That is, the user first discretizes the numerical domain into many bins and then discretizes the value to the boundary of the bin containing the value with a specific probability. Then, the user reports which boundary contains the private value using categorical frequency oracle, for example, GRR [10].

(2) The sampled key is possessed by the user, and the key is perturbed as non-possessed. The existing key disappears after perturbation in this case, and the user simply sets the value to be 0.

(3) The sampled key is non-possessed by the user, and the key is perturbed as possessed. In this case, a "fake" key appears, and the user samples a mean uniformly at random from the current means of all bins as the value.

(4) The sampled key is non-possessed by the user, and the key is perturbed as non-possessed. The user simply set the value to be 0.

After the perturbation, each user reports the obfuscated result, and the server estimates the statistical information: (1) the frequency of all keys, (2) the mean of the value, and (3) the distribution of values. To obtain a more accurate mean estimation, the server leverages virtual iteration technology and further calibrates the estimated mean. In the second phase, the server broadcasts to all users the heavy hitters (the keys with frequencies higher than a given threshold) and their corresponding mean estimated in the first phase, and users and the server repeatedly execute the steps as in the first phase to obtain the statistics estimation, except that the statistics are for the set of heavy hitters instead of all keys. For the rest of the keys, the server averages their statistics to reduce the noise effect since they are non-heavy hitters and the number of samples are insufficient.

3.2.2. Limitations of PrivKVM.* We summarize the limitations of PrivKVM* as follows:

(1) PrivKVM* estimates value distribution using categorical frequency oracle with boundary. In this way, the server can estimate the count of values falling in each bin and obtain the density distribution over the domain. However, values in the numerical domain have meaningful total order, and this method ignores such information in the value due to discretization. What is worse, it faces the challenge of finding the optimal size of bins. Binning causes two sources of errors: (1) LDP noise and (2) bias due to grouping values together. More bins lead to large error due to LDP noise, and fewer bins result in a greater error because of bias. Unfortunately, finding the optimal size of bins is a nontrivial task since the effect of the size depends on both ε and the property of actual data distribution that is unknown to the server [13].

(2) PrivKVM* does not consider the consistency problem in the frequency estimation of the key. That is, the estimated frequencies could not satisfy the basic requirements of frequency: (1) every frequency should be non-negative and (2) all frequencies should be summing-up-to-1. Thus, the estimated frequencies are not the most accurate.

(3) Although PrivKVM* perturbs the key and value in a correlated manner, it does not lead to an improved budget composition for LDP.

In what follows, we elaborate on our mechanism. Some important notations are summarized in Table 1.

Table 1: Notations.

Symbol	Description
$\mathcal{S}$	Domain of key-value data
$\mathcal{S}_i$	Key-value data set of user i
K	Domain of the key, $K = \{1, 2, \ldots, d\}$
K'	Domain of the padded key, $K' = \{1, 2, \ldots, d, d+1, \ldots, d+l\}$
d	Domain size of the key
l	Padding length
d'	Domain size of the padded key, $d' = d + l$
$\mathbf{x}$	Vector encoded by sampled key-value pair
$\widehat{x}$	Randomized output vector
$\mathbf{x}[i]$ or $\widehat{x}[i]$	The i-th element of vector $\mathbf{x}$ or $\widehat{x}$
$k_{\mathbf{x}}^{(i)}$	The key of $\mathbf{x}[i]$
$v_{\mathbf{x}}^{(i)}$	The value of $\mathbf{x}[i]$
$\widehat{k}_{\mathbf{x}}^{(i)}$	The perturbed key of $\mathbf{x}[i]$
$\widehat{v}_{\mathbf{x}}^{(i)}$	The perturbed value of $\mathbf{x}[i]$
f_i	True frequency of the key i
$\hat{f}_i$	Estimated frequency of the key i
$\tilde{f}_i$	Postprocessed frequency of the key i

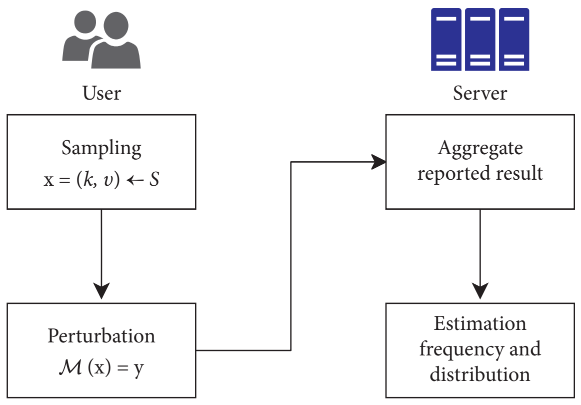

User
Server
Sampling
x = (k, v) ← S
Aggregate reported result
Perturbation
M (x) = y
Estimation frequency and distribution

Figure 1: The overview of our mechanism.

4. Proposed Method

The overview of the proposed method is shown in Figure 1. The idea of our mechanism is as follows. Each user first samples one key-value pair from his personal data (the sampling protocol will be discussed in Section 4.1); then each user privately perturbs the sampled key-value pair by our LDP mechanism (the mechanism will be discussed in Section 4.2). After receiving the reported results, the server aggregates the perturbed data and estimates the key frequency and value distribution, which will be shown in Section 4.3.

4.1. Sampling Protocol.

4.1. Sampling Protocol. In this subsection, we explain why we need to sample before reporting perturbed data and elaborating on our sampling protocol.

4.1.1. Why Sampling Protocol. In practice, each user may have multiple key-value pairs. If the user perturbs all his key-value pairs, then each pair would consume the privacy budget, and the LDP mechanism has to split the total privacy budget. Thus, the noise added to each pair would be too large. To solve this problem, a promising method is to sample and submit one pair, which avoids the privacy budget splitting and improves the utility.

4.1.2. Our Sampling Protocol. Sampling protocols are widely used in existing DP mechanisms for key-value data perturbation [7–9]. However, they either do not support distribution estimation or do not work well on large domain sizes. In particular, PrivKVM* [9] samples from the full domain in the first phase to identify heavy hitters, which does not work well when the domain size is very large and each user only possesses a small number of keys since users rarely report the information about the keys they possess. Therefore, we adopt the padding-and-sampling protocol [8, 14] for key-value data to support frequency and distribution estimation. The advantage of the padding-and-sampling protocol is that it samples from the set of keys users possess instead of from the full domain, and thus, it handles large domains better.

The step of our sampling protocol is as follows. First, all users generate l dummy key-value pairs whose keys are $\{d+1, \ldots, d+l\}$ and values are zeros. For user i whose $|\mathcal{S}_i| < l$, he adds $l - |\mathcal{S}_i|$ different random dummy key-value pairs to $\mathcal{S}_i$ and make the length be l. Without padding, determining the probability that a pair is sampled is difficult, resulting in inaccurate estimation. Therefore, the domain of the key of the padded data is $K' = \{1, 2, \ldots, d'\}$ where $d' = d + l$. Then each user samples one pair from the padded data to perturb and upload. Although some pairs may be unsampled, this case only occurs for infrequent pairs, and the useful information is still reported with high probability. The following shows an example to illustrate the sampling process, and the details are shown in Algorithm 1.

4.1.3. Example. Suppose the domain size $d = 4$ and the padding length $l = 3$. Let the domain of the key $K = \{1, 2, 3, 4\}$, the domain of the dummy key is $\{5, 6, 7\}$, and the domain of the padded key-value pair becomes $K' = \{1, 2, 3, 4, 5, 6, 7\}$. For a user whose key-value pairs are $\{\langle 1, v_1\rangle, \langle 2, v_2\rangle\}$, since he possesses two pairs, he first pads the data by one dummy pair and gets $\{\langle 1, v_1\rangle, \langle 2, v_2\rangle, \langle 6, 0\rangle\}$ (suppose the dummy $\langle 6, 0\rangle$ is picked). Then he picks one pair uniformly at random from the padded result $\{\langle 1, v_1\rangle, \langle 2, v_2\rangle, \langle 6, 0\rangle\}$ to perturb and upload.

We note that the previous mechanism PCKV [8] also adopts a padding-and-sampling protocol for key-value data collection under LDP. We emphasize there is a difference between their protocol and ours. PCKV only supports the mean estimation of values; thus, it discretizes the value into 1 and −1 with particular probability to guarantee the unbiasedness for mean estimation. Our sampling protocol does not adopt the discretizing step because we want to estimate the value distribution and discrete values lose the numerical information and would affect the distribution estimation.

According to the literature [8, 14], padding length l would cause two types of error: (1) *variance* between true

```
**Input:** The set $\mathcal{S}_i$ of key-value pairs, padding length $l$.
**Output:** Sampled key-value pair $\langle k, v\rangle$, where $k \in K'$ and $v \in [-b, 1+b]$
(1) Generate $l$ dummy key-value pairs, whose keys are $d+1, \ldots, d'$ and values are 0.
(2) if $|\mathcal{S}_i| < l$ then
(3)     Add $l - |\mathcal{S}_i|$ random different dummy pairs to $\mathcal{S}_i$.
(4) end if
(5) Pick one key-value pair $\langle k, v\rangle$ uniformly at random from the $\mathcal{S}_i$.
(6) return Sampled key-value pair $\langle k, v\rangle$.
```

Algorithm 1: Sampling protocol.

values and estimated results and (2) *bias* between true values and estimated results. A smaller l would underestimate the key frequency and results in a large bias, and a larger l would enlarge the noise in estimation, thus leading to a large variance. Unfortunately, finding the optimal padding length l that can balance the trade-off between the variance and the bias is a non-trivial task, and it is still an open problem so far [8]. Thus, in this paper, we empirically set the suitable padding length l in the experiments for comparing with other LDP mechanisms.

4.2. Perturbation Mechanism. In this subsection, we introduce our perturbation mechanism. By Algorithm 1, each user samples one key-value pair $\langle k, v\rangle$ as the input of the perturbation mechanism. The basic idea of our perturbation mechanism is to perturb the value according to the perturbed results of the key. If a non-possessed key is perturbed as possessed or a possessed key is perturbed as non-possessed, we generate a fake value for the key to avoid the influence on the distribution estimation of the value. Under this strategy, we find the mechanism provides a tighter privacy budget composition (see Theorem 3), that is, it is shown that the total privacy budget of the combined perturbations (key perturbation and value perturbation) is smaller than the sequential composition. Based on the above idea and two basic LDP mechanisms (UE and SW), we design an LDP mechanism for key-value data collection that can support numerical distribution estimation. Overall, the LDP perturbation is shown in Algorithm 2.

In the UE mechanism, the original input is encoded as a binary vector where the bit at the input-corresponding position is 1 and other bits are 0. Similarly, for key-value data, we encode the sampled key-value pair $\langle k, v\rangle$ as a vector $\mathbf{x}$ where the k-th element $\mathbf{x}[k]$ (corresponding to the key k) is $\langle 1, v\rangle$ and the other element is $\langle 0, 0\rangle$. Then the perturbation can be divided into two steps. Note that each element in the vector has two items (key and value), for brevity, we use notation $k_{\mathbf{x}}^{(i)}$ and $v_{\mathbf{x}}^{(i)}$ to represent the key and value of the i-th element of vector $\mathbf{x}$, respectively, and use $\widehat{k}_{\mathbf{x}}^{(i)}$ and $\widehat{v}_{\mathbf{x}}^{(i)}$ to represent perturbed key and value. First, we perturb the key as follows:

$$\Pr\left(\widehat{k}_{\mathbf{x}}^{(i)} = 1\right) = \begin{cases} a, & \text{if } k_{\mathbf{x}}^{(i)} = 1, \\ c, & \text{if } k_{\mathbf{x}}^{(i)} = 0, \end{cases} \quad \forall i = 1, 2, \ldots, d', \tag{7}$$

where $c \leq 0.5 \leq a$. Given the perturbation result of the key, we then perturb the value. The value perturbation can be divided into three cases as follows:

(i) If the key $\widehat{k}_{\mathbf{x}}^{(i)}$ is perturbed from 1 to 1, the corresponding value $v_{\mathbf{x}}^{(i)}$ is perturbed as $\widehat{v}_{\mathbf{x}}^{(i)}$ such that

$$\Pr\left(\mathcal{M}\left(v_{\mathbf{x}}^{(i)}\right) = \widehat{v}_{\mathbf{x}}^{(i)}\right) = \begin{cases} p, & \text{if } \left|v_{\mathbf{x}}^{(i)} - \widehat{v}_{\mathbf{x}}^{(i)}\right| \leq b, \\ q, & \text{if } \left|v_{\mathbf{x}}^{(i)} - \widehat{v}_{\mathbf{x}}^{(i)}\right| \leq b. \end{cases} \tag{8}$$

(ii) If the key $\widehat{k}_{\mathbf{x}}^{(i)}$ is perturbed from 0 to 1, the fake value drawn from the uniform distribution $U(-b, 1+b)$ is assigned.

(iii) If the key $\widehat{k}_{\mathbf{x}}^{(i)}$ is perturbed from 0 to 0 or from 1 to 0, the key is reported as non-possessed; thus, we set the perturbed value to be 0.

4.2.1. Privacy Analysis of Our Mechanism. In our mechanism, the key is perturbed by the UE mechanism with privacy budget $\varepsilon_1 = \ln(a(1-c)/c(1-a))$ (see Section 2); the value is perturbed by the SW mechanism with privacy budget $\varepsilon_2 = \ln(p/q)$. In our mechanism, the key perturbation and value perturbation are correlated, that is, the value perturbation relies on the key and key perturbation. Generally, the correlated perturbation may leak less privacy than independent perturbation and has a privacy amplification effect [8]. That is, the total privacy budget ε is less than the summation $\varepsilon_1 + \varepsilon_2$. Theorem 3 shows our mechanism satisfies LDP and has a tighter budget composition than sequential composition.

Theorem 3. *Denote the privacy budget for key perturbation and value perturbation are ε_1 and ε_2, respectively; our mechanism satisfies ε-LDP where*

$$\varepsilon = \ln\left[\max\left\{\frac{\varepsilon_2 e^{\varepsilon_2} - e^{\varepsilon_2} + 1}{e^{\varepsilon_2}\left(e^{\varepsilon_2} - \varepsilon_2 - 1\right)}, e^{\varepsilon_2}, e^{\varepsilon_1} \times \frac{e^{\varepsilon_2} - 1}{\varepsilon_2}\right\}\right]. \tag{9}$$

Proof. For a key-value set $\mathcal{S}$, we denote the key-value pairs by $\langle k, v\rangle$ for all $i \in \mathcal{S}$, where $i \in \mathcal{S}$ means the key-value pair $\langle i, \cdot\rangle \in \mathcal{S}$. Suppose the sampled key-value pair is $\langle k, v\rangle$ $(v \in [0, 1])$, we have the perturbed value $\widehat{v}_{\mathbf{x}}^{(k)} = 0$ if the key is drawn from $\{d+1, \ldots, d'\}$. For vector $\mathbf{x}$, only the k-th element is non-zero, and others are zeros. Then we have the probability of outputting a vector $\widehat{x}$ is as follows:

```
      Input: The sampled key-value pair $\langle k, v\rangle$, privacy budget $\varepsilon_1, \varepsilon_2$
      Output: Perturbed result $\widehat{x}$
 (1)  Encode $\langle k, v\rangle$ as vector $\mathbf{x}$
 (2)  Perturb $k_{\mathbf{x}}^{(i)}$ as $\widehat{k}_{\mathbf{x}}^{(i)}$ by (2) $(\forall i = 1,2,\ldots,d')$
 (3)  if $k_{\mathbf{x}}^{(i)}: 1 \longrightarrow 1 (\forall i = 1,2,\ldots,d')$ then
 (4)      Perturb $v_{\mathbf{x}}^{(i)}$ as $\widehat{v}_{\mathbf{x}}^{(i)}$ by (3)
 (5)  end if
 (6)  if $k_{\mathbf{x}}^{(i)}: 0 \longrightarrow 1 (\forall i = 1,2,\ldots,d')$ then
 (7)      Generate fake value $\widehat{v}_{\mathbf{x}}^{(i)} = U(-b, 1+b)$
 (8)  end if
 (9)  if $k_{\mathbf{x}}^{(i)}: 0,1 \longrightarrow 0 (\forall i = 1,2,\ldots,d')$ then
(10)      $\widehat{v}_{\mathbf{x}}^{(i)} = 0$
(11)  end if
(12)  return $\widehat{x} = [\langle \widehat{k}_{\mathbf{x}}^{(i)}, \widehat{v}_{\mathbf{x}}^{(i)}\rangle]_{i=1}^{d'}$.
```

Algorithm 2: Perturbation.

$$\begin{aligned}\Pr[\widehat{x}|\mathcal{S},k] &= \Pr[\widehat{x}[k]|\mathbf{x}[k]]\prod_{i\neq k}\Pr[\widehat{x}[i]|\mathbf{x}[i]=0]\\ &= \frac{\Pr[\widehat{x}[k]|\mathbf{x}[k]]}{\Pr[\widehat{x}[k]|\mathbf{x}[k]=0]}\prod_{i}\Pr[\widehat{x}[i]|\mathbf{x}[i]=0].\end{aligned} \tag{10}$$

Denote the first term by $f(\widehat{x},k)$ and the second term by $g(\widehat{x})$, that is,

$$\begin{aligned}f(\widehat{x},k) &= \frac{\Pr[\widehat{x}[k]|\mathbf{x}[k]]}{\Pr[\widehat{x}[k]|\mathbf{x}[k]=0]},\\ g(\widehat{x}) &= \prod_{i}\Pr[\widehat{x}[i]|\mathbf{x}[i]=0].\end{aligned} \tag{11}$$

Since the second term $g(\widehat{x})$ is same for different inputs, it will be canceled out when we calculate the ratio of $\Pr(\widehat{x}|\mathcal{S}_1)$ to $\Pr(\widehat{x}|\mathcal{S}_2)$ ($\mathcal{S}_1$, $\mathcal{S}_2$ are two different key-value sets). Therefore, we first calculate the $f(\widehat{x},k)$.

According to the perturbation mechanism, we can calculate the numerator as follows:

$$\Pr[\widehat{x}[k]|\mathbf{x}[k]] = \begin{cases} ap, & \text{if } \left|v_{\mathbf{x}}^{(k)} - \widehat{v}_{\mathbf{x}}^{(k)}\right| \leq b,\\ aq, & \text{if } \left|v_{\mathbf{x}}^{(k)} - \widehat{v}_{\mathbf{x}}^{(k)}\right| > b,\\ 1-a, & \text{if } \widehat{v}_{\mathbf{x}}^{(k)} = 0.\end{cases} \tag{12}$$

Based on the result, we can obtain the $f(\widehat{x},k)$ by

$$f(\widehat{x},k) = \begin{cases} \dfrac{ap}{c\times(2b/(1+2b))}, & \text{if}\left|v_{\mathbf{x}}^{(k)} - \widehat{v}_{\mathbf{x}}^{(k)}\right| \leq b,\\ \dfrac{aq}{c\times(2b/(1+2b))}, & \text{if}\left|v_{\mathbf{x}}^{(k)} - \widehat{v}_{\mathbf{x}}^{(k)}\right| > b,\\ \dfrac{1-a}{1-c}, & \text{if } \widehat{v}_{\mathbf{x}}^{(k)} = 0.\end{cases} \tag{13}$$

Then, we discuss the upper and lower bounds of $f(\widehat{x},k)$. Since both the UE mechanism and the SW mechanism have a higher probability of maintaining the input value than that of perturbing the input as other values, we have

$$\begin{aligned}&\frac{\Pr[\widehat{x}[k]=0|\mathbf{x}[k]]}{\Pr[\widehat{x}[k]=0|\mathbf{x}[k]=0]} \leq \frac{\Pr[\widehat{x}[k]=\mathbf{x}[k]|\mathbf{x}[k]]}{\Pr[\widehat{x}[k]|\mathbf{x}[k]=0]}\\ &\Rightarrow \frac{1-a}{1-c} \leq \frac{ap}{c\times(1/(1+2b))}.\end{aligned} \tag{14}$$

Because $(ap/(c\times(1/(1+2b))))$ is greater than $(ap/(c\times(2b/(1+2b))))$ and $(aq/(c\times(1/(1+2b))))$, thus, we have the upper bound f_u and lower bound f_l of $f(\widehat{x},k)$ as follows:

$$\begin{aligned}f_u &= \frac{ap}{c\times(1/(1+2b))},\\ f_l &= \min\left\{\frac{ap}{c\times(2b/(1+2b))}, \frac{aq}{c\times(1/(1+2b))}, \frac{1-a}{1-c}\right\}.\end{aligned} \tag{15}$$

Then, we have the probability of outputting $\widehat{x}$ given a key-value set $\mathcal{S}$ is

$$\begin{aligned}\Pr(\widehat{x}|\mathcal{S}) &= \sum_k \Pr(\widehat{x}|\mathcal{S},k)\times\frac{\Pr(\mathcal{S},k)}{\Pr(\mathcal{S})}\\ &= \sum_k \Pr(\widehat{x}|\mathcal{S},k)\times\Pr(\mathcal{S}|k)\\ &= \sum_k \Pr(\widehat{x}|\mathcal{S},k)\times\frac{1}{|\mathcal{S}|} \leq f_u\times g(\widehat{x}).\end{aligned} \tag{16}$$

Similarly, we also have $\Pr(\widehat{x}|\mathcal{S}) \geq f_l \times g(\widehat{x})$. Thus, the following inequality holds for two different key-value sets $\mathcal{S}_1$ and $\mathcal{S}_2$:

$$\frac{\Pr(\hat{x}|\mathcal{S}_1)}{\Pr(\hat{x}|\mathcal{S}_2)} \leq \frac{f_u \times g(\hat{x})}{f_l \times g(\hat{x})}$$
$$= \frac{ap/(c \times (1/(1+2b)))}{\min\{(ap/(c \times (2b/(1+2b)))), (ap/(c \times (1/(1+2b)))), ((1-a)/(1-c))\}} \tag{17}$$
$$= \max\left\{\frac{\varepsilon_2 e^{\varepsilon_2} - e^{\varepsilon_2} + 1}{e^{\varepsilon_2}(e^{\varepsilon_2} - \varepsilon_2 - 1)}, e^{\varepsilon_2}, e^{\varepsilon_1} \times \frac{e^{\varepsilon_2} - 1}{\varepsilon_2}\right\} = e^{\varepsilon}.$$

The second equality holds because, in SW mechanism, $b = ((\varepsilon_2 e^{\varepsilon_2} - e^{\varepsilon_2} + 1)/2e^{\varepsilon_2}(e^{\varepsilon_2} - \varepsilon_2 - 1))$, $p = (e^{\varepsilon_2}/(2be^{\varepsilon_2} + 1))$ and $q = (1/(2be^{\varepsilon_2} + 1))$.

It is worth noting that the work [8] also proposed a tighter privacy budget composition in their mechanism. However, our tighter privacy budget composition is different from that in [8]. Specifically, the composition theorems hold for different LDP problems. The improved privacy budget composition in [8] holds for the estimation of the key frequency and the mean of the value under LDP. However, our tighter budget composition holds for the estimation of key frequency and the distribution of the value. Moreover, the perturbations are different between our mechanism and [8]. Literature [8] proposed two mechanisms: (1) PCKV-UE and (2) PCKV-GRR. PCKV-UE comprises unary encoding (UE) and randomized response, and PCKV-GRR is based on GRR. However, the components in our mechanism are UE and squared wave (SW) mechanisms. As a result, the privacy budget in our mechanism and literature [8] composes in different ways. The privacy budget is composed as equation (9) in our mechanism. But, in PCKV-UE and PCKV-GRR, the budget is composed as $\max\{\varepsilon_2, \varepsilon_1 + \ln 2/(1 + e^{-\varepsilon_2})\}$ and $\ln((e^{\varepsilon_1+\varepsilon_2} + \lambda)/(\min\{e^{\varepsilon_1}, (e^{\varepsilon_2} + 1)/2\} + \lambda))$, respectively.

Figure 2 shows the (1) basic sequential composition, (2) the tighter composition of our mechanism, and (3) the tighter composition of PCKV (including PCKV-UE and PCKV-GRR). Note that the composition of PCKV-GRR depends on the padding length $l\,(l \geq 1)$ and the larger l results in tighter budget composition. Therefore, we compare PCKV-GRR with varying l. The result shows that the composition of our mechanism is less tight than that of PCKV even under the minimum l, that is, $l = 1$. There is an intuition behind this result. Our mechanism estimates the value distribution under LDP, which needs more information about the data than PCKV that only estimates the mean of the value. Thus, PCKV can bound the privacy loss at a tighter level. □

4.2.2. Privacy Amplification. Figure 2 also shows the relationship between our composition with basic sequential composition, which demonstrates the *privacy amplification* of our mechanism. Compared with a sequential composition where the total privacy budget $\varepsilon = \varepsilon_1 + \varepsilon_2$, our mechanism consumes less privacy budget because $\max\{((\varepsilon_2 e^{\varepsilon_2} - e^{\varepsilon_2} + 1)/e^{\varepsilon_2}(e^{\varepsilon_2} - \varepsilon_2 - 1)), e^{\varepsilon_2}, e^{\varepsilon_1} \times ((e^{\varepsilon_2} - 1)/\varepsilon_2)\} \leq e^{\varepsilon_1+\varepsilon_2}$. In other words, our mechanism has a privacy amplification effect.

4.2.3. No Privacy Amplification Effect from Sampling Protocol. In Theorem 3, the privacy guarantee is independent of the padding length l, which means our mechanism obtains no privacy amplification from the sampling protocol. The main reason is that our mechanism outputs a vector containing multiple keys and multiple positions in the vector are 1. Therefore, even if the sampling protocol is used, the upper bound of the probability ratio in the worst case is independent of the protocol. Here, we take an example that only considers the key perturbation to make this point more clear. Suppose the key domain is $K = \{1, 2, 3, 4\}\,(d = 4)$, $l = 2$, and two key sets $\mathcal{S}_1 = \{1, 2\}$ and $\mathcal{S}_2 = \{3, 4\}$. Note that the output domain is $y = \{0, 1\}^{d+l}$. Then the encoded vector of $\mathcal{S}_1$ is [100000] or [010000], and that of $\mathcal{S}_2$ is [001000] or [000100] (depending on which key is sampled). Since the probability ratio is $\Pr(\mathcal{M}(\mathcal{S}_1) = y)/\Pr(\mathcal{M}(\mathcal{S}_2) = y)$. Thus, in the worst case, we need to maximize the $\Pr(\mathcal{M}(\mathcal{S}_1) = \mathbf{y})$ and minimize $\Pr(\mathcal{M}(\mathcal{S}_2) = \mathbf{y})$. To this end, we select the output vector $y =$ [110000] because, in our mechanism, $1 \longrightarrow 1$ with the highest probability and $0 \longrightarrow 1$ with the smallest probability. Therefore, no matter which key is sampled, the probability of outputting $\mathbf{y}$ is the same, that is, $\Pr(\mathcal{M}(\mathcal{S}_1) = \mathbf{y}) = ac(1-c)^4$ and $\Pr(\mathcal{M}(\mathcal{S}_2) = \mathbf{y}) = c^2(1-a)(1-c)^3$. In other words, there are no privacy benefits from the sampling protocol.

4.2.4. Privacy Budget Allocation. Since our mechanism contains two steps and each of them uses ε_1 and ε_2, allocating a privacy budget for each step is important. A basic and widely used idea to allocate a privacy budget is to calculate the error as the function of ε and then find the optimal ε_1 and ε_2 that minimize the error [8]. However, calculating the error (or distance) between the estimated distribution and the true distribution as the function of ε is a non-trivial task [13]. Thus, we use an empirical allocation method in this paper and leave the strategy of finding the optimal privacy budget allocation method for future work.

In particular, given the total privacy budget ε, we first set $\varepsilon_2 = (\varepsilon/2)$. Then, according to Theorem 3, we can calculate the ε_1 such that the total budget is ε. Specifically, we find the ε_1 such that the term $e^{\varepsilon_1} \times ((e^{\varepsilon_2} - 1)/\varepsilon_2) = \varepsilon$ as the ε_1. This is because, in Theorem 3, the other two terms $\max\{((\varepsilon_2 e^{\varepsilon_2} - e^{\varepsilon_2} + 1)/e^{\varepsilon_2}(e^{\varepsilon_2} - \varepsilon_2 - 1)), e^{\varepsilon_2}\} < \varepsilon$ when $\varepsilon_2 = (\varepsilon/2)$, and only when the third term $e^{\varepsilon_1} \times ((e^{\varepsilon_2} - 1)/\varepsilon_2) = \varepsilon$, the total privacy budget is not violated. Given the privacy budget ε_1, we set the perturbation probability for key perturbation as follows:

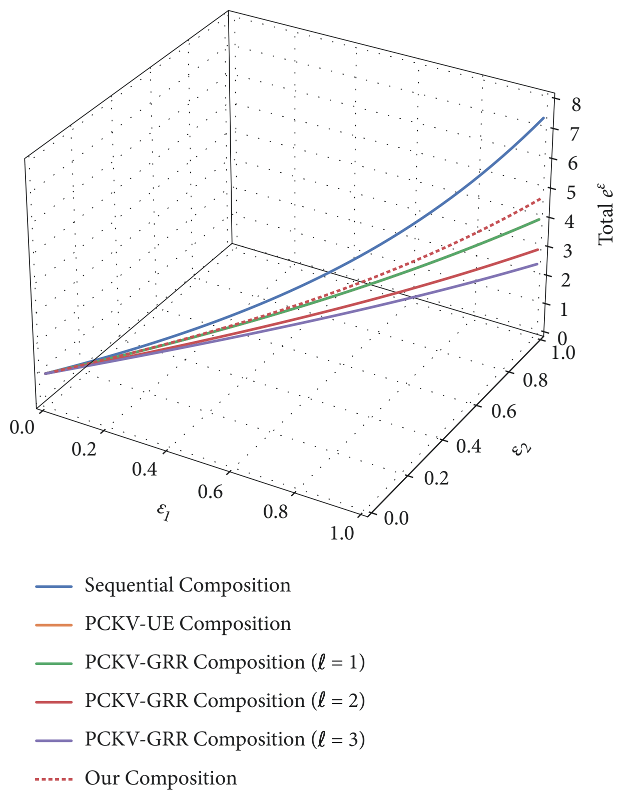


Figure 2: Privacy amplification.

$$a = \frac{1}{2}, \quad c = \frac{1}{e^{\varepsilon_1} + 1}. \tag{18}$$

We note that this perturbation probability is aligned with the optimized unary encoding (OUE) mechanism [10], which achieves the minimum error of frequency estimation under the same privacy budget.

In our experiments, we observe that even under such suboptimal budget allocation, our mechanism is still better than other mechanisms that consider the optimal privacy budget allocation.

4.3. Aggregation and Estimation.

In this subsection, we introduce how to aggregate the perturbed results and get the estimation of the frequency of the key and the estimation of the value distribution. For frequency estimation, an unbiased estimator is proposed in [8, 14]. However, they do not take the prior knowledge of the estimated frequency into account, which reduces the utility. For numerical distribution estimation, the SW mechanism uses the EM algorithm to estimate the distribution. However, due to the fake value in our design (we set $\hat{v}_{\mathbf{x}}^{(i)} = U(-b, 1+b)$ if $k_{\mathbf{x}}^{(i)}$ is perturbed from 0 to 1), directly using the EM algorithm would not get a useful estimation. We use postprocessing methods to address these problems. Note that postprocessing of the output of a DP mechanism does not affect its privacy guarantee [1].

4.3.1. Key Frequency Estimation.

After the server receives the perturbed results from all users, it counts the number of 1's that supports each key i, denoted as $n_i = \text{Count}(\hat{k}_{\mathbf{y}}^{(i)} = 1)$. Then we first use the estimator in [8, 14] to obtain an unbiased frequency estimation $\hat{f}_i$ of key i. Formally,

$$\hat{f}_i = \frac{n_i/(n-c)}{a-c} \times l. \tag{19}$$

Theorem 4. *If the padding length $l \geq |\mathcal{S}_u|$ for all users u, the estimator $\hat{f}_i$ is unbiased, that is, $\mathbb{E}[\hat{f}_i] = f_i$, and the variance is*

$$\text{Var}[\hat{f}_i] = \frac{l^2 c(1-c)}{n(a-c)^2} + \frac{l \times f_i(1-a-c)}{n(a-c)}. \tag{20}$$

Proof. The random variable n_i is the summation of n independent random variables, each of which follows the Bernoulli distribution. For users who input the key i for perturbation (accounting for (f_i/l) of all n users), the variable is drawn from Bernoulli(a), and for users who do not input the key i (accounting for $1-(f_i/l)$ of all n users), the variable is drawn from Bernoulli(c). Thus, we have the expectation of estimator $\hat{f}_i$ that is

$$\begin{aligned}\mathbb{E}[\hat{f}_i] &= \frac{l}{a-c}\mathbb{E}\left[\frac{n_i}{n} - c\right] \\ &= \frac{l[n(f_i/l)a + n(1-(f_i/l))c] - c}{n(a-c)} = f_i,\end{aligned} \tag{21}$$

and the variance of the estimator is

$$\begin{aligned}\text{Var}[\hat{f}_i] &= \frac{l^2}{(a-c)^2}\text{Var}\left[\frac{n_i}{n} - c\right] \\ &= \frac{l^2[n(f_i/l)a(1-a) + n(1-(f_i/l))c(1-c)]}{n^2(a-c)^2} \\ &= \frac{l^2 c(1-c)}{n(a-c)^2} + \frac{l \times f_i(1-a-c)}{n(a-c)}.\end{aligned} \tag{22}$$

□

4.3.2. Improve the Utility with Postprocessing.

The estimator $\hat{f}_i$ only provides unbiasedness in theory. However, the estimation may not be *consistent*. That is, the estimations for many values may be negative, and the total sum of frequencies is not equal to 1. Such inconsistency may reduce the utility of LDP mechanisms [11]. Therefore, we further enforce the following consistency requirements on the estimated results to improve the utility:

(1) The estimated frequencies are non-negative

(2) The sum of the estimated frequencies is 1

To achieve consistency given the estimated results $\hat{f} = [\hat{f}_1, \ldots, \hat{f}_{d'}]$, we solve the following optimization problem and find the postprocessed results $\tilde{f} = [\tilde{f}_1, \ldots, \tilde{f}_{d'}]$ as the estimation:

$$\min_{\tilde{f}} \|\hat{f} - \tilde{f}\|_2^2$$
$$\text{s.t.} \begin{array}{l} \tilde{f} \geq 0 \\ \sum_i \tilde{f}_i = 1. \end{array} \tag{23}$$

Based on the KKT condition [15], we can solve the postprocessed results as follows:

$$\tilde{f}_i = \hat{f}_i + \frac{1}{|A|}\left(1 - \sum_{i\in A}\hat{f}_i\right), \tag{24}$$

where A is the set containing the non-negative frequencies in $\hat{f}$.

We further explain why we use $\|\hat{f} - \tilde{f}\|_2^2$ as the objective function. L_2 norm is used in the objective function because the noise by UE is well approximated by Gaussian noise, and minimizing L_2 norm achieves MLE [13]. Besides, when we enforce the consistency requirement on the estimated results, there are many results that can achieve the consistency requirements. For example, suppose $\hat{f} = [-0.1, 0.5, 0.8]$, the postprocessed results $\tilde{f} = [0, 0.3, 0.7]$ and $\tilde{f} = [0, 0, 1]$ are both consistent. However, the postprocessed results that are far from the estimated results lead to poor utility. This is because the $\hat{f}$ is a useful unbiased estimator for each key, and a large deviation from it results in a large error. Therefore, the postprocessed results should not only be consistent but also be close to the estimated results.

Theorem 5 proves that postprocessing leads to positive bias for frequency estimation.

Theorem 5. *Given the unbiased estimated results $\hat{f}$, the corresponding postprocessed results $\tilde{f}$ solved by (13) lead to positive biases.*

Proof. For each key i, we have the bias that is

$$\begin{aligned} \mathbb{E}\left[\tilde{f}_i - f_i\right] &= \mathbb{E}\left[\tilde{f}_i\right] - f_i \\ &= \mathbb{E}\left[\hat{f}_i\right] + \frac{1}{|A|}\left(1 - \sum_{i\in A}\mathbb{E}\left[\hat{f}_i\right]\right) - f_i \\ &= \frac{1}{|A|}\left(1 - \sum_{i\in A} f_i\right). \end{aligned} \tag{25}$$

Since $\sum_{i\in A} f_i \leq 1$, thus, we have $\mathbb{E}[\tilde{f}_i - f_i] \geq 0$.

In many application domains, the number of users is large, and the true frequency of many keys is far from zero. Thus, few estimated results may be negative, and $|A|$ may be large. Therefore, even though the postprocessing introduces a positive bias, it is sufficiently small in practice.

4.3.3. Numerical Distribution Estimation

The server performs the distribution estimation in a discretized way, that is, the histogram on the domain. For a key i, the server first finds the reported results that support key i, that is, the i-th bit of the perturbed vector $\hat{k}_{\mathbf{x}}^{(i)} = 1$. Then it discretizes the received value (in domain $[-b, 1+b]$) into m buckets and construct a histogram with m bins, where each bin corresponds to the count of values falling in this bin. Since the value of the reported results that support key i is the result perturbed either from the true value or from the fake value (the fake value would affect the distribution estimation), the server then statistically removes the fake value and reconstructs a histogram with m bins (in domain $[0, 1]$) as the estimated value distribution. We denote the histogram constructed by the true values as H (with m bins), the reconstructed histogram as $\tilde{H}$, and the i-th bin of H and $\tilde{H}$ as B_i and $\tilde{B}_i$. Next, we introduce how to statistically remove the fake value and then elaborate on the method of reconstructing the histogram.

Since the fake value is generated by users who do not possess the key i but report as possessed, we first calculate the number of such users. Given the estimated frequency $\tilde{f}_i$ for the key i, we can calculate the count of such users is approximately $n(1 - \tilde{f}_i)c$. Since the fake value is drawn from the uniform distribution $U(-b, 1+b)$, the count of such users in each bin of the histogram is approximately $(n(1 - \tilde{f}_i)c/m)$. Thus, the server can statistically remove the fake value by subtracting $(n(1 - \tilde{f}_i)c/m)$ from each bin in the histogram. Then the server divides each bin by the count of users who really possess the key i, that is, $(n\tilde{f}_i a/l)$, to obtain the frequency of values falling in each bin. Denote this histogram by $\hat{H}$ and the i-th bin of $\hat{H}$ by $\hat{B}_i$, the frequency of values falling in each bin $\hat{B}_i$ can be used to estimate the probability $\Pr(v \in \hat{B}_i)$. Leveraging the probability $\Pr(v \in \hat{B}_i)$ for all $\hat{B}_i$, the server can reconstruct the histogram and obtain the numerical distribution estimation, that is, the probability $\Pr(v \in \tilde{B}_i)$ for each $\tilde{B}_i$. Note that we denote the probability $\Pr(v \in \hat{B}_i)$ by $\hat{P}_{B_i}^{v}$ and $\Pr(v \in \tilde{B}_i)$ by $\tilde{P}_{B_i}^{v}$ for brevity.

Given the histogram of the frequency of values, the server uses the EM algorithm to reconstruct the histogram (in domain $[0, 1]$) and obtain the estimated value distribution. Denote the number of values falling in the $\hat{B}_i$ by $\hat{N}_i$; the overall algorithm is shown in Algorithm 3. □

5. Experiments

5.1. Setup

5.1.1. Data Sets. Four real-world data sets are involved in our evaluation: E-commerce [16], Clothing [17], Amazon [18], and Movie [19]. We summarize the data sets parameters in Table 2, where l is the padding length. All rating values are linearly transformed into the range $[0, 1]$.

5.1.2. Competitors. We compare our mechanism with three existing mechanisms: PrivKVM* [9], PCKV [8], and KVUE [20]. PrivKVM* is elaborated in Section 3.2, and we do not repeatedly introduce it here. PCKV contains two mechanisms, namely PCKV-UE and PCKV-GRR, which are based on optimal unary encoding (OUE) and generalized random response (GRR), respectively, and we compare both in this paper. KVUE is a mechanism proposed to improve the performance of PrivKVM [7], which is the degraded version of PrivKVM*. It treats each key-value pair as a whole entity instead of treating key and value separately and directly perturbs each entity.

```
     Input: Perturbed results, estimated frequency $\tilde{f}$, padding length $l$, number of bins $m$.
     Output: Reconstructed histogram
 (1) Discretize the perturbed results into $m$ bins.
 (2) Subtract $(n(1-\tilde{f}_i)c/m)$ from each bin.
 (3) Divide each bin by $(n\tilde{f}_i a/l)$ to generate $\hat{H}$.
 (4) Initialize $\tilde{P}^{v}_{B_i} = (1/m)$ for all $i = 1, \ldots, m$.
 (5) while not converging do
 (6)     E-step $\forall i = 1, \ldots, m$, $P_i = \tilde{P}^{v}_{B_i} \sum_{j=1}^{m} \hat{N}_j \Pr(v \in \hat{B}_j | v \in B_i) / \sum_{k=1}^{m} \Pr(v \in \hat{B}_k | v \in B_j) \tilde{P}^{v}_{B_k}$
 (7)     M-step $\forall i = 1, \ldots, m$, $\tilde{P}^{v}_{B_i} = (P_i / \sum_{j=1}^{m} P_j)$
 (8) end while
 (9) Reconstructed histogram $\tilde{H} = [\tilde{P}^{v}_{B_1}, \tilde{P}^{v}_{B_2}, \ldots, \tilde{P}^{v}_{B_m}]$
(10) return reconstructed histogram $\tilde{H}$
```

Algorithm 3: Numerical distribution estimation.

Table 2: Data sets.

Data sets	#Ratings	#Users	#Keys	Selected l
E-commerce	23,486	23,486	1,206	3
Clothing	192,544	105,508	5,850	5
Amazon	2,023,070	1,210,271	249,274	5
Movie	20,000,263	138,493	26,744	100

Since only PrivKVM* can support frequency estimation of keys and distribution estimation of values and other mechanisms are only designed for frequency estimation and mean estimation of values, we compare with PrivKVM* on both frequency and distribution estimation tasks and compare with PCKV and KVUE only on frequency and mean estimation.

5.1.3. Evaluation Environments. All mechanisms are implemented using Python 3.6 and Numpy 1.14. All experiments are conducted on an Amax server. The operating system of the machine is Ubuntu 16.04; the CPU is Intel Xeon Silver 4214 2.2 GHz, 24 cores in total; and the memory is DDR4-2666, with a total of 128 GB.

5.2. Metric

5.2.1. Frequency. We evaluate the key frequency by the mean squared error (MSE). Formally, we measure

$$\mathrm{MSE}_{\mathrm{freq}} = \frac{1}{|\mathcal{X}|} \sum_{i \in \mathcal{X}} \left(\tilde{f}_i - f_i\right)^2, \tag{26}$$

where $\mathcal{X}$ is any subset of the key domain K, and we set the default $\mathcal{X}$ to be K.

5.2.2. Distribution Distance. We evaluate the distribution estimation by the average Wasserstein distance. Formally, we measure

$$AW_{\mathrm{dist}} = \frac{1}{|\mathcal{X}|} \sum_{k \in \mathcal{X}} W_k(H, \tilde{H}), \tag{27}$$

where $\mathcal{X}$ is any subset of the key domain K, and we set the default $\mathcal{X}$ to be K. $W_k(H, \tilde{H})$ is the Wasserstein distance between the true value distribution of the key k and the estimated distribution. Formally, given the histogram H constructed by the true value of key k and the reconstructed histogram $\tilde{H}$, the Wasserstein distance is

$$W_k(H, \tilde{H}) = \sum_{j=1}^{m} \left| \sum_{i=1}^{j} \Pr\left(v \in B_i\right) - \sum_{i=1}^{j} \Pr\left(v \in \tilde{B}_i\right) \right|. \tag{28}$$

5.2.3. Mean and Variance. Given the estimated value distribution, we can also calculate the mean and the variance of the value. We also use the MSE to evaluate the mean estimation and variance estimation. Formally, we measure

$$\mathrm{MSE}_{\mathrm{mean}} = \frac{1}{|\mathcal{X}|} \sum_{i \in \mathcal{X}} \left(\tilde{\mu}_i - \mu_i\right)^2, \mathrm{MSE}_{\mathrm{var}} = \frac{1}{|\mathcal{X}|} \sum_{i \in \mathcal{X}} \left(\tilde{\sigma}_i^2 - \sigma_i^2\right)^2, \tag{29}$$

where similar to frequency estimation, $\mathcal{X}$ is any subset of the key domain $\mathcal{K}$ and we set the default $\mathcal{X}$ to be K; $\tilde{\mu}_i$ and μ_i are the estimated mean and the true mean of the value of the key i, respectively; and $\tilde{\sigma}_i^2$ and σ_i^2 are the estimated variance and the true variance of the value of the key i, respectively.

All metrics measure the error between the estimated result and the true result, and the smaller the metric, the more accurate the estimated result. All results are averaged with 50 repeats to make the experiment results stable.

5.3. Key Frequency. We first evaluate the existing LDP mechanisms on key frequency estimation. Here, we analyze these methods on three tasks:

(1) Frequency of individual key: We measure MSE between $[\widetilde{f}_i]_{i\in K}$ and $[f_i]_{i\in K}$. In this task, the $\mathscr{X}$ is the key domain K.

(2) Frequency of most frequent keys: We select the top-T key and measure the MSE between their actual frequencies and the postprocessed ones. Formally, denote the top-T keys by the set $D_T = \{i \in K | f_i$ rank stop $T\}$ and measure MSE between $[\widetilde{f}_i]_{i\in D_K}$ and $[f_i]_{i\in D_K}$. We also set the default $T = 15$ and $\varepsilon = 1$. In this task, $\mathscr{X}$ is the domain of the top-T key.

(3) Frequency of subsets of keys: Estimating the subset of keys plays an important role in the interactive data analysis setting (e.g., estimating which category of products is more popular). We uniformly sample α $(0 \le \alpha \le 1)$ data from the domain of key and measure the MSE between the sum of the actual frequencies and the postprocessed frequencies. Formally, suppose D_α is the random sampled subset of the key that has $\alpha \times K$ keys, we define $f_{D_\alpha} = \sum_{i\in D_\alpha} f_i$ and $\widetilde{f}_{D_\alpha} = \sum_{i\in D_\alpha} \widetilde{f}_i$. We sample D_α 100 times and measure MSE between f_{D_α} and $\widetilde{f}_{D_\alpha}$. We set the default $\alpha = 30\%$ and $\varepsilon = 1$.

5.3.1. Frequency of Individual Key. We first evaluate the performance when querying the frequency of individual keys, and the results are shown in Figure 3. As a result, we conclude that our method is better than any other methods (the MSE of our method is the smallest) on all data sets because we enforce the consistency as postprocessing. Especially when the noise is large $(\varepsilon \le 2)$, our method reduces the MSE of the state-of-the-art solution by about 2 orders of magnitude. This is because the estimated frequencies are prone to be inconsistent under large noise and our postprocessing improves the accuracy significantly. This also happens to the other two tasks for a similar reason (see Figures 4and 5). On data sets E-commerce, Clothing, and Amazon, the MSE results of other existing methods are very similar; this is because the number of users in data sets Clothing and Amazon are large, and it compensates for the impact of the large domain of the key. However, our method shows the smallest MSE in data set Amazon among all four data sets. This is because the number of users on Amazon are largest, which lead to smaller bias and better accuracy (according to the analysis of our postprocessing in Theorem 5). In data set Movies, although all methods do not perform as good as they do on the first three data sets (due to large padding length l leading to large error), our method still performs best among all mechanisms due to the consistency requirement.

5.3.2. Frequency of Most Frequent Keys. The MSE results when querying the top-T frequent keys under varying T and ε values are shown in Figures 4 and 6. Overall, our mechanism significantly reduces the MSE of other methods under all ε values and T values in most cases. Similar to the results when querying the frequency of individual keys, the MSE of our method is also apparently lower than that of other solutions when the noise is large $(\varepsilon \le 2)$. As the ε value grows, the decline in the MSE of our method is becoming stable. Figure 6 represents the MSE of our mechanism is significantly smaller than other mechanisms on all data sets under all T, which actively demonstrates that our method can cope with various queries for top-T frequent keys.

5.3.3. Frequency of Subsets of Key. We show the results for frequency estimation of a subset of keys in Figures 5 and 7. Overall, our method outperforms other mechanisms under all ε values and α values. In Figure 5, the MSEs of all mechanisms decrease as the ε value grows, and the large gap between our method and other methods indicates our method performs much better than other existing methods. Moreover, it is worth noting that in Figure 7, the MSEs of other mechanisms are getting greater as the α grows, but the MSE of our method is symmetric with $\alpha = 50\%$. This is because the individual estimation error accumulates as α increases under other mechanisms, but we enforce consistency on the estimated results and all estimated frequencies are summing-to-1; thus, estimating the frequency of a subset for $\alpha > 50\%$ is equivalent to estimating the rest.

5.4. Distribution. We evaluate existing LDP mechanisms on distribution estimation. Here, we evaluate it from three perspectives: (1) distribution distance, (2) mean, and (3) variance. We compare our mechanism with PrivKVM* on all three tasks and compare with other mechanisms only on mean estimation since they are only designed for mean. We also set the number of buckets $m = 1024$ in our experiments as it has been shown to perform best in most cases for distribution estimation [13].

5.4.1. Distribution Distance. We plot the AW results as the function of ε value in Figure 8. It shows our mechanism outperforms PrivKVM* and achieves a reasonable distribution estimation on all data sets, and the largest AW is only about 10^{-1}. This is because PrivKVM* perturbs numerical values in a discrete manner and does not exploit the ordinal information of the numerical domain. It is also worth noting that the AW results on the first three data sets (E-commerce, Clothing, and Amazon) are similar and lower than that on data set Movies. This is because the padding length l for data set Movies is the largest $(l = 100)$, which leads to a large error for frequency estimation (see Figure 3). Thus, we may get a relatively inaccurate number of users who generate fake values when we statistically remove the fake value, which leads to high AW for distribution estimation.

5.4.2. Mean. The evaluation of mean estimation is shown in Figure 9. As a result, our method performs much better than any other mechanisms under all ε value. Specifically, when ε is relatively small, our mechanism significantly reduces MSEs of all other solutions; when ε is larger than 4, the MSE

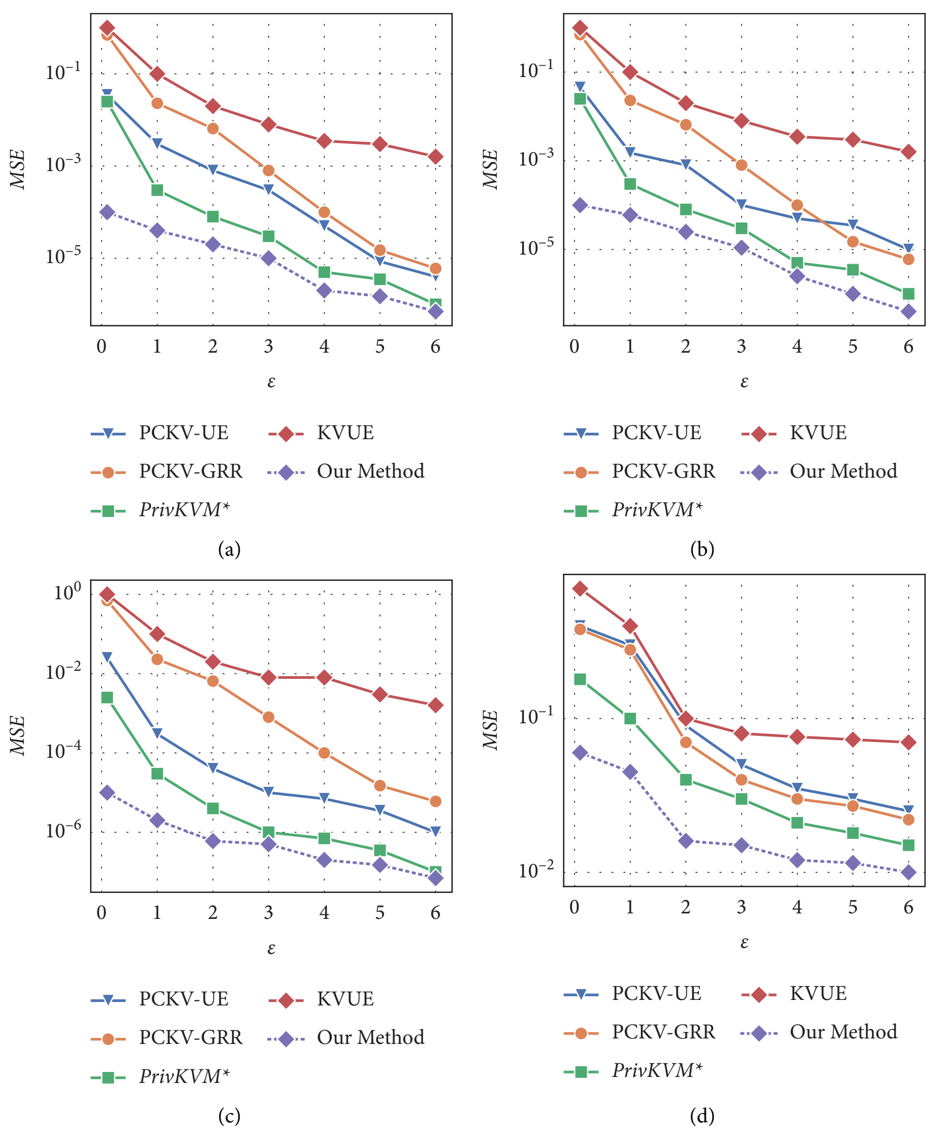


FIGURE 3: MSE results on individual key frequency. (a) E-commerce. (b) Clothing. (c) Amazon. (d) Movies.

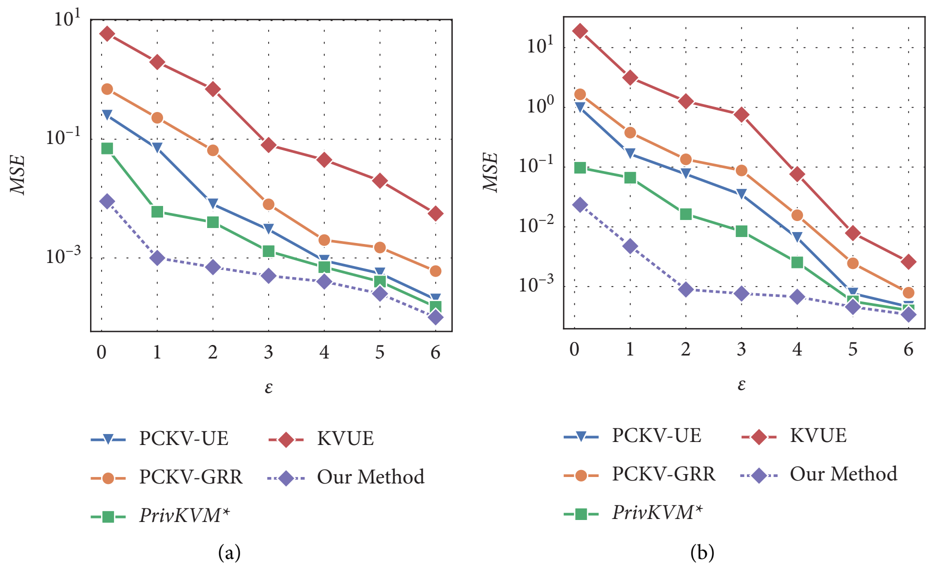
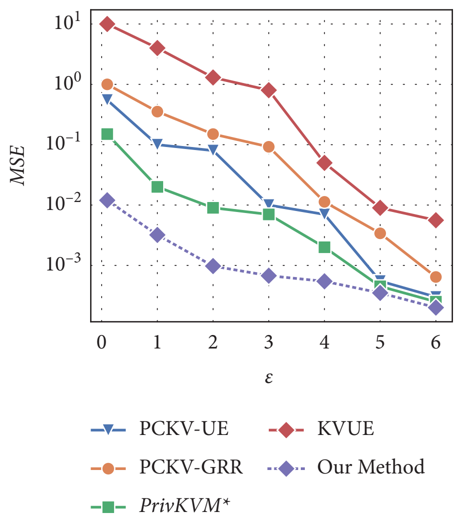
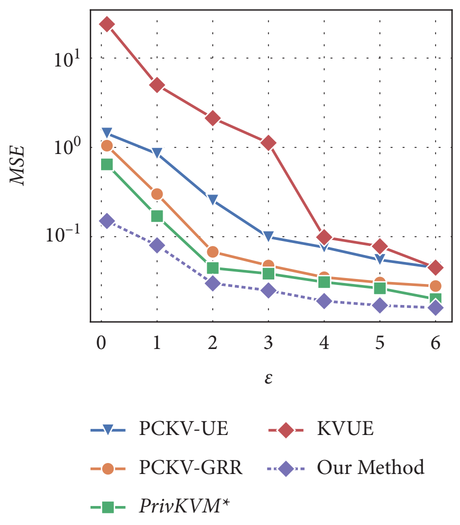


FIGURE 4: Continued.

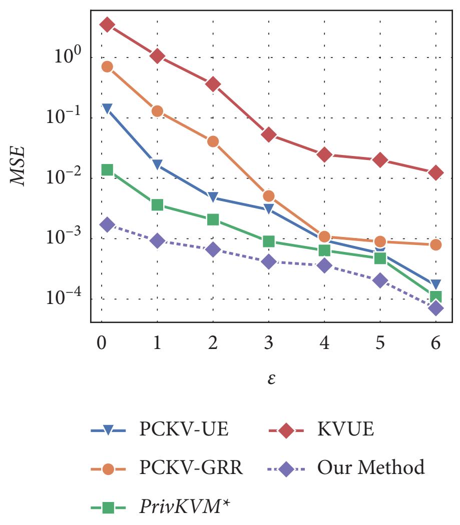


(c)

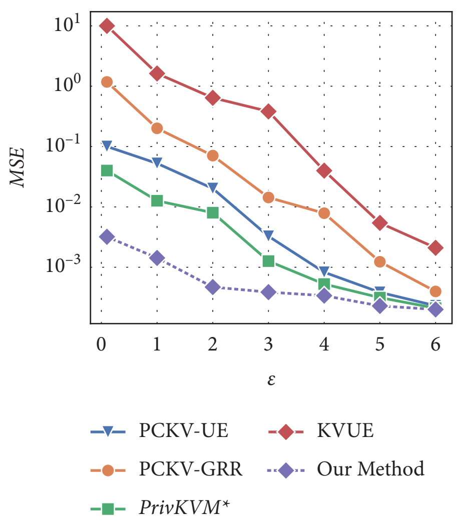


(d)

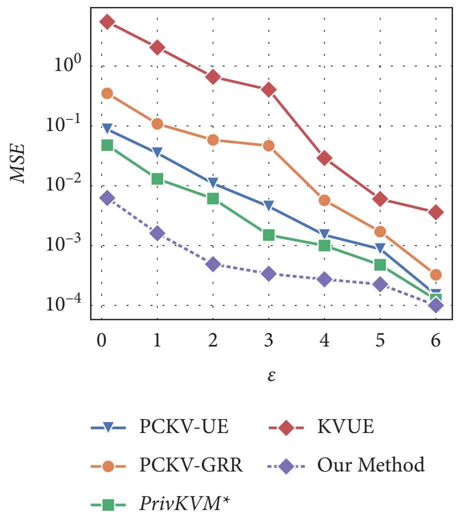
FIGURE 4: MSE results on top-T key frequency varying ϵ from 0.1 to 6, fixing $T = 15$. (a) E-commerce. (b) Clothing. (c) Amazon. (d) Movies.



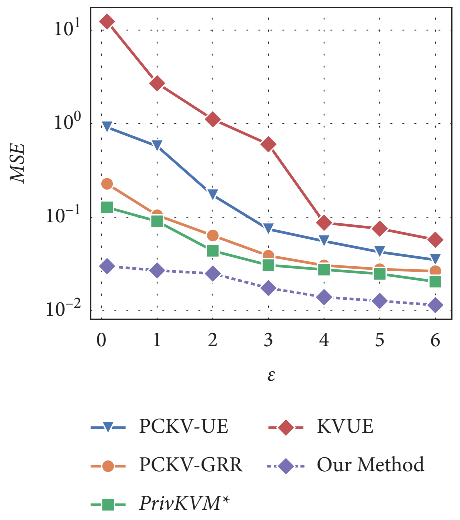
(a)



(b)

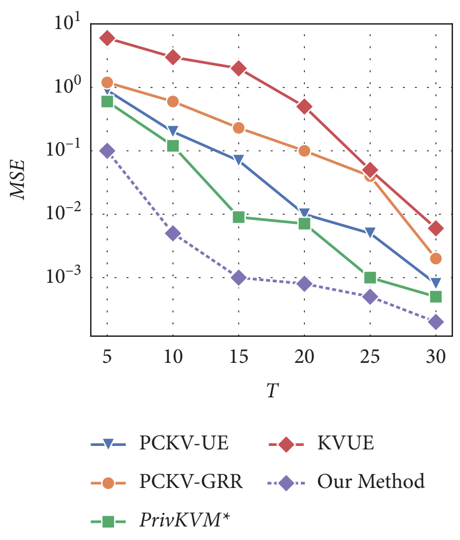


(c)

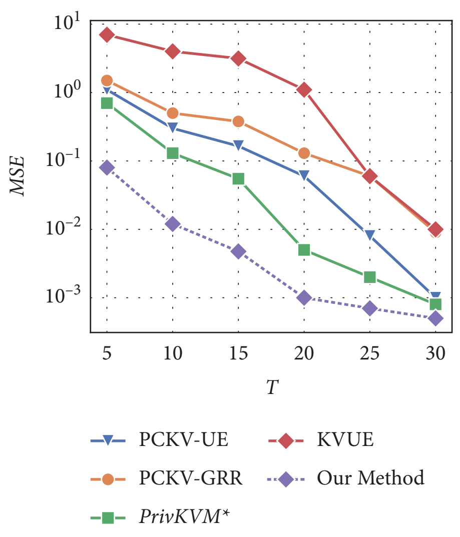


(d)

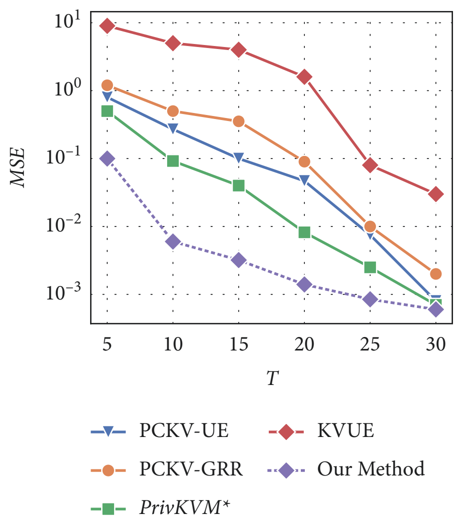
FIGURE 5: MSE results on subset-key frequency varying ϵ from 0.1 to 6, fixing $\alpha = 30\%$. (a) E-commerce. (b) Clothing. (c) Amazon. (d) Movies.



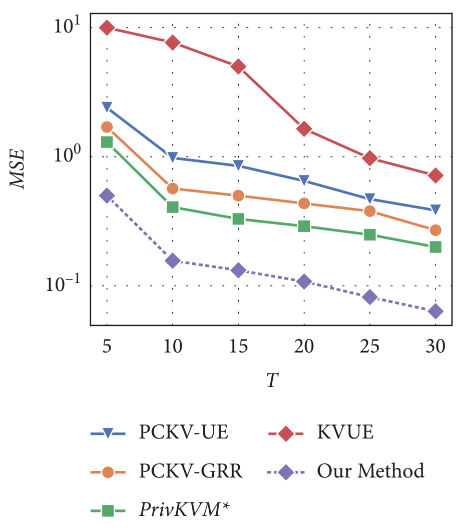
(a)



(b)

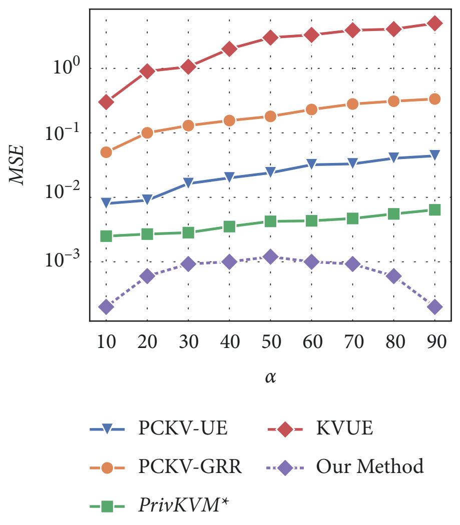


(c)

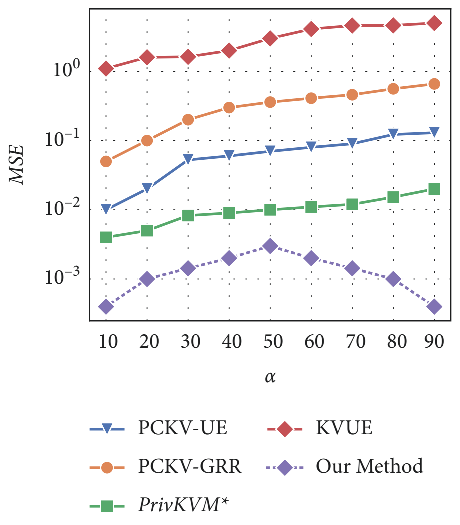


(d)

Figure 6: MSE results on top-T key frequency varying T from 5 to 30, fixing $\epsilon = 1$. (a) E-commerce. (b) Clothing. (c) Amazon. (d) Movies.

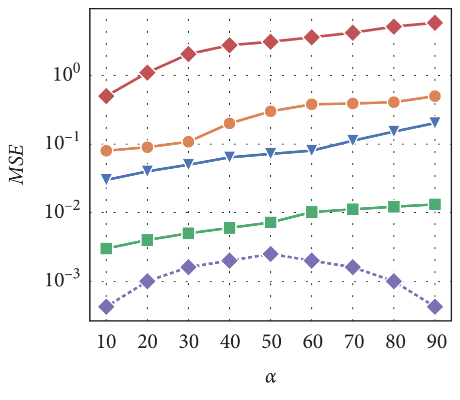


(a)

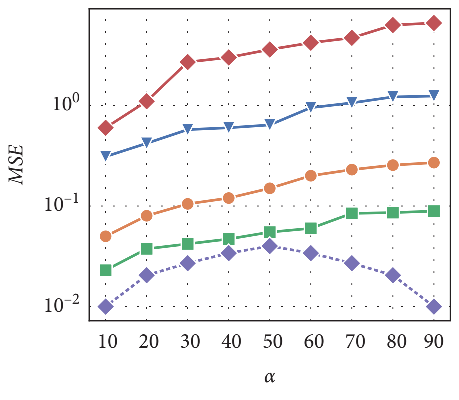


(b)

Figure 7: Continued.

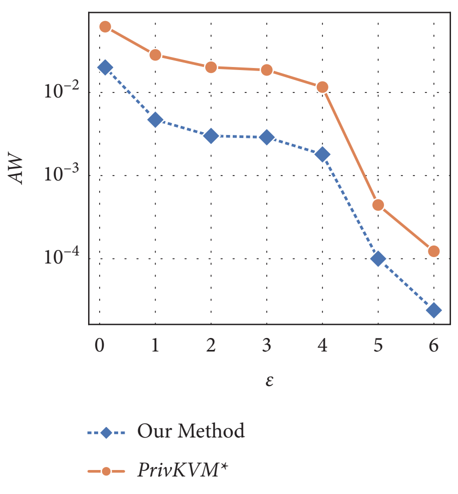


(c)

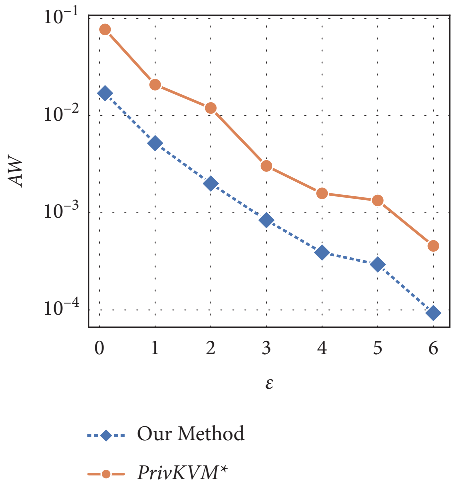


(d)

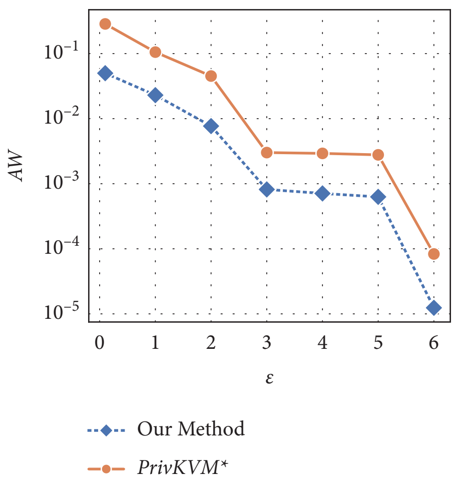
Figure 7: MSE results on subset-key frequency varying α from 10% to 90%, fixing $\epsilon = 1$. (a) E-commerce. (b) Clothing. (c) Amazon. (d) Movies.



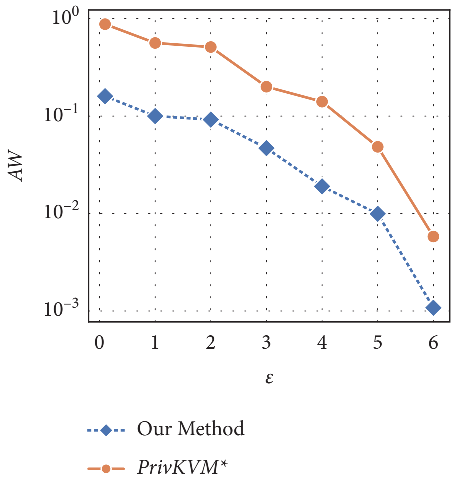
(a)



(b)



(c)



(d)

Figure 8: AW results on distribution. (a) E-commerce. (b) Clothing. (c) Amazon. (d) Movies.

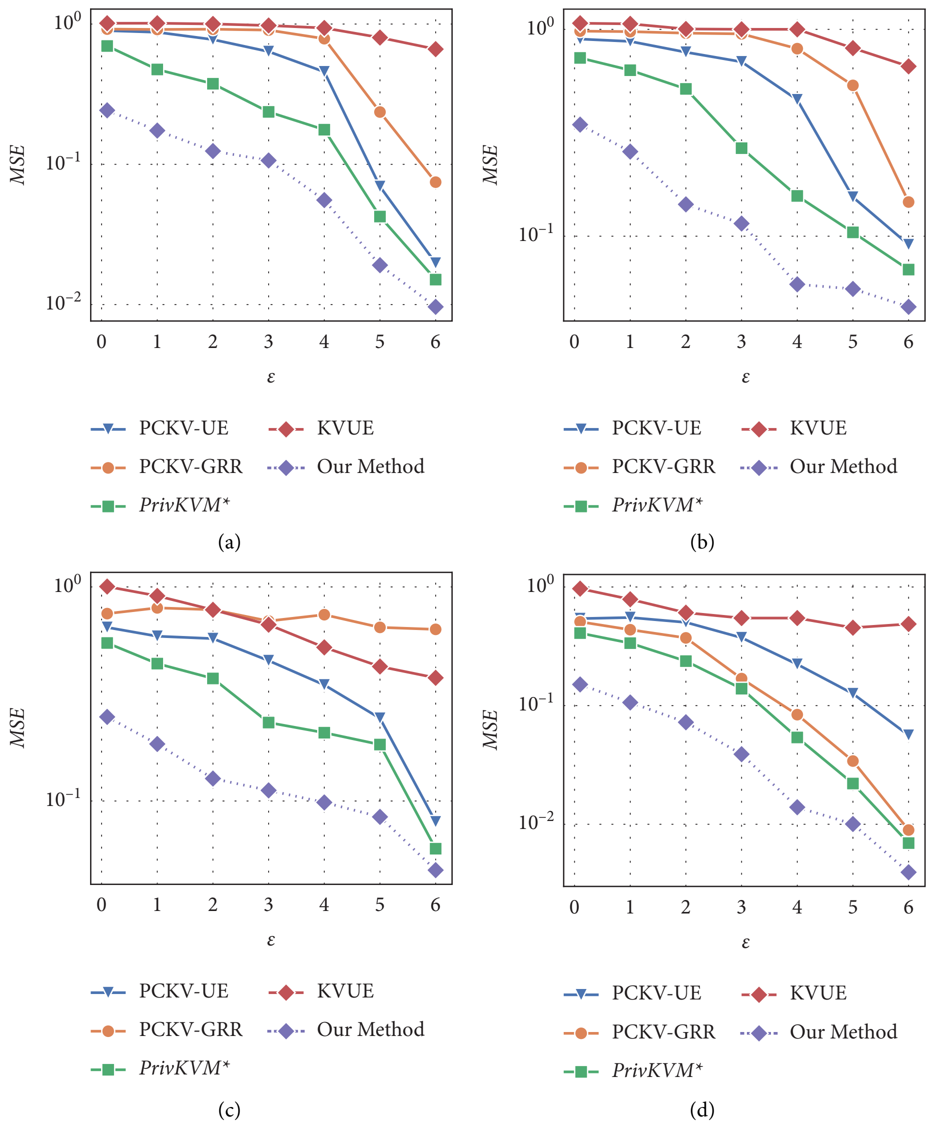


Figure 9: MSE results on mean. (a) E-commerce. (b) Clothing. (c) Amazon. (d) Movies.

of our mechanism is one to two orders of magnitude smaller than most other solutions. This is because our mechanism reports the value closer to the original value with a higher probability than the value far away from the original value. In this way, such perturbed result can carry more useful information about the original value and leads to more accurate results.

5.4.3. *Variance.* Figure 10 plots the MSE results as the function of ε value. Due to the categorical frequency oracle, PrivKVM* underperforms in our experiments. It is worth noting that the MSE on data set Movies is the highest. This is also because the largest padding length $l = 100$ for data set Movies leads to a large error for frequency estimation (see Figure 3) and results in a relatively inaccurate number of users who generate fake values when we statistically remove the fake value.

6. Related Work

Differential privacy has been the de facto standard for privacy-preserving. There are many LDP deployments in the real world: Google Chrome extension [3], spelling prediction of Apple [2], and telemetry collection by Microsoft [21].

6.1. *Frequency Oracle and Distribution Estimation.* Estimating the frequency of values is a basic task in LDP. There have been several mechanisms [3, 10, 22, 23] proposed for this task, and they are often called *frequency oracles*. For example, RAPPOR [3] enables the estimation of the marginal frequencies of a set of strings. However, it needs a dictionary for the candidate strings, which can be very large or unknown in practice. To solve this problem, Fanti et al. [24] use the EM algorithm as a decoder for RAPPOR to enable learning without explicit dictionary knowledge. Based on RAPPOR, Ren et al. [25] propose a novel

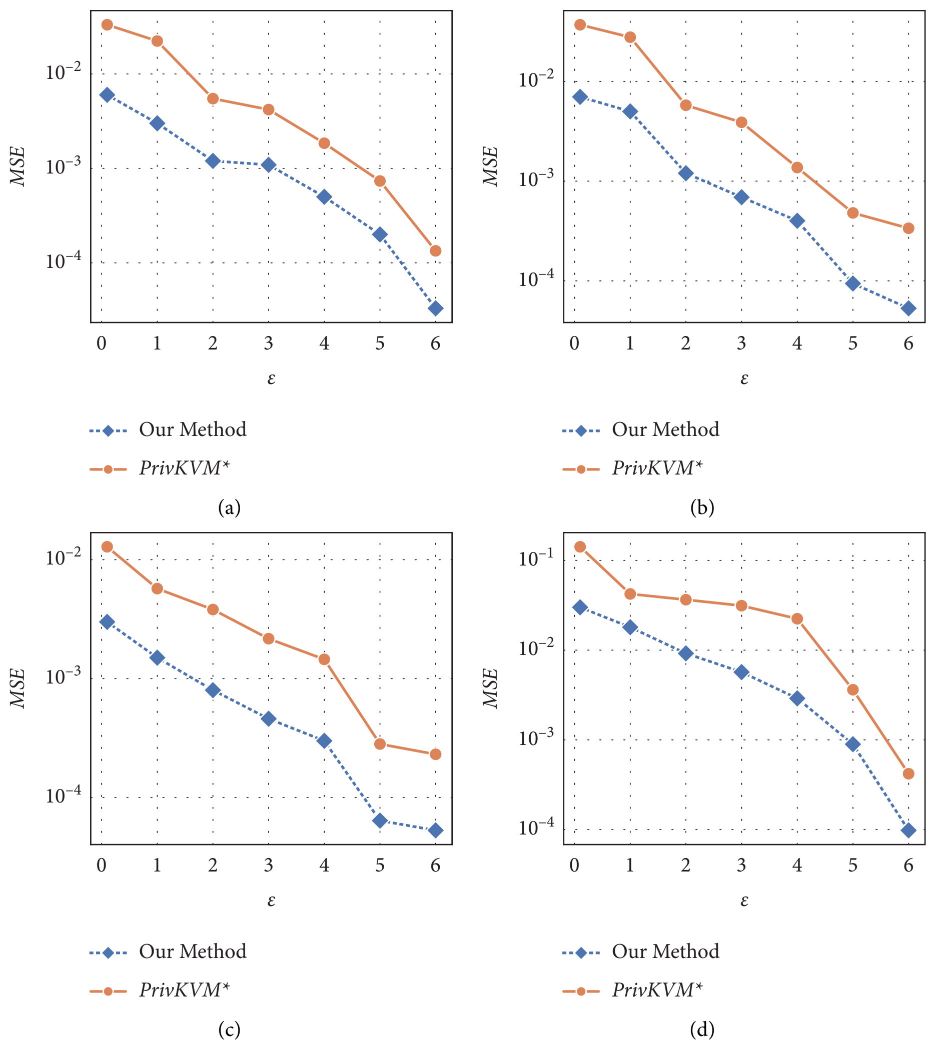


Figure 10: MSE results on variance. (a) E-commerce. (b) Clothing. (c) Amazon. (d) Movies.

mechanism to estimate distribution for high-dimensional data. Instead of the EM algorithm, they use Lasso regression to estimate the distribution in one round. Combining the EM algorithm and Lasso regression, Ren et al. [26] further propose a solution that can generate synthetic data by leveraging the estimated distribution of the data under LDP. Although the above schemes also use the EM algorithm, there are two differences compared to the EM algorithm in our mechanism: (1) our EM algorithm can statistically remove the fake values and (2) it takes the aggregated results and is thus more efficient.

When estimating the distribution of numerical data, a naïve approach is to bucketize the data and apply the categorical frequency oracles listed above. In [4], the authors achieve distribution estimation under LDP but with a strictly weaker privacy guarantee. There are also mechanisms that can handle numerical settings but focus on the specific task of mean estimation, that is, SR [5, 21] and PM [27]. The SW mechanism [13] is the state-of-the-art mechanism for distribution estimation tasks under LDP, which can recover the distribution instead of focusing on a specific task.

Different from existing LDP mechanisms that only focus on simple statistical queries (such as frequency and mean), our paper designs a new LDP mechanism for key-value data collection that considers both key frequency and value distribution simultaneously.

6.2. Postprocessing. For statistic tasks in differential privacy, one can utilize the structural information to postprocess and improve the data accuracy. Following this idea, Hay et al. [28] utilize the structural information and propose an efficient hierarchical method to minimize L_2 difference between the noisy result and the processed result. Besides that, Lee et al. [29] consider the non-negativity constraint and propose to use the alternating direction method of multipliers (ADMM) to obtain a result that achieves maximal likelihood. Wang et al. [11] further improve the data accuracy by enforcing consistency that the frequency should be non-negative and sum-to-one. Jia and Gong [30] use conditional expectation to estimate the true data given the LDP-protected results. This method shows satisfactory results when data approximately follows power-law distribution. EM algorithm is used by [13] to improve the accuracy of histogram data when estimating numerical distribution.

In this paper, we adopt postprocessing for key frequency estimation to further improve the accuracy.

6.3. Key-Value Data Collection. Ye et al. [7] are the first to propose the LDP mechanism to collect key-value data called PrivKV, PrivKVM, and PrivKVM+. PrivKVM iteratively estimates the mean to guarantee unbiasedness. PrivKV is a simple version of PrivKVM, and it can be regarded as PrivKVM with only one iteration. To balance unbiasedness and communication cost, they also propose the advanced version of PrivKVM called PrivKVM+. Sun et al. [20] proposed another estimator for frequency and mean estimation under the PrivKV to achieve better accuracy. They also introduced conditional analysis for key-value data for other complex analysis tasks in machine learning. Gu et al. [8] proposed the framework PCKV. It perturbs the key and value in a correlated manner and provides a tighter privacy budget composition. As a result, PCKV outperforms the above LDP mechanisms in both estimation of the key frequency and the estimation of the value mean. To the best of our knowledge, PrivKVM* [9] is the state-of-the-art mechanism that not only can support more statistical tasks but also can achieve the best accuracy in most cases.

7. Discussion and Conclusion

In this paper, we propose a novel LDP mechanism for private key-value data collection. Due to the consideration of numerical information of the value domain, our mechanism outperforms existing schemes in most cases. The mechanism perturbs the key-value data in a correlated manner and results in the privacy amplification effect. We further improve the accuracy of the frequency estimation by consistency. Finally, we evaluate our mechanism on four real-world data sets and demonstrate our mechanism outperforms existing schemes.

Although our mechanism performs well in our experiments, it still has the following limitations:

(1) We do not consider the optimal padding length l that would lead to more accurate results.

(2) Our mechanism only adopts the suboptimal privacy budget allocation scheme instead of studying the optimal allocation scheme. Although our method still outperforms previous mechanisms, it does not achieve the minimum error.

In future work, we will study the optimal padding length l that can further improve the privacy-utility trade-off and study the optimal privacy budget allocation.

Data Availability

The key-value data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was funded by NSFC (61932015 and 61732022), Shaanxi Innovation Team Project (2018TD-007), and Higher Education Discipline Innovation 111 Project (B16037).

References

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

[2] Apple Team, “Learning with privacy at scale,” *Apple Mach. Learn. J*, vol. 1, no. 8, pp. 1–25, 2017.

[3] U. Erlingsson, V. Pihur, and A. Korolova, “Rappor: Randomized aggregatable privacy-preserving ordinal response,” in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pp. 1054–1067, 2014.

[4] S. Wang, Y. Nie, P. Wang, H. Xu, W. Yang, and L. Huang, “Local private ordinal data distribution estimation,” in *Proceedings of the IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pp. 1–9, IEEE, Atlanta, GA, USA, May 2017.

[5] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, “Minimax optimal procedures for locally private estimation,” *Journal of the American Statistical Association*, vol. 113, no. 521, pp. 182–201, 2018.

[6] N. Wang, X. Xiao, Y. Yang et al., “Collecting and analyzing multidimensional data with local differential privacy,” in *Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 638–649, IEEE, Macao, China, April 2019.

[7] Q. Ye, H. Hu, X. Meng, and H. Zheng, “Privkv: key-value data collection with local differential privacy,” in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, pp. 317–331, IEEE, San Francisco, CA, USA, May 2019.

[8] X. Gu, M. Li, Y. Cheng, L. Xiong, and Y. Cao, “Pckv: locally differentially private correlated key-value data collection with optimized utility,” in *Proceedings of the 29th USENIX Security Symposium (USENIX Security 20)*, pp. 967–984, 2020.

[9] Q. Ye, H. Hu, X. Meng et al., “Privkvm*: revisiting key-value statistics estimation with local differential privacy,” *IEEE Transactions on Dependable and Secure Computing*, 2021.

[10] T. Wang, J. Blocki, N. Li, and S. Jha, “Locally differentially private protocols for frequency estimation,” in *Proceedings of the 26th USENIX Security Symposium (USENIX Security 17)*, pp. 729–745, 2017.

[11] T. Wang, M. Lopuhaa-Zwakenberg, Z. Li, B. Skoric, and N. Li, “Locally differentially private frequency estimation with consistency,” in *Proceedings of the NDSS’20: Proceedings of the NDSS Symposium*, 2020.

[12] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, “Local privacy and statistical minimax rates,” in *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pp. 429–438, IEEE, Berkeley, CA, USA, October 2013.

[13] Z. Li, T. Wang, M. Lopuha-Zwakenberg, N. Li, and B. Skoric, “Estimating numerical distributions under local differential privacy,” in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 621–635, 2020.

[14] T. Wang, N. Li, and S. Jha, "Locally differentially private frequent itemset mining," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*, pp. 127–143, IEEE, San Francisco, CA, USA, May 2018.

[15] H. W. Tucker and A. W. Tucker, "Nonlinear programming," in *Traces and Emergence of Nonlinear Programming*, pp. 247–258, Springer, 2014.

[16] Kaggle, "Ecommerce rating dataset," 2020, https://www.kaggle.com/nicapotato/womens-ecommerce-clothing-reviews.

[17] Kaggle, "Clothing fit and rating dataset," 2020, https://www.kaggle.com/rmisra/clothing-fit-dataset-for-size-recommendation.

[18] Kaggle, "Amazon rating dataset," 2020, https://www.kaggle.com/skillsmuggler/amazon-ratings.

[19] Kaggle, "Movie rating dataset," 2020, https://www.kaggle.com/ashukr/movie-rating-data.

[20] L. Sun, J. Zhao, X. Ye, S. Feng, T. Wang, and T. Bai, "Conditional analysis for key-value data with local differential privacy," 2019, http://arxiv.1907.05014.

[21] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[22] J. Acharya, Z. Sun, and H. Zhang, "Hadamard response: estimating distributions privately, efficiently, and with little communication," in *Proceedings of the The 22nd International Conference on Artificial Intelligence and Statistics. PMLR*, pp. 1120–1129, 2019.

[23] R. Bassily, K. Nissim, U. Stemmer, and A. Thakurta, "Practical locally private heavy hitters," 2017, http://arxiv.1707.04982.

[24] G. Fanti, V. Pihur, and U. Erlingsson, "Building a rappor with the unknown: privacy-preserving learning of associations and data dictionaries," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 3, pp. 41–61, 2016.

[25] X. Ren, C.-M. Yu, W. Yu, S. Yang, X. Yang, and J. A. McCann, "High-dimensional crowdsourced data distribution estimation with local privacy," in *Proceedings of the 2016 IEEE International Conference on Computer and Information Technology (CIT)*, pp. 226–233, IEEE, Nadi, Fiji, December 2016.

[26] X. Ren, C.-M. Yu, W. Yu et al., "Lopub: high-dimensional crowdsourced data publication with local differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2151–2166, 2018.

[27] N. Wang, X. Xiao, Y. Yang et al., "Privtrie: Effective frequent term discovery under local differential privacy," in *Proceedings of the 2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pp. 821–832, IEEE, Paris, France, April 2018.

[28] M. Hay, V. Rastogi, G. Miklau, and D. Suciu, "Boosting the accuracy of differentially private histograms through consistency," *Proceedings of the VLDB Endowment*, vol. 3, no. 1, 2010.

[29] J. Lee, Y. Wang, and D. Kifer, "Maximum likelihood post-processing for differential privacy under consistency constraints," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 635–644, 2015.

[30] J. Jia and N. Z. Gong, "Calibrate: frequency estimation and heavy hitter identification with local differential privacy via incorporating prior knowledge," in *Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 2008–2016, IEEE, Paris, France, May 2019.