PrivKV: Key-Value Data Collection with Local Differential Privacy

Qingqing Ye *, †, Haibo Hu †, Xiaofeng Meng *, Huadi Zheng †
*School of Information, Renmin University of China
{yqqq, xfmeng}@ruc.edu.cn
†Department of Electronic and Information Engineering, Hong Kong Polytechnic University
haibo.hu@polyu.edu.hk, huadi.zheng@connect.polyu.hk

Abstract—Local differential privacy (LDP), where each user perturbs her data locally before sending to an untrusted data collector, is a new and promising technique for privacy-preserving distributed data collection. The advantage of LDP is to enable the collector to obtain accurate statistical estimation on sensitive user data (e.g., location and app usage) without accessing them. However, existing work on LDP is limited to simple data types, such as categorical, numerical, and set-valued data. To the best of our knowledge, there is no existing LDP work on key-value data, which is an extremely popular NoSQL data model and the generalized form of set-valued and numerical data. In this paper, we study this problem of frequency and mean estimation on key-value data by first designing a baseline approach PrivKV within the same "perturbation-calibration" paradigm as existing LDP techniques. To address the poor estimation accuracy due to the clueless perturbation of users, we then propose two iterative solutions PrivKV M and PrivKV M+ that can gradually improve the estimation results through a series of iterations. An optimization strategy is also presented to reduce network latency and increase estimation accuracy by introducing virtual iterations in the collector side without user involvement. We verify the correctness and effectiveness of these solutions through theoretical analysis and extensive experimental results.

I. INTRODUCTION

With the prevalence of big data analytics, service providers become increasingly enthusiastic in collecting and analyzing usage data to improve their services. For instance, Amazon reportedly logs customers’ browsing history to predict product sales and manage inventory to avoid stockouts [2]. However, the collection of user data comes at the price of privacy risks, not only for users but also for service providers who are vulnerable to internal and external data breaches. As an answer to privacy-preserving data collection, local differential privacy (LDP) [11], [15], [17] has been proposed to perturb data at the user side before being collected. Due to its strong privacy guarantee inherited from differential privacy and decentralized nature without the need of a trusted party, it has been adopted in mainstream systems for usage data collection, including Apple [1], Google [21] and Microsoft [14].

Existing work on LDP focuses on collecting basic statistics from simple data types, such as frequency estimation over categorical [21] or set values [33], and mean estimation over numerical values [31]. To the best of our knowledge, no existing work has focused on key-value data model, which is an extremely popular NoSQL data model and a generalized form of set-valued and numerical data. Key-value data are pervasive in big data analytics, and the following two examples show its potential applications.

- Video ads performance analysis. Advertisers are keen to know whether their video ads are attractive to their potential customers. As such, they would like to collect ads viewership data from users in the form of key-value pairs where the key is the ads identifier and the value is the time a user has watched this video ad.
- Mobile app activity analysis. Smartphone manufacturers and third-party apps need to collect daily usage data of mobile apps for a variety of purposes such as optimizing battery and memory management algorithms and identifying daily active users (DAU). These usage data are in the form of key-value pairs where the key is the app identifier and the value is the time or frequency this app appears in the foreground.

In both cases, users are reluctant to provide these usage data, which could disclose their interests, daily activities and other personal particulars.

In this paper, we take the first step on key-value data collection mechanisms that can satisfy local differential privacy. More specifically, we aim at collecting two most fundamental statistics of key-value pairs — frequency of keys and mean of values. There is a naive solution that first divides all key-value pairs into a key set and a value set, and then applies existing LDP methods for categorical data (e.g., RAPPOR [21] or k-RR [24]) to each key, and existing LDP methods for numerical data (e.g., Harmony [31] or MeanEst [17]) to each value. However, this solution does not work as keys and values are correlated. For example, let key-value pairs \{⟨Cancer, 0.6⟩, ⟨HIV, 0.9⟩, ⟨Fever, 0.08⟩\} denote diseases and their diagnostic values. If the key Cancer is perturbed into Fever, then its value 0.6 should be perturbed accordingly in the value domain of Fever, otherwise the perturbed key-value pair would be meaningless.

To address this challenge, we propose PrivKV that partially retains the key-value correlation to improve the accuracy of statistics while still achieving local differential privacy. To further improve the accuracy, we propose PrivKV M and PrivKV M+ that execute PrivKV in multiple iterations. Since the input of each iteration comes from the result of the last iteration, the same privacy budget can be better
TABLE I
DIFFERENCE OF THE PROPOSED SOLUTIONS

<table>
<thead>
<tr>
<th>Solution</th>
<th>Features</th>
<th>Application Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrivKV</td>
<td>lowest comm. cost with one iteration</td>
<td>non-iterative (e.g., no downlink channel)</td>
</tr>
<tr>
<td>PrivKVM</td>
<td>high accuracy with multiple and fixed iterations</td>
<td>sufficient and fixed comm. bandwidth</td>
</tr>
<tr>
<td>PrivKVM+</td>
<td>balanced between accuracy and comm. bandwidth</td>
<td>flexible comm. bandwidth</td>
</tr>
</tbody>
</table>

utilized to perturb data according to LDP while approaching the true statistics. To further reduce the network latency, we also propose an optimization strategy for the data collector to execute virtual PrivKV iterations without user involvement. To summarize, our technical contributions are three-folded.

- We formulate the problem of privacy-preserving key-value data collection for frequency and mean estimation in local setting and design an efficient sanitized mechanism called local perturbation protocol (LPP) to perturb the key-value pair.
- We build three LPP-based solutions, namely PrivKV, PrivKVM and PrivKVM+, that satisfy $\epsilon$-LDP. We theoretically demonstrate the guarantee of convergence and unbiasedness with multiple iterations.
- We present an optimization strategy for the data collector to execute virtual PrivKV iterations, which substantially reduces the network latency and improves the accuracy.

Table I summarizes the difference of our solutions and their application scenarios. The remainder of this paper is organized as follows. Section II reviews the related literature. Section III formulates the problem with preliminary background on LDP. Section IV presents a baseline approach PrivKV. Section V proposes the iterative solutions PrivKVM and PrivKVM+ and elaborates on theoretical analysis on privacy and accuracy. Section VI introduces the virtual iteration strategy. Section VII presents an extensive set of experimental results. Finally, section VIII concludes this paper.

II. RELATED WORK

The notion of differential privacy was first introduced by Dwork in [18]. So far, most existing works focus on the centralized setting, i.e., they assume a trusted central data collector that possesses all the genuine values [20], [27]. As for the local setting, i.e., scenarios without such a collector, Duchi et al. [15] systematically investigate the framework of local differential privacy (LDP) and show an upper bound under LDP based on information theory.

Since then many LDP techniques are proposed for frequency estimation over categorical data. Erlingsson et al. [21] propose RAPPOR, which is the first LDP technique for frequency estimation in real-world applications. The key idea is to transform a sensitive string into a Bloom filter [8] and then apply the randomized response (RR) method [42] to perturb it. A follow-up method proposed by Fanti et al. [22] extends RAPPOR to more complex statistics without explicit dictionary knowledge. As RR only targets at binary variables, Kairouz et al. [24], [25] introduce an extremal mechanism $k$-RR to categorical attributes with arbitrary number of possible values. Kairouz et al. [23] present O-RR and O-RAPPOR by combining cohort-based hashing with $k$-RR and RAPPOR, respectively. In order to reduce the communication cost, Bassily et al. [5] design an efficient protocol SHist, which produces a succinct histogram representation of the input data. Based on a generalized form of RAPPOR and SHist, Wang et al. [39] present a framework of choosing optimal parameters for better estimation accuracy. Wang et al. [41] give theoretical analysis on RR and Laplace mechanism, and prove that the former outperforms the latter in terms of the mean square error. Avent et al. [4] propose a hybrid model of centralized and local differential privacy. Our work is on frequency and mean estimation for key-value data with LDP. It is related to these existing methods as we need to handle both categorical (key) and numerical (value) data in a correlated manner.

Other works use frequency estimation as a primitive to protect data privacy in other domains. For private spatial data aggregation, Chen et al. [10] propose a framework to learn user distribution which can satisfy users’ personalized privacy requirements and Kim et al. [26] use LDP to protect indoor positioning data. Some works focus on data analysis in aspects of mobile crowdsensing [36], weighted histogram [38] and frequent itemset mining [40]. For practical heavy hitters estimation, Bassily et al. [6] give solutions to keep time, space and communication complexity minimal, and Bun et al. [9] propose to strengthen lower bounds on error by incorporating the failure probability. Qin et al. propose LDPMiner [33] to obtain heavy hitters over set-valued data and LDPGen [34] to generate synthetic social graph, both of which split the LDP procedure into two phases. For private learning problem, Smith et al. [37] theoretically analyze the interactive and non-interactive LDP. Besides, some works aim to release multidimensional data [12], [35], [43]. For privacy-preserving system architecture, Prio [13] assumes at least one trusted server in a multi-server setup, whereas Prochlo [7] can support various levels of trust.

As a relevant problem to this paper, mean estimation over numerical data with LDP has also been studied in the literature. Duchi et al. propose MeanEst [16], [17], which incurs high computation and space complexity. To address this, an improved method Harmony is proposed by Nguyễn et al. [31], in which for any numerical input it only gives a binary output according to a certain probability. It has also been shown that Harmony can achieve higher accuracy. Further, Akter et al. [3] present a personalized version of Harmony to meet individual’s privacy requirements. Ding et al. [14] propose mechanisms to continuously collect telemetry data.

III. PRELIMINARIES AND PROBLEM DEFINITION

A. Local Differential Privacy

Centralized differential privacy [19], [30] assumes a trusted data collector that does not steal or leak data owners’ private information. However, in many real-world applications, this assumption does not hold, especially as data are now considered the core assets of businesses. To this end, local differential
privacy (LDP) [15] is proposed for the setting where no one else can get access to the original data except the data owners themselves. In LDP, each data owner locally perturbs her data using a randomized mechanism, and then sends the sanitized version to the untrusted data collector.

Formally, let $D$ denote the whole database. $M$ is a randomized algorithm that takes a data tuple $t$ as input and outputs $t^*$. $\epsilon$-local differential privacy (or $\epsilon$-LDP) is defined on $M$ and a privacy budget $\epsilon > 0$ as follows.

**Definition 1 ($\epsilon$-local differential privacy).** A randomized algorithm $M$ satisfies $\epsilon$-local differential privacy, if and only if for any two input tuples $t, t' \in D$ and for any output $t^*$, the following inequality always holds.

$$\Pr[M(t) = t^*] \leq e^\epsilon \times \Pr[M(t') = t^*].$$

Intuitively $\epsilon$-LDP means that by observing the output $t^*$, the data collector cannot infer whether the input tuple is $t$ or $t'$ with high confidence (controlled by $\epsilon$), which is different from the centralized differential privacy defined on two neighboring datasets that only differ in one record.

As with centralized differential privacy, LDP also has the nice property of sequential composition [29], which guarantees the overall LDP for a sequence of algorithms, each of which satisfies LDP.

**Theorem 3.1:** (Sequential Composition). Given $\epsilon$ randomized algorithms $M_i (1 \leq i \leq c)$, each providing $\epsilon_i$-local differential privacy. Then the sequence of algorithms $M_i (1 \leq i \leq c)$ collectively provides $(\sum \epsilon_i)$-local differential privacy.

According to sequential composition, given a privacy budget $\epsilon$, we can partition it into multiple portions and each portion of budget can be used by one randomized algorithm to collect useful information from the original data. This lays the foundation of our proposed iterative solution, which will be discussed in Section V.

### B. Randomized Response

Randomized response (RR) [42] is a technique developed for the interviewees in a survey to give random answer to a sensitive boolean question so that they can achieve plausible deniability. Specifically, each interviewee gives the genuine answer with probability $p$ and gives the opposite answer with probability $1 - p$.

RR has been the predominant perturbation mechanism for LDP. To adapt RR to satisfy $\epsilon$-LDP, we set $p$ as follows:

$$p = \frac{e^\epsilon}{1 + e^\epsilon}$$

Note that the percentage of “true” (denoted as $f$) directly obtained from all perturbed answers is biased. To correct this, the data collector needs to calibrate it and reports $f'$:

$$f' = \frac{p - 1 + f}{2p - 1}$$

Many state-of-the-art LDP solutions, such as RAPPOR [21] and SHist [5], use RR as the building block and convert input data to a binary form.

### C. Problem Definition

This paper studies the problem of distributed data aggregation over key-value data in the context of LDP. Without loss of generality, let the universe consist of a set of users $U = \{u_1, u_2, ..., u_n\}$, and a set of keys $K = \{1, 2, ..., d\}$ whose value domain $V$ is the continuous domain $[-1, 1]$. The $i$-th user $u_i$ possesses $l_i$ key-value (KV) pairs $S_i = \{(k_j, v_j) | 1 \leq j \leq l_i, k_j \in K, v_j \in V\}$. The main notations are listed in Table II.

An untrusted data collector needs to estimate some statistics of these key-value data from all users. In this paper, we focus on two fundamental estimations: frequency and mean estimation.

- **Frequency estimation.** The frequency of key $k$, $f_k$, is defined as the portion of users who possess a KV pair whose key is $k$. Formally for any key $k$,

$$f_k = \frac{|\{u_i | (k, v_j) \in S_i\}|}{n} \tag{1}$$

- **Mean estimation.** The mean of key $k$, $m_k$, is defined as the mean of all values in KV pairs whose key is $k$, or formally:

$$m_k = \frac{\sum_{i} \sum_{j,k_j=k} v_j}{n \cdot f_k} \tag{2}$$

### IV. PRIVKV: A BASELINE APPROACH

In this section, we present our baseline approach PrivKV for distributed key-value data aggregation with LDP. PrivKV protects key-value data by applying perturbation to keys and values while almost retaining the true frequencies and means. In the following, we will start with a naive protocol that perturbs the keys only. We then show its security flaw and remedy it by a synchronous key and value perturbation protocol LPP that leads to a working PrivKV solution.

#### A. A Flawed Key Perturbation Protocol

As both frequency and mean are associated with a key, a naive approach is to perturb the keys using the classic RR. As such, the first step is to convert the user’s KV pair set $S_i$ to its canonical form $S_i'$, which contains the full set of keys.

Fig. 1 shows this conversion. For $(k, v) \in S_i$, we convert it to $(1, v)$ in $S_i'$; for a KV pair that does not exist, $(0, 0)$, an empty KV pair, is added to $S_i'$. All KV pairs are sorted in ascending order of keys. The canonical form guarantees each
user possesses the same number of KV pairs so that any key can be represented by its index in the key universe. It also addresses the issue of information loss in [33] where each user can only have a fixed number of KV pairs, and all others must be truncated.

Now that all keys are binary after the conversion, we can directly apply RR to perturb them. Specifically, we change the key “1” (resp. “0”) to “0” (resp. “1”) with a certain probability so that the data collector cannot determine whether a user has that KV pair or not. The change on values is based on the perturbation results of keys, which have the following four cases:

- 1 → 1: A KV pair exists before and after the perturbation. In this case, the value should be retained, i.e., (1, v) → (1, v).
- 1 → 0: A KV pair disappears after the perturbation. In this case, we simply set the value to be zero to hide the trace of key perturbation, i.e., (1, v) → (0, 0). Note that this change does not affect the mean estimation of that key.
- 0 → 0: A KV pair does not exist before or after the perturbation. In this case, the whole KV pair is unchanged, i.e., (0, 0) → (0, 0).
- 0 → 1: A new KV pair appears after the perturbation. In this case, we need to assign a value to it. Since the user has no apriori knowledge about the distribution of true values, the value is randomly drawn from the domain of [−1, 1].

The flaw of this naive protocol lies in the fourth case where a random value is assigned. First, as the data collector receives more true values, she is able to distinguish a true value (case 1) from an assigned one (case 4) with high confidence, particularly when the distribution of true values significantly deviates from the uniform distribution of [−1, 1]. Second, a uniform distribution of [−1, 1] leads to a mean of 0, which affects the mean estimation of that key.

### B. Local Perturbation Protocol: A Remedy

A remedy to the naive protocol is to avoid using an assigned value directly — any value is perturbed no matter whether it is a real value or an assigned one.

Numerical value perturbation with LDP for mean estimation has been studied by Nguyễn et al. [31]. They propose Harmony, whose key idea is to discretize a numerical value to a binary one and then perturb it using RR to satisfy ε-LDP. Algorithm 1 shows the major steps: discretization, perturbation and calibration. After discretization (Line 3), the value can only be 1 or −1, which is suitable to apply RR to perturb it (Line 4). As the perturbation causes the mean estimation to be biased, the perturbed value must be calibrated before being sent to the data collector (Line 5).

We adapt Harmony to serve as our value perturbation primitive, as shown in Algorithm 2. The main changes are three-folded. First, we shift the calibration step to the data collector to alleviate the computational cost of a user. In addition, each perturbed value in Harmony is simply scaled d times to counterbalance the effect of sampling, which may also cause bias. To address this, in our setting, the collector precisely counts the sample keys and sums their values to derive the mean. Third, we further add a conditional correction after calibration to remove outliers caused by Harmony. Specifically, if N users participate in a value perturbation (i.e., they all have the same key), after calibration both counts of 1 and −1 should still be in the range of [0, N]. As such, we correct any count less than 0 to 0, and any count greater than N to N. This correction, though possibly making the mean estimation biased, can improve its accuracy, especially for a small privacy budget.

By combining the above key perturbation protocol and the value perturbation primitive, we have the Local Perturbation Protocol (LPP) for a user. The pseudo-code is described in Algorithm 3. It takes as inputs the set of KV pairs of a user Si, the set of all different keys K, and the privacy budgets ε1 and ε2 for key and value perturbation, and returns a perturbed KV pair with its index. In order to reduce the communication cost this change does not affect the mean estimation of that key.

### Algorithm 1 Decomposition of Harmony

**Input:** User u’s set of values \( V \in [-1, 1]^d \)

**Output:** Perturbed set of values \( V^* \)

**Procedure:**
1. Let \( V^* = (0,0, \ldots, 0) \)
2. Sample \( j \) uniformly at random from \([d]\), let \( v = V_j \)
3. Discretization:
\[
\begin{align*}
v^* &= \begin{cases} 
1 & \text{w.p. } \frac{1+w}{2} \\
-1 & \text{w.p. } \frac{1-w}{2} 
\end{cases}
\end{align*}
\]
4. Perturbation:
\[
\begin{align*}
v^* &= \begin{cases} 
v^* & \text{w.p. } \frac{v^*}{1+\epsilon} \\
-v^* & \text{w.p. } \frac{v^*}{1-\epsilon}
\end{cases}
\end{align*}
\]
5. Calibration:
\[
v^* = v^* \cdot \frac{e^\epsilon + 1}{e^\epsilon - 1} \cdot d
\]
6. \( V^* = v^* \)
7. return \( V^* \)

**Algorithm 2 Value Perturbation Primitive**

**Input:** Value \( v \) of a KV pair

**Output:** \( VPP(v, \epsilon) \) is the perturbed value \( v^* \)

**Procedure:**
1. Discretization:
\[
\begin{align*}
v^* &= \begin{cases} 
1 & \text{w.p. } \frac{1+w}{2} \\
-1 & \text{w.p. } \frac{1-w}{2} 
\end{cases}
\end{align*}
\]
2. Perturbation:
\[
\begin{align*}
v^* &= \begin{cases} 
v^* & \text{w.p. } \frac{v^*}{1+\epsilon} \\
-v^* & \text{w.p. } \frac{v^*}{1-\epsilon}
\end{cases}
\end{align*}
\]
3. return \( v^* \)
**Algorithm 3** Local Perturbation Protocol (LPP)

**Input:**
- User $u_i$’s set of KV pairs $S_i$
- The set of keys $K$
- Privacy budgets $\epsilon_1$ and $\epsilon_2$

**Output:**
- $LPP(S_i, K, \epsilon_1, \epsilon_2)$ is the perturbed KV pair \((k_j, v^*)\) of the $j$-th key

**Procedure:**
1: $d = |K|$
2: Sample $j$ uniformly at random from $[d]$
3: if $k_j$ exists in the key set of $S_i$ then
4: \[ v^* = VPP(v_j, e_2) \]
5: Perturbs \((k_j, v^*)\) as:
\[
\begin{align*}
(1, v^*) & \quad \text{w.p. } \frac{e^{\epsilon_1}}{1+e^{\epsilon_1}} \\
(0, 0) & \quad \text{w.p. } \frac{1}{1+e^{\epsilon_1}}
\end{align*}
\]
6: else
7: Randomly draw a value $\tilde{m} \in [-1,1]$
8: $v^* = VPP(\tilde{m}, e_2)$
9: Perturbs \((k_j, v^*)\) as:
\[
\begin{align*}
(1, v^*) & \quad \text{w.p. } \frac{e^{\epsilon_1}}{1+e^{\epsilon_1}} \\
(0, 0) & \quad \text{w.p. } \frac{1}{1+e^{\epsilon_1}}
\end{align*}
\]
10: return $j$ and \((k_j, v^*)\)

from $O(d)$ to $O(1)$, we adopt the same sampling approach as in [5] to select only the $j$-th KV pair \((k_j, v_j)\) for reporting (Line 2).\(^1\) If the user has this KV pair, she first perturbs $v_j$ into $v^*$ using the value perturbation primitive (Line 4), then converts the pair \((k_j, v^*)\) into canonical form \((1, v^*)\), and finally perturbs it to \((1, v^*)\) (resp. \((0, 0)\)) with probability $\frac{e^{\epsilon_1}}{1+e^{\epsilon_1}}$ (resp. $\frac{1}{1+e^{\epsilon_1}}$) (Line 5). The perturbation is similar if the user does not have this KV pair, except that the new value $\tilde{m}$ is first drawn from the domain of $[-1,1]$ before being perturbed (Lines 7-9). For LPP, the change of values is based on the perturbation results of keys. As such, the correlation between keys and values can be well retained.

The following theorem shows LPP satisfies $(\epsilon_1 + \epsilon_2)$-LDP.

**Theorem 4.1:** With privacy budget $\epsilon = \epsilon_1 + \epsilon_2$, the local perturbation protocol satisfies $\epsilon$-LDP.

**Proof:** See Appendix A. \(\square\)

**C. PrivKV: Putting Things Together**

Algorithm 4 describes the complete PrivKV solution for frequency and mean estimation, which consists of user-side perturbation and collector-side calibration.

Each user perturbs her set of KV pairs through LPP, and then sends a sanitized version of the sample KV pair to the data collector (Line 2). Upon receiving all users’ perturbed KV pairs, the collector calculates and calibrates the frequency of each key (Lines 5-6). Mean estimation is slightly more complicated — the collector counts the number of 1’s and −1’s in the set of values for each key and then calibrates them (Lines 7-9). As aforementioned, the most significant difference from Harmony is the extra correction step in Line 10. We treat any count greater than $N$ or less than 0 as outliers and correct them to $N$ and 0, respectively. An enhanced correction scheme based on apriori knowledge will be presented in the next section. By correcting outliers, PrivKV improves the estimation accuracy, especially for a small privacy budget.

**V. PrivKVM: AN ITERATIVE SOLUTION**

PrivKV guarantees LDP at the cost of poor assignment of a new value, which is randomly drawn from $[-1,1]$. As such, it may suffer from low accuracy and instability. In this section, we propose to iterate PrivKV multiple times to assign new value in almost the same distribution as the real values. This leads to two advanced protocols, namely PrivKVM and PrivKVM*.

**A. An Iterative Model**

Fig. 2 illustrates an iterative PrivKV execution model where the discretized estimated mean\(^2\) of the previous iteration $v^*$ becomes the assigned value of this iteration so that the mean estimation can gradually approach the ground truth (as the loop \(2\circ3\circ5\circ6\circ\) shows). Theorem 5.1 below proves that by using this discretized estimated mean instead of a randomly assigned value from $[-1,1]$, the mean estimation becomes unbiased.

\(^1\)Technically we can sample more than one KV pair. However, besides the increasing complexity, the overall estimation accuracy can drop with more KV pairs as each KV pair costs certain privacy budget for perturbation. As such, we use one sample as in [5] throughout this paper.

\(^2\)To protect $\tilde{m}$ from being disclosed to users in step \(5\), the data collector does not send it back directly to them. Instead, in the $r$-th iteration each user just receives a fresh and independently sampled value $v^*$, which is essentially a discretized copy of $\tilde{m}$. It is either 1 or −1 with probabilities $\frac{2\epsilon_1 m}{N}$ and $\frac{1-2\epsilon_1 m}{N}$, respectively. Since these two probabilities are only known to the data collector, no user can infer $\tilde{m}$ unless a large number of users collude and share their received 1’s and −1’s. In such extreme cases, protecting $\tilde{m}$ is no longer necessary as these users can derive it by themselves.
Theorem 5.1: Let \( m_k \) be the true mean of all values in KV pairs with key \( k \), and \( \tilde{m}_k^* \) be the estimated mean by PrivKV (Algorithm 4) when assigning \( v^* = \text{discretization}(\tilde{m}_k) \) to new values, where \( \tilde{m}_k \) is an approximation of \( m_k \). Then, \( \mathbb{E}[\tilde{m}_k^*] = m_k \).

**Proof.** With key perturbation, values are changed accordingly based on the four cases in section IV-A. For key \( k \), let \( \tilde{m}_k \) be the mean of all values after key perturbation, in which we suppose there are \( n_1 \) instances of \((1, v) \rightarrow (1, v^*)\) and \( n_2 \) instances of \((0, 0) \rightarrow (1, v^*)\). Note that \( \sum_{i=1}^{n_1} \sum_{j:k_i=k} v_j \geq \tilde{m}_k \). Thus we have:

\[
\mathbb{E}[\tilde{m}_k] = \mathbb{E}\left[ \frac{n_1 m_k + n_2 \left( 1 + \frac{m_k}{2} - \frac{1 - m_k}{2} \right) - \sum_{i=1}^{n_1} \sum_{j:k_i=k} v_j}{n_1 + n_2} \right]
\]

In VPP, for any value \( v \), it will change to \( \tilde{v} \) accordingly after key perturbation, and let \( \tilde{v} \) be the value after discretization, and \( \tilde{m}_k \) be the mean of all \( \tilde{v} \). Thus we have:

\[
\mathbb{E}[\tilde{v}_j] = 1 \cdot \frac{1 + \tilde{v}_j}{2} + (-1) \cdot \frac{1 - \tilde{v}_j}{2} = \tilde{v}_j
\]

\[
\mathbb{E}[\tilde{m}_k] = \mathbb{E}\left[ \frac{1}{n_1 + n_2} \sum_{i} \sum_{j:k_i=k} \tilde{v}_j \right]
\]

Let \( \tilde{n}_1 \) and \( \tilde{n}_2 \) be the counts of 1 and -1 in the value set after discretization, and \( n_1' \) and \( n_2' \) be the counts after perturbation. According to RR, we have:

\[
\mathbb{E}[n_1'] = \tilde{n}_1 \cdot p + \tilde{n}_2 \cdot (1 - p)
\]

\[
\mathbb{E}[n_2'] = \tilde{n}_2 \cdot p + \tilde{n}_1 \cdot (1 - p)
\]

where \( p = \frac{e^\epsilon}{e^\epsilon + 1} \) with privacy budget \( \epsilon \).

After value perturbation, the counts \( n_1' \) and \( n_2' \) will be calibrated to \( n_1'' \) and \( n_2'' \), respectively.

\[
n_1'' = \frac{(p - 1)(n_1' + n_2') + n_1'}{2p - 1}
\]

\[
n_2'' = \frac{(p - 1)(n_1' + n_2') + n_2'}{2p - 1}
\]

Therefore, the expected mean is:

\[
\mathbb{E}[m_k'^*] = \mathbb{E}\left[ \frac{n_1'' - n_2''}{n_1'' + n_2''} \right] = \frac{n_1' - n_2'}{(2p - 1)(n_1 + n_2)}
\]

In this iterative model, an enhanced scheme for outliers correction based on apriori knowledge can also be devised (as the loop \( \circlearrowleft \circlearrowright \) in Fig. 2 shows). Specifically, data collector records the estimated means of previous iterations. In subsequent iterations, all emerging outliers will be replaced by these recorded means. This correction scheme also helps to improve the accuracy of the iterative model.

B. PrivKV

The iterative model immediately leads to PrivKV, whose details are shown in Algorithm 5. It takes as inputs all users’ sets of KV pairs \( S \), the set of keys \( K \), privacy budget \( \epsilon \) and the number of iterations \( c \). Privacy budget \( \epsilon \) is divided into \( \{\epsilon_{11}, ..., \epsilon_{1c}, \epsilon_{21}, ..., \epsilon_{2c}\} \) for \( c \) iterations by a privacy budget allocation (PBA) strategy (Line 1). In the first iteration, it invokes PrivKV to calculate the frequency \( f^{(1)} \) and mean \( \bar{m}^{(1)} \). Then the data collector uses \( \bar{m}^{(1)} \) to randomly discretize \( v^* \) (as shown in the discretization step of Algorithm 2) for each user to use in the second iteration (Line 3). In each subsequent r-th iteration, it invokes PrivKV', a modified PrivKV algorithm that takes in \( \epsilon_{1r}, \epsilon_{2r} \), and one more parameter \( v^* \). \( v^* \) is needed in the local perturbation protocol and for a non-existent key its value must be randomly discretized with parameter \( \bar{m} \) (see Line 8 of Algorithm 3). In PrivKV, this discretization is performed by the data collector based on the returned values from users in

3Here we use superscripts to denote which iterations the estimated results are calculated from.
The previous iteration (Lines 5-6). Since frequency estimation does not rely on the results of last iteration, it is carried out only in the first iteration.

The PBA strategy for PrivKVM is designed as follows. We first divide the total privacy budget \( c \) equally for perturbing keys and values, i.e., \( \epsilon_1 = \epsilon_2 = \epsilon/2 \). For frequency estimation, since only one iteration is needed, we allocate the privacy budget of \( \epsilon_1 \) only to the first iteration. That is, \( \epsilon_{11} = \epsilon_1 \) and \( \epsilon_{12} = \cdots = \epsilon_{1c} = 0 \). For mean estimation, we allocate the privacy budget of \( \epsilon_2 \) equally to each iteration. That is, \( \epsilon_{21} = \epsilon_{22} = \cdots = \epsilon_{2c} = \epsilon/2 \).

The extension of PrivKVM to multidimensional data is straightforward. If the key is a multidimensional point, we can simply flatten the key universe to one-dimensional. If the value is a multidimensional point, we can treat each dimension as an independent value dimension and perform value perturbation separately. Note that by doing this we will not lose the correlation between different dimensions as PrivKVM well retains the correlation between the key and each value dimension.

Further, PrivKVM can be applied to some other statistics of values, such as the median and percentile. We take the median estimation as an example, and the percentile statistics can be extended in a similar way. According to the property of median, it separates a higher half of a probability distribution. According to the property of median estimation as an example, and the percentile statistics can be extended in a similar way. According to the property of median, it separates a higher half of a probability distribution.

In the first iteration, suppose the initialized approximated mean is \( \tilde{m}_k \), thus we have:

\[
\mathbb{E}[m_k^{(1)}] = \frac{(1 - f_k)(1 - p_k)n \cdot \tilde{m}_k + f_k p_k n \cdot m_k}{f_k n} = f_k p_k - f_k - p_k + 1.
\]

Then, the mean returned from the \( t \)-th iteration is based on that from the \((t - 1)\)-th iteration.

\[
\mathbb{E}[m_k^{(r)}] = \frac{(f_k p_k - f_k - p_k + 1) \cdot \mathbb{E}[m_k^{(r-1)}] + f_k p_k \cdot m_k}{2 f_k p_k - f_k - p_k + 1}.
\]

Thus we have:

\[
|m_k - \mathbb{E}[m_k^{(c)}]| = \frac{f_k p_k - f_k - p_k + 1}{2 f_k p_k - f_k - p_k + 1} |m_k - \mathbb{E}[m_k^{(c-1)}]| = |m_k - \mathbb{E}[m_k^{(c-2)}]| = \cdots = |m_k - \tilde{m}_k|
\]

Note that \( |m_k - \tilde{m}_k| \) is a constant and we have \( p_r \geq 0.5 \) with \( \epsilon_r \geq 0 \). If \( f_k = 0 \), then \( m_k^{(1)} = 0 \), which results in \( m_k^{(c)} = 0 \). For \( f_k \in (0, 1) \), it is obvious that \( \frac{f_k p_k - f_k - p_k + 1}{2 f_k p_k - f_k - p_k + 1} \in [0, 1] \). Hence we have:

\[
\lim_{c \to \infty} |m_k - \mathbb{E}[m_k^{(c)}]| = 0.
\]
Theorem 5.3 provides the guarantee of convergence, regardless of the initialized mean in the first iteration. Further, the following theorem proves the worst case accuracy guarantee of PrivKVM, in terms of the maximum difference between the true mean and any estimated mean, is bounded. The proof is similar to the one in [31].

Theorem 5.4: For any key $k \in K$ and $|K| = d$, let $m_k$ be the true mean of all $n$ values in KV pairs whose key is $k$, and $m^*_k$ be the estimated mean by PrivKVM. With at least $1 - \beta$ probability,

$$\max_{k \in K} |m_k - m^*_k| = O \left( \frac{\sqrt{d \cdot \log(d/\beta)}}{\epsilon \sqrt{n}} \right)$$

D. PrivKVM+: An Adaptive Variant

PrivKVM needs the number of iterations $c$ to be determined in advance. However, in many real-world applications, the optimal $c$ is difficult to obtain. Ideally, to achieve the highest estimation accuracy, $c$ should approach infinity to make the best use of estimated means in previous iterations. Unfortunately, this does not take account of other costs in practice, such as the communication overhead and the execution time. In this section, we present an adaptive PrivKVM protocol, namely PrivKVM+, that determines $c$ adaptively by considering these costs. Specifically, for the $r$-th iteration, we define a cost function $F(r)$ which consists of the accuracy cost $F_1(r)$ and the communication cost $F_2(r)^4$:

$$F(r) = F_1(r) + F_2(r)$$

For $F_1(r)$, we model it using the average of the absolute error of mean among all keys. That is,

$$F_1(r) = \frac{1}{d} \sum_{k \in K} |m_k - m^{(r)}_k|$$

For $F_2(r)$, since the communication cost of each iteration is a constant $A_0$, we have:

$$F_2(r) = A_0 \cdot r$$

Intuitively, as $r$ increases, $F_1(r)$ decreases less and less significantly according to Theorem 5.3, while $F_2(r)$ increases at a constant rate $A_0$. Therefore, $F(r)$ reaches a global minimum when the decrement of $F_1(r)$ can no longer compensate the increase of $A_0$. Formally, $r$ is the first iteration that satisfies

$$F(r) - F(r - 1) = A_0 - \frac{1}{d} \sum_{k \in K} |m_k^{(r)} - m_k^{(r-1)}| \geq 0$$

Here we assume $m_k^{(r)}$ is always closer to $m_k$ than $m_k^{(r-1)}$, i.e., $|m_k - m_k^{(r)}| < |m_k - m_k^{(r-1)}|$. In essence, Eq. 6 provides a termination condition for PrivKVM+ that is easy to implement — only the estimated means of this and last iterations are involved on the right hand side of the equation. If it becomes larger than or equal to 0, PrivKVM+ will be terminated.

Algorithm 6 shows the detailed pseudo-code of PrivKVM+, which does not need the number of iterations $c$ as an input as opposed to PrivKVM. It first allocates the privacy budget using a different strategy PBAt from the one used in PrivKVM where $c$ is known in advance (Line 1). PBAt uses an “exponential decay” strategy that dynamically allocates $\frac{1}{t}$ portion of the remaining privacy budget to the current iteration. In general, $t$ could be any value larger than 1. The smaller $t$ is, the larger privacy budget is allocated to the current iteration and thus the estimation converges faster. Nonetheless, if $t$ is too close to 1, the convergence might be premature as most privacy budget is wasted in the early iterations where the estimated mean is too inaccurate for the next iterations. The termination condition, $F^* = F(r) - F(r - 1) \geq 0$ is calculated in Line 4 (for the first iteration) and Line 10 (for subsequent iterations), and tested in Line 5 before a new iteration.

VI. VIRTUAL ITERATIONS: AN OPTIMIZATION ON LATENCY AND ACCURACY

In this section, we propose an optimization strategy that executes “real” PrivKVM or PrivKVM+ on selective iterations and other iterations are only executed “virtually” by the collector. By virtual iterations, the collector can directly predict the estimated mean without user involvement. It has two advantages. First, it effectively reduces the network transmission overhead between the user and the collector, and therefore improves the latency. Second, since virtual iterations do not cost any privacy budget, real iterations can be allocated more of it, and therefore improves the estimation accuracy. Algorithm 7 shows how $c - 1$ iterations can be virtually executed by the collector and returns a predicated estimated mean vector $\hat{m}^{(c)}$ after the first iteration is really executed with estimated frequency vector $f$ and mean vector $m^{(1)}$ (Line 2). The prediction takes two steps. For each key $k$, the collector calculates an intermediate variable $\theta$ based on frequency $f$ and privacy budget $\epsilon$ that is allocated to the first iteration (Line 4).

---

### Algorithm 6 PrivKVM+: Adaptive PrivKVM

<table>
<thead>
<tr>
<th>Input:</th>
<th>All users' sets of KV pairs $S = {S_1, ..., S_n}$ The set of keys $K$ Privacy budget $\epsilon$ Communication cost of one iteration $A_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>Frequency vector $f^<em>$ Mean vector $m^</em>$</td>
</tr>
<tr>
<td>Procedure:</td>
<td>1: Allocate privacy budget: ${\epsilon_1, \epsilon_2} \leftarrow PBAt(\epsilon)$ 2: Initialize $m = 1$, $v^* = discretization(m)$ 3: $f^<em>, m^</em> = PrivKVM'(S, K, \epsilon_1, \epsilon_2, v^<em>)$ 4: Calculate the bias: $F^</em> = A_0 - \frac{1}{d} \sum_{k \in K}</td>
</tr>
</tbody>
</table>

---

4 As the execution time is dominated by the communication bandwidth, we merge the cost of execution time into communication cost for ease of presentation.
Algorithm 7 Executing Virtual Iterations

**Input:** All users’ sets of KV pairs S = {S_1, ..., S_n}
The set of keys K
Privacy budget \( \epsilon \)
Number of iterations to execute \( c \)

**Output:** Frequency vector \( f \)
Mean vector \( m^{(c)} \)

**Procedure:**
1: Initialize \( \tilde{m} = 1, \nu^* = \text{discretization}(\tilde{m}) \)
2: Calculate the frequency and mean:
   \[
   f, m^{(1)} = \text{PrivKV}'(S, K, \epsilon/2, \nu^*)
   \]
3: for each key \( k \in K \) do
   4: Collector calculates \( \theta = \frac{f_k p_r - f_k - p_r + 1}{2 f_k p_r - f_k - p_r + 1} \), where \( p = \frac{\epsilon^r/2}{1 + \epsilon^{r+2}} \)
   5: Collector predicts \( m_k^{(c)} = \tilde{m}_k + \frac{(m_k^{(c)} - \tilde{m}_k)(1-\theta)}{1-\theta} \), where \( \theta = \frac{f_k p_r - f_k - p_r + 1}{2 f_k p_r - f_k - p_r + 1} \)
   6: return \( f \) and \( m^{(c)} \)

Then the collector can predict the estimated mean for the \( c \)-th iteration \( m_k^{(c)} \) (Line 5).

The following theorem proves the correctness of Algorithm 7.

**Theorem 6.1:** For any key \( k \), let \( \tilde{m}_k \) be the initial mean and \( m_k^{(1)} \) be the estimated mean of the first iteration with privacy budget \( \epsilon \). The expectation of the estimated mean of the \( c \)-th iteration \( E[m_k^{(c)}] \) is:

\[
E[m_k^{(c)}] = \tilde{m}_k + \frac{(E[m_k^{(c)}] - \tilde{m}_k)(1-\theta)}{1-\theta}
\]

where \( \theta = \frac{f_k p_r - f_k - p_r + 1}{2 f_k p_r - f_k - p_r + 1} \) and \( p = \frac{\epsilon^c/2}{1 + \epsilon^{c+2}} \).

**Proof:** We first show the following lemma on the ratio of bias in two consecutive iterations \( r + 1 \).

**Lemma 6.2:** For a key \( k \), let \( E[m_k^{(r-1)}] \), \( E[m_k^{(r)}] \) and \( E[m_k^{(r+1)}] \) be the expected estimation of the first means of three consecutive iterations. The privacy budget \( \epsilon_r \) and \( \epsilon_{r+1} \) are allocated to \( r \)-th and \((r + 1)\)-th iterations, respectively. Then the ratio of bias in these two iterations

\[
\frac{|E[m_k^{(r+1)}] - E[m_k^{(r)}]|}{|E[m_k^{(r-1)}] - E[m_k^{(r)}]|} = \left( \frac{f_k p_r - f_k - p_r + 1}{2 f_k p_r - f_k - p_r + 1} \right)^{r+1}
\]

where \( p_r = \frac{\epsilon^r/2}{1 + \epsilon^{r+2}} \), \( p_{r+1} = \frac{\epsilon^{r+1}/2}{1 + \epsilon^{r+3}} \) and \( f_k \) is the real frequency of key \( k \).

**Proof:** According to the proof of Theorem 5.3, we have:

\[
E[m_k^{(r)}] = \left( f_k p_r - f_k - p_r + 1 \right) \cdot m_k^{(r-1)} + f_k p_r \cdot m_k
\]

Thus we express the bias as follows:

\[
|E[m_k^{(r)}] - E[m_k^{(r-1)}]| = \left| \left( f_k p_r - f_k - p_r + 1 \right) \cdot m_k^{(r-1)} + f_k p_r \cdot m_k - m_k^{(r-1)} \right|
\]

Therefore, the ratio of bias can be derived as:

\[
\frac{|E[m_k^{(r+1)}] - E[m_k^{(r)}]|}{|E[m_k^{(r-1)}] - E[m_k^{(r)}]|} = \left( \frac{f_k p_r - f_k - p_r + 1}{2 f_k p_r - f_k - p_r + 1} \right)^{r+1}
\]

This completes the proof.

Note that the accuracy gain by virtual iterations heavily depends on that of the first iteration. However, if the latter is given only a small privacy budget, e.g., \( \epsilon = 0.05 \), the overwhelmed noise introduced in the first iteration could further accumulate through the virtual iterations. As will be shown in the performance evaluation, this optimization strategy can work well unless for small privacy budgets.

**VII. Experimental Evaluation**

In this section, we evaluate the performance of our proposed solutions. To have a fair comparison with existing LDP techniques that can only handle either categorical (key) or numerical (value) data, we adapt them to the context of key-value data as much as we can. For frequency estimation on keys, we apply the same sampling technique used in LPP to RAPPOR [21], k-RR [24] and SHist [5], i.e., each user randomly selects one key in the key universe, and perturbs it by either RAPPOR, k-RR or SHist if this key exists in her key set, or skips it otherwise. For mean estimation, we replace value perturbation primitive (Algorithm 2) in PrivKVM with Harmony [31] and MeanEst [17], which lead to PrivKVM-Harmony and PrivKVM-MeanEst. They are then compared with PrivKVM.
We implement all methods in Java and conduct experiments on a desktop computer with Intel Core i5-3470 3.20 GHz CPU, 32G RAM running Windows 7 operating system. The testing key-value datasets consist of both synthetic and real ones, and their value domains are all normalized to $[−1, 1]$.

**Parameter settings.** For RAPPOR [21], we set its parameters as system default setting, i.e., set the bloom filter size $h$ to 64, the number of hash functions to 2 and the number of cohorts to 16. The values of three probabilities $f$, $p$ and $q$ totally depend on the given privacy budget. For $SHist$ [5], as larger error bound requires more computation resources, we set it to 0.8, which is the largest value we can support within our computing power.

**Datasets.** We conduct the experiments over six datasets and their parameters are summarized in Table III, including the average and the variance of frequencies and means. The first three are synthetic datasets, whose keys and values follow a gaussian, a power-law and a linear distribution, respectively. The other three are real key-value datasets, obtained from public data sources.

- **AppData app usage data.** This dataset is collected by a leading mobile data service provider 6. It contains app usage data of 2,006,631 devices and 1,134 apps. Each device is associated with a list of installed apps and the total use time of each app in the quarter of Oct. 2017 to Dec. 2017. We treat each app as a key and its use time as the value.
- **TalkingData mobile events**. This dataset is collected from TalkingData SDK which is integrated with many mobile apps. It contains 60,822 devices and 306 categories of apps. There are 32,473,067 events in this dataset, each presenting the category of apps from a certain device gets access to TalkingData SDK. Suppose instead of collecting individual events which is privacy-intrusive, TalkingData collects the frequencies of different events from each device by LDP. In this regard, each category is a key, and its value denotes the number of events of this category.
- **JData shopping records**. This dataset is collected from JD.COM, which contains 50,601,736 sales records in 2016. It records the list of product brands each user has ever purchased. There are 105,180 users and 442 brands in total. We treat each brand as a key, and its value denotes the number of times this user has purchased any product of that brand.

To measure the accuracy of the estimated frequency and mean, we use the following two metrics that are widely used in the literature.

- **Relative Error (RE)** [28]. It measures the relative error of estimated frequencies with respect to real frequencies of all keys. Specifically, for key $k \in K$, let $f^*_k$ and $f_k$ denote real and estimated frequency, respectively.

$$RE = Stat_{k \in K} \frac{|f^*_k - f_k|}{f^*_k},$$

where $Stat$ is a statistical function such as median (default), top 10%, 20%, 30% and 40% percentile, and top 100-th.

- **Mean Square Error (MSE)** [31]. It measures the absolute error of estimated means with respect to real means of all keys. Specifically, for $k \in K$, let $m^*_k$ and $m_k$ denote real and estimated means of all values whose key is $k$, and $d = |K|$.

$$MSE = \frac{1}{d} \sum_{k \in K} (m_k - m^*_k)^2$$

As the absolute MSE value is small, we use logarithm of MSE, i.e., $Log(MSE)$, in the sequel.

---

6The name is anonymous as we have a non-disclosure agreement with this company.

7https://www.kaggle.com/c/talkingdata-mobile-user-demographics

8http://www.datafountain.cn/#/competitions/247/data-intro
A. Overall Results

Fig. 3 shows the mean estimation accuracy of the three competitive methods, with five virtual iterations enabled after the first real iteration. Overall, PrivKVM is more accurate than the other two, especially for cases with small privacy budgets and real datasets. The former is because small privacy budget leads to heavy perturbation, thus causing more outliers. The latter is because outliers are more frequent in real datasets where KV pairs are very sparse. Eventually, these outliers would be corrected by PrivKVM. In terms of the absolute MSE, the latter four are worse than GAUSS and PLAW, as the numbers of samples in the latter four are less than the former.

For frequency estimation, Fig. 4 shows the results of PrivKVM comparing with RAPPOR, k-RR and SHist as the privacy budget increases from 0.1 to 0.8. Due to space limitation, only the results of PLAW and AppData are shown. For PLAW, we measure and plot the RE for the top 10%, 20%, 30%, 40% and 50% percentile, where PrivKVM always outperforms the other three methods. For AppData, since its sampling datasets are sparse, many keys have very noisy and not meaningful frequency estimation. As such, we only measure and plot the RE for the top 100 keys. Nonetheless, similar observation is made that PrivKVM always outperforms the other three methods.

To compare the end-to-end bandwidth cost, we show the number of transferred bits between a user and the data collector for AppData in Table IV.

TABLE IV

<table>
<thead>
<tr>
<th>Method</th>
<th>k-RR &amp; Harmony</th>
<th>SHist &amp; Harmony</th>
<th>RAPPOR &amp; MeanEst</th>
<th>PrivKV (c=10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>17 bits</td>
<td>18 bits</td>
<td>1166 bits</td>
<td>17 bits</td>
</tr>
</tbody>
</table>

B. Scalability

In this subsection, we evaluate the scalability of PrivKVM, i.e., how the size of user or key space affects the accuracy of frequency and mean estimation. For comparison purpose, we also show the results of k-RR and PrivKVM-Harmony.

First, we investigate the effect of user size. We select six subsets of GAUSS of different sizes and plot the RE and MSE in Fig. 5. PrivKVM outperforms the other methods for both frequency and mean estimation. The gain improves with increasing user sizes due to the law of large numbers. When this size reaches $10^6$, the accuracy is high enough ($RE \leq 0.5$ and $Log(MSE) \leq -0.5$) for most data collection tasks.

We then investigate the effect of key space size. We select six subsets of LNR of different number of keys and plot the RE and MSE in Fig. 6. With increasing sizes of key space, although PrivKVM always outperforms the other methods, both frequency and mean estimation become less accurate. This is because the number of samples for each key decreases as the user size is fixed to $10^6$. To compensate for the accuracy loss, we can choose to use a larger user size.

C. Key-value Correlation

As aforementioned, naive solutions such as k-RR&Harmony which uses k-RR and Harmony to perturb keys and values respectively do not work well as they fail to consider the correlation between keys and values. As a comparison, we evaluate how PrivKVM retains this correlation in this subsection.

We use Pearson correlation coefficient [32] as the metric and test it over PLAW and LNR. Fig. 7 plots the coefficient of real data, and that perturbed by PrivKVM and k-RR&Harmony, respectively. With increasing privacy budget, the correlation coefficient of PrivKVM gradually approaches the real one, whereas that of k-RR&Harmony is always close to zero, which indicates its inability to capture the key-value correlation.

To further illustrate how well the key-value correlation is retained across keys with different frequencies, we plot in Fig. 8 3D illustrations of the estimated mean across keys.

![Fig. 4. Results of frequency estimation, varying privacy budget](image-url)
Correlation and Harmony

PLAW Logging from 1 to 10. We observe that in both synthetic datasets, the tail of the distribution (i.e., key id far away from zero) will always achieve the lowest cost by finding the most suitable strategy \(k\)-RR&Harmony completely deviate from the true distribution, which is a consequence of its inability to retain the correlation between keys and values.

D. Impact of Iterations

In this subsection, we evaluate the impact of iterations in \(PrivKVM\) on the accuracy of mean estimation. Fig. 9 shows the results over GAUSS and PLAW. For each privacy budget, we try 10 \(PrivKVM\) runs with the number of iterations varying from 1 to 10. We observe that in both synthetic datasets, \(Log(MSE)\) decreases as the iteration number increases and it converges to a certain value. This value is completely due to the value perturbation as Theorem 5.3 tells us the absolute error between the expected and real mean would converge to zero.

As a comparison, we also plot the impact on PrivKVM-Harmony and PrivKVM-MeanEst in Fig. 10. We only show the results of PLAW due to space limitation. While the trend of PrivKVM-Harmony is similar to PrivKVM, PrivKVM-MeanEst stays almost flat because it is only applicable to the case where the universe consists of just a few keys. PrivKVM outperforms the other two especially for small and medium privacy budgets. This is because PrivKVM has the built-in capability to correct outliers to reasonable values when large noises are added. This correction becomes more accurate through iterations. In case of larger budgets, this advantage is less eminent with fewer outliers.

E. Impact of Cost Function

Fig. 11 plots the total cost of PrivKVM (3 and 6 iterations) and PrivKVM\(^+\) over GAUSS and PLAW based on two different cost functions in Eq. 3. Here the parameter \(t\) of strategy PBA (Algorithm 6) is set to be 2. The communication cost of each iteration \(A_0\) is set to be 0.2 and 0.02, respectively. We observe that when the accuracy cost is small (\(A_0 = 0.2\)), the total cost of PrivKVM-6 is higher than PrivKVM-3 and the situation is reversed when the accuracy cost is large (\(A_0 = 0.02\)). On the other hand, PrivKVM\(^+\) always achieves the lowest cost by finding the most suitable iteration number to minimize the overall cost. Specifically, for \(A_0 = 0.2\), PrivKVM\(^+\) terminates at the second or third iteration for different privacy budget, while for \(A_0 = 0.02\), it terminates at the fourth or fifth iteration.
latter, we also set its iteration number to the budget increases, the accuracy of iteration, which is bad when the budget is too small. When iterations heavily depends on the accuracy of the first real iteration, which is very inaccurate result. This is because the effect of virtual iteration optimization. They are denoted by PRIVKVM and PRIVKVM-0VI, respectively. For the former, we set c, the number of iterations to execute, to 6, which means 5 virtual iterations will be executed after the first real iteration that involves the user. To be fair to the latter, we also set its iteration number to 6. Fig. 12 shows the results over GAUSS and PLAW. We observe that when the privacy budget is extremely small, PRIVKVM returns very inaccurate result. This is because the effect of virtual iterations heavily depends on the accuracy of the first real iteration, which is bad when the budget is too small. When the budget increases, the accuracy of PRIVKVM improves so rapidly that it outperforms PRIVKVM-0VI very soon. This phenomenon is consistent with our analysis in Section VI and shows that virtual iterations can amplify the effect of real iterations, whether it is a good or bad one.

F. Impact of Virtual Iteration Optimization

In this subsection, we evaluate PRIVKVM with and without the virtual iteration optimization. They are denoted by PRIVKVM and PRIVKVM-0VI, respectively. For the former, we set c, the number of iterations to execute, to 6, which means 5 virtual iterations will be executed after the first real iteration that involves the user. To be fair to the latter, we also set its iteration number to 6. Fig. 12 shows the results over GAUSS and PLAW. We observe that when the privacy budget is extremely small, PRIVKVM returns very inaccurate result. This is because the effect of virtual iterations heavily depends on the accuracy of the first real iteration, which is bad when the budget is too small. When the budget increases, the accuracy of PRIVKVM improves so rapidly that it outperforms PRIVKVM-0VI very soon. This phenomenon is consistent with our analysis in Section VI and shows that virtual iterations can amplify the effect of real iterations, whether it is a good or bad one.

VIII. CONCLUSION

This paper proposes a decentralized privacy-preserving mechanism for frequency and mean estimation on key-value data based on local differential privacy. The building block is the local perturbation protocol, based on which we develop three solutions, namely, PRIVKV, PRIVKVM and PRIVKVM+. An optimization strategy that enables the data collector to execute virtual iterations is also presented. Through theoretical and empirical analysis, we show our solutions are effective and robust in terms of accuracy or total cost under various system parameter settings.

As for future work, we plan to study more aggregate statistics on key-value data, such as maximum and minimum estimation. We also plan to explore LDP for privacy-preserving mining tasks (e.g., finding the gradient descent or k-means clustering), and extend this work to queries with relational dependencies (e.g., natural join) and to unknown key spaces.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (Grant No: 91646203, 61532010, 61532016, 61572413 and U1636205), the National Key Research and Development Program of China (Grant No: 2016YFB1000602 and 2016YFB1000603), the Research Grants Council, Hong Kong SAR, China (Grant No: 12200914, 15238116, and C1008-16G), and research grants from PolyU Start-up Fund and Huawei Technologies. (Corresponding author: Xiaofeng Meng)
REFERENCES


APPENDIX

A. Proof of Theorem 4.1

PROOF. (Theorem 4.1) For user $u_1$, let $(k_j, v^*)$ be the output of Algorithm 3. Let $K^*$ be the key set of canonical form outputted by LPP, whose elements are all 0s except the $j$-th one is $k_j$. In the following, we focus on the case when $k_j = 1$; the case when $k_j = 0$ can be analyzed in a similar manner. The probability of observing $K^*$ given key set of its original canonical form $K$ is denoted as $Pr(K^*|K)$. For LDP condition to hold, the ratio of two such conditional probabilities with distinct key sets $K_1$ and $K_2$, needs to be bounded by $\exp(\epsilon_1)$, where the $j$-th keys are denoted as $k_1$ and $k_2$, respectively.

$$Pr(K^*|K_1) = \frac{1}{d} \cdot Pr(k_j|K_1)$$

$$Pr(K^*|K_2) = \frac{1}{d} \cdot Pr(k_j|K_2)$$

$$\leq \frac{1}{d} \cdot Pr(k_j = 1|K_1)$$

$$\leq \frac{1}{d} \cdot Pr(k_j = 1|k_2 = 0)$$

$$= \frac{\epsilon_1}{1 + \epsilon_1} \cdot \frac{1}{1 + \epsilon_1}$$

$$= \epsilon_1$$

Thus the perturbation on keys in LPP satisfies $\epsilon_1$-LDP.

For value perturbation, we consider two kinds of possible outputs: empty and non-empty KV pair. For as non-empty one, let $V^*$ be the value set of canonical form outputted by LPP, whose elements are 0s except the $j$-th one is $v^*$. Similarly,
in the following, we focus on the case when \( v^* = 1 \). With two distinct value sets \( V_1 \) and \( V_2 \), where the \( j \)-th values are denoted as \( v_1 \) and \( v_2 \) respectively, the following ratio needs to be bounded by \( \exp(\epsilon_2) \).

\[
\frac{\Pr(V^*|V_1)}{\Pr(V^*|V_2)} = \frac{1/d \cdot \Pr(v^*|V_1)}{1/d \cdot \Pr(v^*|V_2)} \\
= \frac{1/v_1 \cdot e^{\epsilon_2}}{1/v_1 \cdot e^{\epsilon_2}} + \frac{1/v_1}{1 + v_1 \cdot e^{\epsilon_2}} \\
\leq \min_{v_2} \left\{ \frac{v_1(e^{\epsilon_2} - 1) + e^{\epsilon_2} + 1}{v_2(e^{\epsilon_2} - 1) + e^{\epsilon_2} + 1} \right\}
\]

As for the output of an empty KV pair, it is quite straightforward as follows:

\[
\frac{\Pr(V^*=0|V_1)}{\Pr(V^*=0|V_2)} = \frac{1/d \cdot \Pr(v^* = 0|V_1)}{1/d \cdot \Pr(v^* = 0|V_2)} \\
\leq \frac{1/d \cdot \Pr(v^* = 0|v_1 = 0)}{1/d \cdot \Pr(v^* = 0|v_2 \neq 0)} \\
= \left( \frac{e^{\epsilon_2}}{1 + e^{\epsilon_2}} \right) \left( \frac{1}{1 + e^{\epsilon_2}} \right)
= e^{\epsilon_2}
\]

Thus the perturbation on values in LPP satisfies \( \epsilon_2 \)-LDP.

According to the sequential composition (Theorem 3.1), we conclude that LPP satisfies \( \epsilon \)-LDP, where \( \epsilon = \epsilon_1 + \epsilon_2 \). \( \square \)