PrivKV: Key-Value Data Collection with Local Differential Privacy

Qingqing Ye * †, Haibo Hu †, Xiaofeng Meng *, Huadi Zheng †
*School of Information, Renmin University of China
{*yeqq, 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 $PrivKVM$ and $PrivKVM^+$ 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 backorders [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 ads.
- **Mobile app activity analysis.** Smartphone manufactures 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 $\{\langle Cancer, 0.6\rangle, \langle HIV, 0.9\rangle, \langle Fever, 0.08\rangle\}$ 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 $PrivKVM$ and $PrivKVM^+$ 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

Solution	Features	Application Scenarios
$PrivKV$	lowest comm. cost with one iteration	non-iterative (e.g., no downlink channel)
$PrivKVM$	high accuracy with multiple and fixed iterations	sufficient and fixed comm. bandwidth
$PrivKVM^+$	balanced between accuracy and comm. bandwidth	flexible comm. bandwidth

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 ϵ-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 Nguyên *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. $\mathcal{M}$ is a randomized algorithm that takes a data tuple t as input and outputs t^*. ϵ-local differential privacy (or ϵ-LDP) is defined on $\mathcal{M}$ and a privacy budget $\epsilon > 0$ as follows.

Definition 1 (ϵ-local differential privacy). A randomized algorithm $\mathcal{M}$ satisfies ϵ-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.

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

Intuitively ϵ-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 ϵ), 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 c randomized algorithms $\mathcal{M}_i (1 \leq i \leq c)$, each providing ϵ_i-local differential privacy. Then the sequence of algorithms $\mathcal{M}_i (1 \leq i \leq c)$ collectively provides $(\Sigma \epsilon_i)$-local differential privacy.

According to sequential composition, given a privacy budget ϵ, 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 ϵ-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.

TABLE II
NOTATIONS

Symbol	Description
$\mathcal{U}$	the set of users
n	the number of users, $n = \lvert\mathcal{U}\rvert$
u_i	the i-th user in $\mathcal{U}$
$\mathcal{K}$	the set of keys
d	the number of keys, $d = \lvert\mathcal{K}\rvert$
S_i	the set of KV pairs possessed by u_i
l_i	the number of KV pairs in S_i, $l_i = \lvert S_i\rvert$
$\langle k_j, v_j \rangle$	the j-th KV pair in S_i
f_k	the frequency of key k
m_k	the mean of all values with key k

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 $\mathcal{U} = \{u_1, u_2, ..., u_n\}$, and a set of keys $\mathcal{K} = \{1, 2, ..., d\}$ whose value domain $\mathcal{V}$ is the continuous domain $[-1, 1]$. The i-th user u_i possesses l_i key-value (KV) pairs $S_i = \{\langle k_j, v_j \rangle | 1 \leq j \leq l_i, k_j \in \mathcal{K}, v_j \in \mathcal{V}\}$. 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,

$$f_k = \frac{|\{u_i | \exists \langle k, v \rangle \in S_i\}|}{n} \tag{1}$$

- **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 $\langle k, v \rangle \in S_i$, we convert it to $\langle 1, v \rangle$ in S_i'; for a KV pair that does not exist, $\langle 0, 0 \rangle$, an empty KV pair, is added to S_i'. All KV pairs are sorted in ascending order of keys. The canonical form guarantees each

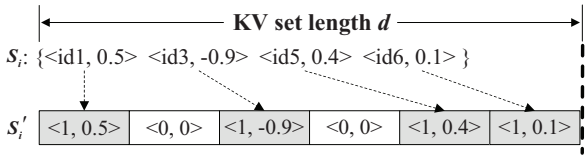


Fig. 1. Conversion of user's set of KV pairs

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:

- $\mathbf{1 \rightarrow 1}$: A KV pair exists before and after the perturbation. In this case, the value should be retained, i.e., $\langle 1, v\rangle \rightarrow \langle 1, v\rangle$.
- $\mathbf{1 \rightarrow 0}$: 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., $\langle 1, v\rangle \rightarrow \langle 0, 0\rangle$. Note that this change does not affect the mean estimation of that key.
- $\mathbf{0 \rightarrow 0}$: A KV pair does not exist before or after the perturbation. In this case, the whole KV pair is unchanged, i.e., $\langle 0, 0\rangle \rightarrow \langle 0, 0\rangle$.
- $\mathbf{0 \rightarrow 1}$: 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 Nguyên *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

Algorithm 1 Decomposition of Harmony

Input: User u_i's set of values $V \in [-1, 1]^d$
Privacy budget ϵ
Output: Perturbed set of values V^*
Procedure:

1: Let $V^* = \langle 0, 0, ..., 0\rangle$
2: Sample j uniformly at random from $[d]$, let $v = V_j$
3: Discretization:

$$v^* = \begin{cases} 1 & \text{w.p. } \frac{1+v}{2} \\ -1 & \text{w.p. } \frac{1-v}{2} \end{cases}$$

4: Perturbation:

$$v^* = \begin{cases} v^* & \text{w.p. } \frac{e^\epsilon}{1+e^\epsilon} \\ -v^* & \text{w.p. } \frac{1}{1+e^\epsilon} \end{cases}$$

5: Calibration:

$$v^* = v^* \cdot \frac{e^\epsilon + 1}{e^\epsilon - 1} \cdot d$$

6: $V_j^* = v^*$
7: return V^*

Algorithm 2 Value Perturbation Primitive

Input: Value v of a KV pair
Privacy budget ϵ
Output: $VPP(v, \epsilon)$ is the perturbed value v^*
Procedure:

1: Discretization:

$$v^* = \begin{cases} 1 & \text{w.p. } \frac{1+v}{2} \\ -1 & \text{w.p. } \frac{1-v}{2} \end{cases}$$

2: Perturbation:

$$v^* = \begin{cases} v^* & \text{w.p. } \frac{e^\epsilon}{1+e^\epsilon} \\ -v^* & \text{w.p. } \frac{1}{1+e^\epsilon} \end{cases}$$

3: return v^*

(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 S_i, the set of all different keys $\mathcal{K}$, and the privacy budgets ϵ_1 and ϵ_2 for key and value perturbation, and returns a perturbed KV pair with its index. In order to reduce the communication cost

Algorithm 3 Local Perturbation Protocol (LPP)

Input: User u_i's set of KV pairs S_i
The set of keys $\mathcal{K}$
Privacy budgets ϵ_1 and ϵ_2
Output: $LPP(S_i, \mathcal{K}, \epsilon_1, \epsilon_2)$ is the perturbed KV pair $\langle k_j, v^* \rangle$ of the j-th key
Procedure:

1: $d = |\mathcal{K}|$
2: Sample j uniformly at random from $[d]$
3: **if** k_j exists in the key set of S_i **then**
4: $v^* = VPP(v_j, \epsilon_2)$
5: Perturbs $\langle k_j, v^* \rangle$ as:

$$\langle k_j, v^* \rangle = \begin{cases} \langle 1, v^* \rangle & \text{w.p. } \frac{e^{\epsilon_1}}{1+e^{\epsilon_1}} \\ \langle 0, 0 \rangle & \text{w.p. } \frac{1}{1+e^{\epsilon_1}} \end{cases}$$

6: **else**
7: Randomly draw a value $\widetilde{m} \in [-1, 1]$
8: $v^* = VPP(\widetilde{m}, \epsilon_2)$
9: Perturbs $\langle k_j, v^* \rangle$ as:

$$\langle k_j, v^* \rangle = \begin{cases} \langle 0, 0 \rangle & \text{w.p. } \frac{e^{\epsilon_1}}{1+e^{\epsilon_1}} \\ \langle 1, v^* \rangle & \text{w.p. } \frac{1}{1+e^{\epsilon_1}} \end{cases}$$

10: return j and $\langle k_j, v^* \rangle$

from $O(d)$ to $O(1)$, we adopt the same sampling approach as in [5] to select only the j-th KV pair $\langle k_j, v_j \rangle$ 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 $\langle k_j, v^* \rangle$ into canonical form $\langle 1, v^* \rangle$, and finally perturbs it to $\langle 1, v^* \rangle$ (resp. $\langle 0, 0 \rangle$) 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 $\widetilde{m}$ 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 ϵ-LDP.

PROOF. See Appendix A. □

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

[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.

Algorithm 4 PrivKV

Input: All users' sets of KV pairs $S = \{S_1, ..., S_n\}$
The set of keys $\mathcal{K}$
Privacy budgets ϵ_1 and ϵ_2
Output: Frequency vector $\boldsymbol{f}^*$
Mean vector $\boldsymbol{m}^*$
Procedure:

1: //**User-side perturbation**
2: Each user perturbs her set and sends the index j and $\langle k_j, v^* \rangle = LPP(S_i, \mathcal{K}, \epsilon_1, \epsilon_2)$ to data collector
3: //**Collector-side calibration**
4: **for** each key $k \in \mathcal{K}$ **do**
5: Collector calculates frequency f_k^*
6: Collector calibrates the frequency as:

$$f_k^* = \frac{p - 1 + f_k^*}{2p - 1}, \text{where } p = \frac{e^{\epsilon_1}}{e^{\epsilon_1} + 1}$$

7: Collector counts 1 and -1 in the set of values:

$$n_1' = Count(1),\ n_2' = Count(-1)$$

8: $N = n_1' + n_2'$
9: Collector calibrates the counts as:

$$n_1^* = \frac{p-1}{2p-1} \cdot N + \frac{n_1'}{2p-1}$$

$$n_2^* = \frac{p-1}{2p-1} \cdot N + \frac{n_2'}{2p-1}, \text{where } p = \frac{e^{\epsilon_2}}{e^{\epsilon_2} + 1}$$

10: Clip n_1^* and n_2^* to $[0, N]$
11: Collector calculates mean $m_k^* = \frac{n_1^* - n_2^*}{N}$
12: return $\boldsymbol{f}^*$ and $\boldsymbol{m}^*$

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 ②③⑤⑥ 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.

[2]To protect $\widetilde{m}$ from being disclosed to users in step ⑥, 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 $\widetilde{m}$. It is either 1 or -1 with probabilities $\frac{1+\widetilde{m}}{2}$ and $\frac{1-\widetilde{m}}{2}$, respectively. Since these two probabilities are only known to the data collector, no user can infer $\widetilde{m}$ unless a large number of users collude and share their received 1's and -1's. In such extreme cases, protecting $\widetilde{m}$ is no longer necessary as these users can derive it by themselves.

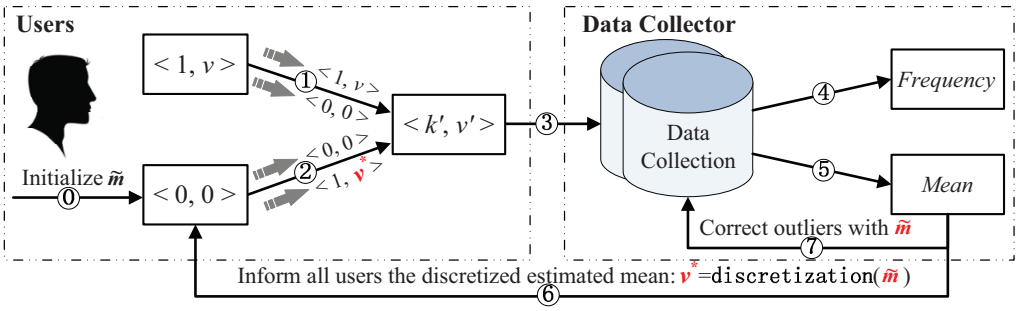


Fig. 2. An Iterative execution model for $PrivKV$

Theorem 5.1: Let m_k be the true mean of all values in KV pairs with key k, and $m_k{}^*$ be the estimated mean by $PrivKV$ (Algorithm 4) when assigning $v^* = discretization(\widetilde{m}_k)$ to new values, where $\widetilde{m}_k$ is an approximation of m_k. Then, $\mathbb{E}[m_k^*] = m_k$.

PROOF. With key perturbation, values are changed accordingly based on the four cases in section IV-A. For key k, let $\bar{m}_k$ be the mean of all values after key perturbation, in which we suppose there are n_1 instances of $\langle 1, v\rangle \to \langle 1, v\rangle$ and n_2 instances of $\langle 0, 0\rangle \to \langle 1, v^*\rangle$. Note that $\sum_{i=1}^{n_1}\sum_{j:k_j=k} v_j = n_1 \cdot m_k$. Thus we have:

$$\begin{aligned}\mathbb{E}[\bar{m}_k] &= \mathbb{E}[\frac{\sum_{i=1}^{n_1}\sum_{j:k_j=k} v_j + \sum_{i=1}^{n_2} v^*}{n_1+n_2}] \\ &= \frac{n_1 m_k + n_2(\frac{1+\widetilde{m}}{2} - \frac{1-\widetilde{m}}{2})}{n_1+n_2} \\ &= \frac{n_1 m_k + n_2 m_k}{n_1+n_2} = m_k\end{aligned}$$

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

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

$$\begin{aligned}\mathbb{E}[\hat{m}_k] &= \mathbb{E}[\frac{1}{n_1+n_2}\sum_i \sum_{j:k_j=k} \hat{v}_j] \\ &= \frac{1}{n_1+n_2}\sum_i \sum_{j:k_j=k} \mathbb{E}[\hat{v}_j] \\ &= \frac{1}{n_1+n_2}\sum_i \sum_{j:k_j=k} \bar{v}_j = \mathbb{E}[\bar{m}_k]\end{aligned}$$

Let $\hat{n}_1$ and $\hat{n}_2$ 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:

$$\begin{aligned}\mathbb{E}[n_1'] &= \hat{n}_1 \cdot p + \hat{n}_2 \cdot (1-p) \\ \mathbb{E}[n_2'] &= \hat{n}_2 \cdot p + \hat{n}_1 \cdot (1-p)\end{aligned}$$

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

After value perturbation, the counts n_1' and n_2' will be calibrated to n_1^* and n_2^*, respectively.

$$\begin{aligned}n_1^* &= \frac{(p-1)(n_1'+n_2') + n_1'}{2p-1} \\ n_2^* &= \frac{(p-1)(n_1'+n_2') + n_2'}{2p-1}\end{aligned}$$

Therefore, the expected mean is:

$$\begin{aligned}\mathbb{E}[m_k^*] &= \mathbb{E}[\frac{n_1^*-n_2^*}{n_1+n_2}] = \mathbb{E}[\frac{n_1'-n_2'}{(2p-1)(n_1+n_2)}] \\ &= \frac{\mathbb{E}[n_1'] - \mathbb{E}[n_2']}{(2p-1)(n_1+n_2)} = \frac{(\hat{n}_1-\hat{n}_2)(2p-1)}{(2p-1)(n_1+n_2)} \\ &= \frac{\hat{n}_1-\hat{n}_2}{\hat{n}_1+\hat{n}_2} = \mathbb{E}[\hat{m}_k] = \mathbb{E}[\bar{m}_k] = m_k \quad \square\end{aligned}$$

In this iterative model, an enhanced scheme for outliers correction based on apriori knowledge can also be devised (as the loop ⑤⑦ 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. $PrivKVM$

The iterative model immediately leads to $PrivKVM$, whose details are shown in Algorithm 5. It takes as inputs all users' sets of KV pairs S, the set of keys $\mathcal{K}$, privacy budget ϵ and the number of iterations c. Privacy budget ϵ 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 $\boldsymbol{f}^{(1)}$ and mean $\boldsymbol{m}^{(1)}$.[3] Then the data collector uses $\boldsymbol{m}^{(1)}$ 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 ϵ_{1r}, ϵ_{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 $\widetilde{\boldsymbol{m}}$ (see Line 8 of Algorithm 3). In $PrivKVM$, this discretization is performed by the data collector based on the returned values from users in

[3]Here we use superscripts to denote which iterations the estimated results are calculated from.

Algorithm 5 PrivKVM: Iterative PrivKV

Input: All users' sets of KV pairs $S = \{S_1, ..., S_n\}$
The set of keys $\mathcal{K}$
Privacy budget ϵ
Number of iterations c

Output: Frequency vector $\boldsymbol{f}^{(1)}$
Mean vector $\boldsymbol{m}^{(c)}$

Procedure:

1: Allocate privacy budget:

$$\{\epsilon_{11}, ..., \epsilon_{1c}, \epsilon_{21}, ..., \epsilon_{2c}\} \leftarrow PBA(\epsilon, c)$$

2: Calculate frequency and mean in the first iteration:

$$\boldsymbol{f}^{(1)}, \boldsymbol{m}^{(1)} = PrivKV(S, \mathcal{K}, \epsilon_{11}, \epsilon_{21})$$

3: Collector sends back $\boldsymbol{v}^* = discretization(\boldsymbol{m}^{(1)})$ to each user
4: **for** $r = 2$ to c **do**
5: Calculate mean: $\boldsymbol{m}^{(r)} = PrivKV'(S, \mathcal{K}, \epsilon_{1r}, \epsilon_{2r}, \boldsymbol{v}^*)$
6: Collector sends back $\boldsymbol{v}^* = discretization(\boldsymbol{m}^{(r)})$ to each user
7: return $\boldsymbol{f}^{(1)}$ and $\boldsymbol{m}^{(c)}$

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 ϵ 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 ϵ_1 only to the first iteration. That is, $\epsilon_{11} = \epsilon_1$ and $\epsilon_{12}, ..., \epsilon_{1c} = 0$. For mean estimation, we allocate the privacy budget of ϵ_2 equally to each iteration. That is, $\epsilon_{21} = \epsilon_{22} = ... = \epsilon_{2c} = \epsilon_2/c$.

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 from a lower half. Since a frequency histogram is essentially a discrete form of the probability distribution, the median can be estimated by dividing the histogram into two halves, each with the same area. To calculate the area, we can use the bar shape of each bin (i.e., we assume the values in each bin follow a uniform distribution). To improve the estimation accuracy and rely less on the uniform assumption, we can further generalize this idea by using multiple bins in the frequency histogram. To enable this, we just need to slightly modify the steps of $discretization$ and $pertubation$ in the value perturbation primitive (Algorithm 2) to expand the perturbed set of values accordingly.

It is noteworthy that the iterative version of $PrivKVM$ is more suitable for data collection of archive or historical data than real-time data, as multiple iterations increase the response time of data collection. Nonetheless, $PrivKVM$ can still handle key-value data changes between iterations, as long as the key or value distribution is not changing.

C. Privacy and Accuracy Analysis

The following two theorems establish the privacy and accuracy guarantee of $PrivKVM$, respectively. Theorem 5.2 proves that $PrivKVM$ satisfies ϵ-LDP, and since the proof is similar to that of Theorem 4.1, we omit it here. Theorem 5.3 proves that the difference between the true mean and the expectation of the estimated mean converges to zero. Based on the unbiasedness by Theorem 5.1, $PrivKVM$ guarantees the convergence with multiple iterations.

Theorem 5.2: $PrivKVM$ satisfies ϵ-LDP.

Theorem 5.3: Let m_k be the true mean of values in KV pairs whose key is k, and $\mathbb{E}[m_k^{(c)}]$ be the expectation of the estimated mean returned from the c-th iteration of $PrivKVM$. $\lim_{c\to\infty} |m_k - \mathbb{E}[m_k^{(c)}]| = 0$.

PROOF. Assuming that a sequence of privacy budgets $\epsilon_{11}, ..., \epsilon_{1c}$ ($\epsilon_1 = \sum_{r=1}^{c} \epsilon_{1r}$) are allocated to $1, ..., c$-th iteration for key perturbation. We expose c intermediate variables $p_1, ..., p_c$ for brevity, where $p_r = \frac{e^{\epsilon_{1r}}}{1+e^{\epsilon_{1r}}} (1 \le r \le c)$.

Let f_k denote the real frequency of key k. Hence, after the conversion of KV pairs, we have $p(k =$"1"$) = f_k$ among all n users. Note that after key perturbation in the r-th iteration, the above frequency will be updated to f'_k:

$$f'_k = f_k p_r + (1 - f_k)(1 - p_r) = 2 f_k p_r - f_k - p_r + 1$$

In the first iteration, suppose the initialized approximated mean is $\widetilde{m}_k$, thus we have:

$$\begin{aligned}\mathbb{E}[m_k^{(1)}] &= \frac{(1 - f_k)(1 - p_1)n \cdot \widetilde{m}_k + f_k p_1 n \cdot m_k}{f'_k n} \\ &= \frac{(f_k p_1 - f_k - p_1 + 1) \cdot \widetilde{m}_k + f_k p_1 \cdot m_k}{2 f_k p_1 - f_k - p_1 + 1}\end{aligned}$$

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

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

Thus we have:

$$\begin{aligned}|m_k - \mathbb{E}[m_k^{(c)}]| &= \frac{f_k p_c - f_k - p_c + 1}{2 f_k p_c - f_k - p_c + 1} |m_k - \mathbb{E}[m_k^{(c-1)}]| \\ &= \prod_{r=c-1}^{c} \frac{f_k p_r - f_k - p_r + 1}{2 f_k p_r - f_k - p_r + 1} |m_k - \mathbb{E}[m_k^{(c-2)}]| \\ &= \prod_{r=1}^{c} \frac{f_k p_r - f_k - p_r + 1}{2 f_k p_r - f_k - p_r + 1} |m_k - \widetilde{m}_k|\end{aligned}$$

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

$$\lim_{c\to\infty} |m_k - \mathbb{E}[m_k^{(c)}]| = 0 \quad \square$$

Algorithm 6 $PrivKVM^+$: Adaptive $PrivKVM$

Input: All users' sets of KV pairs $S = \{S_1, ..., S_n\}$
The set of keys $\mathcal{K}$
Privacy budget ϵ
Communication cost of one iteration A_0
Output: Frequency vector $\boldsymbol{f}^*$
Mean vector $\boldsymbol{m}^*$
Procedure:

1: Allocate privacy budget: $\{\epsilon_1, \epsilon_2\} \leftarrow PBAt(\epsilon)$
2: Initialize $\widetilde{\boldsymbol{m}} = \mathbf{1}$, $\boldsymbol{v}^* = discretization(\widetilde{\boldsymbol{m}})$
3: $\boldsymbol{f}^*, \boldsymbol{m}^* = PrivKV'(S, \mathcal{K}, \epsilon_1, \epsilon_2, \boldsymbol{v}^*)$
4: Calculate the bias: $F^* = A_0 - \frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}} |m_k^* - \widetilde{m}_k|$
5: **while** $F^* < 0$ **do**
6: Collector sends $\boldsymbol{v}^* = discretization(\boldsymbol{m}^*)$ to each user
7: $\widetilde{\boldsymbol{m}} = \boldsymbol{m}^*$
8: $\{\epsilon_1, \epsilon_2\} \leftarrow PBAt(\epsilon - \epsilon_1 - \epsilon_2)$
9: $\boldsymbol{m}^* = PrivKV'(S, \mathcal{K}, \epsilon_1, \epsilon_2, \boldsymbol{v}^*)$
10: $F^* = A_0 - \frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}} |m_k^* - \widetilde{m}_k|$
11: return $\boldsymbol{f}^*$ and $\boldsymbol{m}^*$

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 $k \in \mathcal{K}$ and $|\mathcal{K}| = d$, 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,

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

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 infinite 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]:

$$F(r) = F_1(r) + F_2(r) \tag{3}$$

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

$$F_1(r) = \frac{1}{d}\sum_{k\in\mathcal{K}} |m_k - m_k^{(r)}| \tag{4}$$

[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.

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

$$F_2(r) = A_0 \cdot r \tag{5}$$

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

$$F(r) - F(r-1) = A_0 - \frac{1}{d}\sum_{k\in\mathcal{K}} |m_k^{(r)} - m_k^{(r-1)}| \geq 0 \tag{6}$$

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 $\boldsymbol{m}^{(c)}$ after the first iteration is really executed with estimated frequency vector $\boldsymbol{f}$ and mean vector $\boldsymbol{m}^{(1)}$ (Line 2). The prediction takes two steps. For each key k, the collector calculates an intermediate variable θ based on frequency $\boldsymbol{f}$ and privacy budget ϵ that is allocated to the first iteration (Line 4).

Algorithm 7 Executing Virtual Iterations

Input: All users' sets of KV pairs $S = \{S_1, ..., S_n\}$
The set of keys $\mathcal{K}$
Privacy budget ϵ
Number of iterations to execute c

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

Procedure:

1: Initialize $\widetilde{\boldsymbol{m}} = \mathbf{1}$, $\boldsymbol{v}^* = discretization(\widetilde{\boldsymbol{m}})$
2: Calculate the frequency and mean:

$$\boldsymbol{f}, \boldsymbol{m}^{(1)} = PrivKV'(S, \mathcal{K}, \epsilon/2, \epsilon/2, \boldsymbol{v}^*)$$

3: **for** each key $k \in \mathcal{K}$ **do**
4: Collector calculates $\theta = \frac{f_k p - f_k - p + 1}{2 f_k p - f_k - p + 1}$, where $p = \frac{e^{\epsilon/2}}{1 + e^{\epsilon/2}}$
5: Collector predicts $m_k^{(c)} = \widetilde{m}_k + \frac{(m_k^{(1)} - \widetilde{m}_k)(1 - \theta^c)}{1 - \theta}$
6: return $\boldsymbol{f}$ and $\boldsymbol{m}^{(c)}$

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 $\widetilde{m}_k$ be the initial mean and $m_k^{(1)}$ be the estimated mean of the first iteration with privacy budget ϵ. The expectation of the estimated mean of the c-th iteration $\mathbb{E}[m_k^{(c)}] = \widetilde{m}_k + \frac{(\mathbb{E}[m_k^{(1)}] - \widetilde{m}_k)(1 - \theta^c)}{1 - \theta}$, where $\theta = \frac{f_k p - f_k - p + 1}{2 f_k p - f_k - p + 1}$ and $p = \frac{e^{\epsilon}}{e^{\epsilon} + 1}$.

PROOF. We first show the following lemma on the ratio of bias in two consecutive iterations r and $r + 1$.

Lemma 6.2: For a key k, let $\mathbb{E}[m_k^{(r-1)}]$, $\mathbb{E}[m_k^{(r)}]$ and $\mathbb{E}[m_k^{(r+1)}]$ be the expectation of the estimated means of three consecutive iterations. The privacy budget ϵ_r and ϵ_{r+1} are allocated to r-th and $(r + 1)$-th iterations, respectively. Then the ratio of bias in these two iterations $\frac{|\mathbb{E}[m_k^{(r+1)}] - \mathbb{E}[m_k^{(r)}]|}{|\mathbb{E}[m_k^{(r)}] - \mathbb{E}[m_k^{(r-1)}]|} = \frac{(f_k p_r - f_k - p_r + 1) \cdot p_{r+1}}{(2 f_k p_{r+1} - f_k - p_{r+1} + 1) \cdot p_r}$, where $p_r = \frac{e^{\epsilon_r}}{1 + e^{\epsilon_r}}$, $p_{r+1} = \frac{e^{\epsilon_{r+1}}}{1 + e^{\epsilon_{r+1}}}$ and f_k is the real frequency of key k.[5]

PROOF. According to the proof of Theorem 5.3, we have:

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

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

Thus we express the bias as follows:

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

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

[5]In practice, we substitute f_k with the estimated frequency f_k^*.

Therefore, the ratio of bias can be derived as:

$$\frac{|\mathbb{E}[m_k^{(r+1)}] - \mathbb{E}[m_k^{(r)}]|}{|\mathbb{E}[m_k^{(r)}] - \mathbb{E}[m_k^{(r-1)}]|}$$
$$= \frac{f_k p_{r+1} \cdot |m_k - \mathbb{E}[m_k^{(r)}]|}{f_k p_r \cdot |m_k - \mathbb{E}[m_k^{(r-1)}]|} \cdot \frac{2 f_k p_r - f_k - p_r + 1}{2 f_k p_{r+1} - f_k - p_{r+1} + 1}$$
$$= \frac{(f_k p_r - f_k - p_r + 1) \cdot p_{r+1}}{(2 f_k p_{r+1} - f_k - p_{r+1} + 1) \cdot p_r} \quad \square$$

According to Lemma 6.2, when given the same privacy budget ϵ, p is a constant. So for any three means returned from two consecutive iterations, $\frac{|\mathbb{E}[m_k^{(r+1)}] - \mathbb{E}[m_k^{(r)}]|}{|\mathbb{E}[m_k^{(r)}] - \mathbb{E}[m_k^{(r-1)}]|} = \frac{f_k p - f_k - p + 1}{2 f_k p - f_k - p + 1}$. Let θ denote this ratio.

Then we have the following geometric progression: $\{\mathbb{E}[m_k^{(r+1)}] - \mathbb{E}[m_k^{(r)}]\}$ and its general formula can be written as:

$$\mathbb{E}[m_k^{(r+1)}] - \mathbb{E}[m_k^{(r)}] = (\mathbb{E}[m_k^{(1)}] - \widetilde{m}_k) \cdot \theta^r$$

Thus we have:

$$\begin{aligned}
\mathbb{E}[m_k^{(1)}] - \widetilde{m}_k &= (\mathbb{E}[m_k^{(1)}] - \widetilde{m}_k) \cdot \theta^0 \\
\mathbb{E}[m_k^{(2)}] - \mathbb{E}[m_k^{(1)}] &= (\mathbb{E}[m_k^{(1)}] - \widetilde{m}_k) \cdot \theta^1 \\
\mathbb{E}[m_k^{(3)}] - \mathbb{E}[m_k^{(2)}] &= (\mathbb{E}[m_k^{(1)}] - \widetilde{m}_k) \cdot \theta^2 \\
&\cdots \\
\mathbb{E}[m_k^{(c)}] - \mathbb{E}[m_k^{(c-1)}] &= (\mathbb{E}[m_k^{(1)}] - \widetilde{m}_k) \cdot \theta^{c-1}
\end{aligned}$$

By summing up the above equations, we have:

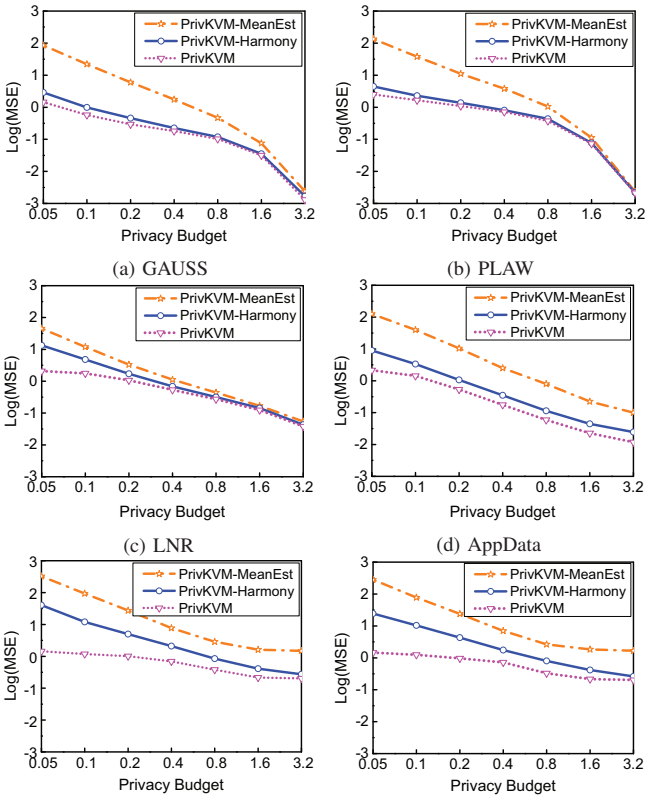
$$\mathbb{E}[m_k^{(c)}] = \widetilde{m}_k + \frac{(\mathbb{E}[m_k^{(1)}] - \widetilde{m}_k)(1 - \theta^c)}{1 - \theta}$$

This completes the proof. $\square$

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 $PrivKVM$.



(a) GAUSS (b) PLAW (c) LNR (d) AppData (e) TalkingData (f) JData

Fig. 3. Results of mean estimation, varying privacy budget

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[7].** 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[8].** 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.

TABLE III
DATASETS

Datasets	Distributions	Users	Keys	Avg.(f)	Var.(f)	Avg.(m)	Var.(m)
GAUSS	gaussian	10^6	100	0.3000	0.0401	0.0207	0.3080
PLAW	power-law	10^6	100	0.1384	0.0167	-0.0723	0.0656
LNR	linear	10^6	1000	0.4003	0.0005	0.0010	0.3330
AppData	–	2006631	1134	0.0013	0.0002	-0.0010	0.0002
TalkingData	–	60822	306	0.0693	0.0255	-0.6780	0.2480
JData	–	105180	442	0.0251	0.0065	-0.6410	0.2690

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 $k \in \mathcal{K}$, let f_k and f_k^* denote real and estimated frequency, respectively.

$$RE = Stat_{k\in\mathcal{K}} \frac{|f_k - f_k^*|}{f_k}, \tag{7}$$

  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 $k \in \mathcal{K}$, let m_k and m_k^* denote real and estimated means of all values whose key is k, and $d = |\mathcal{K}|$.

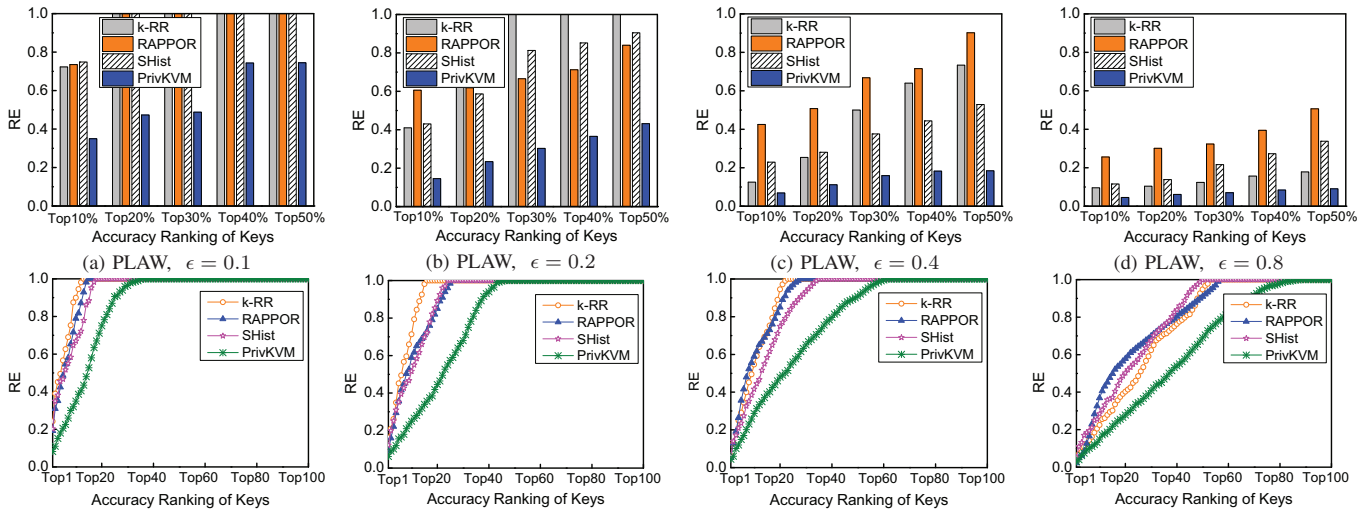
$$MSE = \frac{1}{d} \sum_{k\in\mathcal{K}} (m_k - m_k^*)^2 \tag{8}$$

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

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

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

[8]http://www.datafountain.cn/#/competitions/247/data-intro



(a) PLAW, $\epsilon = 0.1$ (b) PLAW, $\epsilon = 0.2$ (c) PLAW, $\epsilon = 0.4$ (d) PLAW, $\epsilon = 0.8$

(e) AppData, $\epsilon = 0.1$ (f) AppData, $\epsilon = 0.2$ (g) AppData, $\epsilon = 0.4$ (h) AppData, $\epsilon = 0.8$

Fig. 4. Results of frequency estimation, varying privacy budget

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
COMPARISON ON END-TO-END BANDWIDTH COST

Method	*k-RR &Harmony*	*SHist &Harmony*	*RAPPOR &MeanEst*	*PrivKV*	*PrivKVM* (c=10)
Cost	17 bits	18 bits	1166 bits	17 bits	323 bits

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.

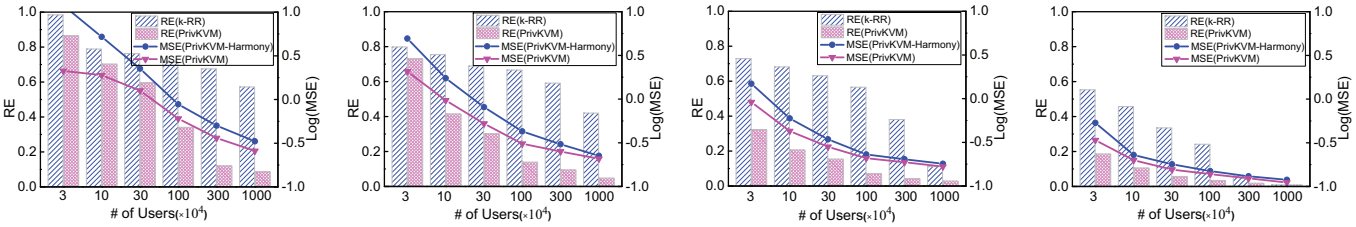
We then investigate the effect of key space size. We select six subsets of *LNR* of different number of keys and plot the RE and MSE in Fig. 6. With increasing sizes of key space, although $PrivKVM$ always outperforms the other methods, both frequency and mean estimation become less accurate. This is 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.

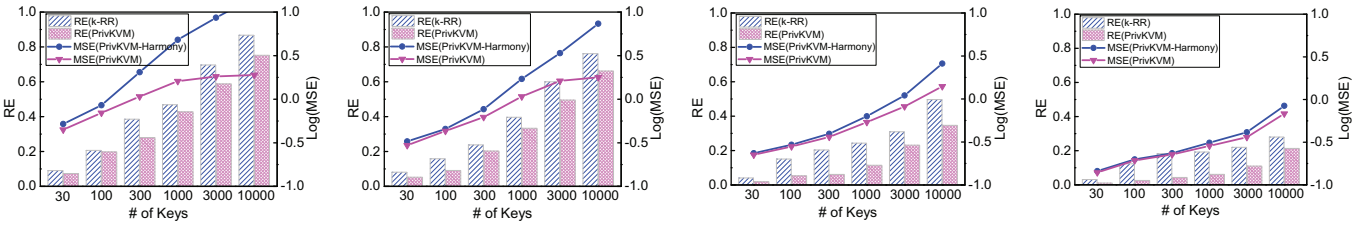
We use *Pearson correlation coefficient* [32] as the metric and test it over *PLAW* and *LNR*. Fig. 7 plots the coefficient of real data, and that perturbed by $PrivKVM$ and k-$RR\&Harmony$, respectively. With increasing privacy budget, the correlation coefficient of $PrivKVM$ gradually approaches the 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



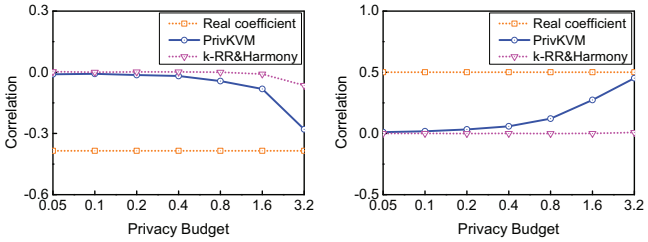
(a) GAUSS, $\epsilon = 0.1$ (b) GAUSS, $\epsilon = 0.2$ (c) GAUSS, $\epsilon = 0.4$ (d) GAUSS, $\epsilon = 0.8$

Fig. 5. Results of frequency and mean estimation, varying user size



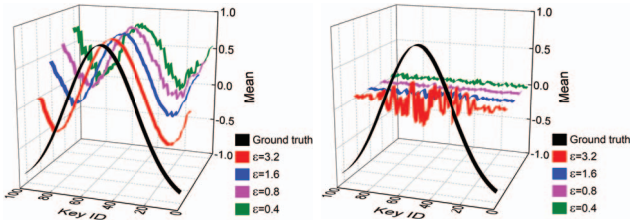
(a) LNR, $\epsilon = 0.1$ (b) LNR, $\epsilon = 0.2$ (c) LNR, $\epsilon = 0.4$ (d) LNR, $\epsilon = 0.8$

Fig. 6. Results of frequency and mean estimation, varying key space



(a) PLAW (b) LNR

Fig. 7. Pearson correlation coefficient, varying privacy budget



(a) GAUSS, $PrivKVM$ (b) GAUSS, k-$RR\&Harmony$

Fig. 8. 3D illustration of estimated mean across different keys and privacy budgets

in the $GAUSS$ distribution under different privacy budgets. We observe that the estimated means of $PrivKVM$ follow a very similar distribution as the real mean, especially for those keys with high frequency (i.e., key id close to 50). The tail of the distribution (i.e., key id far away from 50) has a larger deviation due to the smaller sample size of keys. On the contrary, the estimated means of k-$RR\&Harmony$ completely deviate from the true distribution, which is a consequence of its inability to retain the correlation between keys and values.

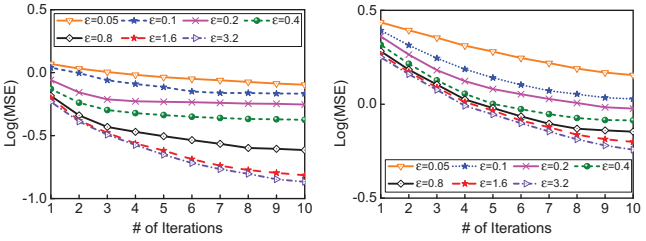
D. Impact of Iterations

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.

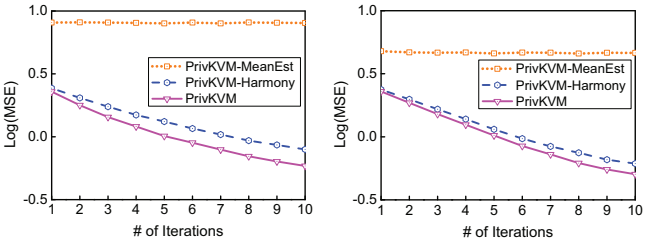
E. Impact of Cost Function

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 $PBAt$ (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.

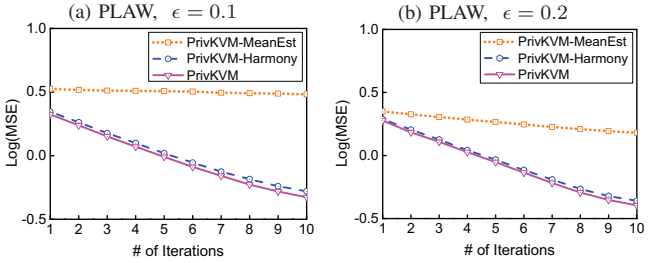


(a) GAUSS (b) PLAW

Fig. 9. Error convergence of $PrivKVM$

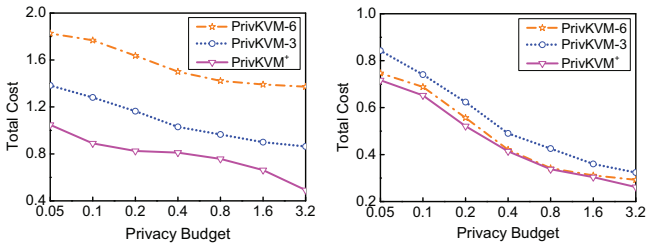


(a) PLAW, $\epsilon = 0.1$ (b) PLAW, $\epsilon = 0.2$

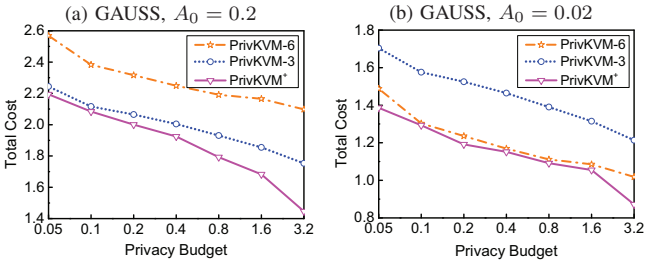


(c) PLAW, $\epsilon = 0.4$ (d) PLAW, $\epsilon = 0.8$

Fig. 10. Results of mean estimation, varying iteration number

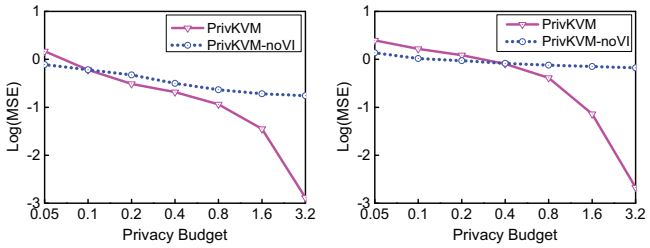


(a) GAUSS, $A_0 = 0.2$ (b) GAUSS, $A_0 = 0.02$



(c) PLAW, $A_0 = 0.2$ (d) PLAW, $A_0 = 0.02$

Fig. 11. Results of total cost, varying cost function



(a) GAUSS (b) PLAW

Fig. 12. Results of mean estimation, with and without virtual iteration

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 $PrivKVM$-$noVI$, 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 $PrivKVM$-$noVI$ 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

[1] Apple's 'differential privacy' is about collecting your data — but not your data. *Wired*, Jun 13, 2016.

[2] Turning big data into big money: How amazon is leveraging big data. *Linkedin*, May 2, 2017.

[3] M. Akter and T. Hashem. Computing aggregates over numeric data with personalized local differential privacy. In *ACISP*, pages 249–260. Springer, 2017.

[4] B. Avent, A. Korolova, D. Zeber, T. Hovden, and B. Livshits. Blender: Enabling local search with a hybrid differential privacy model. In *USENIX Security Symposium*, pages 747–764, 2017.

[5] R. Bassily and A. Smith. Local, private, efficient protocols for succinct histograms. In *STOC*, pages 127–135. ACM, 2015.

[6] R. Bassily, U. Stemmer, A. G. Thakurta, et al. Practical locally private heavy hitters. In *NIPS*, pages 2285–2293, 2017.

[7] A. Bittau, U. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnes, and B. Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *SOSP*, pages 441–459. ACM, 2017.

[8] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[9] M. Bun, J. Nelson, and U. Stemmer. Heavy hitters and the structure of local privacy. In *PODS*, pages 435–447. ACM, 2018.

[10] R. Chen, H. Li, A. Qin, S. P. Kasiviswanathan, and H. Jin. Private spatial data aggregation in the local setting. In *ICDE*, pages 289–300. IEEE, 2016.

[11] G. Cormode, S. Jha, T. Kulkarni, N. Li, D. Srivastava, and T. Wang. Privacy at scale: Local differential privacy in practice. In *SIGMOD*, pages 1655–1658. ACM, 2018.

[12] G. Cormode, T. Kulkarni, and D. Srivastava. Marginal release under local differential privacy. In *SIGMOD*, pages 131–146. ACM, 2018.

[13] H. Corrigan-Gibbs and D. Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *NSDI*, pages 259–282, 2017.

[14] B. Ding, J. Kulkarni, and S. Yekhanin. Collecting telemetry data privately. In *NIPS*, pages 3574–3583, 2017.

[15] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *FOCS*, pages 429–438. IEEE, 2013.

[16] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy, data processing inequalities, and statistical minimax rates. *arXiv:1302.3203*, 2013.

[17] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Privacy aware learning. *Journal of the ACM*, 61(6):1–57, 2014.

[18] C. Dwork. Differential privacy. In *ICALP*, pages 1–12. Springer, 2006.

[19] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284. Springer, 2006.

[20] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[21] Ú. Erlingsson, V. Pihur, and A. Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, pages 1054–1067. ACM, 2014.

[22] G. Fanti, V. Pihur, and Ú. Erlingsson. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *PoPETS*, 2016(3):41–61, 2016.

[23] P. Kairouz, K. Bonawitz, and D. Ramage. Discrete distribution estimation under local privacy. In *ICML*, pages 2436–2444. ACM, 2016.

[24] P. Kairouz, S. Oh, and P. Viswanath. Extremal mechanisms for local differential privacy. In *NIPS*, pages 2879–2887, 2014.

[25] P. Kairouz, S. Oh, and P. Viswanath. Extremal mechanisms for local differential privacy. *Journal of Machine Learning Research*, 17(17):1–51, 2016.

[26] J. W. Kim, D.-H. Kim, and B. Jang. Application of local differential privacy to collection of indoor positioning data. *IEEE Access*, pages 4276–4286, 2018.

[27] N. Li, M. Lyu, D. Su, and W. Yang. Differential privacy: From theory to practice. *Synthesis Lectures on Information Security, Privacy, & Trust*, 8(4):1–138, 2016.

[28] N. Li, W. Qardaji, D. Su, and J. Cao. Privbasis: Frequent itemset mining with differential privacy. *PVLDB*, 5(11):1340–1351, 2012.

[29] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30. ACM, 2009.

[30] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103. IEEE, 2007.

[31] T. T. Nguyên, X. Xiao, Y. Yang, S. C. Hui, H. Shin, and J. Shin. Collecting and analyzing data from smart device users with local differential privacy. *arXiv:1606.05053*, 2016.

[32] K. Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242, 1895.

[33] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *CCS*, pages 192–203. ACM, 2016.

[34] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren. Generating synthetic decentralized social graphs with local differential privacy. In *CCS*, pages 425–438. ACM, 2017.

[35] X. Ren, C.-M. Yu, W. Yu, S. Yang, X. Yang, J. A. McCann, and S. Y. Philip. Lopub: High-dimensional crowdsourced data publication with local differential privacy. *IEEE Transactions on Information Forensics and Security*, 2018.

[36] Y. Sei and A. Ohsuga. Differential private data collection and analysis based on randomized multiple dummies for untrusted mobile crowdsensing. *IEEE Transactions on Information Forensics and Security*, 12(4):926–939, 2017.

[37] A. Smith, A. Thakurta, and J. Upadhyay. Is interaction necessary for distributed private learning? In *S&P*, pages 58–77. IEEE, 2017.

[38] S. Wang, L. Huang, P. Wang, H. Deng, H. Xu, and W. Yang. Private weighted histogram aggregation in crowdsourcing. In *WASA*, pages 250–261. Springer, 2016.

[39] T. Wang, J. Blocki, N. Li, and S. Jha. Locally differentially private protocols for frequency estimation. In *USENIX Security Symposium*, pages 729–745, 2017.

[40] T. Wang, N. Li, and S. Jha. Locally differentially private frequent itemset mining. In *S&P*, pages 127–143. IEEE, 2018.

[41] Y. Wang, X. Wu, and D. Hu. Using randomized response for differential privacy preserving data collection. In *EDBT/ICDT Workshops*, 2016.

[42] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[43] X. Yang, T. Wang, X. Ren, and W. Yu. Copula-based multi-dimensional crowdsourced data synthesis and release with local privacy. In *GLOBECOM*, pages 1–6. IEEE, 2017.

APPENDIX

A. Proof of Theorem 4.1

PROOF. (Theorem 4.1) For user u_i, let $\langle k_j, v^* \rangle$ 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.

$$\begin{aligned}
\frac{\Pr(K^*|K_1)}{\Pr(K^*|K_2)} &= \frac{1/d \cdot \Pr(k_j|K_1)}{1/d \cdot \Pr(k_j|K_2)} \\
&\leq \frac{1/d \cdot \Pr(k_j = 1|k_1 = 1)}{1/d \cdot \Pr(k_j = 1|k_2 = 0)} \\
&= \left(\frac{e^{\epsilon_1}}{1 + e^{\epsilon_1}}\right) \Big/ \left(\frac{1}{1 + e^{\epsilon_1}}\right) \\
&= e^{\epsilon_1}
\end{aligned}$$

Thus the perturbation on keys in LPP satisfies ϵ_1-LDP.

For value perturbation, we consider two kinds of possible outputs: empty and non-empty KV pair. As for 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 $\exp(\epsilon_2)$.

$$\begin{aligned}\frac{\Pr(V^*|V_1)}{\Pr(V^*|V_2)} &= \frac{1/d \cdot \Pr(v^*|V_1)}{1/d \cdot \Pr(v^*|V_2)} \\ &= \frac{\frac{1+v_1}{2} \cdot \frac{e^{\epsilon_2}}{1+e^{\epsilon_2}} + \frac{1-v_1}{2} \cdot \frac{1}{1+e^{\epsilon_2}}}{\frac{1+v_2}{2} \cdot \frac{e^{\epsilon_2}}{1+e^{\epsilon_2}} + \frac{1-v_2}{2} \cdot \frac{1}{1+e^{\epsilon_2}}} \\ &\leq \frac{\max_{v_1}\{v_1(e^{\epsilon_2}-1)+e^{\epsilon_2}+1\}}{\min_{v_2}\{v_2(e^{\epsilon_2}-1)+e^{\epsilon_2}+1\}} \\ &= e^{\epsilon_2}\end{aligned}$$

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

$$\begin{aligned}\frac{\Pr(V^*|V_1)}{\Pr(V^*|V_2)} &= \frac{1/d \cdot \Pr(v^*=0|V_1)}{1/d \cdot \Pr(v^*=0|V_2)} \\ &\leq \frac{1/d \cdot \Pr(v^*=0|v_1=0)}{1/d \cdot \Pr(v^*=0|v_2\neq 0)} \\ &= (\frac{e^{\epsilon_2}}{1+e^{\epsilon_2}}) \Big/ (\frac{1}{1+e^{\epsilon_2}}) \\ &= e^{\epsilon_2}\end{aligned}$$

Thus the perturbation on values in LPP satisfies ϵ_2-LDP.

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