The following publication Q. Ye et al., "PrivKVM*: Revisiting Key-Value Statistics Estimation With Local Differential Privacy," in IEEE Transactions on Dependable and Secure Computing, vol. 20, no. 1, pp. 17-35, Jan.-Feb. 2023 is available at https://doi.org/10.1109/TDSC.2021.3107512.

PrivKVM*: Revisiting Key-Value Statistics Estimation with Local Differential Privacy

Qingqing Ye, *Member, IEEE*, Haibo Hu, *Senior Member, IEEE*, Xiaofeng Meng, *Member, IEEE*, Huadi Zheng, Kai Huang, Chengfang Fang, Jie Shi

Abstract—A key factor in big data analytics and artificial intelligence is the collection of user data from a large population. However, the collection of user data comes at the price of privacy risks, not only for users but also for businesses who are vulnerable to internal and external data breaches. To address privacy issues, local differential privacy (LDP) has been proposed to enable an untrusted collector to obtain accurate statistical estimation on sensitive user data (e.g., location, health, and financial data) without actually accessing the true records. As key-value data is an extremely popular NoSQL data model, there are a few works in the literature that study LDP-based statistical estimation on key-value data. However, these works have some major limitations, including supporting small key space only, fixed key collection range, difficulty in choosing an appropriate padding length, and high communication cost. In this paper, we propose a two-phase mechanism $PrivKVM^*$ as an optimized and highly-complete solution to LDP-based key-value data collection and statistics estimation. We verify its correctness and effectiveness through rigorous theoretical analysis and extensive experimental results.

Index Terms—Key-value data, local differential privacy, privacy-preserving data collection, statistics estimation, histogram.

1 Introduction

With the prevalence of big data analytics, businesses become keen to collect data from users, from which they can benefit and even monetize. For instance, Amazon reportedly logs customers' browsing history to predict product sales and manage inventory to avoid backorders [1]. As another example, open banking was mandated in UK from January 2018 to allow FinTech companies and banks to share customers' financial data. PwC forecast that by 2022 the financial industry in UK will gain more than 9 billion from such data [2]. However, the collection of user data comes at the price of privacy risks, not only for users but also for businesses who are vulnerable to internal and external data breaches. As an answer to privacy-preserving data collection, local differential privacy (LDP) [3], [4] 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, LDP has been adopted in mainstream systems for usage data collection, including Apple [5], Google [6] and Microsoft [7].

- *Qingqing Ye, Haibo Hu and Huadi Zheng are with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong, China. Qingqing Ye was also with the School of Information, Renmin University of China, Beijing, China. Haibo Hu is also with Shenzhen Research Institute, The Hong Kong Polytechnic University. Xiaofeng Meng is with the School of Information, Renmin University of China, Beijing, China. Kai Huang is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China. Fangcheng Fang and Jie Shi are with Huawei International, Shanghai, China.*
E-mail: {qqing.ye, haibo.hu}@polyu.edu.hk, xfmeng@ruc.edu.cn, huadi.zheng@connect.polyu.hk, ustkhuang@ust.hk, {fang.chengfang, shi.jie1}@huawei.com

Manuscript received Jul 18, 2020; revised — –, —.

Although LDP has been attempted in social network and graph data analysis [8], [9], [10], the majority of existing work on LDP focuses on collecting some basic statistics, such as frequency estimation over categorical [6], [11], [12], [13] or set values [14], [15], [16], and mean estimation over numerical values [4], [7], [17], [18]. Recently, key-value data has been studied under LDP [19], [20]. As a popular NoSQL data model, key-value data are pervasive in big data analytics, and the following two examples show its potential applications.

- **Internet of things data analysis.** Wearable and internet-of-things (IoT) devices are generating enormous streaming data, most of which are in the form of key-value pairs, i.e., $\langle sensor\_id, sensor\_value\rangle$, or $\langle timestamp, sensor\_value\rangle$. To analyze these data, a fundamental task is to collect the statistics, such as frequency count and mean value, from a particular sensor or during a specific time period.
- **Usage activity analysis.** Software and service providers such as Microsoft and Apple need to collect usage data to improve user experience, for example, the daily number of accesses to a function. These usage data are commonly in the form of key-value pairs where the key is the function identifier and the value is the timestamp of access or frequency of this function being accessed in a time period.

However, such key-value data contain sensitive information about the users, which could disclose their interests, daily activities and other personal particulars. In this paper, we study the problem of privacy-preserving key-value data collection under LDP. The key challenge lies in preserving the inherent correlation between keys and values. As such, applying existing LDP perturbation protocols to keys and values separately can only twist data from their original correlated distributions. To address this problem, in our previous work [19], we propose *PrivKVM* protocol that perturbs keys and values in a correlated manner for frequency and mean estimation. However, *PrivKVM* has two limitations. First, it cannot scale to large key domains, especially when the user population is limited and the "full sampling" strategy used in *PrivKVM* leads to very few samples for each key, causing large estimation variance. Second, it only supports the two basic statistics of key-value data, namely, full-domain key frequency and mean estimation.

In this paper, we propose a two-phase mechanism $PrivKVM^*$ as an optimized and highly-complete solution to LDP-based key-value data collection and statistics estimation. On the one hand, $PrivKVM^*$ adopts phase-wise "adaptive sampling" to address the challenge of large key domain. On the other hand, $PrivKVM^*$ supports more complex statistics such as range query and histogram release, which underpins in-depth analytics on key-value data. The experimental evaluations show that $PrivKVM^*$ outperforms state-of-the-art key-value solutions [19], [20] in almost all datasets and settings, especially for large key domains. To summarize, our technical contributions in this paper are three-folded.

- We propose a generalized value perturbation primitive (GVPP) for numerical value perturbation, which supports a wide range of statistics, including frequency and mean estimation, range queries and histogram release over key-value pairs.
- Based on GVPP, we propose a two-phase solution $PrivKVM^*$ that can well handle large key domain, while achieving better estimation accuracy than all state-of-the-art solutions.
- We rigorously verify the correctness and effectiveness of our solution through both theoretical analysis on privacy and accuracy, and extensive experimental evaluations.

The remainder of this paper is organized as follows. Section 2 reviews the related literature. Section 3 formulates the problem and introduces the preliminaries on LDP. Section 4 briefly reviews the two state-of-the-art solutions for key-value perturbation with LDP. Section 5 presents our proposed solution $PrivKVM^*$. Section 6 elaborates on theoretical analysis on privacy and accuracy. Section 7 presents an extensive set of experimental results. Finally, Section 8 concludes this paper.

2 Related Work

The notion of differential privacy was first introduced by Dwork in [21]. 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 [22], [23]. As for the *local setting*, i.e., scenarios without such a collector, local differential privacy (LDP) [3], [24], [25] was proposed as a new and promising framework for private data collection and analysis.

Since then many LDP techniques are proposed for frequency estimation over categorical data. Erlingsson *et al.* [6] propose *RAPPOR*, which is the first LDP technique for frequency estimation in real-world applications. A follow-up method proposed by Fanti *et al.* [26] extends *RAPPOR* to more complex statistics without explicit dictionary knowledge. As Ranodomized Response (RR) [27] only targets at binary variables, Kairouz *et al.* [11] introduce generalized Randomized Response (*GRR*) to categorical attributes with arbitrary number of possible values. In order to reduce the communication cost, Bassily *et al.* [12] design an efficient protocol *SHist*, which produces a succinct histogram representation of the input data. By choosing optimal parameters, Wang *et al.* [13] propose Optimized Unary Encoding (OUE) and Optimized Local Hashing (OLH) to achieve better estimation accuracy and lower communication cost, respectively.

Other works use frequency estimation as a primitive to protect data privacy in other domains, including heavy hitter identification [12], [13], [16], [28], [29], [30], itemset mining [14], [15], [31], marginal release [32], [33], graph data analysis [8], [9], [34], spatiotemporal data aggregation [35], [36], time-series data collection [37], distribution estimation [38] and range query [39]. For high-dimensional data analysis and learning, there are works on LDP for correlated data collection [40], joint distribution estimation [41], query processing with different types of predicates and aggregation functions [42], [43], and private learning [44], [45], [46], [47].

As a relevant problem to this paper, mean estimation over numerical data with LDP has also been studied in the literature. Duchi *et al.* are the first to propose a solution for this problem, which may incur high computation and space complexity [4]. To address this, an improved method [17] is proposed to perturb any numerical input into a binary output according to a certain probability. More recently, Wang *et al.* [18] propose Piecewise Mechanism (PM) and Hybrid Mechanism (HM) to improve estimation accuracy.

3 Preliminaries and Problem Definition

3.1 Local Differential Privacy

Centralized differential privacy [48], [49] 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 owners. To this end, local differential privacy (LDP) [3] is proposed to let each data owner locally perturb her data using a randomized mechanism, and then to send 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 that is defined on two neighboring datasets that only differ in one record.

TABLE 1
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
Ψ_k^i	mean value of KV pairs with key k in the i-th bucket
Φ_k^i	count of KV pairs with key k in the i-th bucket
g	the number of boundary points
$\boldsymbol{x}$	a set of boundary points, $\boldsymbol{x} = \{x_1, x_2, ..., x_g\}$
$\mathbb{X}$	a set of discretized/perturbed values in GVPP
L	the number of perturbed values in GVPP, $L = \lvert\mathbb{X}\rvert$
δ	the threshold of key frequency in the second phase

As with centralized differential privacy, LDP also has the nice property of sequential composition [50] as below, 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 information from the original data. This guarantees the privacy of our proposed two-phase solution, which will be discussed in Section 6.

3.2 Randomized Response

Randomized response (RR) [27] is a technique developed for interviewees in a survey to give random answer to a sensitive boolean question so that they can achieve plausible deniability while collectively giving away an approximate answer. 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 $\tilde{f}$:

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

Many state-of-the-art LDP solutions, such as *RAPPOR* [6] and *SHist* [12], use RR as the building block and thus convert input data to a binary form.

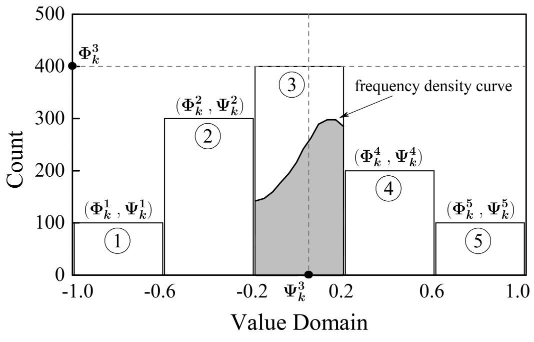


Fig. 1. Illustration of multi-bucket aggregation query

3.3 Problem Definition

This paper studies the problem of distributed statistic estimation 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 1.

An untrusted data collector needs to estimate some statistics of these key-value data from all users. In this paper, we assume the statistics are in the form of a **multi-bucket aggregation query** [51], [52], which has been a popular statistical query type supported by mainstream analytics engine such as Elasticsearch and Apache Spark. The query is defined upon $g-1$ non-overlapping buckets with g boundary points $\boldsymbol{x} = \{x_1, x_2, ..., x_g\} \in [-1, 1]$ in ascending order, where $x_1 = -1$ and $x_g = 1$. The results of a multi-bucket aggregation query on key k, $\boldsymbol{R}_k$, have $g-1$ components $\{R_k^1, R_k^2, ..., R_k^{g-1}\}$, each $R_k^i = (\Phi_k^i, \Psi_k^i)$ denoting the count and mean value of KV pairs of key k that fall in the i-th bucket. Formally,

$$\Phi_k^i = \left|\left\{u_j | \exists\langle k, v\rangle \in S_j, v \in [x_i, x_{i+1}]\right\}\right| \tag{2}$$

$$\Psi_k^i = \frac{\sum_{\mathcal{U}} \sum_{j:k_j=k, v_j \in [x_i, x_{i+1}]} v_j}{\Phi_k^i} \tag{3}$$

Fig. 1 illustrates the physical meaning of this query by an example, where $g = 6$ and the boundary points $\boldsymbol{x} = \{-1, -0.6, -0.2, 0.2, 0.6, 1\}$. Φ_k^i is simply the height of bucket i whereas $x = \Psi_k^i$ is the bisector of area under the frequency density curve.

The multi-bucket aggregation query is a very general type of statistical query, which can be degeneralized to the following three popular statistics, namely, full-domain key frequency and mean, range aggregation query, and histogram release.

- **Full-domain key frequency and mean.** The full-domain frequency f_k of key k is defined as the portion of users who possess any KV pair whose key is k. And the mean Ψ_k is defined as the mean of all values in the full domain whose keys are k. To degeneralize to this statistics, we set $g = 2$ and $\boldsymbol{x} = \{-1, 1\}$, i.e., there is only one bucket $[-1, 1]$. Formally, given querying result $R_k = (\Phi_k, \Psi_k)$,

$$f_k = \frac{\Phi_k}{n} = \frac{|\{u_j | \exists \langle k, v\rangle \in S_j\}|}{n} \tag{4}$$

$$\Psi_k = \frac{\sum_{\mathcal{U}} \sum_{j:k_j=k} v_j}{n \cdot f_k} \tag{5}$$

- **Range aggregation query**. Given key k and a value range $[\alpha, \beta]$ $(-1 < \alpha < \beta < 1)$, the result of the range query is $R_k = (\Phi_k, \Psi_k)$, where Φ_k is the count of KV pairs of key k whose values fall in $[\alpha, \beta]$, and Ψ_k is the mean of these values. To degeneralize to this statistical query, we set $g = 4$ and $\boldsymbol{x} = \{-1, \alpha, \beta, 1\}$. Formally,

$$\Phi_k = \left| \left\{ u_j | \exists \langle k, v\rangle \in S_j, v \in [\alpha, \beta] \right\} \right| \tag{6}$$

$$\Psi_k = \frac{\sum_{\mathcal{U}} \sum_{j:k_j=k, v_j \in [\alpha,\beta]} v_j}{\Phi_k} \tag{7}$$

- **Histogram Release**. Histogram is essentially the same as multi-bucket aggregation without returning the mean values. Without loss of generality, in this paper we assume a histogram is equal-width. To release a b-bucket histogram for key k, we set $g = b + 1$ and $\boldsymbol{x} = \{\frac{2(j-1)}{b} - 1 | 1 \le j \le b + 1\}$. Formally,

$$\Phi_k^i = \left| \left\{ u_j | \exists \langle k, v\rangle \in S_j, v \in t_i \right\} \right| \tag{8}$$

where $t_1 = [-1, \frac{2}{b} - 1]$, and $t_i = (\frac{2(i-1)}{b} - 1, \frac{2i}{b} - 1]$ for $2 \le i \le g$.

4 Existing Key-Value Perturbation with LDP

In this section, we introduce two state-of-the-art LDP perturbation protocols for key-value data. The main challenge of such protocols lies in the preservation of inherent correlation between keys and values, which is essential to key-value data analysis.

4.1 $PrivKVM$

Our previous work *PrivKVM* [19] is the first solution to key-value data perturbation with LDP. *PrivKVM* protects key-value data by adopting a synchronous key and value perturbation protocol, which will be elaborated in Section 5.3. To preserve the correlation between keys and values, *PrivKVM* adopts multiple iterations to collect KV pairs from users, where the results from the last iteration are fed to the next iteration as inputs.

Since each user possesses more than one KV pair, to avoid dividing privacy budget among them, *PrivKVM* adopts a sampling technique [12] for each user to randomly select one index $j \in \{1, 2, ..., |\mathcal{K}|\}$ from the key domain $\mathcal{K}$ before perturbation, whether or not this user possesses this KV pair with j-th key. The perturbed KV pair is then sent to the untrusted data collector. By projecting each user's key set to the entire key domain, *PrivKVM* avoids the information loss of unpopular keys and guarantees the estimation of frequency and mean is unbiased. It has been shown to achieve satisfying estimation accuracy when the key domain is not very large. For large key domain, however, *PrivKVM* may cause high estimation error due to insufficient samples.

4.2 PCKV

More recently, PCKV [20] has been proposed as an alternative key-value perturbation protocol. To address the problem of large key domain in *PrivKVM*, PCKV adapts the Padding-and-Sampling protocol [15] to sample one KV pair from each user's KV set. If a user has fewer than l KV pairs, dummy pairs will be padded until she has l KV pairs. Then, PCKV randomly selects one pair, perturbs it by UE or GRR (namely, PCKV-UE or PCKV-GRR), and sends it to the data collector.

The main challenge of PCKV is an appropriate choice of the padding length l. Theoretically, l should be set according to the data distribution. A small l underestimates the frequency counting of keys while a large l increases the number of dummy KV pairs and thus increases the estimation error. Unfortunately, the optimal choice of l is infeasible as learning the data distribution is the objective of PCKV in the first place.

Besides the above, both variants of PCKV have some other limitations. For PCKV-UE, the communication bandwidth cost is proportional to the key domain size, which can be unacceptable in many real-world applications. As for PCKV-GRR, the estimation accuracy of GRR degrades rapidly with the key domain size, even with an optimization technique called privacy amplification [15]. These issues of PCKV will be demonstrated by experimental evaluation in Section 7.

5 $PrivKVM^*$

In this section, we propose our solution $PrivKVM^*$ to distributed key-value data aggregation with LDP, which addresses the issues of both *PrivKVM* and PCKV. We will first introduce the key idea of $PrivKVM^*$ and then overview its workflow. Then we present the implementation details of each step. Finally, we summarize it with an overall algorithm.

5.1 Key Idea

The design principle of $PrivKVM^*$ is to get the best of both worlds from *PrivKVM* and PCKV. On the one hand, *PrivKVM* inherently causes less noise in the estimation result than PCKV-UE as the former has a sensitivity of 1 whereas the latter has a sensitivity of 2. On the other hand, an advanced sampling protocol like Padding-and-Sampling in PCKV is crucial for large-key-domain scenarios because it can effectively increase valid samples to achieve high estimation accuracy. Unfortunately, the Padding-and-Sampling protocol itself cannot be adapted to *PrivKVM*, because this protocol requires each user to sample a key she possesses, while *PrivKVM* asks a user to sample a key from a uniform key domain, whether or not the key is in her KV sets. In a word, the sampling domains of *PrivKVM* and Padding-and-Sampling protocol are different. In addition, although PCKV with Padding-and-Sampling protocol is still a good alternative for applications with frequent key estimation, for those infrequent ones, there may be very few valid samples for the frequency estimation, which also causes the overall evaluation unsatisfying.

To this end, we design $PrivKVM^*$ as a two-phase solution, which finds frequent keys in the first phase and then estimates the statistics of associated KV pairs in the second phase by a new sampling technique called *adaptive sampling*. The key idea is to tighten the key space for sampling, i.e., more samples for each popular key. In this sense, it effectively combines the advantages of full-domain sampling in *PrivKVM* and Padding-and-Sampling in PCKV. As another enhancement, apart from frequency and full-domain mean estimation, $PrivKVM^*$ is designed to support more statistical tasks, e.g., range aggregation query and histogram release.

5.2 Overview of $PrivKVM^*$

The two-phase workflow of $PrivKVM^*$ is shown in Fig. 2. In the first phase, each user randomly selects a KV pair from the canonical form (as shown in Fig. 3) of her KV set by a "full-domain sampling", i.e., the key is uniformly sampled from the full domain $\mathcal{K}$. Then the selected KV pair is perturbed by taking into accounts of key, value, and key-value correlation. The detailed perturbation protocol will be presented in Section 5.3, and its building block — a generalized value perturbation protocol — will be presented in Section 5.4. At the collector side, upon receiving all users' perturbed KV pairs, the data collector estimates the frequency and mean of each key (see Section 5.5), and executes virtual iterations (see Section 5.6) to derive more accurate mean estimation over all values. Based on the estimated frequency, the collector broadcasts to all users $\mathcal{K}_\delta$, a set of popular keys whose frequencies exceed a threshold δ, and Ψ_δ, the estimated mean of each popular key. The collector also obtains the querying statistics (e.g., range query or histogram) based on the aforementioned frequency and mean estimation over all keys.

In the second phase, instead of full-domain sampling, each user adaptively samples a key from the popular key set $\mathcal{K}_\delta$. Adaptive sampling can obtain more valid KV pairs (i.e., pairs whose keys actually exist) than full-domain sampling, and therefore can obtain more accurate frequency and mean estimation of popular keys. Similar to the first phase, each user then perturbs the selected KV pair, based on which the collector estimates the querying statistics and executes virtual iterations to derive more accurate estimation.

Essentially, the estimated results from the second phase is a subset of those in the first phase, but are more accurate. As such, the former overrides the latter when results in both phases are merged. Furthermore, to improve the accuracy of the latter, we assume unpopular keys have similar value distributions so that the collector can merge the samples of such keys and average the statistics of values among all these unpopular keys. The details of the overall $PrivKVM^*$ algorithm will be presented in Section 5.7.

5.3 A Correlated Perturbation Protocol

In this subsection, we describe Correlated Perturbation Protocol (CPP), the building block of $PrivKVM^*$. It is a protocol that synchronously perturbs key and value to preserve their correlation. CPP itself relies on the generalized value perturbation primitive GVPP, which will be elaborated in Section 5.4. In this subsection, we simply treat GVPP as a blackbox protocol.

As shown in Fig. 3, the first step of CPP is to convert a user u_i's KV pair set S_i to its *canonical* form S_i', which contains the full key universe $\mathcal{K}$. For each key $k \in \mathcal{K}$, if there is pair $\langle k, v\rangle \in S_i$, we convert it to $\langle 1, v\rangle$ in S_i'; otherwise the key does not exist in S_i, we add an empty key pair $\langle 0, 0\rangle$ to S_i'. The canonical form guarantees each user possesses the same number of KV pairs.

Now that all KV pairs are either $\langle 1, v\rangle$ or $\langle 0, 0\rangle$, we can apply RR to perturb keys, which are now binary. Specifically, we flip key "1" (resp. "0") to "0" (resp. "1") with a probability specified by RR. As such, the data collector cannot determine whether a user possesses a key or not. The perturbation on values is based on the perturbation results of keys, which have the following four cases:

- $\langle 1, v\rangle \rightarrow \langle 1, \tilde{v}\rangle$: An existing key is preserved after the key perturbation. In this case, value v is perturbed to $\tilde{v}$ by the generalized value perturbation primitive GVPP (see Section 5.4).
- $\langle 1, v\rangle \rightarrow \langle 0, 0\rangle$: An existing key disappears after the key perturbation. In this case, we simply set the value to 0 to be consistent with the next case and thus hide the trace of key perturbation.
- $\langle 0, 0\rangle \rightarrow \langle 0, 0\rangle$: A non-existent key is still non-existent after the key perturbation. In this case, the whole KV pair remains unchanged.
- $\langle 0, 0\rangle \rightarrow \langle 1, \tilde{v}\rangle$: A new key appears after the key perturbation. In this case, to ensure unbiasedness, we first set its value to be the current mean $\widehat{\Psi}$, and then perturb it by GVPP, i.e., $\tilde{v} \leftarrow GVPP(\widehat{\Psi})$.

Algorithm 1 Correlated Perturbation Protocol (CPP)

```
Input:      User $u_i$'s set of KV pairs $S_i$
            Key set $\mathcal{K}$
            Privacy budget $\epsilon$
            Current range means $\widehat{\mathbf{\Psi}}$ for each key and each bucket
Output:     $CPP(S_i, \mathcal{K}, \epsilon, \widehat{\Psi})$ is the perturbed KV pair
Procedure:
1: Sample a key $k$ uniformly from $\mathcal{K}$, where $j$ is the key index
2: if $k$ exists in the key set of $S_i$ then
3:     Perturb its value $v$: $\tilde{v} = GVPP(v, \epsilon/2)$
4:     Perturb $\langle k, \tilde{v}\rangle$ as:
           $\langle k, \tilde{v}\rangle = \begin{cases} \langle 1, \tilde{v}\rangle & \text{w.p. } \frac{e^{\epsilon/2}}{1+e^{\epsilon/2}} \\ \langle 0, 0\rangle & \text{w.p. } \frac{1}{1+e^{\epsilon/2}} \end{cases}$
5: else
6:     Sample a $\widehat{\Psi}_k^i$ from $\widehat{\mathbf{\Psi}}_k = \{\widehat{\Psi}_k^1, ..., \widehat{\Psi}_k^{g-1}\}$ uniformly at random
7:     Perturb the sampled value: $\tilde{v} = GVPP(\widehat{\Psi}_k^i, \epsilon/2)$
8:     Perturb $\langle k, \tilde{v}\rangle$ as:
           $\langle k, \tilde{v}\rangle = \begin{cases} \langle 0, 0\rangle & \text{w.p. } \frac{e^{\epsilon/2}}{1+e^{\epsilon/2}} \\ \langle 1, \tilde{v}\rangle & \text{w.p. } \frac{1}{1+e^{\epsilon/2}} \end{cases}$
9: return $j$ and $\langle k, \tilde{v}\rangle$
```

The pseudo-code of CPP is described in Algorithm 1. It takes as inputs the set of KV pairs of a user S_i, the key set $\mathcal{K}$, the privacy budget ϵ, and the current means $\widehat{\mathbf{\Psi}}$ for each key and each bucket, and returns a perturbed KV pair. The first step is to let each user randomly sample a KV pair $\langle k, v\rangle$ for reporting (Line 1). Depending on whether or not $\mathcal{K}$ is the full key domain, in phase 1 CPP adopts "full-domain sampling" and in phase 2 it adopts "adaptive sampling". If the user

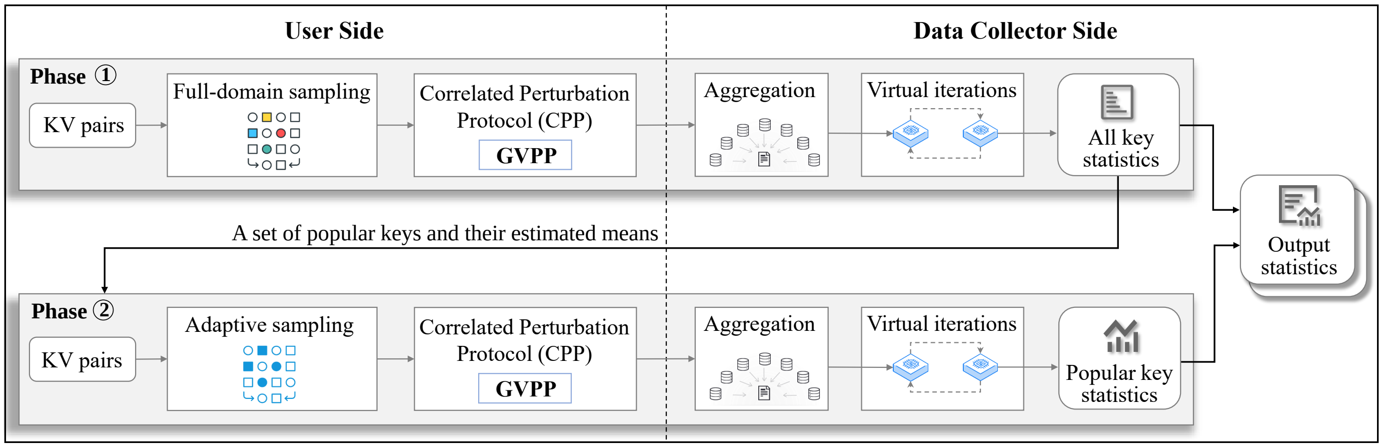


Fig. 2. An overview of $PrivKVM^*$

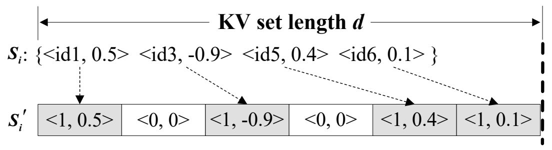


Fig. 3. Conversion of user's set of KV pairs

has this KV pair, she first perturbs the value v into $\tilde{v}$ by GVPP (Line 3), then converts the pair $\langle k, \tilde{v}\rangle$ into canonical form $\langle 1, \tilde{v}\rangle$, and finally perturbs it to $\langle 1, \tilde{v}\rangle$ (resp. $\langle 0, 0\rangle$) with probability $\frac{e^{\epsilon/2}}{1+e^{\epsilon/2}}$ (resp. $\frac{1}{1+e^{\epsilon/2}}$) (Line 4). The perturbation is similar if the user does not have this KV pair, except that the value $\widehat{\Psi}_k^i$ is randomly selected from current means $\{\widehat{\Psi}_k^1, ..., \widehat{\Psi}_k^{g-1}\}$ of $g-1$ buckets before being perturbed (Lines 6-8). Since the perturbed value always depends on the key, CPP effectively preserves the correlation between keys and values. Note that in CPP, the current means $\widehat{\boldsymbol{\Psi}}$ in the first phase are initialized locally by each user (see Lines 3-4 in Algorithm 5), which incurs no communication cost. In the second phase, only those means of popular keys $\mathcal{K}_\delta$ are sent back by the data collector to each user, so the communication cost is limited to $O((g-1)|\mathcal{K}_\delta|)$.

5.4 Generalized Value Perturbation Primitive

Numerical value perturbation primitive (VPP) for mean estimation has been widely studied in LDP literature [7], [17]. The key idea is to discretize a numerical value to a binary one and then perturb it by RR to satisfy ϵ-LDP. Algorithm 2 shows the major steps: discretization, perturbation and calibration. After discretization (Line 1), the value can only be 1 or -1, which is suitable to apply RR to perturb it (Line 2). As the perturbation causes the mean estimation to be biased, the perturbed value must be calibrated before being sent to the data collector (Line 3).

However, the above VPP protocol only works for value domain $[-1, 1]$, and cannot be applied to arbitrary range, which is needed for the multi-bucket aggregation query. In this subsection, we develop a Generalized Value Perturbation Primitive (GVPP) for multiple buckets specified by a set of boundary points $\boldsymbol{x} = \{x_1, x_2, ..., x_g\}$.

Algorithm 2 Value Perturbation Primitive (VPP)

```
Input:      An original value $v \in [-1, 1]$
            Privacy budget $\epsilon$
Output:     Perturbed value $\tilde{v}$
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:
       $\tilde{v} = \begin{cases} v^* & \text{w.p. } \frac{e^\epsilon}{1+e^\epsilon} \\ -v^* & \text{w.p. } \frac{1}{1+e^\epsilon} \end{cases}$
3: Calibration:
       $\tilde{v} = \tilde{v} \cdot \frac{e^\epsilon + 1}{e^\epsilon - 1}$
4: return $\tilde{v}$
```

Discretization. For each value $v \in [-1, 1]$, GVPP first finds its enclosing interval $[x_i, x_{i+1}]$, such that $x_i < v \leq x_{i+1}$.[1] Then for this value v, GVPP discretizes it to v^* as

$$\Pr[v^*|v] = \frac{1}{x_{i+1} - x_i} \begin{cases} x_{i+1} - v & \text{if } v^* = x_i \\ v - x_i & \text{if } v^* = x_{i+1} \end{cases} \tag{9}$$

Note that $\mathbb{E}[v^*] = x_i \cdot \frac{x_{i+1}-v}{x_{i+1}-x_i} + x_{i+1} \cdot \frac{v-x_i}{x_{i+1}-x_i} = \mathbb{E}[v]$, therefore v^* is an unbiased estimation of v.

After discretization, each value is transformed to one of boundary points. Note that for each internal boundary point $x_i \in \{x_2, x_3, ..., x_{g-1}\}$, there are two intervals whose values may be discretized to it, i.e., $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$. To guarantee an unbiased mean estimation for each interval, we need to distinguish these two. We use x_i^- (resp. x_i^+) to denote a value in interval $[x_{i-1}, x_i]$ (resp. $[x_i, x_{i+1}]$) being discretized to x_i. Therefore, there are $2(g-1)$ possible discretized values, denoted by $\mathbb{X} = \{x_1^+, x_2^-, x_2^+, ..., x_{g-1}^-, x_{g-1}^+, x_g^-\}$. In the sequel, we always use $L = |\mathbb{X}|$ to denote the cardinality of $\mathbb{X}$. So for the multi-bucket aggregation query, $L = 2(g-1)$.

Perturbation. Then GVPP perturbs a discretized value. Here we adopt two existing methods, i.e., Generalized Randomized Response (GRR) [11] and Unary Encoding (UE) (a.k.a., basic RAPPOR) [6], which are sub-optimal in the

1. If $x_i = -1$, the condition is $x_i \leq v \leq x_{i+1}$.

high and low privacy budget ϵ scenarios, respectively [53]. In what follows, we first show how to apply GRR or UE, and then derive a hybrid scheme of both to reduce the estimation error.

In GRR, a discretized value $v^* \in \mathbb{X}$ is perturbed to $\tilde{v} \in \mathbb{X}$ as

$$\Pr[\tilde{v}|v^*] = \frac{1}{L-1+e^\epsilon}\begin{cases} e^\epsilon & \text{if } \tilde{v} = v^* \\ 1 & \text{if } \tilde{v} \neq v^* \end{cases} \tag{10}$$

In UE, a discretized value $v^* \in \mathbb{X}$ is first encoded into an L-bit binary vector b^*, where only the index position of v^* in $\mathbb{X}$ is 1, and all other bits are 0. Then we can perturb each bit $b_j^* \in b^*$ to $\tilde{v}_j \in \tilde{v}$ as

$$\Pr[\tilde{v}_j = 1|b_j^*] = \begin{cases} p^* & \text{if } b_j^* = 1 \\ q^* & \text{if } b_j^* = 0 \end{cases} \tag{11}$$

where $p^* = \frac{1}{2}$ and $q^* = \frac{1}{1+e^\epsilon}$, which was proved as an optimal setting [13] to minimize the estimation variance. Note that the perturbed output $\tilde{v}$ in Eq. 11 is a binary vector, while in Eq. 10 it is a scalar.

According to [53], GRR and UE are suitable for various ϵ ranges. Therefore, we propose a hybrid perturbation strategy as follows. If $\epsilon \geq \ln(L/2)$, GVPP-GRR achieves better estimation accuracy than GVPP-UE, and is thus adopted. Otherwise, GVPP-UE is adopted. The pseudo-code of complete GVPP procedure is described in Algorithm 3.

Algorithm 3 Generalized Value Perturbation Primitive (GVPP)

```
Input:      An original value $v$
            Privacy budget $\epsilon$
Output:     the perturbed result $\tilde{v}$
Procedure:
1: Derive $L$ from query type and the set of boundary points $\boldsymbol{x} = \{x_1, x_2, ..., x_g\}$.
2: Find an interval $[x_i, x_{i+1}]$ such that $x_i < v \leq x_{i+1}$
3: Discretize $v$ to $v^*$ as Eq. 9
4: if $\epsilon \geq \ln(L/2)$ then
5:     Perturb $v^*$ to $\tilde{v}$ as Eq. 10, with privacy budget $\epsilon$
6: else
7:     Encode $v^*$ and perturb to $\tilde{v}$ as Eq. 11, with privacy budget $\epsilon$
8: return $\tilde{v}$
```

In Line 1, the GVPP algorithm derives L. So far we consider the general form, i.e., multi-bucket aggregation query only, so $L = 2(g-1)$. For the specific statistical queries listed in Section 3.3, we can derive more suitable L for them as follows.

- **Frequency and mean estimation.** For this query, $g = 2$ and $\boldsymbol{x} = \{-1, 1\}$. Since the perturbed value set $\mathbb{X} = \{-1, 1\}$, $L = |\mathbb{X}| = 2$. Obviously, $\epsilon \geq \ln(L/2) = 0$ always holds. So GVPP-GRR is always adopted.
- **Range aggregation query.** For a range aggregation query in $[\alpha, \beta](-1 < \alpha < \beta < 1)$, $g = 4$ and $\boldsymbol{x} = \{-1, \alpha, \beta, 1\}$. Since intervals $[-1, \alpha]$ and $(\beta, 1]$ do not contribute to the query results, we can merge them and further reduce the perturbed value set $\{-1^+, \alpha^-, \alpha^+, \beta^-, \beta^+, 1^-\}$ to $\mathbb{X} = \{\alpha^+, \beta^-, x_{others}\}$. As such, $L = 3$.
- **Histogram release.** For a histogram with b bins, we have $g = b + 1$ and $\boldsymbol{x} = \{\frac{2(i-1)}{b} - 1 | 1 \leq i \leq b+1\}$. Since this query only estimates counts, not mean values, of bins, we can directly set the perturbed value set $\mathbb{X} = \{1, 2, ..., b\}$, one for each bin.

5.5 Collector-Side Aggregation

Upon receiving all perturbed KV pairs from users, the data collector first calibrates the frequency of each key and their associated values, and then aggregates them according to the statistical query type.

Calibration. For each key k, as the observed frequency is biased, the collector calibrates it to $\tilde{f}_k$ as in Eq. 1. As for the values associated with key k, the observed counts of all perturbed value, i.e., $\widetilde{\mathbb{C}} = \{\tilde{c}_1^+, \tilde{c}_2^-, \tilde{c}_2^+, ..., \tilde{c}_{g-1}^-, \tilde{c}_{g-1}^+, \tilde{c}_g^-\}$, are also biased. The data collector needs to calibrate them into unbiased ones $\mathbb{C} = \{c_1^+, c_2^-, c_2^+, ..., c_{g-1}^-, c_{g-1}^+, c_g^-\}$.

For GVPP-GRR, according to Eq. 10, the observed count $\tilde{c} \in \widetilde{\mathbb{C}}$ of each boundary point can be estimated by

$$\tilde{c} = \frac{c \cdot e^\epsilon}{L-1+e^\epsilon} + \frac{n_k - c}{L-1+e^\epsilon} \tag{12}$$

where n_k is the number of sampled users involved in value perturbation for each key k, and c is the count of a given boundary point after value discretization. By solving c from Eq. 12, we derive the calibration as

$$c = \frac{(L-1+e^\epsilon)\tilde{c} - n_k}{e^\epsilon - 1} \tag{13}$$

Note that all such calibrated c constitute $\mathbb{C}$.

For GVPP-UE, according to Eq. 11, each observed count $\tilde{c} \in \widetilde{\mathbb{C}}$ can be estimated by

$$\tilde{c} = \frac{1}{2}c + (n_k - c)\frac{1}{1+e^\epsilon} \tag{14}$$

By solving c from Eq. 14, we can derive the calibration as

$$c = \frac{2(e^\epsilon+1)\tilde{c} - 2n_k}{e^\epsilon - 1} \tag{15}$$

Aggregation. For each key k, after we calibrate the observed counts $\widetilde{\mathbb{C}}$, we can estimate the count and mean value of KV pairs that fall in the i-th bucket, i.e., $\widetilde{\Phi}_k^i$ and $\widetilde{\Psi}_k^i$. Recall that a key may appear after key perturbation, with its mean value assigned by the current mean of a bucket. As such, the count of any key k is incremented by $\frac{n(1-f_k)(1-p)}{g-1}$, where f_k is the real frequency of key k and can be approximated by the estimated one $\tilde{f}_k$, and $p = \frac{e^{\epsilon/2}}{1+e^{\epsilon/2}}$ is the key's the perturbation probability. Then it is further inflated (i.e., scaled up) by n/n_k due to sampling. Specifically,

$$\Phi_k^i \cdot p + \frac{n(1-\tilde{f}_k)(1-p)}{g-1} = (c_i^+ + c_{i+1}^-) \cdot \frac{n}{n_k} \tag{16}$$

Then the count of the i-th bucket (i.e., $(x_i, x_{i+1}]$) of key k can be estimated by solving Φ_k^i from Eq. 16 as

$$\widetilde{\Phi}_k^i = \frac{(c_i^+ + c_{i+1}^-)n}{n_k \cdot p} - \frac{n(1-\tilde{f}_k)(1-p)}{(g-1)\cdot p} \tag{17}$$

On the other hand, the mean value of the i-th bucket of key k can be simply estimated by the value counts in c_i^+ and c_{i+1}^-. Specifically,

$$\widetilde{\Psi}_k^i = \frac{x_i \cdot c_i^+ + x_{i+1} \cdot c_{i+1}^-}{c_i^+ + c_{i+1}^-} \tag{18}$$

The accuracy guarantee of the above estimation will be analyzed and proved by Theorem 6.2 in Section 6.

The pseudo-code of collector-side aggregation is described in Algorithm 4. Based on all users' perturbed KV pairs, the data collector first calculates the frequency of each key $\tilde{f}_k$ and then calibrates it according to Eq. 1 (Line 2). As for value calibration, if the received data is perturbed by GVPP-GRR ($\epsilon/2 \geq \ln(L/2)$, where key and value consumes $\epsilon/2$ respectively), the algorithm first counts all L possible perturbed values $\{x_1^+, x_2^-, x_2^+, ..., x_{g-1}^-, x_{g-1}^+, x_g^-\}$ and then calibrates it according to Eq. 13 (Lines 5-6); otherwise the received data is perturbed by GVPP-UE, and for each bit $\tilde{v}_j (1 \leq j \leq L)$, the algorithm counts the number of $\tilde{v}_j = 1$ in all perturbed vectors and then calibrates it according to Eq. 15 (Lines 9-10). After the calibration, we can estimate the count and mean value of each of $g-1$ buckets by Eqs. 17 and 18 respectively (Line 12).

Algorithm 4 Collector-Side Aggregation

```
Input:      A set of perturbed KV pairs $S'$
            The set of keys $\mathcal{K}$
            Privacy budgets $\epsilon$
            Number of boundary points $g$
Output:     Querying results $\widetilde{\boldsymbol{f}}, \widetilde{\boldsymbol{\Phi}}, \widetilde{\boldsymbol{\Psi}} = Aggre(S', \mathcal{K}, \epsilon, g, c)$
Procedure:
1:  for key $k \in \mathcal{K}$ do
2:      Collector calculates frequency $\tilde{f}_k$ and calibrate it as Eq. 1
3:      if $\epsilon/2 \geq \ln(L/2)$ then
4:          //$\tilde{v}$ is a scalar perturbed by GVPP-GRR, $L = 2(g-1)$
5:          for value $\tilde{v} \in \mathbb{X} = \{x_1^+, x_2^-, x_2^+, ..., x_{g-1}^-, x_{g-1}^+, x_g^-\}$ do
6:              Calculate the count of $\tilde{v}$ and calibrates it as Eq. 13
7:      else
8:          //$\tilde{v}$ is a $L$-bit binary vector perturbed by GVPP-UE
9:          for $i \in \{1, 2, ..., L\}$ do
10:             Calculate the count of $\tilde{v}_j = 1$ and calibrates it as Eq. 15
11:     for $i \in \{1, 2, g-1\}$ do
12:         Calculate count $\widetilde{\Phi}_k^i$ and mean $\widetilde{\Psi}_k^i$ as Eqs. 17 and 18
13: return  Frequency $\widetilde{\boldsymbol{f}}$, count $\widetilde{\boldsymbol{\Phi}}$ and mean $\widetilde{\boldsymbol{\Psi}}$
```

5.6 An Iterative Model with Virtual Iterations

There is a remaining problem in CPP to determine the current means of values. Obviously, the closer the current mean to the ground truth, the more accurate this protocol. It has been proved that using an approximation of the real mean as this initialization makes the estimated mean becomes unbiased [19].

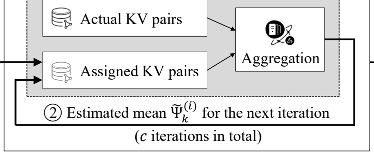

Actual KV pairs
Assigned KV pairs
Aggregation
① Initialized mean $\widetilde{\Psi}_k^{(0)}$
② Estimated mean $\widetilde{\Psi}_k^{(i)}$ for the next iteration
(c iterations in total)
③ Estimated mean $\widetilde{\Psi}_k^{(c)}$

Fig. 4. An iterative model for mean estimation

In this subsection, we show how to apply an iterative model to derive a good approximation of this mean. As shown in Fig. 4, for each key k, the iterative model takes as input an initialized mean $\widetilde{\Psi}_k^{(0)}$ (step ①), and consists of multiple iterations, where the estimated means of the previous iteration becomes the input of this iteration (step ②). Since actual KV pairs are always involved in each iteration, estimation can gradually approach the ground truth. After c iterations, this iterative model releases the estimated mean $\widetilde{\Psi}_k^{(c)}$ (step ③).

The following two theorems show the accuracy guarantee of this iterative model for mean estimation. Specifically, we first provide Theorem 5.1 to show that the estimated mean after c iterations goes to the true mean as c increases if we ignore the perturbation noise. Further, when considering the effect of perturbation, Theorem 5.2 shows that the expectation of the estimated mean $\widetilde{\Psi}_k^{(c)}$ converges to the true mean as c increases. The proof of Theorem 5.2 is similar to the one in [19] and is therefore omitted.

Theorem 5.1. Let Ψ_k denote the true mean of values in any range $[\alpha, \beta] (-1 \leq \alpha \leq \beta \leq 1)$, and $\widetilde{\Psi}_k^{(c)}$ denote the estimated one with c iterations. When ignoring the effect of perturbation noise, we have $\lim_{c \to \infty} |\Psi_k - \widetilde{\Psi}_k^{(c)}| = 0$.

PROOF. For each key k, let $\widetilde{\Psi}_k^{(0)}$ be the initialized mean, n be the number of users and f_k be the frequency. The estimated mean from the c-iteration is

$$\begin{aligned}\widetilde{\Psi}_k^{(c)} &= \frac{\sum_{i=1}^{nf_k} v_i + \sum_{j=1}^{n(1-f_k)} \widetilde{\Psi}_k^{(c-1)}}{n} \\ &= \frac{nf_k \cdot \Psi_k + n(1-f_k) \cdot \widetilde{\Psi}_k^{(c-1)}}{n} \\ &= f_k \cdot \Psi_k + (1-f_k) \cdot \widetilde{\Psi}_k^{(c-1)}\end{aligned}$$

Then we have

$$\begin{aligned}|\Psi_k - \widetilde{\Psi}_k^{(c)}| &= (1-f_k) \cdot |\Psi_k - \widetilde{\Psi}_k^{(c-1)}| \\ &= (1-f_k)^c \cdot |\Psi_k - \widetilde{\Psi}_k^{(0)}|\end{aligned}$$

Note that $|\Psi_k - \widetilde{\Psi}_k^{(0)}|$ is a constant and $0 < 1 - f_k < 1$, hence we have $\lim_{c \to \infty} |\Psi_k - \widetilde{\Psi}_k^{(c)}| = 0$. □

Theorem 5.2. Let Ψ_k denote the true mean of values in any range $[\alpha, \beta] (-1 \leq \alpha \leq \beta \leq 1)$, and $\widetilde{\Psi}_k^{(c)}$ denote the estimated one by CPP with c iterations, then $\lim_{c \to \infty} |\Psi_k - \mathbb{E}[\widetilde{\Psi}_k^{(c)}]| = 0$.

However, this iterative model has two disadvantages. First, since users need to report key-value pair in each iteration, according to sequential composition (Theorem 3.1), the privacy budget must be split and allocated to each iteration, which may incur extra noise to the final estimated value. Second, multiple iterations also lead to high communication bandwidth. To address these issues, we propose to adopt the virtual iteration technique [19]. Specifically, after the first "real" iteration that involves all users, the data collector executes "virtual" iterations to directly predict the estimated mean without user involvement, i.e., $c-1$ iterations of the step ② in Fig. 4 are executed virtually. As such, it effectively reduces the network bandwidth to only one iteration and avoid privacy budget split and allocation to virtual iterations.

The virtual iteration technique works as follows. It first chooses an initial mean value as $\widetilde{\Psi}_k^{(0)}$, using simple mean or some domain knowledge. $\widetilde{\Psi}_k^{(1)}$ is set to the estimated mean after the first (real) iteration, and then the estimated mean after c (one real and $c-1$ virtual) iterations is determined as

$$\Psi_k^{(c)} = \widetilde{\Psi}_k^{(0)} + \frac{(\widetilde{\Psi}_k^{(1)} - \widetilde{\Psi}_k^{(0)})(1-\theta^c)}{1-\theta} \tag{19}$$

where $\theta = \frac{n(1-f_k)(1-p)}{n(1-f_k)(1-p)+(g-1)\Phi_k \cdot p}$, $p = \frac{e^{\epsilon/2}}{1+e^{\epsilon/2}}$ and Φ_k is the count of values in $[\alpha, \beta]$. In Theorem 6.3 of Section 6.2, we will prove that by adopting Eq. 19 for virtual iterations, the estimated mean is the same as that estimated from real iterations in terms of expectation.

While virtual iterations can reduce communication bandwidth and improve the accuracy of mean estimation in most cases, we admit it has some limitations. First, it only works for mean estimation. Second, as can be seen from Eq. 19, the accuracy gain by virtual iterations heavily relies on that of the first iteration. That is, if the first iteration is given only a very small privacy budget (e.g., $\epsilon \leq 0.1$), the overwhelming noise introduced in the first iteration could accumulate through the virtual iterations. To address this, we can either reduce the number of virtual iterations or change virtual iterations to real ones as in *PrivKVM* [19] when the overall privacy budget is tight.

5.7 Overall Algorithm

Algorithm 5 summarizes the workflow of $PrivKVM^*$ in two phases. In the first phase, $PrivKVM^*$ initializes the mean of each bucket to the average of the upper and the lower bound of that bucket $\widehat{\Psi}^i = (x_i + x_{i+1})/2$ (Lines 3-4). Then each user invokes the CPP protocol with "full-domain sampling", and perturbs her KV set with privacy budget $\epsilon/2$ before sending it to the data collector (Line 5). Note that for simplicity we just divide the total privacy budget ϵ equally for both phases. Based on the perturbed KV pairs S' returned from users, the collector calibrates and aggregates to obtain estimation on frequency of all keys $\widetilde{\boldsymbol{f}}$, count $\widetilde{\boldsymbol{\Phi}}$ and mean value $\widetilde{\boldsymbol{\Psi}}$ for any key k and bucket i (Line 6), and further execute c virtual iterations to derive a more accurate mean $\widetilde{\boldsymbol{\Psi}}$ (Line 7). After that, $PrivKVM^*$ prepares for the second phase by accepting only popular keys (i.e., those whose estimated frequency exceeds the threshold δ) to a reduced key set $\mathcal{K}_\delta$. This key set, together with the corresponding estimated mean $\widehat{\boldsymbol{\Psi}}_\delta$ of $g-1$ buckets for each key, is broadcast to all users for the second phase (Lines 8-11).

In the second phase, each user invokes CPP protocol again to perturb her KV set with the "adaptive sampling" of key set $\mathcal{K}_\delta$ and $\widehat{\boldsymbol{\Psi}}_\delta$ as the current means (Line 14). Then the data collector calibrates, aggregates and executes virtual iterations to obtain the same statistics estimation as in the first phase (Lines 15-16), except that they are for the reduced key set and override those in the first phase (Lines 17-18). For the rest of keys, the data collector averages their statistics in the first phase to avoid noisy estimation due to insufficient samples (Lines 19-23).

Comparing with *PrivKVM*, as $PrivKVM^*$ adopts different key domains in two phases, it is able to achieve better performance than *PrivKVM* in most real-world applications where there exist popular keys whose frequencies are much higher than the others. However, since $PrivKVM^*$ uses half of the privacy budget in the second phase to get more accurate estimation on frequent keys (whose frequencies exceed a given threshold), this strategy is not effective if the frequencies of all keys are either very small (i.e., smaller than the threshold) or very large (i.e., larger than the threshold). For the former, the second phase is not triggered, so $PrivKVM^*$ degrades to $PrivKVM$ except that only half of the privacy budget is used for perturbation. For the latter, although the second phase is triggered, the key domain is the same as the first phase, which is equivalent to executing $PrivKVM$ twice with half of the privacy budget. In both cases and their less extreme variants, $PrivKVM^*$ may not achieve better performance. Nevertheless, for these cases, we can also properly set a frequency threshold so that $PrivKVM^*$ can outperform $PrivKVM$.

It is noteworthy that the extension of $PrivKVM^*$ to other similar data type is straightforward. For example, two-dimensional data which should consider their data correlation, can be treated as key-value data and perturb it by $PrivKVM^*$. For multi-dimensional (i.e., more than two) data, a similar manner can be applied, except that the given privacy budget should be allocated to these dimensions and the perturbation algorithm needs to consider their correlations, i.e., the perturbation of a dimension should be based on the perturbation results of other dimensions.

Algorithm 5 $PrivKVM^*$

```
Input:     Users' sets of KV pairs $\{S_1, S_2, ..., S_n\}$
           The set of keys $\mathcal{K}$
           Privacy budgets $\epsilon$
           A threshold of frequency $\delta$
           A set of boundary points $\boldsymbol{x} = \{x_1, x_2, ..., x_g\}$
           Number of virtual iterations in each phase $c$
Output:    Count and mean estimation results $\boldsymbol{R} = \{\boldsymbol{R}_1, ..., \boldsymbol{R}_{|\mathcal{K}|}\}$,
           where $\boldsymbol{R}_k = \{R_k^1, ..., R_k^{g-1}\}$
Procedure:
1:  // The first phase
2:  $\widehat{\Psi} = \emptyset$, $\widehat{\Psi}_\delta = \emptyset$, $\mathcal{K}_\delta = \emptyset$
3:  for $i \in \{1, 2, ..., g-1\}$ do
4:      Initialize range mean $\widehat{\Psi}^i = (x_i + x_{i+1})/2$
5:  Each user $u_i$ perturbs her KV set, i.e., $CPP(S_i, \mathcal{K}, \epsilon/2, \widehat{\boldsymbol{\Psi}})$, and
    sends the perturbed data to the collector
6:  Collector-side aggregation: $\widetilde{\boldsymbol{f}}, \widetilde{\boldsymbol{\Phi}}, \widetilde{\boldsymbol{\Psi}} = Aggre(S', \mathcal{K}, \epsilon/2, g, c)$,
    where $S'$ is the perturbed KV set from all users
7:  Collector executes $c$ virtual iterations to refine mean $\widetilde{\boldsymbol{\Psi}}$ as Eq. 19
8:  for key $k \in \mathcal{K}$ do
9:      if $\tilde{f}_k > \delta$ then
10:         $\mathcal{K}_\delta \leftarrow \mathcal{K}_\delta \cup k$
11:         $\widehat{\Psi}_\delta \leftarrow \widehat{\Psi}_\delta \cup \widetilde{\Psi}_k$, where $\widetilde{\Psi}_k = \{\widetilde{\Psi}_k^1, \widetilde{\Psi}_k^2, ..., \widetilde{\Psi}_k^{g-1}\}$
12: Collector broadcasts $\mathcal{K}_\delta$ and $\widehat{\boldsymbol{\Psi}}_\delta$ to all users

13: // The second phase
14: Each user $u_i$ perturbs her KV set, i.e., $CPP(S_i, \mathcal{K}_\delta, \epsilon/2, \widehat{\boldsymbol{\Psi}}_\delta)$, and
    sends the perturbed data to the collector
15: Collector-side aggregation: $\widetilde{\boldsymbol{f}}', \widetilde{\boldsymbol{\Phi}}', \widetilde{\boldsymbol{\Psi}}' = Aggre(S', \mathcal{K}_\delta, \epsilon/2, g, c)$,
    where $S'$ is the perturbed KV set from all users
16: Collector executes $c$ virtual iterations to refine mean $\widetilde{\boldsymbol{\Psi}}'$ as Eq. 19
17: for key $k \in \mathcal{K}_\delta$ do
18:     $\tilde{f}_k = \tilde{f}'_k$, $\widetilde{\Phi}_k = \widetilde{\Phi}'_k$, $\widetilde{\Psi}_k = \widetilde{\Psi}'_k$
19: for key $k \in \mathcal{K}/\mathcal{K}_\delta$ do
20:     $\tilde{f}_k = \frac{1}{|\mathcal{K}|-|\mathcal{K}_\delta|} \sum_{k \in \mathcal{K}/\mathcal{K}_\delta} \tilde{f}_k$
21:     for $i \in \{1, 2, g-1\}$ do
22:         $\widetilde{\Phi}_k^i = \frac{1}{|\mathcal{K}|-|\mathcal{K}_\delta|} \sum_{k \in \mathcal{K}/\mathcal{K}_\delta} \widetilde{\Phi}_k^i$
23:         $\widetilde{\Psi}_k^i = \frac{1}{|\mathcal{K}|-|\mathcal{K}_\delta|} \sum_{k \in \mathcal{K}/\mathcal{K}_\delta} \widetilde{\Psi}_k^i$
24: for key $k \in \mathcal{K}$ do
25:     $\boldsymbol{R}_k = \{R_k^1, R_k^2, ..., R_k^{g-1}\}$, where $R_k^i = (\widetilde{\Phi}_k^i, \widetilde{\Psi}_k^i)$
26: return $\boldsymbol{R}$
```

6 Privacy and Accuracy Analysis

In this section, we provide theoretical analysis on privacy and accuracy of $PrivKVM^*$. In particular, we will first show the privacy and accuracy guarantee of GVPP for value perturbation, followed by the correctness proof of virtual iterations. Finally, the privacy analysis of $PrivKVM^*$ is provided.

6.1 Analysis of GVPP

For GVPP protocol, Theorems 6.1 and 6.2 below establish the privacy and accuracy guarantee, respectively.

Theorem 6.1. Given privacy budget ϵ and a set of boundary points $\{x_1, x_2, ..., x_g\}$, GVPP protocol satisfies ϵ-LDP.

Proof. Recall that for an input value $v \in [-1, 1]$, GVPP protocol first discretizes it into one of $2(g-1)$ values, and then perturbs the discretized value by GRR or UE. Below we separately prove the privacy guarantee of GVPP-GRR and GVPP-UE protocols.

(1) **GVPP-GRR**. It perturbs v into one of $L = 2(g-1)$ values in $\mathbb{X} = \{x_1^+, x_2^-, x_2^+, ..., x_{g-1}^-, x_{g-1}^+, x_g^-\}$. The probability of observing any perturbed value $\tilde{v} \in \mathbb{X}$ given v is denoted by $\Pr(\tilde{v}|v)$. Without loss of generality, we only show below the derivation when $\tilde{v} = x_i^+$, as the derivation is similar if $\tilde{v} = x_i^-$. To derive $\Pr(\tilde{v} = x_i^+|v)$, we consider two cases, namely, $v \in (x_i, x_{i+1}]$ and $v \notin (x_i, x_{i+1}]$.

Case 1. If $v \in (x_i, x_{i+1}]$, then

$$\begin{aligned}\Pr(\tilde{v} = x_i^+|v) &= \frac{x_{i+1} - v}{x_{i+1} - x_i} \cdot \frac{e^\epsilon}{L-1+e^\epsilon} + \frac{v - x_i}{x_{i+1} - x_i} \cdot \frac{1}{L-1+e^\epsilon} \\ &= \frac{(1 - e^\epsilon)v + x_{i+1}e^\epsilon - x_i}{(x_{i+1} - x_i)(L - 1 + e^\epsilon)}\end{aligned}$$

Obviously, $\Pr(\tilde{v} = x_i^+|v)$ is monotonically decreasing with respect to $v \in (x_i, x_{i+1}]$. Therefore,

$$\Pr(\tilde{v} = x_i^+|v)_{max} < \frac{(1 - e^\epsilon)x_i + x_{i+1}e^\epsilon - x_i}{(x_{i+1} - x_i)(L - 1 + e^\epsilon)} = \frac{e^\epsilon}{L - 1 + e^\epsilon}$$

$$\Pr(\tilde{v} = x_i^+|v)_{min} = \frac{(1 - e^\epsilon)x_{i+1} + x_{i+1}e^\epsilon - x_i}{(x_{i+1} - x_i)(L - 1 + e^\epsilon)} = \frac{1}{L - 1 + e^\epsilon}$$

Case 2. If $v \notin (x_i, x_{i+1}]$, and $v \in (x_j, x_{j+1}]$ where $j \neq i$, then

$$\begin{aligned}\Pr(\tilde{v} = x_i^+|v) &= \frac{x_{j+1} - v}{x_{j+1} - x_j} \cdot \frac{1}{L-1+e^\epsilon} + \frac{v - x_j}{x_{j+1} - x_j} \cdot \frac{1}{L-1+e^\epsilon} \\ &= \frac{1}{L - 1 + e^\epsilon}\end{aligned}$$

Therefore, for any two input values $v_1, v_2 \in [-1, 1]$, and any possible output $\tilde{v} \in \mathbb{X}$, the ratio of two conditional probabilities $\Pr(\tilde{v}|v_1)$ and $\Pr(\tilde{v}|v_2)$ is

$$\frac{\Pr(\tilde{v}|v_1)}{\Pr(\tilde{v}|v_2)} \leq \frac{\Pr(\tilde{v}|v)_{max}}{\Pr(\tilde{v}|v)_{min}} < \frac{e^\epsilon/(L - 1 + e^\epsilon)}{1/(L - 1 + e^\epsilon)} = e^\epsilon$$

As such, GVPP-GRR protocol satisfies ϵ-LDP.

(2)**GVPP-UE**. It encodes v by a $2(g-1)$-bit binary vector b^*, and then perturbs b^* into $\tilde{b}$ by UE. Given any input v, the probability of observing $\tilde{b}_i$, the i-th bit in $\tilde{b}$, is denoted by $\Pr(\tilde{b}_i|v)$. Without loss of generality, we assume now $\tilde{b}_i = 1$. To derive $\Pr(\tilde{b}_i = 1|v)$, we consider two cases, namely, $v \in (x_i, x_{i+1}]$ and $v \notin (x_i, x_{i+1}]$.

Case 1. If $v \in (x_i, x_{i+1}]$, then

$$\begin{aligned}\Pr(\tilde{b}_i = 1|v) &= \frac{x_{i+1} - v}{x_{i+1} - x_i} \cdot \frac{1}{2} + \frac{v - x_i}{x_{i+1} - x_i} \cdot \frac{1}{1 + e^\epsilon} \\ &= \frac{(1 + e^\epsilon)x_{i+1} - 2x_i - (e^\epsilon - 1)v}{2(1 + e^\epsilon)(x_{i+1} - x_i)}\end{aligned}$$

$$\Pr(\tilde{b}_i = 1|v)_{min} = \frac{(1+e^\epsilon)x_{i+1} - 2x_i - (e^\epsilon - 1)x_{i+1}}{2(1 + e^\epsilon)(x_{i+1} - x_i)} = \frac{1}{1+e^\epsilon}$$

$$\Pr(\tilde{b}_i = 1|v)_{max} < \frac{(1 + e^\epsilon)x_{i+1} - 2x_i - (e^\epsilon - 1)x_i}{2(1 + e^\epsilon)(x_{i+1} - x_i)} = \frac{1}{2}$$

Case 2. If $v \notin (x_i, x_{i+1}]$, and $v \in (x_j, x_{j+1}]$ where $j \neq i$, then

$$\begin{aligned}\Pr(\tilde{b}_i = 1|v) &= \frac{x_{j+1} - v}{x_{j+1} - x_j} \cdot \frac{1}{1 + e^\epsilon} + \frac{v - x_j}{x_{j+1} - x_j} \cdot \frac{1}{1 + e^\epsilon} \\ &= \frac{1}{1 + e^\epsilon}\end{aligned}$$

Therefore, for any two input values $v_1, v_2 \in [-1, 1]$, and any possible perturbed vector $\tilde{b}$, the ratio of two conditional probabilities $\Pr(\tilde{b}|v_1)$ and $\Pr(\tilde{b}|v_2)$ is

$$\begin{aligned}\frac{\Pr(\tilde{b}|v_1)}{\Pr(\tilde{b}|v_2)} &= \frac{\prod_{i=1}^{2(g-1)} \Pr(\tilde{b}_i|v_1)}{\prod_{i=1}^{2(g-1)} \Pr(\tilde{b}_i|v_2)} = \frac{\Pr(\tilde{b}_i|v_1)\Pr(\tilde{b}_j|v_1)}{\Pr(\tilde{b}_i|v_2)\Pr(\tilde{b}_j|v_2)} \\ &\leq \frac{\Pr(\tilde{b}_i = 0|v_1)\Pr(\tilde{b}_j = 1|v_1)_{max}}{\Pr(\tilde{b}_i = 0|v_2)\Pr(\tilde{b}_j = 1|v_2)_{min}} \\ &= \frac{(1 - \frac{1}{1+e^\epsilon}) \cdot \frac{1}{2}}{(1 - \frac{1}{2}) \cdot \frac{1}{1+e^\epsilon}} = e^\epsilon\end{aligned}$$

As such, GVPP-UE protocol also satisfies ϵ-LDP. □

Theorem 6.2. Given privacy budget ϵ and a set of boundary points $\boldsymbol{x} = \{x_1, x_2, ..., x_g\}$, for KV pairs with key k and values in the range $(x_i, x_{i+1}]$, the estimated count by GVPP (i.e., Eq. 17) is unbiased, i.e., $\mathbb{E}[\widetilde{\Phi}_k^i] = \Phi_k^i$, and the expectation of estimated mean $\mathbb{E}[\widetilde{\Psi}_k^i]$ can be approximated by $\Psi_k^i(1 + 2\gamma) - (x_i + x_{i+1})\gamma$, where $\gamma = \frac{2g-6+e^\epsilon}{(e^\epsilon-1)^2(\Phi_k^i \cdot \frac{p}{n} + \frac{(1-f_k)(1-p)}{g-1})^2 \cdot n_k}$, and their variances are

$$Var[\widetilde{\Phi}_k^i] = \frac{n^2(L - 2 + e^\epsilon)}{n_k^4(e^\epsilon - 1)^2 p^2} - \frac{n(1 - p)^3}{(g - 1)^2 p(2p - 1)^2}$$

$$Var[\widetilde{\Psi}_k^i] \approx \frac{2(2g - 6 + e^\epsilon)(1 - (x_i + x_{i+1})\Psi_k^i + (\Psi_k^i)^2)}{(\Phi_k^i \cdot \frac{p}{n} + \frac{(1-f_k)(1-p)}{g-1})^2(e^\epsilon - 1)^2 n_k}.$$

where p is the key's perturbation probability.

Proof. For ease of presentation, we assume GVPP-GRR throughout this proof, and GVPP-UE can be proved in a similar manner. After key perturbation, the count of values falling in the i-th bucket $(x_i, x_{i+1}]$ can be estimated as

$$c^* = \Phi_k^i \cdot \frac{n_k}{n} \cdot p + \frac{n_k(1 - f_k)(1 - p)}{g - 1} \quad (20)$$

Note that the first item in Eq. 20 comes from the sampled users who really possess the key k, while the second item is due to assigned ones.

Recall that in GVPP-GRR, $v \in (x_i, x_{i+1}]$ is first discretized into x_i^+ or x_{i+1}^-. Among all n_k sampled users for key k, let $N(x_i^+)$ and $N(x_{i+1}^-)$ be the numbers of x_i^+ and x_{i+1}^- respectively after key perturbation. Formally,

$$\begin{aligned}\mathbb{E}[N(x_i^+)] &= \sum_{v\in(x_i,x_{i+1}]} \frac{x_{i+1}-v}{x_{i+1}-x_i}\\ &= \frac{c^*\cdot x_{i+1} - \sum_{v\in(x_i,x_{i+1}]} v}{x_{i+1}-x_i}\end{aligned}$$

Similarly,

$$\mathbb{E}[N(x_{i+1}^-)] = \frac{\sum_{v\in(x_i,x_{i+1}]} v - c^*\cdot x_i}{x_{i+1}-x_i}.$$

After perturbation by GVPP-GRR, the expectation of the observed count of x_i^+ is

$$\mathbb{E}[\tilde{c}_i^+] = \frac{\mathbb{E}[N(x_i^+)]\cdot e^\epsilon}{L-1+e^\epsilon} + \frac{n_k - \mathbb{E}[N(x_i^+)]}{L-1+e^\epsilon}.$$

and n_k is the number of samples for key k.

Then after calibration by Eq. 13, the expectation of the calibrated counts of x_i^+ is

$$\mathbb{E}[c_i^+] = \frac{(L-1+e^\epsilon)\mathbb{E}[\tilde{c}_i^+] - n_k}{e^\epsilon - 1} = \mathbb{E}[N(x_i^+)]$$

Similarly, we can also derive $\mathbb{E}[c_{i+1}^-] = \mathbb{E}[N(x_{i+1}^-)]$. Therefore, the expectation of the estimated count is

$$\begin{aligned}\mathbb{E}[\widetilde{\Phi}_k^i] &= \mathbb{E}[\frac{(c_i^+ + c_{i+1}^-)n}{n_k\cdot p} - \frac{n(1-\tilde{f}_k)(1-p)}{(g-1)\cdot p}]\\ &= (\mathbb{E}[N(x_i^+)] + \mathbb{E}[N(x_{i+1}^-)])\frac{n}{n_k\cdot p} - \frac{n(1-p)}{(g-1)p}\mathbb{E}[1-\tilde{f}_k]\\ &= c^*\cdot\frac{n}{n_k\cdot p} - \frac{n(1-f_k)(1-p)}{(g-1)p}\\ &= \Phi_k^i\end{aligned}$$

As such, the estimated count by GVPP-GRR is unbiased.

Further, the variance of the estimated count of GVPP-GRR can be derived as

$$\begin{aligned}Var[\widetilde{\Phi}_k^i] &= Var[\frac{(c_i^+ + c_{i+1}^-)n}{n_k\cdot p} - \frac{n(1-\tilde{f}_k)(1-p)}{(g-1)\cdot p}]\\ &= \frac{n^2}{n_k^2p^2}\cdot\frac{L-2+e^\epsilon}{n_k^2(e^\epsilon-1)^2} - \frac{n^2(1-p)^2}{(g-1)^2p^2}\cdot\frac{p(1-p)}{n(2p-1)^2}\\ &= \frac{n^2(L-2+e^\epsilon)}{n_k^4(e^\epsilon-1)^2p^2} - \frac{n(1-p)^3}{(g-1)^2p(2p-1)^2}\end{aligned}$$

Now we prove the expectation of the estimated mean. To simply the notation, let $\lambda = c_i^+ + c_{i+1}^-$ and $\mu = x_i\cdot c_i^+ + x_{i+1}\cdot c_{i+1}^-$, then we have

$$\begin{aligned}\mathbb{E}[\lambda] &= \mathbb{E}[c_i^+] + \mathbb{E}[c_{i+1}^-]\\ &= \mathbb{E}[N(x_i^+)] + \mathbb{E}[N(x_{i+1}^-)]\\ &= c^*\\ \mathbb{E}[\mu] &= x_i\cdot\mathbb{E}[c_i^+] + x_{i+1}\cdot\mathbb{E}[c_{i+1}^-]\\ &= x_i\cdot\mathbb{E}[N(x_i^+)] + x_{i+1}\cdot\mathbb{E}[N(x_{i+1}^-)]\\ &= \sum_{v\in(x_i,x_{i+1}]} v\end{aligned}$$

By applying the multivariate Taylor Expansions to Eq. 18, the expectation of the estimated mean is

$$\begin{aligned}\mathbb{E}[\widetilde{\Psi}_k^i] &= \mathbb{E}[\frac{\mu}{\lambda}] \approx \frac{\mathbb{E}[\mu]}{\mathbb{E}[\lambda]} - \frac{Cov_{\mu,\lambda}}{\mathbb{E}[\lambda]^2} + \frac{\mathbb{E}[\mu]}{\mathbb{E}[\lambda]^3}\cdot Var[\lambda]\\ &= \frac{\sum_{v\in(x_i,x_{i+1}]} v}{\Phi_k^i\cdot\frac{n_k}{n}\cdot p + \frac{n_k(1-f_k)(1-p)}{g-1}}\\ &\quad - \frac{(x_i+x_{i+1})(2g-6+e^\epsilon)n_k}{(e^\epsilon-1)^2(\Phi_k^i\cdot\frac{n_k}{n}\cdot p + \frac{n_k(1-f_k)(1-p)}{g-1})^2}\\ &\quad + \frac{\sum_{v\in(x_i,x_{i+1}]} v}{(\Phi_k^i\cdot\frac{n_k}{n}\cdot p + \frac{n_k(1-f_k)(1-p)}{g-1})^3}\cdot\frac{2g-6+e^\epsilon}{(e^\epsilon-1)^2}2n_k\\ &= \Psi_k^i(1+2\gamma) - (x_i+x_{i+1})\gamma,\end{aligned}$$

where $\gamma = \frac{2g-6+e^\epsilon}{(e^\epsilon-1)^2(\Phi_k^i\cdot\frac{p}{n}+\frac{(1-f_k)(1-p)}{g-1})^2\cdot n_k}$.

Further, the variance of the estimated mean is

$$\begin{aligned}Var[\widetilde{\Psi}_k^i] &= Var[\frac{\mu}{\lambda}] \approx \frac{Var[\mu]}{\mathbb{E}[\lambda]^2} - \frac{2\mathbb{E}[\mu]Cov_{\mu,\lambda}}{\mathbb{E}[\lambda]^3} + \frac{\mathbb{E}[\mu]^2Var[\lambda]}{\mathbb{E}[\lambda]^4}\\ &= \frac{2(2g-6+e^\epsilon)n_k}{(\Phi_k^i\cdot\frac{n_k}{n}\cdot p + \frac{n_k(1-f_k)(1-p)}{g-1})^2(e^\epsilon-1)^2}\\ &\quad - \frac{\sum_{v\in(x_i,x_{i+1}]} v\cdot(x_i+x_{i+1})2(2g-6+e^\epsilon)n_k}{(\Phi_k^i\cdot\frac{n_k}{n}\cdot p + \frac{n_k(1-f_k)(1-p)}{g-1})^3(e^\epsilon-1)^2}\\ &\quad + \frac{(\sum_{v\in(x_i,x_{i+1}]} v)^2}{(\Phi_k^i\cdot\frac{n_k}{n}\cdot p + \frac{n_k(1-f_k)(1-p)}{g-1})^4}\cdot\frac{2(2g-6+e^\epsilon)n_k}{(e^\epsilon-1)^2}\\ &= \frac{2(2g-6+e^\epsilon)\cdot(1-(x_i+x_{i+1})\Psi_k^i + (\Psi_k^i)^2)}{(\Phi_k^i\cdot\frac{p}{n} + \frac{(1-f_k)(1-p)}{g-1})^2(e^\epsilon-1)^2n_k}\end{aligned}$$

□

6.2 Analysis of Virtual Iterations

The following Theorem 6.3 provides the correctness proof of virtual iterations. That is, the estimated mean is the same as that estimated from real iterations in terms of expectation.

Theorem 6.3. In virtual iterations, the expectation of the estimated mean of any values in range $[\alpha,\beta](-1\le\alpha\le\beta)$ with key k after the c-th iteration $\mathbb{E}[\Psi_k^{(c)}]$ is the same as $\mathbb{E}[\widetilde{\Psi}_k^{(c)}]$, the expectation after c real iterations. Formally,

$$\mathbb{E}[\Psi_k^{(c)}] = \mathbb{E}[\widetilde{\Psi}_k^{(c)}] = \widetilde{\Psi}_k^{(0)} + \frac{(\mathbb{E}[\widetilde{\Psi}_k^{(1)}] - \widetilde{\Psi}_k^0)(1-\theta^c)}{1-\theta},$$

where $\theta = \frac{n(1-f_k)(1-p)}{n(1-f_k)(1-p)+(g-1)\Phi_k\cdot p}$ and p is the key's perturbation probability. .

PROOF. Given privacy budget ϵ, the estimated range mean of the r-th real iteration $\widetilde{\Psi}_k^{(r)}$ is based on $\widetilde{\Psi}_k^{(r-1)}$ from the $(r-1)$-th iteration, and the real range mean Ψ_k, or formally

$$\mathbb{E}[\widetilde{\Psi}_k^{(r)}] = \frac{\frac{n(1-f_k)}{g-1}(1-p)\cdot\mathbb{E}[\widetilde{\Psi}_k^{(r-1)}] + \Phi_k\cdot p\cdot\Psi_k}{\frac{n(1-f_k)}{g-1}(1-p) + \Phi_k\cdot p}$$

where Φ_k and Ψ_k are the real count and mean of values in $[\alpha,\beta]$, respectively. Therefore, we have

$$\frac{|\Psi_k - \mathbb{E}[\widetilde{\Psi}_k^{(r)}]|}{|\Psi_k - \mathbb{E}[\widetilde{\Psi}_k^{(r-1)}]|} = \frac{\frac{n(1-f_k)}{g-1}(1-p)}{\frac{n(1-f_k)}{g-1}(1-p) + \Phi_k\cdot p}$$

Then the bias of any two consecutive iterations is

$$|\mathbb{E}[\widetilde{\Psi}_k^{(r)}] - \mathbb{E}[\widetilde{\Psi}_k^{(r-1)}]| = \frac{\Phi_k \cdot p \cdot |\Psi_k - \mathbb{E}[\widetilde{\Psi}_k^{(r-1)}]|}{\frac{n(1-f_k)}{g-1}(1-p) + \Phi_k \cdot p}$$

$$|\mathbb{E}[\widetilde{\Psi}_k^{(r+1)}] - \mathbb{E}[\widetilde{\Psi}_k^{(r)}]| = \frac{\Phi_k \cdot p \cdot |\Psi_k - \mathbb{E}[\widetilde{\Psi}_k^{(r)}]|}{\frac{n(1-f_k)}{g-1}(1-p) + \Phi_k \cdot p}$$

Therefore, the ratio of bias can be derived as:

$$\begin{aligned}\frac{|\mathbb{E}[\widetilde{\Psi}_k^{(r+1)}] - \mathbb{E}[\widetilde{\Psi}_k^{(r)}]|}{|\mathbb{E}[\widetilde{\Psi}_k^{(r)}] - \mathbb{E}[\widetilde{\Psi}_k^{(r-1)}]|} &= \frac{|\Psi_k - \mathbb{E}[\widetilde{\Psi}_k^{(r)}]|}{|\Psi_k - \mathbb{E}[\widetilde{\Psi}_k^{(r-1)}]|} \\ &= \frac{\frac{n(1-f_k)}{g-1}(1-p)}{\frac{n(1-f_k)}{g-1}(1-p) + \Phi_k \cdot p} \\ &= \frac{n(1-f_k)(1-p)}{n(1-f_k)(1-p) + (g-1)\Phi_k \cdot p}\end{aligned}$$

To simplify the notation, we use $\theta = \frac{n(1-f_k)(1-p)}{n(1-f_k)(1-p)+(g-1)\Phi_k \cdot p}$ to denote this ratio. Then we derive a geometric progression $\{\mathbb{E}[\widetilde{\Psi}_k^{(r+1)}] - \mathbb{E}[\widetilde{\Psi}_k^{(r)}]\}$ and the general formula can be written as:

$$\mathbb{E}[\widetilde{\Psi}_k^{(r+1)}] - \mathbb{E}[\widetilde{\Psi}_k^{(r)}] = (\mathbb{E}[\widetilde{\Psi}_k^{(1)}] - \widetilde{\Psi}_k^{(0)}) \cdot \theta^r$$

With r varying from 0 to $c-1$, we have:

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

By summing up the above equations, we have:

$$\mathbb{E}[\widetilde{\Psi}_k^{(c)}] = \widetilde{\Psi}_k^{(0)} + \frac{(\mathbb{E}[\widetilde{\Psi}_k^{(1)}] - \widetilde{\Psi}_k^{(0)})(1-\theta^c)}{1-\theta}$$

□

6.3 Analysis of $PrivKVM^*$

The following two theorems provide the privacy and accuracy guarantee of $PrivKVM^*$. Specifically, Theorems 6.4 shows $PrivKVM^*$ achieves ϵ-LDP, and Theorem 6.5 further provides the estimation variance of $PrivKVM^*$.

Theorem 6.4. Given privacy budget ϵ, $PrivKVM^*$ satisfies ϵ-LDP.

PROOF. For user u who possesses a set of KV pairs S, let $\langle\tilde{k}, \tilde{v}\rangle$ be the sampled and perturbed KV pair by CPP. In each phase, key and value perturbations consume half of the given privacy budget and satisfy $\frac{\epsilon}{4}$- or $\frac{\epsilon}{4}$-LDP, respectively. Let $\Pr(\langle\tilde{k}, \tilde{v}\rangle|S)$ denote the probability of observing a perturbed KV pair $\langle\tilde{k}, \tilde{v}\rangle$ given a set of KV pairs S.

In the first phase, "full-domain sampling" randomly selects a KV pair from the full domain, denoted by $\langle k_1, v_1\rangle$ and $\langle k_2, v_2\rangle$, from S_1 and S_2 respectively. According to Theorem 6.1,

$$\begin{aligned}\frac{\Pr(\langle\tilde{k}, \tilde{v}\rangle|S_1)}{\Pr(\langle\tilde{k}, \tilde{v}\rangle|S_2)} &= \frac{1/d \cdot \Pr(\langle\tilde{k}, \tilde{v}\rangle|\langle k_1, v_1\rangle)}{1/d \cdot \Pr(\langle\tilde{k}, \tilde{v}\rangle|\langle k_2, v_2\rangle)} \\ &= \frac{\Pr(\tilde{k}|k_1)}{\Pr(\tilde{k}|k_2)} \cdot \frac{\Pr(\tilde{v}|v_1)}{\Pr(\tilde{v}|v_2)} \\ &\le e^{\epsilon/4} \cdot e^{\epsilon/4} = e^{\epsilon/2}\end{aligned}$$

In the second phase, "adaptive sampling" randomly selects a KV pair from the reduced key domain in $\mathcal{K}_\delta$, whose estimated frequencies exceed δ. According to Theorem 6.1,

$$\begin{aligned}\frac{\Pr(\langle\tilde{k}, \tilde{v}\rangle|S_1)}{\Pr(\langle\tilde{k}, \tilde{v}\rangle|S_2)} &= \frac{1/d_\delta \cdot \Pr(\langle\tilde{k}, \tilde{v}\rangle|\langle k_1, v_1\rangle)}{1/d_\delta \cdot \Pr(\langle\tilde{k}, \tilde{v}\rangle|\langle k_2, v_2\rangle)} \\ &= \frac{\Pr(\tilde{k}|k_1)}{\Pr(\tilde{k}|k_2)} \cdot \frac{\Pr(\tilde{v}|v_1)}{\Pr(\tilde{v}|v_2)} \\ &\le e^{\epsilon/4} \cdot e^{\epsilon/4} = e^{\epsilon/2}\end{aligned}$$

where $d_\delta = |\mathcal{K}_\delta|$ denotes the number of keys in $\mathcal{K}_\delta$.

Therefore, according to the sequential composition in Theorem 3.1, $PrivKVM^*$ with two-phase perturbation satisfies ϵ-LDP. □

The following Theorem 6.5 shows the variance of the mean estimation by $PrivKVM^*$. As it is related to the random initialized mean, here for simplicity we just assume it to be 0.

Theorem 6.5. By virtual iterations with initialized mean 0, for any range $(x_i, x_{i+1}] \in [-1, 1]$, the variance of estimated mean $\widetilde{\Psi}_k^{(c)}$ by $PrivKVM^*$ can be approximated as $\frac{2(2f_kp-f_k-p+1)^2(2g-6+e^{\epsilon/4})(1-(x_i+x_{i+1})\Psi_k^i+(\Psi_k^i)^2)}{(\Phi_k^i\frac{p}{n}+\frac{(1-f_k)(1-p)}{g-1})^2(e^{\epsilon/4}-1)^2 n_k f_k^2 p^2}$, where $p = \frac{e^{\epsilon/4}}{1+e^{\epsilon/4}}$ and ϵ is the given privacy budget.

PROOF. By setting the initialized mean of a key to 0, we have

$$\begin{aligned}Var[\widetilde{\Psi}_k^{(c)}] &= Var[\widetilde{\Psi}_k^{(0)} + \frac{(\widetilde{\Psi}_k^{(1)} - \widetilde{\Psi}_k^{(0)})(1-\theta^c)}{1-\theta}] \\ &= Var[\widetilde{\Psi}_k^{(1)} \cdot \frac{1-\theta^c}{1-\theta}]\end{aligned}$$

From the multivariate Taylor Expansions of functions of random variables [54], the variance of product of any two random variables X and Y can be approximated by

$$\begin{aligned}Var[XY] \approx\ &\mathbb{E}[Y]^2 \cdot Var[X] + 2\mathbb{E}[X]\mathbb{E}[Y] \cdot Cov(X, Y) \\ &+ \mathbb{E}[X]^2 \cdot Var[Y]\end{aligned}$$

For convenience, let $X = \widetilde{\Psi}_k^{(1)}$ and $Y = \frac{1-\theta^c}{1-\theta}$, then according to the proof of Theorem 6.2, we have

$$\begin{aligned}\mathbb{E}[X] = \mathbb{E}[\widetilde{\Psi}_k^{(1)}] &\approx \frac{\sum_{v\in(x_i,x_{i+1}]} v}{\Phi_k^i \frac{n_k}{n} p + \frac{n_k(1-f_k)(1-p)}{g-1}} \\ &= \frac{\Phi_k^i \cdot \Psi_k (g-1) n}{\Phi_k^i (g-1)p + n(1-f_k)(1-p)} \\ \mathbb{E}[Y] = \mathbb{E}[\frac{1-\theta^c}{1-\theta}] &< \mathbb{E}[\frac{1}{1-\theta}] \approx \frac{2f_kp - f_k - p + 1}{f_kp} \\ Var[Y] = Var[\frac{1-\theta^c}{1-\theta}] &\approx c^2 Var[\frac{\ln\theta}{1-\theta}]\end{aligned}$$

Therefore, the variance of the estimated mean is

$$\begin{aligned}Var[\widetilde{\Psi}_k^{(c)}] \approx\ &\mathbb{E}[\frac{1-\theta^c}{1-\theta}]^2 \cdot Var[\widetilde{\Psi}_k^{(1)}] \\ &+ 2\mathbb{E}[\widetilde{\Psi}_k^{(1)}]\mathbb{E}[\frac{1-\theta^c}{1-\theta}] \cdot Cov(\widetilde{\Psi}_k^{(1)}, \frac{1-\theta^c}{1-\theta}) \\ &+ \mathbb{E}[\widetilde{\Psi}_k^{(1)}]^2 \cdot Var[\frac{1-\theta^c}{1-\theta}] \\ \approx\ &\frac{2(2f_kp-f_k-p+1)^2(2g-6+e^{\epsilon/4})(1-(x_i+x_{i+1})\Psi_k^i+(\Psi_k^i)^2)}{(\Phi_k^i\frac{p}{n}+\frac{(1-f_k)(1-p)}{g-1})^2(e^{\epsilon/4}-1)^2 n_k f_k^2 p^2}\end{aligned}$$

□

TABLE 2
Statistics of datasets

Datasets	Users	Keys	Avg. frequency	Var. of frequency	Avg. mean	Var. of mean
GAUSS	10^6	100	0.1463	0.0775	-0.4766	0.2517
PLAW	10^6	100	0.0307	0.0115	-0.8698	0.0548
TalkingData	60,822	306	0.0693	0.0255	-0.6780	0.2480
JData	105,180	442	0.0251	0.0065	-0.6410	0.2690
Clothing	105,508	5,850	3.1139×10^{-4}	6.4646×10^{-7}	0.7513	0.0355
Amazon	1,210,271	249,274	6.7058×10^{-6}	8.8794×10^{-10}	0.5746	0.2742

7 Experimental Evaluation

In this section, we evaluate the performance of our proposed solution $PrivKVM^*$ against existing LDP solutions for key-value data.

7.1 Experimental Setting

We introduce our experimental setting in this subsection, including experiment design, datasets and performance measurements.

Experiment design. We design two sets of experiments. The first set estimates the full-domain key frequency and mean, which is supported by $PrivKVM^*$ and all existing LDP solutions, namely, *PrivKVM* [19], PCKV-UE [20] and PCKV-GRR [20]. For PCKV-UE and PCKV-GRR, we set the padding length l to the average number of KV pairs of each user, which is the most favorable setting [20]. The second set estimates statistics of range query and histogram release, which is only supported by $PrivKVM^*$. For comparison purpose, we also replace our value perturbation protocol GVPP in $PrivKVM^*$ with Square Wave (SW) mechanism [38] and Piecewise Mechansim (PM) [18], the state-of-the-art solutions for LDP-based distribution and mean estimation over numerical values, which lead to $PrivKVM^*$*-SW* and $PrivKVM^*$*-PM*, respectively. They are then compared with $PrivKVM^*$-$GVPP$.

Datasets. We use six datasets, whose statistics regarding their key frequencies and value means are summarized in Table 2. GAUSS and PLAW are synthetic datasets whose keys, values, and KV pair numbers all follow Gaussian and power-law distributions, respectively. The other four are real-world key-value datasets obtained from public sources, among which *TalkingData* and *JData* have been used in [19], and *Clothing* and *Amazon* have been used in [20]. The value domains of all datasets are normalized to $[-1, 1]$.

- **TalkingData mobile events.**[2] This dataset is collected from TalkingData SDK which has been used by many mobile apps. It contains 60,822 users (devices) and 306 categories of apps. There are 32,473,067 events in this dataset, each of which records one app category of a certain device accessing the SDK. For each user, we treat a category as a key, and the number of events associated with this category and device as a value.
- **JData shopping records**[3]**.** This dataset is collected from JD.COM. It contains 50,601,736 sales records in 2016, each of which is a list of product brands a user has ever purchased. There are 105,180 users and 442 brands in total. For each user, we treat a brand as a key, and the number of records associated with this brand and user as a value.
- **Clothing fit ratings**[4]**.** This dataset is collected from a E-Commerce website *RentTheRunWay* for women to rent clothes. It contains 192,544 self-reported reviews and ratings from customers. For each user, we treat a product category as a key, and the associated rating as a value.
- **Amazon beauty product ratings**[5]**.** This is a dataset of 2 million customer reviews of beauty-related products sold on Amazon.com. For each user, we treat each product as key, and the associated rating as a value.

According to the average number of users per key (i.e., $\frac{\#(\text{user})}{\#(\text{key})}$), these datasets can be categorized into three groups — dense population (GAUSS and PLAW), medium population (TalkingData and JData) and sparse population (Clothing and Amazon).

Performance measurements. To measure the accuracy of the estimated frequency and mean, we use **Mean Square Error** (MSE) that is widely used in the literature. It measures the mean square error of estimated frequency (resp. means) with respect to real frequency (resp. means) of all keys. Specifically, for $k \in \mathcal{K}$, let f_k and $\tilde{f}_k$ denote real and estimated frequency, Φ_k and $\widetilde{\Phi}_k$ denote real and estimated counts of values in $[\alpha, \beta]$, and Ψ_k and $\widetilde{\Psi}_k$ denote real and estimated mean of values in $[\alpha, \beta]$, and $d = |\mathcal{K}|$.

$$MSE_f = \frac{1}{d} \sum_{k \in \mathcal{K}} (f_k - \tilde{f}_k)^2$$
$$MSE_\Phi = \frac{1}{d} \sum_{k \in \mathcal{K}} (\Phi_k - \widetilde{\Phi}_k)^2$$
$$MSE_\Psi = \frac{1}{d} \sum_{k \in \mathcal{K}} (\Psi_k - \widetilde{\Psi}_k)^2$$

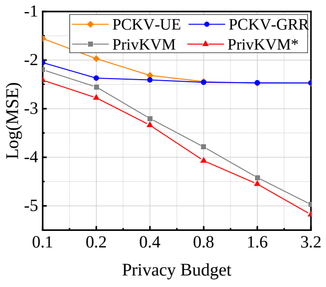
As the absolute MSE has very large variance, we use logarithm of MSE, i.e., $Log(MSE)$, in the sequel.

We conduct experiments on a desktop computer with Intel Core i9-9900K 3.60 GHz CPU, 64G RAM running Windows 10 Enterprise.

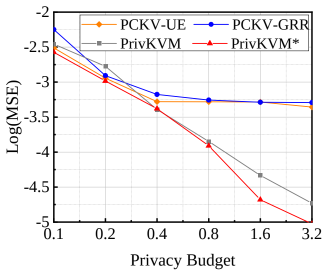
7.2 Results of Frequency and Mean Estimation

Fig. 5 shows the accuracy of frequency estimation by $PrivKVM^*$, *PrivKVM*, PCKV-UE and PCKV-GRR as the privacy budget increases from 0.1 to 3.2 (for the first four datasets) or 6.4 (for Clothing and Amazon). For $PrivKVM^*$ and *PrivKVM*, we set $c = 6$ (5 virtual iterations after the first real one). For $PrivKVM^*$, we set the frequency threshold $\delta = 0.5\%$. As will be shown at the end of this subsection, under the privacy budget up to 3.2, the choice of δ does not have much impact on the performance of $PrivKVM^*$. Overall, $PrivKVM^*$ outperforms the other three in most cases. For densely-populated datasets GAUSS and PLAW, *PrivKVM* can achieve comparable accuracy to

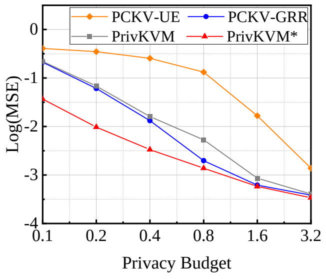
2. https://www.kaggle.com/c/talkingdata-mobile-user-demographics
3. http://www.datafountain.cn/#/competitions/247/data-intro
4. https://www.kaggle.com/rmisra/clothing-fit-dataset-for-size-recommendation
5. https://www.kaggle.com/skillsmuggler/amazon-ratings



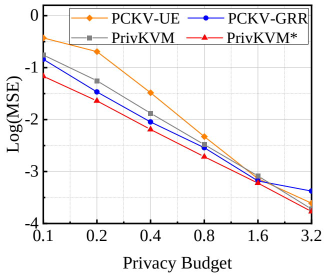
(a) GAUSS



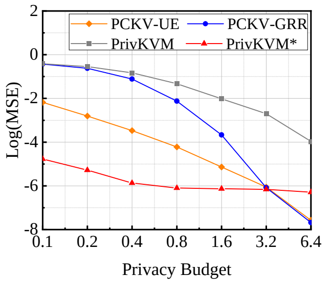
(b) PLAW



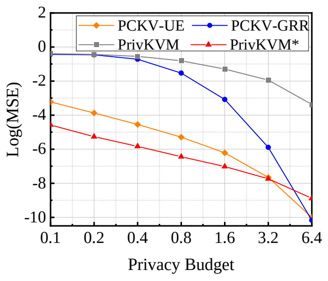
(c) TalkingData



(d) JData

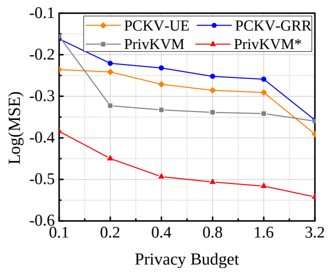


(e) Clothing

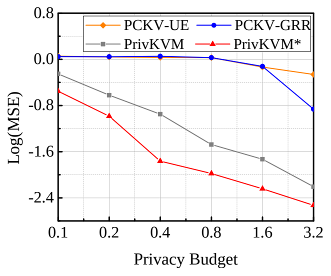


(f) Amazon

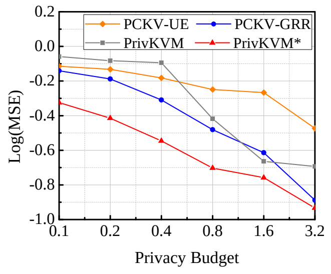
Fig. 5. Results of frequency estimation



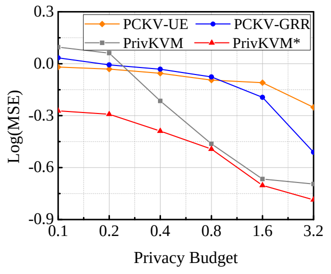
(a) GAUSS



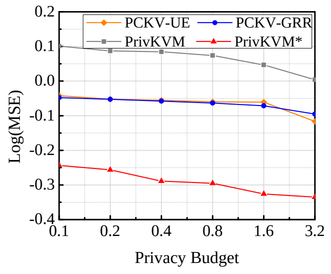
(b) PLAW



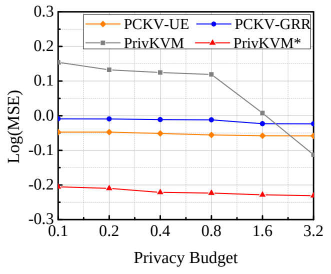
(c) TalkingData



(d) JData



(e) Clothing



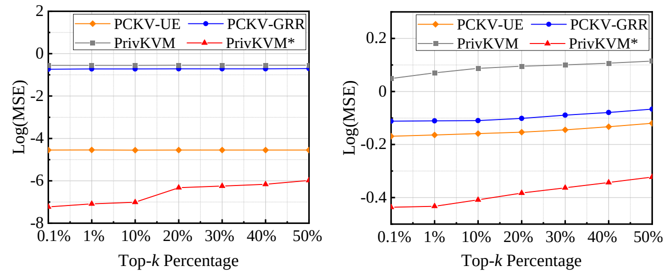
(f) Amazon

Fig. 6. Results of mean estimation

$PrivKVM^*$, and outperforms both PCKV-UE and PCKV-GRR. For medium-populated datasets TalkingData and JData, PCKV-GRR performs the second best due to the privacy amplification effect [15], followed by *PrivKVM* and PCKV-UE. For sparsely-populated datasets Clothing and Amazon, the performance gain of $PrivKVM^*$ is the most eminent when $\epsilon < 3.2$, which is mainly attributed to the two-phase adaptive sampling. On the other hand, PCKV-UE becomes the second best for $\epsilon < 3.2$, because the estimation accuracy of PCKV-UE dose not rely on the key space while PCKV-GRR suffers from an extremely large one. *PrivKVM*, on the other hand, performs the worst as it adopts a full-domain sampling and is thus not suitable for large key space. We also observe that, as the privacy budget becomes large, the accuracy gain of $PrivKVM^*$ tends to saturate, e.g., $\epsilon > 0.8$ for Clothing. For sparsely-populated datasets, when given a very large privacy budget, e.g., $\epsilon = 6.4$, PCKV-UE and PCKV-GRR even outperform $PrivKVM^*$. This is because we use the same frequency threshold for different ϵ, and therefore not many keys will be selected in the second phase even when the privacy budget is large. As such, the accuracy gain of $PrivKVM^*$ is not so eminent with the increasing privacy budget. Inspired by this, in real applications, when given a large privacy budget, we may properly decrease the frequency threshold δ to let more keys enter the second phase so that overall performance can be improved. Note that there are some discrepancy between the above results and those in [20], because in the latter only the results of popular keys are measured.

Fig. 6 shows the results of mean estimation. As with frequency estimation, the frequency threshold of $PrivKVM^*$ is $\delta = 0.5\%$. The results are similar to those in frequency estimation. First, $PrivKVM^*$ always outperforms the other three protocols. Second, *PrivKVM* achieves better accuracy than PCKV-UE and PCKV-GRR for densely-populated datasets. Third, PCKV-UE achieves better accuracy than the other two for sparsely-populated datasets, especially for the most sparse one Amazon, but the gap with PCKV-GRR is small. Fourth, for sparsely-populated datasets, the privacy budget ϵ has little impact on the accuracy, i.e., the curves in Figs. 6(e) and (f) are more flattened than the others. This is because of two reasons. First, a large key space and the sparsity of data lead to fewer valid samples for estimation. Second, even with increasing privacy budget, only the most popular keys can be estimated accurately and the MSE is still dominated by a majority of unpopular keys.

To further understand the performance of frequent keys, Fig. 7 shows the results of frequency and mean estimation over top-k frequent keys on the Amazon dataset, where k is set to 0.1%, 1%, 10%, 20%, 30%, 40% and 50% of the key domain size, and ϵ is set to 0.4. We observe that $PrivKVM^*$ outperforms the other three methods whereas *PrivKVM* performs the worst as the extremely large key domain reduces the valid samples for the estimation, even for frequent keys. We believe the reason why PCKV-UE and PCKV-GRR,



(a) Frequency estimation (b) Mean estimation

Fig. 7. Results of top-k frequent keys

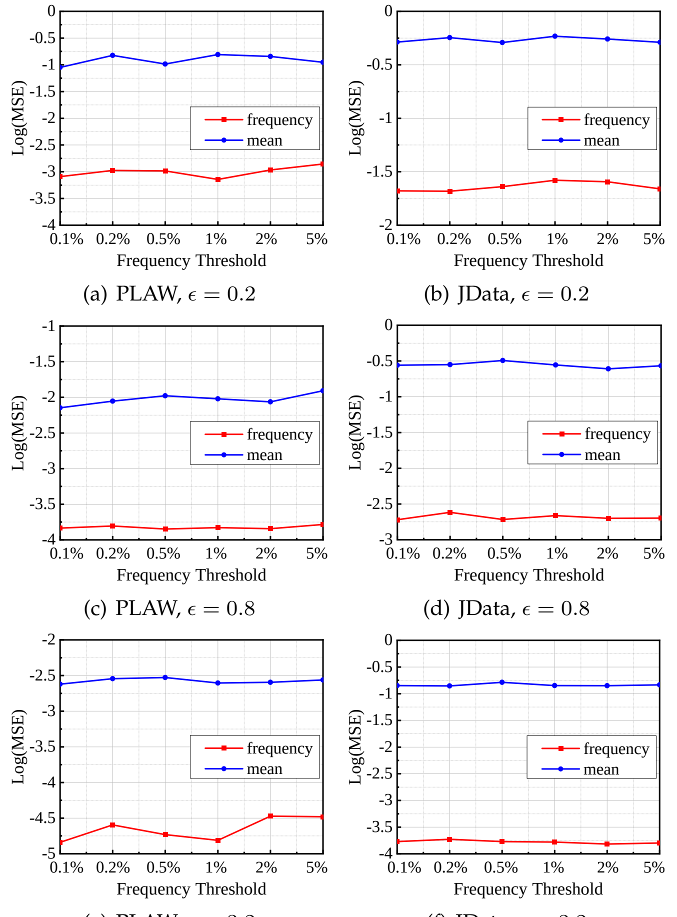
TABLE 3
Comparison on end-to-end bandwidth cost (in bits)

Datasets	PCKV-UE	PCKV-GRR	PrivKVM	PrivKVM* ($\delta = 0.5\%$)
GAUSS	230	8	8	289
PLAW	206	8	8	442
TalkingData	654	10	10	209
JData	906	10	10	1,334
Clothing	11,704	14	14	158
Amazon	498,552	19	19	2,586

originally favored by top-k frequent key queries, cannot achieve better performance than $PrivKVM^*$ is due to the difficulty of setting an optimal padding length for the Padding-and-Sampling protocol. This length is set to 2 in their original paper [20], regardless of k, which we adopt in the experiment, but obviously it should be adaptive to k.

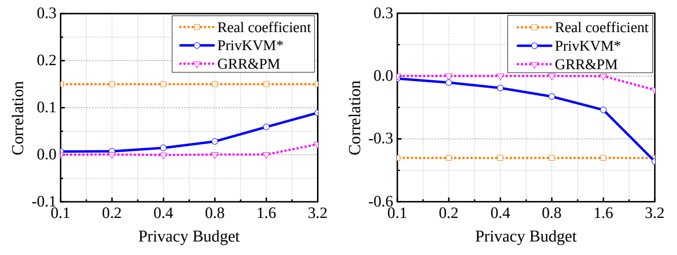
Communication bandwidth cost. Table 3 compares the end-to-end bandwidth cost in terms of the bits transferred between a user and the data collector. We observe that both PCKV-GRR and *PrivKVM* have equivalently the lowest bandwidth cost. On the other hand, the bandwidth cost of PCKV-UE dramatically increases with a larger key space. In particular, for the Amazon dataset, this cost reaches around 0.5Mb. For $PrivKVM^*$, the bandwidth cost is always kept at a moderate level between 158b and 2.5Kb, which is controlled by the number of popular keys.

Impact of frequency threshold δ. In Fig. 8, we investigate the impact of δ on the accuracy of frequency and mean estimation, varying the threshold δ from 0.1% to 5%. In the interest of space, we only show the results on synthetic dataset PLAW and real dataset JData. Overall, with increasing δ, both MSEs of frequency and mean estimation have only slight fluctuation, which indicates δ has little impact on the estimation accuracy. The fluctuation in JData is even less eminent than that in PLAW as the key distribution of the former is less skewed.



(a) PLAW, $\epsilon = 0.2$ (b) JData, $\epsilon = 0.2$
(c) PLAW, $\epsilon = 0.8$ (d) JData, $\epsilon = 0.8$
(e) PLAW, $\epsilon = 3.2$ (f) JData, $\epsilon = 3.2$

Fig. 8. Impact of δ on frequency and mean estimation



(a) GAUSS (b) PLAW

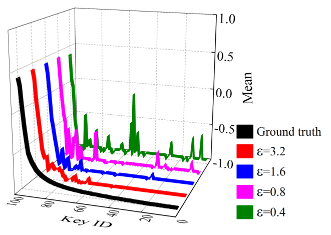
Fig. 9. Pearson correlation coefficient, varying privacy budget

7.3 Key-Value Correlation

In this subsection, we evaluate how well $PrivKVM^*$ retains the correlation between keys and values. As a comparison, we also show the result of the naive method that separately applies perturbation to keys (by GRR [11]) and values (by PM [18]) .

We use *Pearson correlation coefficient* [55] as the performance metric and test it over GAUSS and PLAW. Fig. 9 plots the results of real data, perturbed data by $PrivKVM^*$ and by $GRR\&PM$, respectively. We observe that with increasing privacy budget, the correlation coefficient of $PrivKVM^*$ gradually approaches the real one, whereas that of $GRR\&PM$ is always close to zero, which indicates its incapability 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. 10 3D illustrations of the estimated mean across keys in $PLAW$ under different privacy budgets. While the real means (the black line) across keys follow a power-law distribution, the estimated means of $PrivKVM^*$ follow a very similar distribution. Nonetheless, the tail of the distribution has larger fluctuation due to smaller sampling size of keys. On the contrary, the estimated means of $GRR\&PM$ are almost flat, which completely deviates from the true distribution and indicates the great loss of key-value correlation


(a) PLAW, $PrivKVM^*$

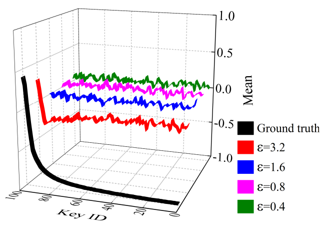
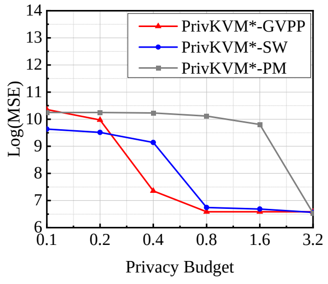
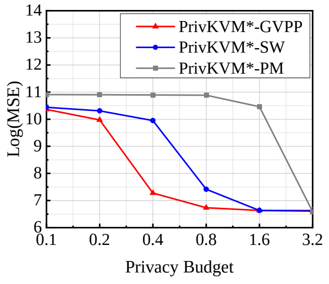
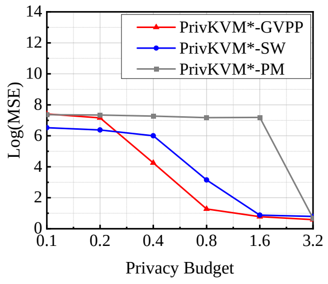

(b) PLAW, $GRR\&PM$

Fig. 10. 3D illustration of estimated mean across different keys and privacy budgets


(a) GAUSS, [0, 0.4]


(b) GAUSS, [0, 0.8]


(c) TalkingData, [0, 0.4]

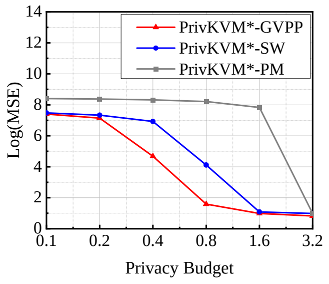

(d) TalkingData, [0, 0.8]

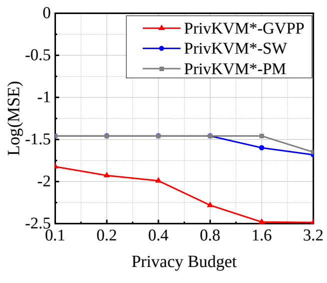
Fig. 11. Results of range query on count

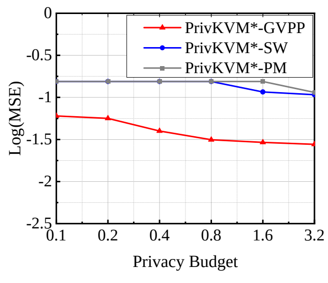
information.

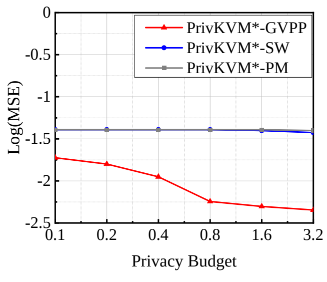
7.4 Range Query and Histogram Release

Finally, we conduct experiments of range query and histogram release over a synthetic dataset GAUSS and a real one TalkingData to evaluate the performance of $PrivKVM^*$. For range query, as the data is distributed in $[-1, 1]$, we choose two query lengths, i.e., $[0, 0.4]$ and $[0, 0.8]$, and then measure the MSEs of count and mean estimation of $PrivKVM^*$-$GVPP$, $PrivKVM^*$-SW and $PrivKVM^*$-PM, respectively. Fig. 11 plots the results of these three methods with privacy budget ranging from 0.1 to 3.2. Overall, $PrivKVM^*$-$GVPP$ outperforms the other two in most cases, except for extremely small privacy budgets, e.g., $\epsilon < 0.2$ for the range $[0, 0.4]$. $PrivKVM^*$-SW performs the second best, which is mainly because SW is designed for distribution estimation while PM is for mean estimation. Thus, $PrivKVM^*$-PM almost stays flat except given a large privacy budget. Note that for large privacy budget, e.g., $\epsilon = 3.2$, these three methods have very similar performance.

Fig. 12 shows the results of mean estimation. Overall, regardless of query length and privacy budget, $PrivKVM^*$-$GVPP$ always outperforms the other two by 1-2 orders of magnitude in most cases, and the advantage becomes more eminent with larger privacy budget. This is mainly attributed to the fact that the perturbation protocol GVPP always focuses on a specific range while SW or PM pays more attention to the whole domain range (overall distribution or mean respectively).


(a) GAUSS, [0, 0.4]


(b) GAUSS, [0, 0.8]


(c) TalkingData, [0, 0.4]

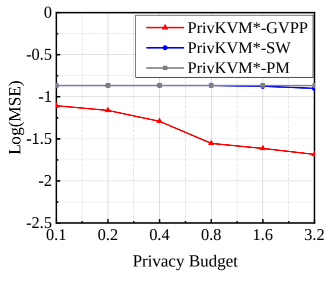
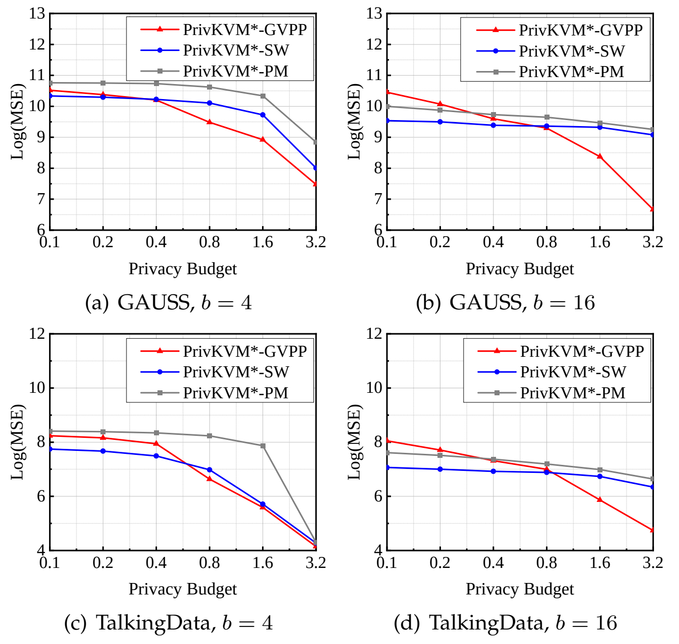

(d) TalkingData, [0, 0.8]

Fig. 12. Results of range query on mean

Fig. 13 shows the results of histogram release, where we only show the results of 4 bins and 16 bins due the space limitation. As for the performance metric, for each key, we first calculate the mean square error (MSE) of the estimated b histogram counts against ground truth. Then for the two datasets GAUSS and TalkingData, we further average all MSEs among keys to show the performance. Overall, $PrivKVM^*$-$GVPP$ and $PrivKVM^*$-SW achieve the comparable performance to each other, where the former has higher accuracy for large privacy budget, and $PrivKVM^*$-PM has the highest estimation error in most cases. Specifically, the accuracy of $PrivKVM^*$-$GVPP$ outperforms the other two for $\epsilon > 0.4$ or 0.8. The advantage of $PrivKVM^*$-$GVPP$ becomes more eminent for more bins when given a large privacy budget, e.g., $\epsilon = 3.2$. On the other hand, $PrivKVM^*$-PM has the highest estimation error, which is mainly due to its insufficiency to deal with histogram release which focuses on specific ranges. We also observe that when given a small privacy budget, e.g., $\epsilon = 0.1$, $PrivKVM^*$-$GVPP$ cannot achieve satisfying performance due to the heavy perturbation to boundaries points in GVPP, especially when there are more bins/boundary points.

8 Conclusion

This paper proposes $PrivKVM^*$, a two-phase statistics estimation mechanism on key-value data with local differential privacy. The building blocks are the correlated perturbation protocol (CPP) and generalized value perturbation protocol (GVPP). An optimization strategy that enables the data collector to execute virtual iterations is also presented. Through theoretical and empirical analysis, we



(a) GAUSS, $b = 4$

(b) GAUSS, $b = 16$

(c) TalkingData, $b = 4$

(d) TalkingData, $b = 16$

Fig. 13. Results of histogram release

show $PrivKVM^*$ is superior to state-of-the-art solutions, including $PrivKVM$ and $PCKV$ in terms of generality, accuracy, robustness, and communication cost under various datasets and 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 complicated privacy-preserving learning tasks (e.g., building some machine learning models and finding the gradient descent), and extend this work to key-value-specific learning tasks in these machine learning models.

Acknowledgments

We would like to thank authors of [20], especially Xiaolan Gu, for their provision of PCKV source codes. This work was supported by National Natural Science Foundation of China (Grant No: 62072390 and 61941121), and the Research Grants Council, Hong Kong SAR, China (Grant No: 15238116, 15222118, 15218919, 15203120 and C1008-16G).

References

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

[2] P. LLP, "How to seize the open banking opportunity," 2018.

[3] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *FOCS*. IEEE, 2013, pp. 429–438.

[4] ——, "Privacy aware learning," *Journal of the ACM*, vol. 61, no. 6, pp. 1–57, 2014.

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

[6] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *CCS*. ACM, 2014, pp. 1054–1067.

[7] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *NIPS*, 2017, pp. 3574–3583.

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

[9] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, "Towards locally differentially private generic graph metric estimation," in *ICDE*. IEEE, 2020, pp. 1922–1925.

[10] C. Wei, S. Ji, C. Liu, W. Chen, and T. Wang, "Asgldp: Collecting and generating decentralized attributed graphs with local differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3239–3254, 2020.

[11] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," in *NIPS*, 2014, pp. 2879–2887.

[12] R. Bassily and A. Smith, "Local, private, efficient protocols for succinct histograms," in *STOC*. ACM, 2015, pp. 127–135.

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

[14] 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*. ACM, 2016, pp. 192–203.

[15] T. Wang, N. Li, and S. Jha, "Locally differentially private frequent itemset mining," in *S&P*. IEEE, 2018, pp. 127–143.

[16] N. Wang, X. Xiao, Y. Yang, T. D. Hoang, H. Shin, J. Shin, and G. Yu, "PrivTrie: Effective frequent term discovery under local differential privacy," in *ICDE*. IEEE, 2018, pp. 821–832.

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

[18] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. C. Hui, H. Shin, J. Shin, and G. Yu, "Collecting and analyzing multidimensional data with local differential privacy," in *ICDE*, 2019.

[19] Q. Ye, H. Hu, X. Meng, and H. Zheng, "PrivKV: Key-value data collection with local differential privacy," in *S&P*. IEEE, 2019, pp. 317–331.

[20] X. Gu, M. Li, Y. Cheng, L. Xiong, and Y. Cao, "PCKV: locally differentially private correlated key-value data collection with optimized utility," in *USENIX Security Symposium*, 2020.

[21] C. Dwork, "Differential privacy," in *ICALP*. Springer, 2006, pp. 1–12.

[22] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[23] N. Li, M. Lyu, D. Su, and W. Yang, "Differential privacy: From theory to practice," *Synthesis Lectures on Information Security, Privacy, & Trust*, vol. 8, no. 4, pp. 1–138, 2016.

[24] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What can we learn privately?" *SIAM Journal on Computing*, vol. 40, no. 3, pp. 793–826, 2011.

[25] Q. Ye and H. Hu, "Local differential privacy: Tools, challenges, and opportunities," in *WISE*. Springer, 2020, pp. 13–23.

[26] G. Fanti, V. Pihur, and Ú. Erlingsson, "Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries," *PoPETS*, vol. 2016, no. 3, pp. 41–61, 2016.

[27] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.

[28] R. Bassily, U. Stemmer, A. G. Thakurta *et al.*, "Practical locally private heavy hitters," in *NIPS*, 2017, pp. 2285–2293.

[29] M. Bun, J. Nelson, and U. Stemmer, "Heavy hitters and the structure of local privacy," in *PODS*. ACM, 2018, pp. 435–447.

[30] S. Kim, H. Shin, C. H. Baek, S. Kim, and J. Shin, "Learning new words from keystroke data with local differential privacy," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[31] A. G. Thakurta, A. H. Vyrros, U. S. Vaishampayan, G. Kapoor, J. Freudinger, V. V. Prakash, A. Legendre, and S. Duplinsky, "Emoji frequency detection and deep link frequency," Jul. 11 2017, uS Patent 9,705,908.

[32] G. Cormode, T. Kulkarni, and D. Srivastava, "Marginal release under local differential privacy," in *SIGMOD*. ACM, 2018, pp. 131–146.

[33] Z. Zhang, T. Wang, N. Li, S. He, and J. Chen, "CALM: Consistent adaptive local marginal for marginal release under local differential privacy," in *CCS*. ACM, 2018, pp. 212–229.

[34] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, "LF-GDPR:graph metric estimation with local differential privacy," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

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

[36] Y. Cao, Y. Xiao, L. Xiong, and L. Bai, "Priste: from location privacy to spatiotemporal event privacy," in *ICDE*. IEEE, 2019, pp. 1606–1609.

[37] Q. Ye, H. Hu, N. Li, X. Meng, H. Zheng, and H. Yan, "Beyond value perturbation: Local differential privacy in the temporal setting," in *INFOCOM*. IEEE, 2021, pp. 1–10.

[38] Z. Li, T. Wang, M. Lopuhaä-Zwakenberg, N. Li, and B. Škoric, "Estimating numerical distributions under local differential privacy," in *SIGMOD*, 2020, pp. 621–635.

[39] G. Cormode, T. Kulkarni, and D. Srivastava, "Answering range queries under local differential privacy," *Proceedings of the VLDB Endowment*, vol. 12, no. 10, pp. 1126–1138, 2019.

[40] R. Du, Q. Ye, Y. Fu, and H. Hu, "Collecting high-dimensional and correlation-constrained data with local differential privacy," in *SECON*, 2021.

[41] 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.

[42] T. Wang, B. Ding, J. Zhou, C. Hong, Z. Huang, N. Li, and S. Jha, "Answering multi-dimensional analytical queries under local differential privacy," in *SIGMOD*. ACM, 2019, pp. 159–176.

[43] M. Xu, T. Wang, B. Ding, J. Zhou, C. Hong, and Z. Huang, "Dpsaas: multi-dimensional data sharing and analytics as services under local differential privacy," *Proceedings of the VLDB Endowment*, vol. 12, no. 12, pp. 1862–1865, 2019.

[44] A. Smith, A. Thakurta, and J. Upadhyay, "Is interaction necessary for distributed private learning?" in *S&P*. IEEE, 2017, pp. 58–77.

[45] Y. Wang, Y. Tong, and D. Shi, "Federated latent dirichlet allocation: A local differential privacy based framework," in *AAAI*, 2020, pp. 6283–6290.

[46] H. Zheng, Q. Ye, H. Hu, C. Fang, and J. Shi, "BDPL: A boundary differentially private layer against machine learning model extraction attacks," in *ESORICS*. Springer, 2019, pp. 66–83.

[47] ——, "Protecting decision boundary of machine learning model with differentially private perturbation," *IEEE Transactions on Dependable and Secure Computing*, 2020.

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

[49] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *FOCS*. IEEE, 2007, pp. 94–103.

[50] F. McSherry, "Privacy integrated queries: an extensible platform for privacy-preserving data analysis," in *SIGMOD*. ACM, 2009, pp. 19–30.

[51] F. Buccafurri, D. Rosaci, L. Pontieri, and D. Saccá, "Improving range query estimation on histograms," in *ICDE*, 2002.

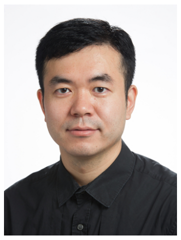
[52] A. Dobra, "Histograms revisited: when are histograms the best approximation method for aggregates over joins?" in *PODS*, 2005, pp. 228—-237.

[53] P. Kairouz, K. Bonawitz, and D. Ramage, "Discrete distribution estimation under local privacy," in *ICML*. ACM, 2016, pp. 2436–2444.

[54] G. Casella and R. L. Berger, *Statistical inference*. Cengage Learning, 2021.

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

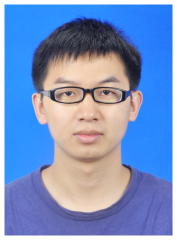
Qingqing Ye is a research assistant professor in the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University. She received her PhD degree in Computer Science from Renmin University of China in 2020. She has received several prestigious awards, including China National Scholarship, Outstanding Doctoral Dissertation Award, and IEEE S&P Student Travel Award. Her research interests include data privacy and security, and adversarial machine learning.



Haibo Hu is an associate professor in the Department of Electronic and Information Engineering, Hong Kong Polytechnic University. His research interests include cybersecurity, data privacy, internet of things, and machine learning. He has published over 80 research papers in refereed journals, international conferences, and book chapters. As principal investigator, he has received over 12 million HK dollars of external research grants from Hong Kong and mainland China. He is the recipient of a number of titles and awards, including IEEE MDM 2019 Best Paper Award, WAIM Distinguished Young Lecturer, VLDB Distinguished Reviewer, ACM-HK Best PhD Paper, Microsoft Imagine Cup, and GS1 Internet of Things Award.

Xiaofeng Meng is a professor in School of Information, Renmin University of China. He is a CCF Fellow and the vice chair of the Special Interesting Group on Privacy of China Confidentiality Association(CCA). He has served on the program committee SIGMOD, ICDE, CIKM, MDM, DASFAA, etc., and editorial board of JCST, FCS, JoS, CRAD, etc. His research interests include web data management, cloud data management, mobile data management, and privacy protection. He has published over 200 papers in refereed international journals and conference proceedings including IEEE TKDE,VLDBJ , VLDB, SIGMOD, ICDE, EDBT, ACM GIS etc.

Huadi Zheng receives the BEng degree from the School of Data and Computer Science, Sun Yat-sen University, China, in 2012. Currently he is pursuing a PhD degree in the Department of Electronic and Information Engineering, Hong Kong Polytechnic University. His research interests include mobile side-channel security, data privacy and machine learning.



Kai Huang is a postdoc in the Department of Electronic and Information Engineering, Hong Kong Polytechnic University. He received his PhD in School of Computer Science from Fudan University in 2020, and BEng degree in Software Engineering from East China Normal University in 2014. His research interests include graph database and privacy-aware data management.

Chengfang Fang obtained his Ph.D. degree from National University of Singapore before joining Huawei. He has been working on security and privacy protection in several areas including machine learning, internet of things, mobile device and biometrics for more than 10 years. He has published over 20 research papers and obtained 15 patents in the domain. He is currently a principle researcher in Huawei Singapore Research Center.

Jie Shi is a Principal Researcher in Huawei Singapore Research Center. His research interests include trustworthy AI, machine learning security, data security and privacy, IoT security and applied cryptography. He has over 10 years' research experience and has published over 30 research papers in refereed journals and international conferences. He received his Ph.D degree from Huazhong University of Science and Technology, China.