Local Differentially Private Heavy Hitter Detection in Data Streams with Bounded Memory

XIAOCHEN LI, The State Key Laboratory of Blockchain and Data Security in Zhejiang University, China
WEIRAN LIU, Alibaba Group, China
JIAN LOU, Zhejiang University, China
YUAN HONG, University of Connecticut, USA
LEI ZHANG, Alibaba Group, China
ZHAN QIN*, The State Key Laboratory of Blockchain and Data Security in Zhejiang University, China
KUI REN, The State Key Laboratory of Blockchain and Data Security in Zhejiang University, China

Top-k frequent items detection is a fundamental task in data stream mining. Many promising solutions are proposed to improve memory efficiency while still maintaining high accuracy for detecting the Top-k items. Despite the memory efficiency concern, the users could suffer from privacy loss if participating in the task without proper protection, since their contributed local data streams may continually leak sensitive individual information. However, most existing works solely focus on addressing either the memory-efficiency problem or the privacy concerns but seldom jointly, which cannot achieve a satisfactory tradeoff between memory efficiency, privacy protection, and detection accuracy.

In this paper, we present a novel framework HG-LDP to achieve accurate Top-k item detection at bounded memory expense, while providing rigorous local differential privacy (LDP) protection. Specifically, we identify two key challenges naturally arising in the task, which reveal that directly applying existing LDP techniques will lead to an inferior "accuracy-privacy-memory efficiency" tradeoff. Therefore, we instantiate three advanced schemes under the framework by designing novel LDP randomization methods, which address the hurdles caused by the large size of the item domain and by the limited space of the memory. We conduct comprehensive experiments on both synthetic and real-world datasets to show that the proposed advanced schemes achieve a superior "accuracy-privacy-memory efficiency" tradeoff, saving 2300× memory over baseline methods when the item domain size is $41,270$. Our code is anonymously open-sourced via the link.[1]

CCS Concepts: • **Security and privacy → Data anonymization and sanitization**.

Additional Key Words and Phrases: Local differential privacy, heavy hitter, data stream processing

ACM Reference Format:
Xiaochen Li, Weiran Liu, Jian Lou, Yuan Hong, Lei Zhang, Zhan Qin, and Kui Ren. 2024. Local Differentially Private Heavy Hitter Detection in Data Streams with Bounded Memory. *Proc. ACM Manag. Data* 2, 1 (SIGMOD), Article 30 (February 2024), 27 pages. https://doi.org/10.1145/3639285

*Zhan Qin is the corresponding author.

[1]https://github.com/alibaba-edu/mpc4j/tree/main/mpc4j-dp-service

Authors' addresses: Xiaochen Li, xiaochenli@zju.edu.cn, The State Key Laboratory of Blockchain and Data Security in Zhejiang University, China; Weiran Liu, weiran.lwr@alibaba-inc.com, Alibaba Group, China; Jian Lou, jian.lou@zju.edu.cn, Zhejiang University, China; Yuan Hong, yuan.hong@uconn.edu, University of Connecticut, USA; Lei Zhang, zongchao.zl@taobao.com, Alibaba Group, China; Zhan Qin, qinzhan@zju.edu.cn, The State Key Laboratory of Blockchain and Data Security in Zhejiang University, China; Kui Ren, kuiren@zju.edu.cn, The State Key Laboratory of Blockchain and Data Security in Zhejiang University, China.

ACM 2836-6573/2024/2-ART30
https://doi.org/10.1145/3639285

1 INTRODUCTION

Detecting Top-k frequent items in data streams is one of the most fundamental problems in streaming data analysis [10, 16, 25, 39, 47]. It forms the foundation for a multitude of critical applications across various domains, such as anomaly detection in data mining [15], click analysis in web analysis [55], and topic mining in social networks [64]. In the typical decentralized setting as illustrated in Figure 1, the users send local item counts to the server in a streaming fashion, and the s
the i
all it
dom
billi
minu
serv
men
dom

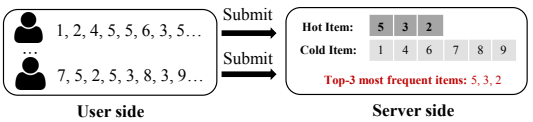

Fig. 1. An example of Top-3 frequent items detection.

Furthermore, users' submitted streaming data often contain sensitive individual information, e.g., click analysis may reveal online behavior and topic mining may reveal political opinions. The privacy of users is under severe threat if they submit local data streams without proper privacy protection. In particular, the privacy concern has a unique characteristic in the Top-k detection problem. That is, the cold items (i.e., items with low frequencies, as depicted in Figure 1) are not statistical targets, but constitute the majority of the data domain and are particularly sensitive, as they reveal highly personal information specific to certain user groups. Due to its central role in streaming data analysis, Top-k frequent items detection has attracted significant research attention in recent years. However, most existing works pursue the memory efficiency or privacy protection goals separately but seldom jointly.

On the memory efficiency side, a series of approaches have been proposed to improve the memory efficiency with decent accuracy for detecting the Top-k items [44, 45, 48, 49, 62, 65]. The key rationale of the memory-saving stems from the fact that most items are cold while only a few items are hot in practical data streams [19, 53]. Accurately recording the information of massive cold items not only wastes much memory, but also incurs non-trivial errors in hot item estimation when the memory is tight. Thus, existing methods seek to design a compact data structure to keep and guard the items and their frequencies of hot items, while possibly evicting cold items. One of the most widely adopted and effective data structures addressing this challenge is *HeavyGuardian* [62]. It introduces the *separate-and-guard-hot* design principle, which effectively segregates hot items from cold items, preserving the accuracy of hot item estimations. *HeavyGuardian* further delineates a specific strategy called *Exponential Decay* (ED) to guard the hot items by exponentially decreasing the probability that the possible cold items remain in the heavy part of the data structure. However, despite achieving a promising balance between accuracy and memory efficiency, none of these methods simultaneously account for privacy concerns.

On the privacy protection side, Differential Privacy (DP) has been regarded as a *de facto* standard by both academia and industry [27, 30]. In the decentralized data analytics setting, Local Differential Privacy (LDP) is the state-of-the-art approach extended from DP to the local setting, which has been widely deployed in industry, e.g., Google Chrome browser [31] and Apple personal data collection [37]. In LDP, each user perturbs his/her data with a local randomization mechanism before sending it to the server. The server could still derive general statistics from the perturbed submissions with a certain accuracy decrease. General randomization mechanisms for frequency estimation such as Generalized Randomized Response (GRR), Optimal Local Hash (OLH) [57], and Hadamard Response (HR) [6], can be applied to Top-k items detection as baseline methods. There also exist many works designed specifically for heavy hitter estimation under LDP, including estimates over the single-valued data [11, 22, 36, 60], and set-valued data [51, 59]. However, it is noteworthy that these works neither address the data stream setting nor tackle the issue of memory efficiency.

In this paper, our objective is to bridge the gap between memory-efficient heavy hitter tracking in data streams and LDP privacy protection. To achieve this, we introduce the HG-LDP framework designed for tracking the Top-k heavy hitters within data streams. This framework comprises three essential modules. First, the *randomization module* is responsible for randomizing the streaming data generated by users, ensuring event-level LDP privacy that is more suitable for the streaming data [28, 50]. Second, the *storage module* records the incoming data on the server side. To this end, we integrate the *HeavyGuardian* data structure, and significantly optimize its implementations, i.e., dynamic parameter configuration, and sampling optimization (see details in Appendix A.7 in [46]) to facilitate the heavy hitter tasks and processing of LDP-protected noisy data. Finally, the *response module* processes and publishes the statistical results of heavy hitters.

It is worth noting that directly applying existing LDP techniques cannot achieve satisfactory accuracy or would be even functionally infeasible, primarily due to the following two new challenges.

Challenge (1): Incompatibility of Space-Saving Strategy and Large Domain Size for LDP. To highlight this challenge, we instantiate a basic scheme BGR as a baseline (detailed in Section 3.2), which directly uses the Generalized Randomized Response (GRR) mechanism [57] in the randomization module. The large domain size incurs two problems that jointly fail BGR: 1) the noise variance introduced by the GRR will increase as the data domain increases; 2) the space-saving strategy of the data structure introduces additional underestimation error to the noise items, which will be further amplified by the debiasing operation, required by LDP. Although existing mechanisms such as Optimal Local Hash (OLH) [57] and Hadamard Response (HR) [6] in the LDP field aim to alleviate the impact of large data domains on randomized results' accuracy, it is crucial to emphasize that we still confront a unique and unaddressed challenge. We identified that the core idea of the LDP field in addressing this problem is to encode the large data domain into a smaller one for randomization. However, the decoding of randomized data on the server side inevitably produces a multiple of diverse collision data, which can significantly disrupt the decision-making of the space-saving strategy.

Challenge (2): Dynamically Changing Hot/Cold Items. Notably, cold items often constitute the majority of the data domain, and indiscriminately randomizing data across the entire domain can result in an unnecessary waste of privacy budget. The ability to distinguish between hot items and cold items during the randomization process is crucial for enhancing the accuracy of hot item estimation. However, since the labels of hot and cold items may dynamically change as the data stream evolves, randomizing data based on the previous timestamp's state may introduce a huge bias towards the prior state. This poses several new challenges, e.g., how to strike a balance between reducing unnecessary privacy budget expenditure on cold items, and how to manage such dynamically emerging bias. Addressing this challenge also mandates novel LDP mechanism designs.

Contribution. In this paper, we initiate a baseline method and propose three novel advanced LDP designs under the hood of a framework HG-LDP to address these hurdles. First, we present a baseline method that directly combines the GRR mechanism with *HeavyGuardian* data structure. Second, we propose a newly designed LDP mechanism. It is based on the observation that the ED strategy does not need to know the specific item of the incoming data in most cases if it is not recorded in the data structure. Third, we adjust the noise distribution by dividing the privacy budget to achieve higher accuracy. Finally, we utilize the light part of *HeavyGuardian* to elect current cold items before they become new hot items, which further improves the accuracy of the estimated result. The main contributions are summarized as follows.

- To our best knowledge, this paper is the first to track the Top-k frequent items from data streams in a bounded memory space while providing LDP protection for the sensitive streaming data. We present a general framework called HG-LDP to accommodate any proper LDP randomization mechanisms on the users' side into the space-saving data structures on the server side for the task.
- By investigating the failure of naïvely combining existing LDP techniques with HG-LDP, we design three new LDP schemes, which achieve a desired tradeoff performance between accuracy, privacy, and memory efficiency.
- We comprehensively evaluate the proposed schemes on both synthetic and real-world datasets in terms of accuracy and memory consumption, which shows that the proposed schemes achieve higher accuracy and higher memory efficiency than baseline methods. For instance, when the size of the domain size reaches $41,270$, the proposed schemes save about $2300\times$ size of memory over baselines.

2 PRELIMINARIES

2.1 Problem Statement

We consider the setting of finding Top-k items in data streams under Local Differential Privacy (LDP). Given n users, each user generates a private infinite data stream. Denote $v_i^t \in \Omega$ as the data generated by the user u_i at timestamp t. The user only sends data at the timestamp when data is generated. A server collects values from users at each timestamp t. Note that the server can only maintain a data structure with a length much smaller than the size d of data domain Ω due to its limited memory space. Whenever a query is received, the server needs to publish the Top-k items up to the latest timestamp and their counts.

2.2 Privacy Definitions

In this paper, we provide event-level privacy guarantee [14, 17, 28, 50, 58]. Specifically, the event-level LDP ensures the indistinguishability of any pairs of elements in streams, e.g., every single transaction remains private in a user's long-term transactions:

Definition 2.1 (Local Differential Privacy (LDP) [41]). An algorithm $\mathcal{M}$ satisfies ϵ-LDP, where $\epsilon \geq 0$, if and only if for any input $v, v' \in \mathbb{D}$, and any output $y \in \mathit{Range}(\mathcal{M})$, we have

$$\Pr\left[\mathcal{M}(v) = y\right] \leq e^{\epsilon} \Pr\left[\mathcal{M}(v') = y\right].$$

The parameter ϵ is called the *privacy budget*, whereby smaller ϵ reflects stronger privacy guarantees. We say $\mathcal{M}$ satisfies ϵ-LDP if for different data v and v', the ratio of distribution of output $\mathcal{M}(v)$ and that of $\mathcal{M}(v')$ are not greater than e^{ϵ}.

2.3 LDP Mechanisms

The Randomized Response (RR) mechanism [61] is considered to be the first LDP mechanism that supports binary response. It allows each user to provide a false answer with a certain probability so as to provide plausible deniability to users. The Generalized Randomized Response (GRR) mechanism [57] is an extension of Randomized Response (RR) [61], which supports multi-valued domain response. Denote d as the size of the domain $\mathbb{D}$. Each user with private value $v \in \mathbb{D}$ reports the true value $v' = v$ with probability p and reports a randomly sampled value $v' \in \mathbb{D}$ where $v' \neq v$ with probability q. The probability p and q are defined as follows

$$\begin{cases} p = \dfrac{e^{\epsilon}}{e^{\epsilon} + d - 1}, \\ q = \dfrac{1}{e^{\epsilon} + d - 1}. \end{cases} \tag{1}$$

where d is the size of the data domain. It is straightforward to prove ϵ-LDP for GRR, i.e., $p/q \leq e^{\epsilon}$ [57]. Assuming that each of the n users reports one randomized value. Let $\hat{c}_i$ be the number of value i occurs in randomized values, the estimation of true number $\tilde{c}_i$ of value i can be computed with $\tilde{c}_i = \frac{\hat{c}_i - nq}{p-q}$. The variance of the estimated result $\tilde{c}_i$ is $Var[\tilde{c}_i] = n \cdot \frac{d-2+e^{\epsilon}}{(e^{\epsilon}-1)^2}$.

As shown above, the variance of the estimation result of the GRR mechanism increases linearly with the increase of d. Some other mechanisms, e.g., Optimal Local Hash (OLH) [57] and Hadamard Response (HR) [6], are proposed to randomize data in a large data domain. Essentially, they map the data to a smaller domain before randomizing it to avoid the large variance caused by a large data domain. We defer their details to Appendix A.1 in [46].

2.4 Space-Saving Data Structure

Counter-based data structures [48, 49, 62, 65] and sketches [7, 18, 23, 45] are two kinds of mainstream memory-efficient data structures. While sketches have been extensively studied as compressed data structures for frequency estimation, they may not be the optimal choice when it comes to heavy hitter estimation in data streams, particularly in scenarios characterized by limited storage space and real-time response requirements. This preference is underpinned by two key reasons: Firstly, sketches record counts for all items, whereas heavy hitter tasks only concern hot items. This equally treated recording of all counts results in unnecessary memory consumption. For example, the Count-Min sketch (CMS) necessitates a minimum of $O(\frac{N}{\alpha} \times \log(1/\delta))$ space to guarantee that the probability of error in the estimated count of each item being less than α is no less than $1 - \delta$, with N representing the total data count [23]. Furthermore, as highlighted by Cormode and Hadjieleftheriou in [20], sketches require additional storage for finding the counts of hot items. For instance, $O(\frac{N}{\alpha} \log d \log \delta)$ space increase is incurred when using group testing to find hot items, or a minimum of $O(d)$ computational overhead is needed for hot item retrieval.

Thus, in this paper, we choose to employ a counter-based data structure called *HeavyGuardian* proposed by Yang et al. [62] as the foundation for our framework. It identifies and records the high-frequency items in subsequent data streams based on observations of historical streaming data. The basic version of *HeavyGuardian* is a hash table with each bucket storing several KV pairs ($\langle ID, count \rangle$) and small counters. Specifically, each bucket is divided into two parts: a heavy part with a length of λ_h ($\lambda_h > 0$) to precisely store counts of hot items, and a light part with a length of λ_l (λ_l can be 0) to approximately store counts of cold items. For each incoming item e, *HeavyGuardian* needs to decide whether and how to insert it into the heavy part of a bucket according to a strategy called Exponential Decay (ED). There are three cases when inserting an item e into the heavy part of *HeavyGuardian*.

Case 1: The KV pair of e has been stored in the heavy part, it increments the corresponding *count* by 1.

Case 2: The KV pair of e is not in the heavy part, and there are still empty buckets. It inserts the KV pair of e into the heavy part and sets the *count* to 1.

Case 3: The KV pair of e is not in the heavy part, and there is no empty bucket. It decays 1 from the current least *count* in the heavy part with probability $\mathcal{P} = b^{-c}$, where b is a predefined constant number (b=1.08 in [62]), and c is the *count* value. After decay, if the *count* becomes 0, it replaces this KV pair (the weakest KV pair) with e's KV pair, and sets the *count* to 1.

If e is not successfully inserted into the heavy part, it is recorded in the light part. Since the heavy hitter tasks only focus on Top-k items and their counts, we set the parameters of *HeavyGuardian* as the number of buckets $w = 1$, the length of the heavy part $\lambda_h = k$, and the length of light part $\lambda_l = 0$ (except in one of the proposed scheme CNR). For simplicity of description, we denote the data structure of *HeavyGuardian* as $\mathcal{HG}$ in the following sections. We use $\mathcal{HG}[i]$ to denote the i^{th} key pair in $\mathcal{HG}$, and use $\mathcal{HG}[i].ID$ and $\mathcal{HG}[i].C$ to denote the ID and the count of an item, respectively.

3 HG-LDP FOR HEAVY HITTERS TRACKING

In this section, we first introduce the HG-LDP framework for tracking heavy hitters in data streams with bounded memory space. Then, we instantiate a baseline to highlight key obstacles for achieving a satisfactory "*accuracy-privacy-memory efficiency*" tradeoff.

3.1

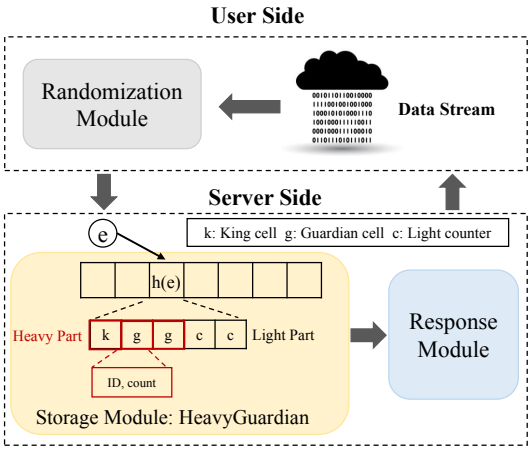

Fig. 2. The overview of HG-LDP.

Figure 2 illustrates the framework for HG-LDP, which contains three modules: *randomization module*, *storage module*, and *response module*. The *randomization module* runs on the user side to randomize the users' sensitive streaming data. The *storage module* and the *response module* run on the server side, where the *storage module* utilizes a space-saving data structure.

In this paper, we aim to adapt and optimize the *HeavyGuardian* ($\mathcal{HG}$) data structure due to its popularity and simplicity, but expect our LDP designs to be generalizable to more sophisticated space-saving data structures in the future. The randomized streaming data continuously reported

by users is stored in $\mathcal{HG}$ following the *ED strategy*, and the statistical results are released by the *response module* after *debiasing*. Specifically, the functions of the three modules can be summarized as the following three algorithmic components:

- Randomize. It is executed in the *randomization module*. It takes raw data v_i^t of the i^{th} user at timestamp t as input, and outputs a randomized data r_i^t that satisfies LDP.
- Insert. It is executed in the *storage module*. It inserts the randomized data r_i^t into $\mathcal{HG}$ following the *ED strategy*, and updates the counts of the KV pairs in $\mathcal{HG}$.
- Response. It is executed in the *response module*. It obtains the hot items and their corresponding counts from $\mathcal{HG}$ when receiving a request. Then it maps them to a list for publishing after debiasing all counts.

In the following sections, we first instantiate a baseline scheme, and then propose three advanced schemes based on this framework by elaborately designing algorithms for the three modules.

3.2 A Baseline Scheme: BGR

We first discuss a baseline scheme BGR (Basic Scheme Combining GRR) that directly integrates an existing LDP scheme: GRR.

Algorithms. The BGR algorithm is outlined in Algorithm 1. At timestamp t, the data v_i^t of a user u_i is randomized using GRR, and the resulting randomized value r_i^t is then transmitted to the server. Subsequently, the server incorporates r_i^t into the data structure $\mathcal{HG}$ following the *ED strategy*. Note that the counts stored within $\mathcal{HG}$ are consistently biased noisy values. To mitigate this, the server debiases all counts in the *response module* following the standard GRR debiasing approach [57] before publishing the statistical outcomes.

Algorithm 1 BGR (baseline)

Input: timestamp t, data domain Ω with size d, data structure $\mathcal{HG}$, number of the received data *num*.
Output: *ResponseList*

Randomize
Obtain the current raw data v_i^t;
$r_i^t \leftarrow \text{GRR}(v_i^t, \epsilon)$ ▷ Randomize data with GRR.
Insert
Receive an incoming data r_i^t;
$num \leftarrow num + 1$
Insert r_i^t into $\mathcal{HG}$ following ED strategy;
if the least count $\mathcal{HG}[k].C \leq 0$ **then**
    Replace the weakest KV pair with new KV pair $< r_i^t, 1 >$
Response
$p = \frac{e^\epsilon}{e^\epsilon + d - 1}$, $q = \frac{1}{e^\epsilon + d - 1}$
if receive a Top-k query **then**
    for each $\mathcal{HG}[j] \in \mathcal{HG}$ **do**
        $ResponseList[j].ID \leftarrow \mathcal{HG}[j].ID$
        $ResponseList[j].C \leftarrow (\mathcal{HG}[j].C - num \cdot q)/(p - q)$
return *ResponseList*

Theoretical Analysis. Next, we theoretically analyze the error bound of the frequency estimated by BGR. Part of the error comes from the exponential decay of the counts on the server when the coming data is not recorded in $\mathcal{HG}$. Another part of the error comes from the noise introduced by

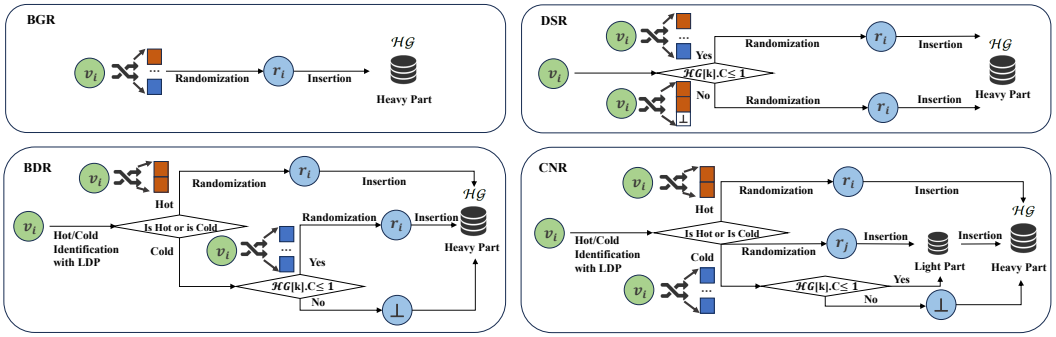

Fig. 3. The flowcharts of the randomization and storage modules in BGR (baseline) and three advanced schemes. *Note that each subfigure only shows the procedures of one scheme for a single user. The system procedures are more complicated since a large number of users would frequently/concurrently submit data to the server and update the $\mathcal{HG}$ (the domain for randomization frequently changes). A debiasing procedure is also included in the response module of the server.* DSR employs a strategy of randomization within a reduced domain when there are no imminent hot item evictions. BDR mitigates the impact of expansive cold domains on the accuracy of hot item estimates by splitting the privacy budget, which also eliminates the need for switching randomization strategies in DSR. CNR fully utilizes the idle privacy budget in BDR and elects new hot items with more potential to enter $\mathcal{HG}$.

the *randomization module* to perturb the data with the GRR. We first give the error analysis for the *ED strategy* of $\mathcal{HG}$ provided by Yang et al. [62] in Lemma 3.1 below.

Lemma 3.1. *Given a stream prefix S_t with t items in Ω, it obeys an arbitrary distribution and $|\Omega| = d$. We assume that there are w buckets to store the hottest λ items mapped to them, each item is mapped to a bucket with the probability of $\frac{1}{w}$. Let v_i be the i^{th} hottest item, f_i be the real frequency of v_i, and $\tilde{f}_i$ be the estimated frequency of v_i. Given a small positive number α, we have*

$$Pr[f_i - \tilde{f}_i \geq \alpha t] \leq \frac{1}{2\alpha t}(f_i - \sqrt{f_i^2 - \frac{4P_{weak}E(V)}{b-1}})$$

where $P_{weak} = e^{-(i-1)/w} \times (\frac{i-1}{w})^{l-1}/(l-1)!$, $E(V) = \frac{1}{w}\sum_{j=i+1}^{d} f_j$.

Our theoretical analysis follows the conclusion provided in Lemma 3.1. In fact, Lemma 3.1 only considers the bias caused by *exponential decays* after the items are recorded as hot items, ignoring the count loss before items are recorded. However, this count loss is strongly related to the distribution of the data stream and the order of the data arrival, so it's difficult to be theoretically analyzed. Besides, as we mentioned in Section 2.4, we set the number of buckets $w = 1$ in this paper since we only track Top-k heavy hitters and k is a small constant. Therefore, we only use the result when $w = 1$ in Lemma 3.1 and we show the error bound of BGR in Theorem 3.2.

Theorem 3.2. *Given a stream prefix $\hat{S}_t$ with t items randomized by BGR satisfying ϵ-LDP and there is a data structure $\mathcal{HG}$ to store the Top-k items. Let v_i be the i^{th} hottest item, f_i be the real frequency of v_i, $\tilde{f}_i$ be the final estimated frequency of v_i. We have*

$$\begin{aligned} Pr[f_i - \tilde{f}_i \leq (\sqrt{2t\log(2/\beta)} + \alpha t) \cdot \frac{e^\epsilon + d - 1}{e^\epsilon - 1}] \\ \geq (1-\beta)(1 - \frac{1}{2\alpha}(1 - \sqrt{1 - \frac{4P_{weak}E(V)}{b-1}})) \end{aligned}$$

where $P_{weak} = \frac{(i-1)!(d-k)!}{(d-1)!(i-k)!}$, $E(V) = \sum_{j=i+1}^{d} f_j$, α *and* β *are small positive numbers with* $\alpha, \beta \in (0, 1)$.

Proof. We assume that $\hat{f}_i$ is the frequency of noisy data recorded in $\mathcal{HG}$ according to the *ED strategy*. Meanwhile, due to the ED strategy introducing additional errors during the recording of $\hat{f}_i$, the frequency used for debiasing by the GRR mechanism before publication is denoted as $\bar{f}_i$. Then the error bound of final debiased frequency $\tilde{f}_i$ compared to f_i can be obtained by combining the error bounds of $\hat{f}_i - \bar{f}_i$ and $f_i - \frac{\hat{f}_i - tq}{p-q}$. The detailed proof is deferred to Appendix A.2 in [46]. □

Problems with Existing LDP mechanisms. Theorem 3.2 shows that the error bound of BGR grows proportionally to the size of the data domain d. While BGR is sufficient for solving the task of finding Top-k items in streaming data with a small data domain, it can inevitably fall into the dilemma that the error is too large when dealing with a large data domain. From the proof of Theorem 3.2, we can find that the excessive error caused by the large data domain mainly comes from the randomization process of the GRR.

Several LDP mechanisms have been proposed to address data randomization in large data domains. However, directly integrating these mechanisms with *HeavyGuardian* is still problematic in practice. The core concept behind these mechanisms revolves around mapping data from the large data domain to a smaller data domain using techniques such as hash functions [57], Hadamard matrix encoding [5, 6], or Bloom filter encoding [32]. Subsequently, data is randomized within this reduced data domain.

There are several issues with these approaches. Firstly, decoding a single randomized data on the server side entails an exhaustive scan of the entire data domain, which becomes computationally expensive for large data domains. Furthermore, this approach implies that the server must store the entirety of the data domain, which may contradict the requirement for bounded memory consumption on the server side. Additionally, these mechanisms introduce collisions when decoding randomized data for analysis. While such collisions are typically manageable in general frequency estimation tasks due to their uniform distribution, they can render strategies like the ED strategy and other space-saving techniques unusable. Assuming that data is mapped from a large domain of size d to a smaller data domain of size g, the average number of collision data generated by decoding a data point is d/g. In essence, the arrival density of an item directly impacts its potential to be recorded within the data structure as a hot item. If decoded data is mixed with $d/g - 1$ different data points, the true hot item may lose its advantage in being recorded within $\mathcal{HG}$. In scenarios where the domain size d is extremely large, such that d/g surpasses the size k of $\mathcal{HG}$, the entire scheme becomes untenable, and all data points are indiscriminately recorded with equal probability.

Consequently, it is desirable to develop novel LDP mechanisms capable of effectively randomizing data within large data domains and addressing challenges posed by the dynamically changing hot/cold items while optimizing the performance of *HeavyGuardian*.

4 ADVANCED LDP MECHANISM DESIGNS

In this section, we propose three novel advanced schemes to address the aforementioned problem in BGR by designing new randomization methods, which are outlined in Figure 3.

4.1 DSR (Domain-Shrinkage Randomization)

Tasks involving heavy hitter estimation in streams often assume that the streaming data follows a Zipf distribution [20, 24, 48]. This assumption aligns well with the distribution observed in various real-world scenarios, such as purchased goods and popular songs. In these contexts, the data domain predominantly consists of a few frequently occurring hot items, while most items are relatively

rare or never appear. However, the GRR mechanism in BGR randomizes a large number of hot items to these rare items for ensuring LDP, which leads to poor performance of the *ED strategy*. Furthermore, the protection of these rare items is critical since they often contain highly sensitive information. For instance, an individual might not be concerned about others knowing they have watched popular movies but may be apprehensive about revealing their interest in niche films, as it could inadvertently expose their personal preferences and hobbies. It is based on these observations that we have designed our advanced algorithm, DSR.

Specifically, we refer to the items recorded in $\mathcal{HG}$ as the hot items, and the items not in $\mathcal{HG}$ as the cold items. We observe that the *ED strategy* of $\mathcal{HG}$ does not need to know the specific value of the cold items in most cases. It only needs to reduce the count of the KV pair with the lowest frequency in $\mathcal{HG}$ by 1 with a certain probability when it receives a cold item. The *ED strategy* needs to know the specific value of the cold item to replace the KV pair in $\mathcal{HG}$ only when the weakest KV pair (with the lowest frequency) is going to be evicted. A direct idea is to represent all cold items as "$\perp$", and randomize the data on the domain $\{\mathcal{HG}.C\} \cup \{\perp\}$. When the weakest KV pair in $\mathcal{HG}$ is about to be evicted, it changes back to BGR to randomize the data on the entire domain. In this way, the size of the data domain can be reduced from d to $k+1$ when there is no KV pair in $\mathcal{HG}$ going to be replaced, which alleviates the low utility caused by a large domain.

Algorithm 2 DSR (RANDOMIZE)

Input: timestamp t, privacy budget ϵ, data domain Ω with size d, data structure $\mathcal{HG}$.
Output: r_i^t

Obtain the current raw data v_i^t;
if the least count $\mathcal{HG}[k].C \leq 1$ **then**
    $r_i^t \leftarrow \text{GRR}(v_i^t, \epsilon)$ ▷ Randomize data in Ω with GRR.
else
    Let $b \leftarrow Ber(\frac{e^\epsilon}{e^\epsilon + k})$
    if $b == 1$ **then** ▷ Randomize data in reduced domain.
        $r_i^t = v_i^t$
    else
        $r_i^t = v'$, where $v' \in \{\mathcal{HG}.ID\} \cup \{\perp\}$ and $v' \neq v_i^t$
return r_i^t

Algorithms. The RANDOMIZE Algorithm of DSR is presented in Algorithm 2. In the general case, the user randomizes data on the shrinking domain $\mathcal{HG}.C \cup \perp$. If the server receives a " $\perp$ ", it reduces the count of the weakest KV pair by 1 with a certain probability. However, this approach poses a challenge when the count of the weakest KV pair reduces to 0, as it becomes uncertain which cold item should replace the weakest KV pair. Furthermore, requiring the user to re-randomize the data across the entire domain can potentially violate ϵ-Local Differential Privacy (ϵ-LDP). To solve this problem, DSR requires users to switch to BGR for randomization on the entire data domain when the count of the weakest KV pair reaches 1 or less. In this case, as long as the count of the weakest KV pair is reduced by 1, it can be replaced by a new KV pair with a cold item directly. Users can subsequently switch back to randomizing data on the reduced domain once the new KV pair stabilizes (i.e., reaches a count > 1).

The INSERT and RESPONSE algorithms are shown in Algorithms 3 and 4, respectively. Due to the switch between two mechanisms with different parameters in the RANDOMIZE algorithm, a complex debiasing process is initiated during the insertion and response phases. Each switch between mechanisms necessitates debiasing of all the counts of KV pairs stored in $\mathcal{HG}$ using the

Algorithm 3 DSR (INSERT)

Input: timestamp t, privacy budget ϵ, data domain Ω with size d, reduced domain $\Omega_s = \{\mathcal{HG}.ID\} \cup \{\perp\}$, data structure $\mathcal{HG}$, number of the received data $\in \Omega$ num_{entire}, number of the received data $\in \Omega_s$ $num_{reduced}$.
Output: Updated $\mathcal{HG}$

$p_1 = \frac{e^\epsilon}{e^\epsilon+d-1}$, $q_1 = \frac{1}{e^\epsilon+d-1}$, $p_2 = \frac{e^\epsilon}{e^\epsilon+k}$, $q_2 = \frac{1}{e^\epsilon+k}$
Receive an incoming data r_i^t;
$\mathcal{HG} \leftarrow$ DSR_Insert(r_i^t, $\mathcal{HG}$, p_1, q_1, p_2, q_2, num_{entire}, $num_{reduced}$)
return Updated $\mathcal{HG}$

Algorithm 4 DSR (RESPONSE)

Input: timestamp t, privacy budget ϵ, data domain Ω with size d, data structure $\mathcal{HG}$, number of the received data *num*.
Output: *ResponseList*

$p_1 = \frac{e^\epsilon}{e^\epsilon+d-1}$, $q_1 = \frac{1}{e^\epsilon+d-1}$, $p_2 = \frac{e^\epsilon}{e^\epsilon+k}$, $q_2 = \frac{1}{e^\epsilon+k}$
if receive a Top-k query **then**
    $\mathcal{HG} \leftarrow$ DSR_FinalDebias($\mathcal{HG}$, p_1, q_1, p_2, q_2)
    for each $\mathcal{HG}[j] \in \mathcal{HG}$ **do**
        $ResponseList[j].ID \leftarrow \mathcal{HG}[j].ID$
        $ResponseList[j].C \leftarrow \mathcal{HG}[j].C$
return *ResponseList*

debiasing formula of the current mechanism. To prevent redundant debiasing of cumulative counts, it is imperative to multiply all the counts by the denominator of the debiasing formula of the new mechanism. For the sake of readability, the debiasing functions DSR_Insert in the INSERT algorithm and DSR_FinalDebias in the RESPONSE algorithm are deferred to Appendix A.5 in [46].

Theoretical Analysis. Theoretical analysis demonstrates that the error bound of DSR in the worst-case scenario aligns with that of BGR, as illustrated in Theorem 3.2. This can be attributed to the frequent replacement of the weakest KV pair for certain data distributions, compelling users to randomize data over the entire data domain for the majority of instances. However, DSR's improvement over BGR is expected to be more substantial for datasets exhibiting a more concentrated data distribution.

4.2 BDR (Budget-Division Randomization)

We present a novel scheme, BDR, that further enhances accuracy beyond DSR. Although DSR demonstrates improvement over BGR, it still predominantly randomizes data similarly to BGR when there are frequent changes to the items in $\mathcal{HG}$. Additionally, the complexity of debiasing is increased due to the transition between two randomization mechanisms with distinct parameters. Since the current cold value cannot be randomized and sent repeatedly, resulting in the waste of the privacy budget while awaiting new cold items. To address this problem, we designed a budget-division-based scheme (BDR) that efficiently avoids switching between different randomization mechanisms and mixing randomized data from different output data domains. Besides, we observe that the hot items stored by $\mathcal{HG}$ after initialization may not be true hot items. Through adjustments in the allocation of the privacy budget, BDR reduces the impact of the initial $\mathcal{HG}$ on the final result, with the probability of $\frac{k}{e^\epsilon+k}$ that any other item be randomized to the current "hot items".

We divide the privacy budget into two parts and run three sub-randomization mechanisms $\mathcal{M}_{judge}$, $\mathcal{M}_{hot}$, and $\mathcal{M}_{cold}$. Specifically, the $\mathcal{M}_{judge}$ mechanism is used to randomize whether the data is a hot item. If the $\mathcal{M}_{judge}$ mechanism determines that the data is a hot item, the $\mathcal{M}_{hot}$ mechanism is used to randomize the data in the data domain covered by items recorded in $\mathcal{HG}$.

Algorithm 5 BDR (RANDOMIZE)

Input: timestamp t, privacy budget ϵ, data domain Ω with size d, data structure $\mathcal{HG}$.
Output: r_i^t

Divide ϵ into ϵ_1 and ϵ_2, where $\epsilon_1 + \epsilon_2 = \epsilon$;
Obtain the current raw data v_i^t;
$Flag \leftarrow \mathcal{M}_{judge}(v_i^t, \epsilon_1)$ ▷ Determine whether v_i^t is hot or cold
if Flag==1 **then** ▷ v_i^t is determined as hot
    $r_i^t \leftarrow \mathcal{M}_{hot}(v_i^t, \epsilon_2)$
else if $Flag == 0$ and $\mathcal{HG}[k].C \leq 1$ **then** ▷ v_i^t is determined as cold and an item in $\mathcal{HG}$ is about to be evicted
    $r_i^t \leftarrow \mathcal{M}_{cold}(v_i^t, \epsilon_2)$
else
    $r_i^t = \perp$
return r_i^t

Algorithm 6 BDR (RANDOMIZE-$\mathcal{M}_{judge}$)

Input: raw data v_i^t, privacy budget ϵ_1, data structure $\mathcal{HG}$
Output: $Flag$

Let $b \leftarrow Ber(\frac{e^{\epsilon_1}}{e^{\epsilon_1}+1})$
if $b == 1$ **then**
    if $v_i^t \in \mathcal{HG}$ **then**
        $Flag = 1$
    else
        $Flag = 0$
else
    if $v_i^t \in \mathcal{HG}$ **then**
        $Flag = 0$
    else
        $Flag = 1$
return $Flag$

Algorithm 7 BDR (RANDOMIZE-$\mathcal{M}_{hot}$)

Input: raw data v_i^t, $Flag$, privacy budget ϵ_2, data structure $\mathcal{HG}$.
Output: r_i^t

Let $b \leftarrow Ber(\frac{e^{\epsilon_2}}{e^{\epsilon_2}+k-1})$
if $v_i^t \in \mathcal{HG}$ **then**
    if $b == 1$ **then**
        $r_i^t = v_i^t$
    else
        $r_i^t = v'$, where $v' \in \mathcal{HG}$ and $v' \neq v_i^t$
else
    $r_i^t = v'$, where v' is uniform random sampled from $\mathcal{HG}$
return r_i^t

The $\mathcal{M}_{cold}$ mechanism randomizes the items determined to be cold by the $\mathcal{M}_{judge}$ mechanism when an item in $\mathcal{HG}$ is about to be evicted. We show the overall flow of the RANDOMIZE algorithm in Algorithm 5, and the $\mathcal{M}_{judge}$, $\mathcal{M}_{hot}$, and $\mathcal{M}_{cold}$ mechanisms in Algorithm 6, Algorithm 7, and Algorithm 8, respectively.

Algorithm 8 BDR (RANDOMIZE-$\mathcal{M}_{cold}$)

Input: raw data v_i^t, privacy budget ϵ_2, data domain Ω with size d, data structure $\mathcal{HG}$.
Output: r_i^t

Let $b \leftarrow Ber(\frac{e^{\epsilon_2}}{e^{\epsilon_2}+d-k-1})$
if $v_i^t \notin \mathcal{HG}$ **then**
  if $b == 1$ **then**
    $r_i^t = v_i^t$
  else
    $r_i^t = v'$, where $v' \in \Omega/\mathcal{HG}$ and $v' \neq v_i^t$
else
  $r_i^t = v'$, where v' is uniform random sampled from $\Omega/\mathcal{HG}$
return r_i^t

Algorithms. At timestamp t, the user obtains the current raw data v_i^t, which is a hot item or a cold item. Note that the server can write the currently recorded hot items to a bulletin board in real time or the users can obtain the set of hot items from the *response module* at any time. Therefore, users can always know the current hot items and cold items when randomizing their data. Firstly, the user randomizes whether v_i^t is a hot item using the $\mathcal{M}_{judge}$ mechanism, which is a binary flip. The error introduced by the $\mathcal{M}_{judge}$ mechanism is independent of the size of the data domain. If the $\mathcal{M}_{judge}$ mechanism determines that v_i^t is a hot item, then v_i^t needs to be randomized on the data domain covered by items recorded in $\mathcal{HG}$ with the $\mathcal{M}_{hot}$ mechanism. If v_i^t is a hot item, $\mathcal{M}_{hot}$ mechanism randomizes it in the data domain covered by items recorded in $\mathcal{HG}$ as the general GRR. If v_i^t is actually a cold item, $\mathcal{M}_{hot}$ mechanism uniformly and randomly maps it to any item contained in $\mathcal{HG}$. Otherwise, the user sends "$\perp$" to the server if the $\mathcal{M}_{judge}$ mechanism determines that v_i^t is a cold item.

We consider a special case where the $\mathcal{M}_{judge}$ mechanism determines that v_i^t is a cold item, but the count of the weakest KV pair in $\mathcal{HG}$ is reduced to 0 by the *ED strategy*. Then the server would need this cold value to replace the item in $\mathcal{HG}$. Therefore, we provide the $\mathcal{M}_{cold}$ mechanism, similar to the $\mathcal{M}_{hot}$ mechanism, randomizing the data in the data domain covered by the cold items. When the user observes that the count of the weakest KV pair in $\mathcal{HG}$ is equal to or smaller than 1, the user uses $\mathcal{M}_{cold}$ mechanism to randomize v_i^t and then sends it to the server when v_i^t is determined to be cold. The $\mathcal{HG}$ has a high probability of replacing the weakest item with a cold item in this case. Note that the privacy budget consumed by $\mathcal{M}_{cold}$ is the remaining budget ϵ_2 at timestamp t, and the total privacy budget for v_i^t is still limited to ϵ. Figure 4 shows an example at 6 timestamps to illustrate the randomization process.

Next, we discuss how the *response module* on the server debiases the counts of hot items stored in $\mathcal{HG}$. Denote p_1 as the probability $\frac{e^{\epsilon_1}}{e^{\epsilon_1}+1}$, q_1 as the probability $\frac{1}{e^{\epsilon_1}+1}$, p_2 as the probability $\frac{e^{\epsilon_2}}{e^{\epsilon_2}+k-1}$, q_2 as the probability $\frac{1}{e^{\epsilon_2}+k-1}$. Let *num* denote the total number of data received by the server from the beginning of the statistics to the current timestamp, and γ_h denote the proportion of hot items. Let $\bar{f_v}$ be the noisy recorded count of item v, then the debiased estimation result $\tilde{f_v}$ is calculated as

$$\tilde{f_v} = \frac{\bar{f_v} - \gamma_h \cdot num(p_1 q_2 - q_1/k) - num \cdot q_1/k}{p_1(p_2 - q_2)} \tag{2}$$

Here, γ_h can be obtained from the warm-up round or the prior knowledge of data distribution, which is discussed in detail in Section 6. We show the details of the RESPONSE algorithm in Algorithm 9. Besides, we omit the details of the INSERT algorithm here since it is the same as that of BGR shown in Algorithm 1.

Algorithm 9 BDR (Response)

Input: timestamp t, privacy budget ϵ_1, ϵ_2, data domain Ω with size d, data structure $\mathcal{HG}$, number of the received data *num*.
Output: *ResponseList*
1: $p_1 = \frac{e^{\epsilon_1}}{e^{\epsilon_1}+1}$, $p_2 = \frac{e^{\epsilon_2}}{e^{\epsilon_2}+k-1}$, $q_1 = \frac{1}{e^{\epsilon_1}+1}$, $q_2 = \frac{1}{e^{\epsilon_2}+k-1}$
2: **if** receive a Top-k query **then**
3: **for** each $\mathcal{HG}[j] \in \mathcal{HG}$ **do**
4: $ResponseList[j].ID \leftarrow \mathcal{HG}[j].ID$
5: I
6: **return** l

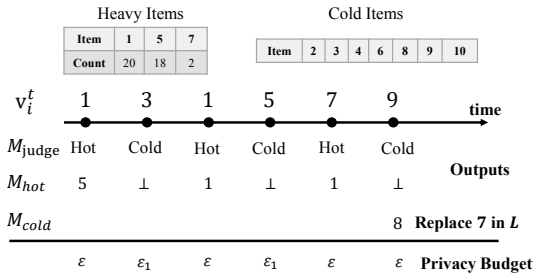

Fig. 4. An example of BDR.

Theoretical Analysis. We show that BDR satisfies ϵ-LDP as below.

Theorem 4.1. *BDR satisfies ϵ-LDP.*

Proof. Firstly, $\mathcal{M}_{judge}$ satisfies ϵ-LDP since $p_1/q_1 = e^{\epsilon_1}$. Secondly, M_{hot} satisfies ϵ_2-LDP since $p_2 k \leq p_2/q_2 = e^{\epsilon_2}$. Similarly, M_{cold} also satisfies ϵ_2-LDP. Therefore, BDR satisfies $(\epsilon_1 + \epsilon_2)$-LDP. The detailed proof is deferred to Appendix A.4 in [46]. □

Then we show the error bound of BDR in Theorem 4.2.

Theorem 4.2. *Given a stream prefix $\hat{S}_t$ with t items randomized by BDR satisfying ϵ-LDP and there is a data structure $\mathcal{HG}$ to store the Top-k items. Let v_i be the i^{th} hottest item, f_i be the real frequency of v_i, $\tilde{f}_i$ be the final estimated frequency of v_i. We have*

$$Pr[f_i - \tilde{f}_i \leq (3\sqrt{\frac{t\log(3/\beta)}{2}} + \alpha t) \cdot \frac{(e^{\epsilon_1}+1)(e^{\epsilon_2}+k-1)}{e^{\epsilon_1}(e^{\epsilon_2}-1)}]$$

$$\geq (1-\beta)(1 - \frac{1}{2\alpha}(1 - \sqrt{1 - \frac{4P_{weak}E(V)}{b-1}}))$$

where $P_{weak} = \frac{(i-1)!(d-k)!}{(d-1)!(i-k)!}$, $E(V) = \sum_{j=i+1}^{d} f_j$, α and β are small positive numbers with $\alpha, \beta \in (0,1)$.

Proof. The approach of the proof is similar to that of Theorem 3.2, the error bound of final debiased frequency $\tilde{f}_i$ compared to f_i can be obtained by combining the error bounds of $\hat{f}_i - \bar{f}_i$ and $f_i - \frac{\hat{f}_i - N_h p_1 q_2 - (t-N_h)\cdot\frac{q_1}{k}}{p_1(p_2-q_2)}$, where N_h is the number of hot items and $N_h \leq t$. The detailed proof is deferred to Appendix A.3 in [46]. □

The result of Theorem 4.2 shows that BDR significantly reduces the impact of the large data domain on the accuracy of the statistical results compared to BGR and DSR (Theorem 3.2).

Algorithm 10 CNR (RANDOMIZE)

Input: timestamp t, privacy budget ϵ, data domain Ω with size d, data structure $\mathcal{HG}$.
Output: r_i^t

Divide ϵ into ϵ_1 and ϵ_2, where $\epsilon_1 + \epsilon_2 = \epsilon$;
Obtain the current raw data v_i^t;
$Flag \leftarrow \mathcal{M}_{judge}(v_i^t, \epsilon_1)$ ▷ Determine whether v_i^t is hot or cold
if Flag==1 **then** ▷ v_i^t is determined as hot
    $r_i^t \leftarrow \mathcal{M}_{hot}(v_i^t, \epsilon_2)$
else ▷ v_i^t is determined as cold
    $r_i^t \leftarrow \mathcal{M}_{cold}(v_i^t, \epsilon_2)$
return r_i^t

Algorithm 11 CNR (INSERT)

Input: timestamp t, data domain Ω with size d, data structure $\mathcal{HG}$, number of the received data num.
Output: Updated $\mathcal{HG}$

Receive an incoming data r_i^t;
$num \leftarrow num + 1$
Insert r_i^t into Heavy part of $\mathcal{HG}$ following ED strategy;
if r_i^t not in Heavy part of $\mathcal{HG}$ **then**
    Insert r_i^t into Light part of $\mathcal{HG}$ following ED strategy;
if the least count in Heavy part $\mathcal{HG}[k].C \leq 0$ **then**
    Replace the weakest KV pair in Heavy part with the king KV pair in Light part, where their counts are set to 1.
return Updated $\mathcal{HG}$

4.3 CNR (Cold-Nomination Randomization)

In BDR, we find that the privacy budget ϵ_2 is unexploited when the data is determined to be a cold item and there is no item in $\mathcal{HG}$ that is about to be evicted, which can be observed in Figure 4. Besides, there is a light part in the original data structure of $\mathcal{HG}$ used to store the counts of cold items (see Figure 2). The length of this part λ_l is set to 0 in BGR, DSR, and BDR. Driven by these observations, we propose a new scheme CNR, which uses these two idle resources to further improve the accuracy over BDR.

Algorithms. Algorithm 10 shows the RANDOMIZE algorithm of CNR, similar to that of BDR. All the data determined as cold items by $\mathcal{M}_{judge}$ mechanism are randomized to specific cold items on the cold domain using $\mathcal{M}_{cold}$ mechanism, rather than calling $\mathcal{M}_{cold}$ mechanism only when there is a hot item to be evicted. Here, $\mathcal{M}_{judge}$, $\mathcal{M}_{hot}$, and $\mathcal{M}_{cold}$ are the same as Algorithms 6, 7, and 8 in BDR. When inserting the randomized items into $\mathcal{HG}$, the cold items that cannot be inserted into the heavy part are inserted into the light part following the ED strategy. Then the light part helps to provide a more accurate potential hot item to become a new hot item when a value in the heavy part is about to be evicted. Note that the light part only provides selected cold items, and its count is set to 1 when a cold item enters the heavy part, just the same as BDR. Thus, the debiasing formula of the counts in the heavy part is the same as that of the BDR, avoiding debiasing the randomized counts from different output domains like DSR. We show the INSERT in Algorithm 11, and the RESPONSE is the same as Algorithm 9.

Theoretical Analysis. Firstly, CNR still satisfies ϵ-LDP, and the privacy budget consumed by randomizing data is $\epsilon_1 + \epsilon_2 = \epsilon$. Then, the error bound of counts recorded in the heavy part is the same as Theorem 4.2 shown in BDR, since CNR only provides a better cold item to become a

new hot item when there is an item to be evicted. Note that all theoretical analyses for the error bound of the counts we provide only consider the error of the recorded counts without considering whether the items are true hot items. Since the accuracy of the hot items tracked by the scheme is influenced by both initial $\mathcal{HG}$ and data distributions, we evaluate it by conducting a comprehensive evaluation in Section 5.

Besides, CNR has no specific requirement for the length of the light part λ_l, as long as it satisfies $\lambda_l > 0$. The longer light part can provide more accurate new hot items to the heavy part. The setting of λ_h can refer to the original $\mathcal{HG}$ [62], or set a small constant according to the specific requirements. In our experiments, setting $\lambda_l = 5$ for finding Top-20 items on a concentrated data distribution can observe a significant improvement for small ϵ. Furthermore, the counters in the light part of $\mathcal{HG}$ are tailored for cold items, and the counter size is very small, e.g., 4 bits. Therefore, CNR does not increase too much additional memory consumption compared to the other schemes and still meets high memory efficiency.

5 EXPERIMENTAL EVALUATION

In this section, we design experiments to evaluate our proposed schemes. The evaluation mainly includes four aspects: (1) the accuracy of the heavy hitters via the proposed schemes; (2) the accuracy achieved by the proposed schemes compared with the baselines; (3) the impact of the key parameters on the accuracy of the proposed schemes; (4) the memory size consumed by the proposed schemes compared with the baselines. Towards these goals, we conduct experiments on both synthetic and real-world datasets, and simulate to collect streaming data from users at continuous timestamps for heavy hitter analysis. Besides, we introduce different metrics to evaluate the accuracy of the results from three different aspects.

To better guide the application of the schemes in practice, we also conduct supplementary experiments on more datasets and test the computation and communication overheads. Please refer to Appendix A.8 in [46] for details.

5.1 Setup

Datasets. We run experiments on the following datasets:

- Several synthetic datasets are generated with two different distributions and three domain sizes. One kind of datasets are generated by randomly sampling data from a Normal distribution with variance $\sigma = 5$, and others are generated from an Exponential distribution with variance $\sigma = 10$. There are $n = 100,000$ values in each dataset.
- Retail dataset [1] contains the retail market basket data from an anonymous Belgian retail store with around 0.9 million values and $16k$ distinct items.
- Kosarak dataset [3] contains the click streams on a Hungarian website, with around 8 million values and $42k$ URLs.
- Webdocs dataset [2] is constructed from a collection of web HTML documents, which comprises around 300 million records, and 5.26 million distinct items.

Metrics. In reality, various applications focus on different aspects of the heavy hitter estimation results. Therefore, we have to comprehensively evaluate the quality of the results from three aspects: (1) how accurately that $\mathcal{HG}$ captures the actual heavy hitters; (2) how accurately that the ordering of the heavy hitters in $\mathcal{HG}$; (3) how accurately that $\mathcal{HG}$ captures the actual counts of heavy hitters. We use the following three metrics to cover each aspect:

Precision. It measures the accuracy of the actual heavy hitters captured by $\mathcal{HG}$. It is the number of actual heavy hitters divided by the number of all items in $\mathcal{HG}$, as given by

$$Precision = \frac{\#Actual\ heavy\ hitters\ in\ \mathcal{HG}}{\#Heavy\ hitters}.$$

Normalized Discounted Cumulative Gain (NDCG). It measures the ordering quality of the heavy hitters captured by $\mathcal{HG}$, which is a common effectiveness in recommendation systems and other related applications. *NDCG* is between 0 and 1 for all k, and the closer it is to 1 means the ordering quality of $\mathcal{HG}$ is higher. The formulas for calculating NDCG is deferred to Appendix A.6 in [46].

Average Absolute Error (AAE). It measures the error of the counts of the actual Top-k items with their estimated counts recorded in $\mathcal{HG}$, which can be calculated as

$$AAE_k = \frac{1}{k}\sum_{i=1}^{k} |f_{actual}(v_i) - f_{estimated}(v_i)|.$$

If an actual hot item is not recorded by $\mathcal{HG}$, its AAE is calculated by setting the estimated count as 0. For consistent and fair comparisons, we post-process all counts recorded by $\mathcal{HG}$ to 0 when calculating AAE. All results in experiments are averaged with 20 repeats.

5.2 Implementation Details

We fully implemented our schemes and all baselines in Java to provide unified concrete performance comparisons. For all schemes, we separately implement the server and the client side, and the perturb data for communication are serialized to 'byte[]'. This makes our implementation easier to be deployed in practice, in which the server and clients would communicate via network channels using byte strings. In our experiments, focus more on the effectiveness of our schemes so that we run the server and the client on a single process. All experiments are run on Ubuntu 20.04 with 96 Intel Xeon 2.20 GHz CPU and 256 GB RAM. Our source code is available for public request. Besides, we have some improvements compared with the original implementation in our re-implementation for both LDP mechanisms and original *HeavyGuardian*. More implementation details are deferred to Appendix A.7 in [46].

5.3 Analysis of Experimental Results

Comparison of Accuracy. We compare the accuracy of the baseline scheme and three advanced schemes with the non-private *HeavyGuardian* and two LDP mechanisms: Generalized Randomized Response (GRR) and Hadamard Response (HR) (HR performs the best in our evaluation, see Figure 11 in Appendix A.7 in [46]). We evaluate all schemes on the Synthetic, Retail, Kosarak, and Webdocs datasets. The results for three metrics: NDCG, Precision, and AAE are shown in Figure 5, Figure 6, and Figure 7, respectively. Since running GRR and HR exceeds the computing or storage capabilities of our server, we only show the results of our schemes on the Webdocs dataset. In each figure, we vary the privacy budget ϵ within a range of $[0.5, 5]$. All schemes involve a warm-up stage for fairness of the comparison.

Firstly, we observe that the accuracy of the proposed schemes BGR, DSR, BDR, and CNR improves sequentially. The improvement of DSR compared with BGR is more obvious as ϵ increases, and the advantage of CNR over BDR is more significant as ϵ decreases. We think the reason is that when ϵ is large, i.e., $\epsilon > 1$, the randomized hot items are still concentrated and there are fewer times to randomize on the entire domain to provide specific cold items for replacing with the weakest hot items in $\mathcal{HG}$, thus the improvement achieved by DSR is relatively significant. When ϵ is small, i.e., $\epsilon < 1$, the distribution of the randomized data is relatively uniform, thus the weakest hot item in $\mathcal{HG}$ always need to be replaced. In this case, the advantage of CNR compared to BDR in

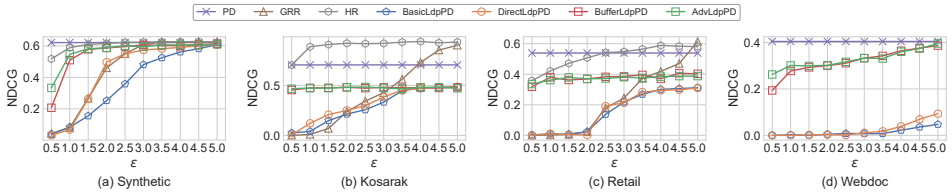

Fig. 5. Evaluation of NDCG for Top-20 on both synthetic and real-world datasets while taking 1% data for warm-up stage.

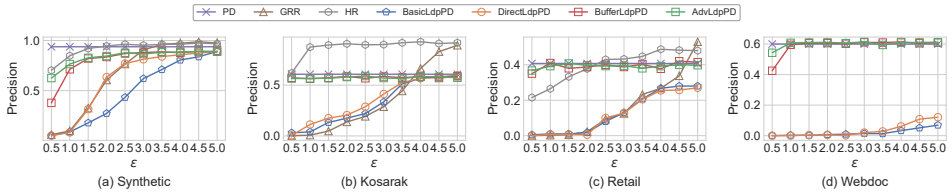

Fig. 6. Evaluation of Precision for Top-20 on both synthetic and real-world datasets while taking 1% data for warm-up stage.

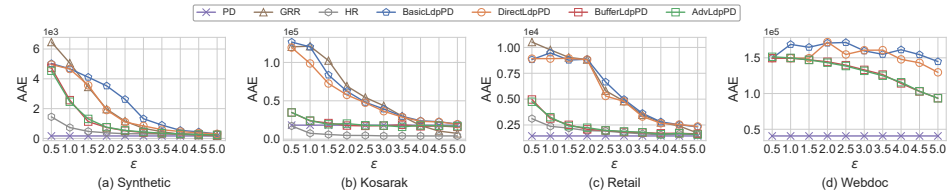

Fig. 7. Evaluation of AAE for Top-20 on both synthetic and real-world datasets while taking 1% data for warm-up stage.

providing more potential cold items to enter $\mathcal{HG}$ can be more obvious. Besides, we find that these observations are not pronounced on two real-world datasets. The reason is that those real-world datasets have large data domains and irregular data distributions. Therefore, $\mathcal{HG}$ needs to replace the items frequently even if the ϵ is relatively large. This means that DSR always randomizes the data on the entire domain in the same way as BGR. In addition, the large data domain can also lead to low accuracy in the light part of $\mathcal{HG}$. Then the performance of the CNR is similar to BDR in this case.

Secondly, compared with the non-private *HeavyGuardian* and memory-unlimited LDP randomization mechanisms, BDR and CNR outperform GRR on all datasets in terms of all metrics when $\epsilon < 3$. Moreover, their accuracy on the synthetic dataset is close to HR, and the accuracy on all datasets is close to non-private *HeavyGuardian*. In all three datasets, BDR and CNR are set to $\epsilon_1/\epsilon_2 = 0.5$, and their parameter γ_h is calculated during the warm-up stage. We also observe that the performance of BGR and DSR gradually dominates that of GRR as the size of the data domain increases when $\epsilon < 3.5$. However, their accuracy is much lower than that of BDR and CNR when the domain size is extremely large.

Finally, we observe that the NDCG of all schemes is slightly lower than their Precision on all datasets. The main reason is that NDCG considers the ordering weights of the hit items in addition to whether the true hot items are hit or not. Besides, the comparison results of all schemes in terms of AAE on all datasets are consistent with the comparison of NDCG and Precision. The AAE of the statistical results of BGR, DSR, BDR, and CNR decreases in turn.

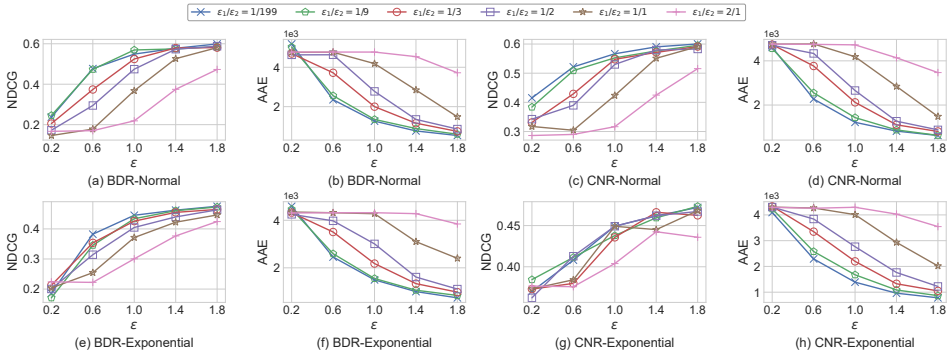

Fig. 8. Accuracy of BDR and CNR vs. different allocations of privacy budget; Conducted on the two synthetic datasets with Normal distribution and Exponential distribution, where the domain size $d = 1000$, and taking 1% data for warm-up stage.

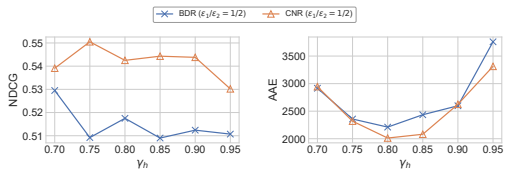

Fig. 9. The impact of parameter γ_h on accuracy of BDR and CNR. The evaluation is conducted on the synthetic dataset with normal distribution, where taking 1% data for warm-up stage and $\epsilon = 0.6$.

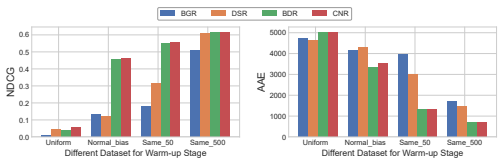

Fig. 10. The impact of the warm-up stage on the accuracy of the proposed schemes for tracking Top-k items on synthetic dataset with normal distribution, where $\epsilon = 2$.

Impact of Key Parameters on the Accuracy. We evaluate the impact of key parameters on the accuracy of the proposed schemes by varying them within a certain range. In order to eliminate the interference of irregular distribution on the evaluation, we conduct experiments on several synthetic datasets. Due to limited space, we only present the NDCG and Precision of the statistical results on two synthetic datasets with Normal distribution and Exponential distribution. The results of all metrics on more synthetic datasets with different domain size are deferred to Appendix A.8 in [46].

Firstly, Figure 8 shows the impact of the allocation method of privacy budget ϵ on the accuracy of BDR and CNR. We observe that BDR and CNR allocate less privacy budget to ϵ_1 and more privacy budget to ϵ_2 can obtain higher accuracy of the statistical results. The improvement of NDCG is significant when ϵ_1/ϵ_2 decreases from $2/1$ to $1/9$, and the increase slows down after ϵ_1/ϵ_2 is less than $1/9$. We think the reason is that a hot item recorded in $\mathcal{HG}$ is randomized to a cold item with a greater probability when ϵ_1 is small, and the number of data that is a hot item is larger than data that is a cold item, which leads to the items in $\mathcal{HG}$ are easier to be evicted. Meanwhile, increasing ϵ_2 can improve the correctness of the orders of the items recorded in $\mathcal{HG}$. Therefore, reducing

Table 1. Comparison of memory size (KB) consumed by schemes on different datasets.

Scheme \ Dataset	Synthesize	Kosarak	Webdocs
HeavyGuardian	2.40	2.40	2.41
GRR	153.33	6154.99	-
HR	153.41	6249.77	-
BGR	**2.66**	**2.66**	**2.67**
DSR	**2.73**	**2.73**	**2.75**
BDR	**2.69**	**2.68**	**2.70**
CNR	**3.43**	**3.09**	**3.43**

ϵ_1/ϵ_2 can increase the probability that the hot items with a small count are replaced by other cold items, so that the real hot items can occupy the $\mathcal{HG}$ faster. This is also consistent with the experimental results in original $\mathcal{HG}$ (Figure 4(a), [62]). The accuracy of the result increases when the parameter b is reduced to make it easier for the new item to enter $\mathcal{HG}$, but the improvement becomes no longer obvious when b is reduced to a certain small value. Therefore, we recommend setting $\epsilon_1/\epsilon_2 = 1/9$ to get near-optimal accuracy in the actual deployment of BDR and CNR. We also conduct the evaluations on synthetic datasets with different domain sizes, and obtain the consistent observations with the above. The results are shown in Figure 12-Figure 17 in Appendix A.8 in [46]. Moreover, we find that increasing the domain size has some impact on the accuracy of the schemes, but we can still improve the accuracy by adjusting the privacy budget allocation.

Then Figure 9 shows the impact of the parameter γ_h on the accuracy of the BDR and CNR. We calculate the exact $\gamma_h \simeq 0.92$. As a debiasing parameter, γ_h directly affects the counts of the statistical result, so the impact of γ_h can be clearly observed from the AAE of the result. However, the indirect impact on NDCG is not obvious, the lines in the figure are fluctuating. An interesting phenomenon can be observed from the AAE of the results. More accurate γ_h does not necessarily give more accurate count of the result. The reason is that the ED strategy continuously reduces the counts of the weakest item with a certain probability, which causes the statistical results to be underestimated. According to debiasing Equation 2, reducing γ_h can cause the debiased result to be over-estimated, thereby offsetting part of the bias introduced by the ED strategy.

Finally, Figure 10 shows the impact of the warm-up stage on the accuracy of the baseline BGR and the proposed three schemes. We compared their accuracy using five different datasets for the warm-up stage. The five datasets include a uniformly random dataset with the size of 50, a dataset with the size of 50 and distribution skewed from the true normal distribution, and two datasets with the true normal distribution with sizes of 50 and 500. We can observe that their accuracy increases as the distribution of the dataset used in the warm-up stage approaches the true distribution and as the size of the dataset increases. Specifically, BDR and CNR set $\epsilon_1/\epsilon_2 = 0.5$, and they are least affected by the warm-up stage among all schemes. We think the reason is that the current cold items in BDR and CNR are easier to enter $\mathcal{HG}$ to become new hot items, which reduces the impact of the accuracy of the initial $\mathcal{HG}$ on the final statistical result. Similar observations can be obtained on the real-world datasets, and the results are shown in Figure 18 in Appendix A.8 in [46].

Comparison of Memory Consumption. We then evaluate the total memory size consumed by all schemes when tracking Top-20 heavy hitters on the four different datasets. We present the results in Table 1. The proposed schemes show a significant advantage in memory consumption when the data domain is large, such as in the Kosarak and Webdocs datasets. We can observe that the memory size consumed by the GRR and HR increases linearly as the domain size d increases. In contrast, the memory consumed by all space-saving schemes is only related to the number of

tracked heavy hitters k, and k is usually much smaller than d. Note that the GRR and HR do not have memory consumption available for the Webdocs dataset since the computation or memory requirements of these schemes exceeded the capacity of the server used for testing. Additionally, we conduct tests to evaluate the computation and communication overhead of all schemes. The detailed results of these tests can be found in Appendix A.8 in [46].

6 DISCUSSION

In this section, we supplementally discuss more details about the practical implementation and the potential extension of these schemes.

6.1 System Parameters & Implementations

Warm-up Stage. In practice, our framework along with the theoretical designs and analyses are applied to a steady state where the $\mathcal{HG}$ are filled during previous timestamps, rather than dealing with the cold-start scenario where $\mathcal{HG}$ is empty. Therefore, to simulate such a steady state where $\mathcal{HG}$ is properly warm-started, we consider all the proposed schemes include a warm-up round at the beginning of the statistics. Note that although CNR needs to use the light part in $\mathcal{HG}$ structure, only the heavy part should be filled in the warm-up stage like other schemes. The data for the warm-up stage can be a priori dataset stored on the server or data voluntarily contributed by users in the first round of the statistics. There is no specific requirement for the number of data in the warm-up stage. The only requirement is that the data should at least be able to fill the $\mathcal{HG}$. Besides, the closer the distribution of the priori dataset to the real data distribution, or the larger the number of data that users voluntarily contribute, the higher the accuracy of $\mathcal{HG}$ in the subsequent statistics.

Parameter γ_h in BDR and CNR. The debiasing formula for both BDR and CNR contains a parameter λ_h, which is the proportion of data that is the hot item in the stream. The server actually does not know the specific value of γ_h, but it can be theoretically calculated based on prior knowledge about the data distribution. If the server has no prior knowledge about the data distribution, γ_h can also be statistically obtained from the initial $\mathcal{HG}$ after the warm-up stage. Certainly, γ_h obtained by the above two methods both inevitably introduce additional errors to the estimated results, and the impact is evaluated in the experiments. However, the current design of schemes cannot avoid it, and we leave it for future work.

Privacy Parameters ϵ_1, ϵ_2 in BDR and CNR. Next, we analyze how to split the privacy budget ϵ into ϵ_1 and ϵ_2 in BDR and CNR, based on insights from our theoretical and experimental results.

Our theoretical analysis in Theorem 4.2, provides an error bound for estimating the count of hot items in BDR, which is equally applicable to CNR. It shows that allocating a larger portion of the privacy budget to ϵ_2 leads to a reduced error bound, which is further corroborated by our experimental results in Figure 8(b)(d)(f)(h). In fact, the count error only focuses on the accuracy of counts for hot items already identified by the data structure. This calculation excludes errors coming from the misclassification of hot items due to randomization with ϵ_1.

However, the estimation is complex when considering the impact of ϵ_1 and ϵ_2 on the precision of the data structure $\mathcal{HG}$ in capturing the true hot items. Increasing the privacy budget allocated to ϵ_1 does reduce the probability of determining hot data as cold and simultaneously enhances the probability that currently recorded items remain within $\mathcal{HG}$. Nevertheless, this does not necessarily get an improved precision in capturing items within $\mathcal{HG}$. The setting of parameter b in $\mathcal{HG}$ [62] faces the same dilemma. Increasing b will reduce the probability of the current cold values entering $\mathcal{HG}$, and vice versa. Multiple factors collaboratively impact the precision of $\mathcal{HG}$ in capturing hot items. For instance, when the initial $\mathcal{HG}$ captures inaccurate hot items, a higher probability of eviction among recorded items within $\mathcal{HG}$ can lead to improved precision; if the true hot items

are concentrated in the first half of the data stream, a higher probability of retention for items within $\mathcal{HG}$ can result in higher precision. It can also be observed from the experimental results that the Precision and NDCG of the results on some data streams are not as regular as those of AAE as ϵ_1 and ϵ_2 change, i.e., when hot items are distributed in a more dispersed manner within the Exponential distribution as opposed to the Normal distribution, the NDCG depicted in Figure 8 emphasize that allocating a smaller fraction of ϵ_1 does not confer any discernible advantage. In [62], they provide an empirical value, i.e., $b = 1.08$. Based on our comprehensive evaluations, we suggest setting $\epsilon_1/\epsilon_2 = 0.5$ in most scenarios can achieve promising accuracy.

Guidance on Scheme Selection. In this paper, we introduce three enhanced schemes, each making distinct trade-offs between accuracy, computational overhead, and memory usage. According to our theoretical and experimental results, we summarize a table, as detailed in Table 2, including three advanced designs and a baseline in terms of accuracy, computation overhead, and memory consumption. From the baseline BGR to DSR, BDR, and CNR, there is a sequential improvement in the accuracy of the results. Meanwhile, this enhancement comes at the cost of increased computational complexity on the client side or memory consumption on the server side. In practical deployment, we recommend selecting a scheme based on the specific performance requirements of the task.

Table 2. Performance comparison of baseline method and proposed schemes.

	BGR	DSR	BDR	CNR
Accuracy	4th	3rd	2nd	1st
Computation Overheads	1st	2nd	3rd	4th
Memory Consumption	1st	1st	1st	2nd

6.2 Extensions

w-Event-Level and User-Level Privacy. While the schemes proposed in this paper offer event-level privacy guarantees, they possess the flexibility to be extended to offer enhanced privacy protection, including w-event-level privacy and user-level privacy. Specifically, w-event-level privacy ensures ϵ-LDP within any sliding window of size w, while user-level privacy guarantees ϵ-LDP for all streaming data contributed by an individual user.

To achieve w-event-level privacy and user-level privacy for finite data streams, we could distribute the privacy budget evenly across each timestamp. This entails changing the privacy budget used for randomizing each streaming data point from ϵ to ϵ/w and ϵ/l, where l represents the length of the finite data stream. We have to mention that while there are existing methods that outperform the average allocation approach [34, 35, 42], applying them to our proposed schemes presents certain challenges. The primary obstacle lies in the variation of privacy budgets used to randomize each streaming data, which can impede the server to debias the accumulated counts in the heavy list. This complication also obstructs the application of the schemes to provide user-level privacy for infinite data streams. An intuitive approach to address this issue is that the server to independently debias each incoming streaming data point using the privacy budget transmitted by the user concurrently. However, this approach may introduce increased computational complexity on the server's end and heightened communication complexity for the user. We leave this challenge for future research and exploration.

Other Tasks. Since the proposed framework HG-LDP focuses on the heavy hitter estimation task, only CNR involves the Light part of the data structure $\mathcal{HG}$ to store the counts of part of

cold items. When CNR extends its functionality to store the counts of all cold items in the Light part as in [62], it can also support other tasks supported in [62], such as frequency estimation and frequency distribution estimation. It's essential to note that these tasks, even functionally supported, encounter a challenge related to accuracy when randomizing within large data domains. The new LDP randomization mechanisms in this paper are designed by utilizing the characteristics of the heavy hitter tasks to only ensure the accuracy of hot items. We intend to delve deeper into this aspect as part of our future research efforts.

7 RELATED WORK

An extended Related Work is in Appendix A.9 in [46].

Differential Private Data Stream Collection The earliest studies in differential privacy for streaming data collection originate from continuous observation of private data [8, 26, 29, 32, 33]. Recent works on differential private data stream collection mainly focus on Centralized Differential Privacy (CDP). Some works study how to publish the summation of the streaming data privately [34, 42, 50, 58]. Some works study the release of correlated streaming data [9, 56] propose a correlated Gaussian noise mechanism. Some recent works focus on data stream collection with Local Differential Privacy (LDP) [40, 52, 58].

Tracking Heavy Hitters in Data stream Mining streaming data faces three principal challenges: *volume*, *velocity*, and *volatility* [43]. The existing heavy hitters estimation algorithms in the data stream can be divided into three classes: Counter-based algorithms, Quantile algorithms, and Sketch algorithms [20]. Counter-based algorithms track the subset of items in the stream, and they quickly determine whether to record and how to record with each new arrival data [48, 49, 62, 65]. The Quantile algorithms [38, 54] focus on finding the item which is the smallest item that dominates ϕn items from the data stream. Sketch algorithms [7, 18, 23, 45] record items with a data structure, which can be thought of as a linear projection of the input, hash functions are usually used to define that. However, the sketch algorithms involve a large number of hash operations, which cannot meet the timeliness requirements of streaming data. Besides, all items are recorded and additional information needs to be stored for retrieval, which leads to unnecessary memory consumption [20]. Our design is based on Counter-based algorithms with an extended setting where streaming data is protected by LDP.

8 CONCLUSION

In this paper, we proposed a framework HG-LDP for tracking the Top-k heavy hitters on data streams at bounded memory expense, while providing rigorous LDP protection. A baseline and three advanced schemes with new LDP randomization mechanisms are designed under the hood of the framework. We implement all the proposed schemes and evaluate them on both synthetic and real-world datasets in terms of accuracy and memory consumption. The experimental results demonstrated that the proposed schemes achieve a satisfactory "accuracy-privacy-memory efficiency" tradeoff. For future work, we will extend the framework to be compatible with more diverse selections of memory-efficiency data structures as well as broader types of statistical tasks to enhance its flexibility.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China under Grant 2021YFB3100300, and the National Natural Science Foundation of China under Grant U20A20178 and 62072395. Weiran Liu is supported in part by the Major Programs of the National Social Science Foundation of China under Grant 22&ZD147. Yuan Hong is supported in part by

the National Science Foundation under Grants CNS-2308730, CNS-2302689, CNS-2319277, CMMI-2326341 and the Cisco Research Award. Additionally, we would like to express our sincere gratitude to the anonymous reviewers for their valuable time, insightful comments, and constructive feedback, which greatly contributed to the enhancement of the quality and rigor of this paper.

REFERENCES

[1] 1999. Frequent Itemset Mining Dataset Repository, retail. http://fimi.uantwerpen.be/data/.
[2] 2004. Frequent Itemset Mining Dataset Repository, webdocs. http://fimi.uantwerpen.be/data/webdocs.dat.gz.
[3] 2015. Frequent Itemset Mining Dataset Repository, kosarak. http://fimi.uantwerpen.be/data/.
[4] 2023. How Many Videos Are on YouTube: Exploring the Vast Digital Landscape. https://www.techpluto.com/how-many-videos-are-on-youtube/.
[5] Jayadev Acharya and Ziteng Sun. 2019. Communication complexity in locally private distribution estimation and heavy hitters. In *International Conference on Machine Learning*. PMLR, 51–60.
[6] Jayadev Acharya, Ziteng Sun, and Huanyu Zhang. 2019. Hadamard response: Estimating distributions privately, efficiently, and with little communication. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 1120–1129.
[7] Noga Alon, Yossi Matias, and Mario Szegedy. 1999. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences* 58, 1 (1999), 137–147.
[8] Nikhil Bansal, Don Coppersmith, and Maxim Sviridenko. 2008. Improved approximation algorithms for broadcast scheduling. *SIAM J. Comput.* 38, 3 (2008), 1157–1174.
[9] Ergute Bao, Yin Yang, Xiaokui Xiao, and Bolin Ding. 2021. CGM: an enhanced mechanism for streaming data collection with local differential privacy. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2258–2270.
[10] Ran Ben Basat, Gil Einziger, Isaac Keslassy, Ariel Orda, Shay Vargaftik, and Erez Waisbard. 2022. Memento: Making Sliding Windows Efficient for Heavy Hitters. *IEEE/ACM Trans. Netw.* 30, 4 (2022), 1440–1453. https://doi.org/10.1109/TNET.2021.3132385
[11] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. 2017. Practical Locally Private Heavy Hitters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 2288–2296.
[12] Ran Ben-Basat, Gil Einziger, Roy Friedman, and Yaron Kassner. 2016. Heavy hitters in streams and sliding windows. In *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016*. IEEE, 1–9. https://doi.org/10.1109/INFOCOM.2016.7524364
[13] Vladimir Braverman, Stephen R. Chestnut, Nikita Ivkin, and David P. Woodruff. 2016. Beating CountSketch for heavy hitters in insertion streams. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, Daniel Wichs and Yishay Mansour (Eds.). ACM, 740–753. https://doi.org/10.1145/2897518.2897558
[14] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. 2010. Private and Continual Release of Statistics. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 6199)*, Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis (Eds.). Springer, 405–417. https://doi.org/10.1007/978-3-642-14162-1_34
[15] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Comput. Surv.* 41, 3 (2009), 15:1–15:58. https://doi.org/10.1145/1541880.1541882
[16] Lisi Chen and Gao Cong. 2015. Diversity-Aware Top-k Publish/Subscribe for Text Stream. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, Timos K. Sellis, Susan B. Davidson, and Zachary G. Ives (Eds.). ACM, 347–362. https://doi.org/10.1145/2723372.2749451
[17] Yan Chen, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2017. PeGaSus: Data-Adaptive Differentially Private Stream Processing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM, 1375–1388. https://doi.org/10.1145/3133956.3134102
[18] Yun Chi, Haixun Wang, Philip S Yu, and Richard R Muntz. 2004. Moment: Maintaining closed frequent itemsets over a stream sliding window. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*. IEEE, 59–66.
[19] Graham Cormode. 2011. Sketch techniques for approximate query processing. *Foundations and Trends in Databases. NOW publishers* (2011), 15.
[20] Graham Cormode and Marios Hadjieleftheriou. 2008. Finding frequent items in data streams. *Proc. VLDB Endow.* 1, 2 (2008), 1530–1541. https://doi.org/10.14778/1454159.1454225

[21] Graham Cormode, Flip Korn, S. Muthukrishnan, and Divesh Srivastava. 2003. Finding Hierarchical Heavy Hitters in Data Streams. In *Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003, Berlin, Germany, September 9-12, 2003*, Johann Christoph Freytag, Peter C. Lockemann, Serge Abiteboul, Michael J. Carey, Patricia G. Selinger, and Andreas Heuer (Eds.). Morgan Kaufmann, 464–475. https://doi.org/10.1016/B978-012722442-8/50048-3

[22] Graham Cormode, Samuel Maddock, and Carsten Maple. 2021. Frequency Estimation under Local Differential Privacy. *Proc. VLDB Endow.* 14, 11 (2021), 2046–2058. https://doi.org/10.14778/3476249.3476261

[23] Graham Cormode and Shan Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms* 55, 1 (2005), 58–75.

[24] Graham Cormode and S. Muthukrishnan. 2005. What's hot and what's not: tracking most frequent items dynamically. *ACM Trans. Database Syst.* 30, 1 (2005), 249–278. https://doi.org/10.1145/1061318.1061325

[25] Gautam Das, Dimitrios Gunopulos, Nick Koudas, and Nikos Sarkas. 2007. Ad-hoc Top-k Query Answering for Data Streams. In *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*, Christoph Koch, Johannes Gehrke, Minos N. Garofalakis, Divesh Srivastava, Karl Aberer, Anand Deshpande, Daniela Florescu, Chee Yong Chan, Venkatesh Ganti, Carl-Christian Kanne, Wolfgang Klas, and Erich J. Neuhold (Eds.). ACM, 183–194.

[26] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting telemetry data privately. *Advances in Neural Information Processing Systems* 30 (2017).

[27] Cynthia Dwork. 2006. Differential Privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 4052)*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer, 1–12. https://doi.org/10.1007/11787006_1

[28] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. 2010. Differential privacy under continual observation. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, Leonard J. Schulman (Ed.). ACM, 715–724. https://doi.org/10.1145/1806689.1806787

[29] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. 2010. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*. 715–724.

[30] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3-4 (2014), 211–407. https://doi.org/10.1561/0400000042

[31] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM, 1054–1067. https://doi.org/10.1145/2660267.2660348

[32] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1054–1067.

[33] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. 2003. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 211–222.

[34] Farhad Farokhi. 2020. Temporally discounted differential privacy for evolving datasets on an infinite horizon. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 1–8.

[35] Shuya Feng, Meisam Mohammady, Han Wang, Xiaochen Li, Zhan Qin, and Yuan Hong. 2023. DPI: Ensuring Strict Differential Privacy for Infinite Data Streaming. *arXiv preprint arXiv:2312.04738* (2023).

[36] Jennifer Gillenwater, Matthew Joseph, Andres Muñoz Medina, and Mónica Ribero Diaz. 2022. A Joint Exponential Mechanism For Differentially Private Top-k. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.). PMLR, 7570–7582.

[37] Andy Greenberg. 2016. Apple's 'differential privacy'is about collecting your data—but not your data. *Wired, June* 13 (2016).

[38] Michael Greenwald and Sanjeev Khanna. 2001. Space-efficient online computation of quantile summaries. *ACM SIGMOD Record* 30, 2 (2001), 58–66.

[39] Cheqing Jin, Ke Yi, Lei Chen, Jeffrey Xu Yu, and Xuemin Lin. 2010. Sliding-window top-k queries on uncertain streams. *VLDB J.* 19, 3 (2010), 411–435. https://doi.org/10.1007/s00778-009-0171-0

[40] Matthew Joseph, Aaron Roth, Jonathan Ullman, and Bo Waggoner. 2018. Local differential privacy for evolving data. *Advances in Neural Information Processing Systems* 31 (2018).

[41] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2008. What Can We Learn Privately?. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*. IEEE Computer Society, 531–540. https://doi.org/10.1109/FOCS.2008.27

[42] Georgios Kellaris, Stavros Papadopoulos, Xiaokui Xiao, and Dimitris Papadias. 2014. Differentially Private Event Sequences over Infinite Streams. *Proc. VLDB Endow.* 7, 12 (2014), 1155–1166. https://doi.org/10.14778/2732977.2732989

[43] Georg Krempl, Indre Žliobaite, Dariusz Brzeziński, Eyke Hüllermeier, Mark Last, Vincent Lemaire, Tino Noack, Ammar Shaker, Sonja Sievi, Myra Spiliopoulou, et al. 2014. Open challenges for data stream mining research. *ACM SIGKDD explorations newsletter* 16, 1 (2014), 1–10.

[44] Haoyu Li, Qizhi Chen, Yixin Zhang, Tong Yang, and Bin Cui. 2022. Stingy Sketch: A Sketch Framework for Accurate and Fast Frequency Estimation. *Proc. VLDB Endow.* 15, 7 (2022), 1426–1438.

[45] Jizhou Li, Zikun Li, Yifei Xu, Shiqi Jiang, Tong Yang, Bin Cui, Yafei Dai, and Gong Zhang. 2020. WavingSketch: An Unbiased and Generic Sketch for Finding Top-k Items in Data Streams. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 1574–1584. https://doi.org/10.1145/3394486.3403208

[46] Xiaochen Li, Weiran Liu, Jian Lou, Yuan Hong, Lei Zhang, Zhan Qin, and Kui Ren. 2023. Local Differentially Private Heavy Hitter Detection in Data Streams with Bounded Memory (full version). *arXiv preprint arXiv:2311.16062* (2023).

[47] Hongyan Liu, Xiaoyu Wang, and Yinghui Yang. 2010. Comments on "an integrated efficient solution for computing frequent and top-k elements in data streams". *ACM Trans. Database Syst.* 35, 2 (2010), 15:1–15:4. https://doi.org/10.1145/1735886.1735894

[48] Gurmeet Singh Manku and Rajeev Motwani. 2002. Approximate Frequency Counts over Data Streams. In *Proceedings of 28th International Conference on Very Large Data Bases, VLDB 2002, Hong Kong, August 20-23, 2002*. Morgan Kaufmann, 346–357. https://doi.org/10.1016/B978-155860869-6/50038-X

[49] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. 2005. Efficient Computation of Frequent and Top-k Elements in Data Streams. In *Database Theory - ICDT 2005, 10th International Conference, Edinburgh, UK, January 5-7, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3363)*, Thomas Eiter and Leonid Libkin (Eds.). Springer, 398–412. https://doi.org/10.1007/978-3-540-30570-5_27

[50] Victor Perrier, Hassan Jameel Asghar, and Dali Kaafar. 2018. Private continual release of real-valued data streams. *arXiv preprint arXiv:1811.03197* (2018).

[51] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2016. Heavy Hitter Estimation over Set-Valued Data with Local Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM, 192–203. https://doi.org/10.1145/2976749.2978409

[52] Xuebin Ren, Liang Shi, Weiren Yu, Shusen Yang, Cong Zhao, and Zongben Xu. 2022. LDP-IDS: Local Differential Privacy for Infinite Data Streams. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 1064–1077. https://doi.org/10.1145/3514221.3526190

[53] Pratanu Roy, Arijit Khan, and Gustavo Alonso. 2016. Augmented Sketch: Faster and More Accurate Stream Processing. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, Fatma Özcan, Georgia Koutrika, and Sam Madden (Eds.). ACM, 1449–1463. https://doi.org/10.1145/2882903.2882948

[54] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. 2004. Medians and beyond: new aggregation techniques for sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*. 239–249.

[55] Charles Steinfield, Thomas Adelaar, and Fang Liu. 2005. Click and Mortar Strategies Viewed from the Web: A Content Analysis of Features Illustrating Integration Between Retailers' Online and Offline Presence. *Electron. Mark.* 15, 3 (2005), 199–212. https://doi.org/10.1080/10196780500208632

[56] Hao Wang and Zhengquan Xu. 2017. CTS-DP: publishing correlated time-series data via differential privacy. *Knowledge-Based Systems* 122 (2017), 167–179.

[57] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally differentially private protocols for frequency estimation. In *USENIX Security Symposium*. 729–745.

[58] Tianhao Wang, Joann Qiongna Chen, Zhikun Zhang, Dong Su, Yueqiang Cheng, Zhou Li, Ninghui Li, and Somesh Jha. 2021. Continuous release of data streams under both centralized and local differential privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 1237–1253.

[59] Tianhao Wang, Ninghui Li, and Somesh Jha. 2018. Locally Differentially Private Frequent Itemset Mining. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 127–143. https://doi.org/10.1109/SP.2018.00035

[60] Tianhao Wang, Ninghui Li, and Somesh Jha. 2021. Locally Differentially Private Heavy Hitter Identification. *IEEE Trans. Dependable Secur. Comput.* 18, 2 (2021), 982–993. https://doi.org/10.1109/TDSC.2019.2927695

[61] Stanley L Warner. 1965. Randomized response: A survey technique for eliminating evasive answer bias. *J. Amer. Statist. Assoc.* 60, 309 (1965), 63–69.

[62] Tong Yang, Junzhi Gong, Haowei Zhang, Lei Zou, Lei Shi, and Xiaoming Li. 2018. HeavyGuardian: Separate and Guard Hot Items in Data Streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Yike Guo and Faisal Farooq (Eds.). ACM, 2584–2593. https://doi.org/10.1145/3219819.3219978

[63] Ke Yi and Qin Zhang. 2009. Optimal tracking of distributed heavy hitters and quantiles. In *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*, Jan Paredaens and Jianwen Su (Eds.). ACM, 167–174. https://doi.org/10.1145/1559795.1559820

[64] Xi Zhang, Jian Cheng, Ting Yuan, Biao Niu, and Hanqing Lu. 2013. TopRec: domain-specific recommendation through community topic mining in social network. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, Daniel Schwabe, Virgílio A. F. Almeida, Hartmut Glaser, Ricardo Baeza-Yates, and Sue B. Moon (Eds.). International World Wide Web Conferences Steering Committee / ACM, 1501–1510. https://doi.org/10.1145/2488388.2488519

[65] Yang Zhou, Tong Yang, Jie Jiang, Bin Cui, Minlan Yu, Xiaoming Li, and Steve Uhlig. 2018. Cold Filter: A Meta-Framework for Faster and More Accurate Stream Processing. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 741–756. https://doi.org/10.1145/3183713.3183726

Received July 2023; revised October 2023; accepted November 2023