A APPENDICES

A.1 LDP Mechanisms

Optimal Local Hash. The Optimal Local Hash (OLH) mechanism [55] is designed for Randomizing private values in a large domain. It maps the value with a randomly selected hash function to a new data domain with the size of $g << d$ before randomizing the value. The randomization method is the same as GRR with p' and q' as follows

$$\begin{cases} p' = \dfrac{e^\epsilon}{e^\epsilon + g - 1}, \\ q' = \dfrac{1}{e^\epsilon + g - 1}. \end{cases} \tag{5}$$

It can also use Equation 2 to compute $\tilde{c}_i$ with $p = p'$ and $q = \frac{1}{g}p' + \frac{g-1}{g}q' = \frac{1}{g}$. When the size of the new data domain $g = e^\epsilon + 1$, the variance of $\tilde{c}_i$ can be minimized as

$$Var[\tilde{c}_i] = n \cdot \frac{4e^\epsilon}{(e^\epsilon - 1)^2} \tag{6}$$

Hadamard Response. The Hadamard Response (HR) mechanism [4, 5] encodes private values with a $K \times K$ Hadamard matrix, where $K = 2^{\lceil \log_2(d+1) \rceil}$. Expect for the first row of the matrix (all values are '1'), each other row corresponds to a value in the data domain. When encoding the i^{th} value in the data domain, the output value is randomly selected from column indices that have '1' in the $(i+1)^{th}$ row with the probability of p, and selected from other indices (columns have '0') with the probability of q, where

$$\begin{cases} p = \frac{e^\epsilon}{1+e^\epsilon} \\ q = \frac{1}{1+e^\epsilon} \end{cases} \tag{7}$$

Then $\tilde{c}_i$ can be calculated as follows

$$\tilde{c}_i = \frac{2(e^\epsilon + 1)}{e^\epsilon - 1}(\hat{c}_i - \frac{n}{2}) \tag{8}$$

The variance of the estimation result $\tilde{c}_i$ is

$$Var[\tilde{c}_i] = n \cdot \frac{4(e^\epsilon + 1)^2}{(e^\epsilon - 1)^2} \tag{9}$$

A.2 Proof of Theorem 3.2

Proof. We assume that $\hat{f}_i$ is the frequency of noisy data recorded in $\mathcal{HG}$ according to the ED strategy in BGR. Meanwhile, the ED strategy would introduce additional error when recording $\hat{f}_i$, while the frequency actually recorded and used for debiasing by the GRR mechanism before publishing is $\bar{f}_i$. According to Lemma 3.1, we have the upper bound of $\hat{f}_i - \bar{f}_i$ is

$$\begin{aligned} &Pr[\hat{f}_i - \bar{f}_i \geq \alpha t] \leq \frac{1}{2\alpha t}(\hat{f}_i - \sqrt{\hat{f}_i^2 - \frac{4P_{weak}E(V)}{b-1}}) \\ &\Rightarrow Pr[\hat{f}_i \leq \bar{f}_i + \alpha t] \geq (1 - \frac{1}{2\alpha t}(\hat{f}_i - \sqrt{\hat{f}_i^2 - \frac{4P_{weak}E(V)}{b-1}})) \end{aligned}$$

The distribution of $\hat{f}_i$ can be decomposed into $Bin(f_i, p) + Bin(t - f_i, q)$. Denote $Bin(f_i, p)$ as P_1 and $Bin(t - f_i, q)$ as P_2, according to the hoeffding inequality, we have

$$Pr[f_i p - P_1 \leq \zeta] \geq 1 - e^{-\frac{2\zeta^2}{f_i}}, \text{and}$$
$$Pr[(t - f_i)q - P_2 \leq \zeta] \geq 1 - e^{-\frac{2\zeta^2}{t - f_i}}$$

Then, according to Bonferroni inequality, we have

$$\begin{aligned} &Pr[P_1 + P_2 \geq f_i p + (t - f_i)q - 2\zeta] \geq 1 - e^{-\frac{2\zeta^2}{f_i}} - e^{-\frac{2\zeta^2}{t-f_i}} \\ &\Rightarrow Pr[\hat{f}_i \geq f_i p + (t - f_i)q - 2\zeta] \geq 1 - e^{-\frac{2\zeta^2}{f_i}} - e^{-\frac{2\zeta^2}{t-f_i}} \\ &\Rightarrow Pr[\frac{\hat{f}_i - tq}{p - q} - f_i \geq \frac{-2\zeta}{p-q}] \geq 1 - e^{-\frac{2\zeta^2}{f_i}} - e^{-\frac{2\zeta^2}{t-f_i}} \\ &\Rightarrow Pr[f_i - \frac{\hat{f}_i + \alpha t - tq}{p-q} \leq f_i - \frac{\hat{f}_i - tq}{p-q} \leq \frac{2\zeta}{p-q}] \\ &\geq (1 - e^{-\frac{2\zeta^2}{f_i}} - e^{-\frac{2\zeta^2}{t-f_i}})(1 - \frac{f_i}{2\alpha t}(1 - \sqrt{1 - \frac{4P_{weak}E(V)}{\hat{f}_i^2(b-1)}})) \end{aligned}$$

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

where $\zeta = \sqrt{t\log(2/\beta)/2}$, $P_{weak} = \frac{(i-1)!(d-k)!}{(d-1)!(i-k)!}$, and $E(V) = \sum_{j=i+1}^{d} f_j$. □

A.3 Proof of Theorem 3.4

Proof. We assume that $\hat{f}_i$ is the frequency of noisy data recorded in $\mathcal{HG}$ according to the ED strategy in BDR. Meanwhile, the ED strategy would introduce additional error when recording $\hat{f}_i$, while the frequency actually recorded and used for debiasing by the randomization mechanism before publishing is $\bar{f}_i$. According to Lemma 3.1, we have the upper bound of $\hat{f}_i - \bar{f}_i$ is

$$\begin{aligned} &Pr[\hat{f}_i - \bar{f}_i \geq \alpha t] \leq \frac{1}{2\alpha t}(\hat{f}_i - \sqrt{\hat{f}_i^2 - \frac{4P_{weak}E(V)}{b-1}}) \\ &\Rightarrow Pr[\hat{f}_i \leq \bar{f}_i + \alpha t] \geq (1 - \frac{1}{2\alpha t}(\hat{f}_i - \sqrt{\hat{f}_i^2 - \frac{4P_{weak}E(V)}{b-1}})) \end{aligned}$$

The distribution of $\hat{f}_i$ can be decomposed into $Bin(f_i, p_1p_2)$+$Bin(N_h - f_i, p_1q_2)$ + $Bin(t - N_h, q_1/k)$, where N_h is the number of hot items, $N_h \leq t$. Denote $Bin(f_i, p_1p_2)$ as P_1, $Bin(N_h - f_i, p_1q_2)$ as P_2, and $Bin(t - N_h, q_1/k)$ as P_3, according to the hoeffding inequality, we

have

$$Pr[f_i p_1 p_2 - P_1 \le \zeta] \ge 1 - e^{-\frac{2\zeta^2}{f_i}},$$

$$Pr[(N_h - f_i) p_1 q_2 - P_2 \le \zeta] \ge 1 - e^{-\frac{2\zeta^2}{N_h - f_i}}, \text{and}$$

$$Pr[(t - N_h) \cdot \frac{q_1}{k} - P_3 \le \zeta] \ge 1 - e^{-\frac{2\zeta^2}{t - N_h}}.$$

Then, according to Bonferroni inequality, we have

$$\begin{aligned}
&Pr[f_i p_1 p_2 - P_1 + (N_h - f_i) p_1 q_2 - P_2 + (t - N_h) \cdot \frac{q_1}{k} \\
&- P_3 \le 3\zeta] \ge 1 - e^{-\frac{2\zeta^2}{f_i}} + 1 - e^{-\frac{2\zeta^2}{N_h - f_i}} + 1 - e^{-\frac{2\zeta^2}{t - N_h}} - 2 \\
&\Rightarrow Pr[P_1 + P_2 + P3 \ge f_i p_1 p_2 + (N_h - f_i) p_1 q_2 \\
&+ (t - N_h) \cdot \frac{q_1}{k} - 3\zeta] \ge 1 - e^{-\frac{2\zeta^2}{f_i}} - e^{-\frac{2\zeta^2}{N_h - f_i}} - e^{-\frac{2\zeta^2}{t - N_h}} \\
&\Rightarrow Pr[\hat{f_i} \ge f_i p_1 p_2 + (N_h - f_i) p_1 q_2 + (t - N_h) \cdot \frac{q_1}{k} - 3\zeta] \\
&\ge 1 - e^{-\frac{2\zeta^2}{f_i}} - e^{-\frac{2\zeta^2}{N_h - f_i}} - e^{-\frac{2\zeta^2}{t - N_h}}
\end{aligned}$$

$$\begin{aligned}
&\Rightarrow Pr[\frac{\hat{f_i} - N_h p_1 q_2 - (t - N_h) \cdot \frac{q_1}{k}}{p_1(p_2 - q_2)} - f_i \ge \frac{-3\zeta}{p_1(p_2 - q_2)}] \\
&\ge 1 - e^{-\frac{2\zeta^2}{f_i}} - e^{-\frac{2\zeta^2}{N_h - f_i}} - e^{-\frac{2\zeta^2}{t - N_h}} \\
&\Rightarrow Pr[f_i - \frac{\overline{f_i} + \alpha t - N_h p_1 q_2 - (t - N_h) \cdot \frac{q_1}{k}}{p_1(p_2 - q_2)} \\
&\le f_i - \frac{\hat{f_i} - N_h p_1 q_2 - (t - N_h) \cdot \frac{q_1}{k}}{p_1(p_2 - q_2)} \le \frac{3\zeta}{p_1(p_2 - q_2)}] \\
&\ge (1 - e^{-\frac{2\zeta^2}{f_i}} - e^{-\frac{2\zeta^2}{N_h - f_i}} - e^{-\frac{2\zeta^2}{t - N_h}}) \cdot (1 \\
&- \frac{f_i}{2\alpha t}(1 - \sqrt{1 - \frac{4P_{weak}E(V)}{\hat{f_i}^2(b-1)}})) \\
&\Rightarrow Pr[f_i - \tilde{f_i} \le \frac{3\zeta + \alpha t}{p_1(p_2 - q_2)}] \ge (1 - 3e^{-\frac{2\zeta^2}{N_h}}) \cdot (1 \\
&- \frac{1}{2\alpha}(1 - \sqrt{1 - \frac{4P_{weak}E(V)}{b-1}})) \\
&\Rightarrow Pr[f_i - \tilde{f_i} \le (3\sqrt{\frac{N_h \log(3/\beta)}{2}} + \alpha t) \\
&\cdot \frac{(e^{\epsilon_1} + 1)(e^{\epsilon_2} + k - 1)}{e^{\epsilon_1}(e^{\epsilon_2} - 1)}] \\
&\ge (1 - \beta)(1 - \frac{1}{2\alpha}(1 - \sqrt{1 - \frac{4P_{weak}E(V)}{b-1}}))
\end{aligned}$$

□

A.4 Proof of Theorem 3.3

Proof. Denote v and v' as two raw data, o_1, o_2 and o_3 as the output of mechanisms, Ω_h as the domain of hot items, Ω_c as the domain of cold items. Firstly, $\mathcal{M}_{judge}$ satisfies ϵ_1-LDP, we have

$$\frac{Pr[o = \text{“}Hot\text{”} | v \in \Omega_h]}{Pr[o = \text{“}Hot\text{”} | v' \in \Omega_c]} \le \frac{p_1}{q_1} = e^{\epsilon_1}$$

The same result can be proved when o =“Cold".

Secondly, we prove that $\mathcal{M}_{hot}$ mechanism satisfies ϵ_2-LDP. The probability ratio of v and v' to get the same randomized item $o_2 \in \Omega_h$ is

$$\frac{Pr[o_2 | v \in \Omega_h]}{Pr[o_2 | v' \in \Omega_c]} \le \frac{p_2}{1/k} \le \frac{p_2}{q_2} = e^{\epsilon_2}$$

Similarly, the $\mathcal{M}_{cold}$ mechanism satisfies

$$\frac{Pr[o_3 | v \in \Omega_c]}{Pr[o_3 | v' \in \Omega_h]} \le e^{\epsilon_2}$$

According to the composition theorem of DP [30], BDR satisfies $(\epsilon_1 + \epsilon_2)$-LDP at each timestamp where $\epsilon_1 + \epsilon_2 = \epsilon$. Therefore, BDR satisfies ϵ-LDP. □

Algorithm 14 DSR_Insert($v, \mathcal{HG}, p_1, q_1, p_2, q_2$)

$\Omega_s = \{\mathcal{HG}.ID\} \cup \{\perp\}$
if the least count in $\mathcal{HG}$ changed from > 1 to ≤ 1 **then**
  for each $\mathcal{HG}[j] \in \mathcal{HG}$ **do**
    $\mathcal{HG}[j] \leftarrow (\mathcal{HG}[j].C - q_1)(p_2 - q_2)/(p_1 - q_1)$
  end for
  Insert v into $\mathcal{HG}$ following ED strategy; ▷ $r_i^t \in \Omega$
  $num_{entire} \leftarrow num_{entire} + 1$
  if the least count in $\mathcal{HG} \le 0$ **then**
    Replace the weakest KV pair with $< r_i^t, 1 - p_1 \cdot num_{entire}/(p_1 - q_1) - p_2 \cdot num_{reduced}/(p_2 - q_2) >$
  end if
else if the least count in $\mathcal{HG}$ changed from ≤ 1 to > 1 **then**
  for each $\mathcal{HG}[j] \in \mathcal{HG}$ **do**
    $\mathcal{HG}[j] \leftarrow (\mathcal{HG}[j].C - q_2)(p_1 - q_1)/(p_2 - q_2)$
  end for
  Insert v into $\mathcal{HG}$ following ED strategy; ▷ $r_i^t \in \Omega_s$
  $num_{reduced} \leftarrow num_{reduced} + 1$
else if the least count in $\mathcal{HG} > 1$ **then**
  for each $\mathcal{HG}[j] \in \mathcal{HG}$ **do**
    $\mathcal{HG}[j] \leftarrow \mathcal{HG}[j].C - q_1$
  end for
  Insert v into $\mathcal{HG}$ following ED strategy; ▷ $r_i^t \in \Omega_s$
  $num_{reduced} \leftarrow num_{reduced} + 1$
else
  for each $\mathcal{HG}[j] \in \mathcal{HG}$ **do**
    $\mathcal{HG}[j] \leftarrow \mathcal{HG}[j].C - q_2$
  end for
  Insert v into $\mathcal{HG}$ following ED strategy; ▷ $r_i^t \in \Omega$
  $num_{entire} \leftarrow num_{entire} + 1$
  if the least count in $\mathcal{HG} \le 0$ **then**
    Replace the weakest KV pair with $< r_i^t, 1 - p_1 \cdot num_{entire}/(p_1 - q_1) - p_2 \cdot num_{reduced}/(p_2 - q_2) >$
  end if
end if
return Updated $\mathcal{HG}$

Algorithm 15 DSR_FinalDebias($\mathcal{HG}$, p_1, q_1, p_2, q_2)

```
# Complete final debias based on the state of the previous HG.
if the least count in HG changed from > 1 to ≤ 1 then
    for each HG[j] ∈ HG do
        HG[j] ← HG[j].C/(p2 − q2)
    end for
else if the least count in HG changed from ≤ 1 to > 1 then
    for each HG[j] ∈ HG do
        HG[j] ← HG[j].C/(p1 − q1)
    end for
else if the least count in HG > 1 then
    for each HG[j] ∈ HG do
        HG[j] ← HG[j].C/(p1 − q1)
    end for
else
    for each HG[j] ∈ HG do
        HG[j] ← HG[j].C/(p2 − q2)
    end for
end if
return Updated HG
```

A.5 Algorithm of Function DSR_Insert and DSR_FinalDebias

A.6 Extended Related Work

Differential Private Data Stream Collection The earliest studies in differential privacy for streaming data collection originate from continuous observation of private data [7, 26, 29, 32, 33]. Long-term data collection from users can be regarded as the collection of data streams. These studies mainly consider the degradation of privacy guarantee due to the repeated appearance of private data when the user's state not changing for a period of time.

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. To avoid the overestimation of sensitivity of the streaming data caused by outliers, Perrier et al. [48] propose truncating the data exceeding a threshold to reduce the sensitivity. Wang et al. [56] point out that the threshold should be dynamically adjusted for different distributions instead of using the fixed quantile value. Therefore, they propose to use an exponential mechanism to get a more reasonable sensitivity. Some works are devoted to solving the problem that the privacy guarantee is continuously degraded due to the repeated use of data in streams in consecutive periods. kellaris et al. [41] propose w-event DP, regarding the statistical results published in a sliding window as an event. Farokhi et al. [34] propose to address the problem of the exploding privacy budget by reducing the privacy guarantee provided for data that appeared in the past over time. Some works study the release of correlated streaming data. Wang et al. [54] propose a correlated Laplace noise mechanism, and Bao et al. [8] propose a correlated Gaussian noise mechanism.

There are also some recent works on data stream collection with Local Differential Privacy (LDP) [39, 50, 56]. Joseph et al. [39] design a protocol to submit LDP-protected data only when the streaming data has changed and can have a greater impact on the statistical results. Ren et al. [50] propose a privacy budget segmentation framework that provides w-event LDP protection to prevent the degradation of privacy due to continuous data collection. Besides, Wang et al. [56] extend the proposed truncation-based CDP mechanism to LDP for the release of streaming data.

Tracking Heavy Hitters in Data stream Mining streaming data faces three principal challenges: *volume*, *velocity*, and *volatility* [42]. 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. Manku et al. [46] propose two algorithms Sticky Sampling and Lossy Counting, which only record the item and its counts whose estimated counts exceed the threshold. Metwally et al. [47] design an algorithm called Space-Saving and record data with a data structure called Stream-Summary, which achieves rapid deletion, update, and insertion for each new arrival data. Subsequently, Yang et al. [60] propose a new algorithm called HeavyGuardian to improve Space-Saving. Zhou et al. [63] also propose a framework called Cold Filter (CF) to improve Space-Saving. The Quantile algorithms such as the GK algorithm [37] and QDigest algorithm [52], focus on finding the item which is the smallest item that dominates ϕn items from the data stream. Sketch algorithms 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 the linear projection. Some existing works include Count Sketch[6], Count-Min Sketch[23], WavingSketch[44], and Moment[18], etc. However, the sketch algorithms may involve a large number of hash operations, which cannot meet the timeliness requirements of streaming data. Besides, the unimportant low-frequent items are all recorded, which leads to unnecessary memory consumption. Our design is based on Counter-based algorithms with an extended setting where streaming data is protected by LDP.

A.7 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. Specifically, let $V = \{v_1, v_2, ..., v_k\}$ as the Top-k heavy hitters in $\mathcal{HG}$. If v_i is one of a true Top-k heavy hitter, the relevance score rel_i is

$$rel_{v_i} = |k - |rank_{actual}(v_i) - rank_{estimated}(v_i)||.$$

If v_i is not a true Top-k heavy hitter, we directly set its rel_i as 0. Then, the Discounted Cumulative Gain (DCG) is

$$DCG_k = rel_{v_1} + \sum_{i=2}^{k} \frac{rel_{v_i}}{\log_2(i)}.$$

Finally, we normalize the DCG of $\mathcal{HG}$ by comparing it with the Ideal DCG ($IDCG$), which is the DCG when $\mathcal{HG}$ records an actual list of Top-k heavy hitters.

$$NDCG_k = \frac{DCG_k}{IDCG_k}.$$

$NDCG_k$ is between 0 and 1 for all k, and the closer it is to 1 means the ordering quality of $\mathcal{HG}$ is higher.

A.8 Implementation Details

A.8.1 Re-implementation for LDP Mechansims. We treat existing LDP frequency estimation approaches as privacy-preserving baselines. Specifically, we estimate the frequency of all items under LDP, and output the Top-k counts as the heavy hitter. Cormode, Maddock, and Maple [22] placed various LDP frequency estimation approaches into a common framework, and performed an series of experiments in Python[1]. Their work offered a starting point of our implementations.

We carefully studied the source codes, and fully re-implemented all baseline LDP mechanisms with the following optimizations.

Data serialization. In [22], the client outputs the perturbed data as an object, which server takes as its input to do data aggregation. In practice, the server and the client would communicate via a network channel. This requires object serializations and introduces additional communication and computation costs. In our implementation, we manually serialize the perturbed data to 'byte[]' based on the underlying approaches. If the client outputs bit strings (e.g., OUE and RAPPOR), we compress the output bit string by representing each 8 bits into 1 byte. If the client outputs integers (e.g., OLH, HR), we represent the integer with the minimal byte length, i.e., 1-4 byte for integers in range $[0, 2^8)$, $[0, 2^{16})$, $[0, 2^{24})$, and $[0, 2^{32})$, respectively.

Choices of the hash. Some frequency estimation approaches leverage (non-cryptographic) hash to map input to Boolean (BLH) or integer(s) (RAPPOR, OLH, HCMS). The performances of these approaches are greately affected by the efficiency of the underlying hash. Meanwhile, HeavyGuardian also leverages (non-cryptographic) hash to partition data into buckets. Note that [22] and [60] respectively use xxHash and BobHash. We invoke BobHash in all schemes since our test shows that BobHash are more efficient[2].

Besides, we find that de-biasing the randomized data before storing it into *HeavyGuardian* can avoid the bias introduced by the ED strategy being amplified by the de-biasing process. However, if a complete de-biasing is performed every time randomized data comes, it can cause the previously accumulated count to be de-biased repeatedly, e.g., divided by denominator $p - q$ of the de-biasing formula repeatedly in BGR. Therefore, we perform partial de-biasing when collecting and dividing all counts by the denominator of the de-biasing formula only before publishing the results.

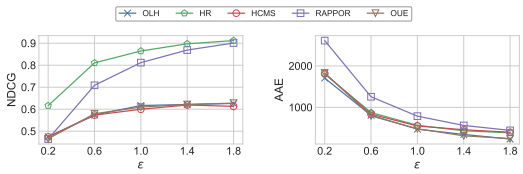

Figure 11: The comparison of different LDP mechanisms on Synthetic dataset.

A.8.2 Re-implementation for HeavyGuardian. We treat the original *HeavyGuardian* as the non-private baseline. We carefully studied existing open-source C/C++ codes provided by Yang et al.[3] and fully re-implemented the *HeavyGuardian* using Java. Our *HeavyGuardian* re-implementation has some improvements compared with the original implementation. First, the original implementation contains some hard-coded parameters for different tasks, while our re-implementation allows developers to dynamically config for different tasks. Second, the ED strategy in *HeavyGuardian* contains a Bernoulli sampling procedure, i.e., sampling a Boolean value with probability $\mathcal{P} = b^{-C}$ being True, where $b > 1$ is a predefined constant number, and C is a counting value. The naive method of sampling used in original *HeavyGuardian* implementation is to randomly sample $r \in [0, 1)$ and test whether $r < b^{-C}$. However, since finite computers cannot faithfully represent real numbers, the naive method would not produce the Boolean value with the correct distribution. In our implementation, we parse $b^{-C} = \exp(-C \cdot \ln(b))$ and leverage the method of Bernoulli$(\exp(-\gamma))$ proposed by Ganonne et al. [13] to do the sampling with no loss in accuracy.

Recall that the basic version of *HeavyGuardian* is a hash table with $w \geq 1$ buckets storing KV pairs ($\langle ID, count \rangle$). Each bucket is divided into the heavy part with size $\lambda_h > 0$ and the light part with size $\lambda_l \geq 0$. Because heavy hitter detection focus on only hot items, Yang et al. [60] recommend setting $\lambda_l = 0$ when using *HeavyGuardian* for heavy hitter detection tasks. Our experiments follow this recommendation and set $\lambda_l = 0$ (except CNR). The basic version of *HeavyGuardian* also allows using different λ_h and different numbers of buckets w when counting the most frequent k items in the heavy hitter task. Although our implementation also allows setting w and λ_h, our experiments focus on the basic cases, i.e., $w = 1$ and $\lambda_h = k$, to better demonstrate the effectiveness of our schemes.

We obtain memory consumption by measuring the deep sizes (i.e., the size of an object including the size of all referred objects, in addition to the size of the object itself) of Objects packaging the *HeavyGuardian* and our schemes. The tool we use is the JOL (Java Object Layout) library[4]. Although the error bounds we give in the theoretical analyses are de-biasing after storing, the actual error in our implementation is still bounded by and even lower than the theoretical results.

A.9 Supplementary Experiments

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

[1] https://github.com/Samuel-Maddock/pure-LDP
[2] Our test shows that on MacBook M1, xxHash takes about 0.01us to provide an output while BobHash takes about 0.1us.
[3] https://github.com/Gavindeed/HeavyGuardian
[4] http://hg.openjdk.java.net/code-tools/jol

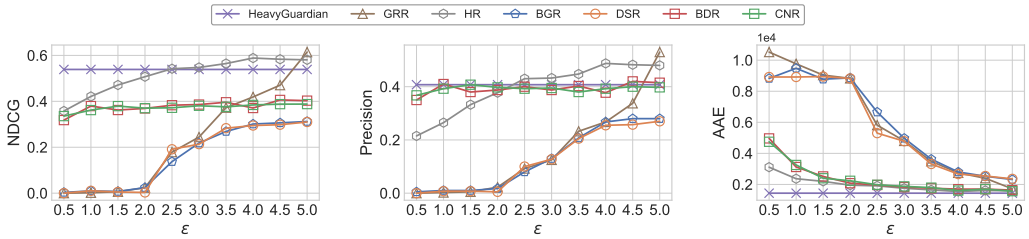

Figure 12: Evaluation on Retail dataset with $n = 908,576$ and $d = 16,469$, varying ϵ while taking 3% data for warm-up stage.

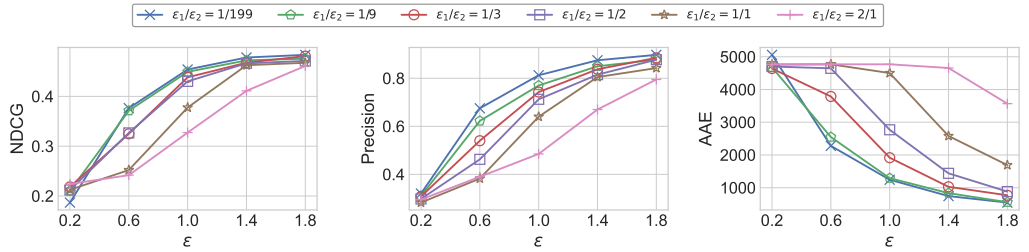

Figure 13: The impact of different allocations of privacy budget on accuracy of BDR. The evaluation is conducted on the Synthetic dataset (Normal distribution) with domain size 500, while taking 1% data for warm-up stage.

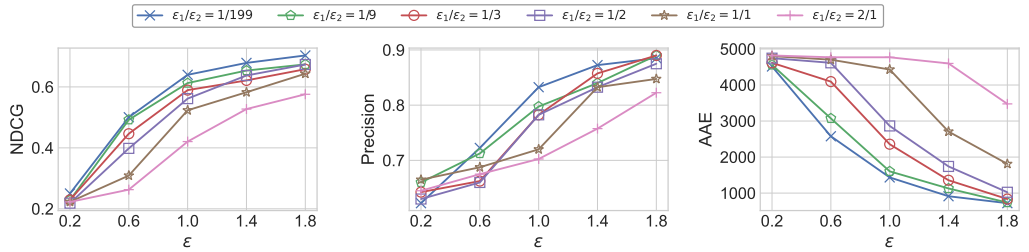

Figure 14: The impact of different allocations of privacy budget on accuracy of CNR. The evaluation is conducted on the Synthetic dataset (Normal distribution) with domain size 500, while taking 1% data for warm-up stage.

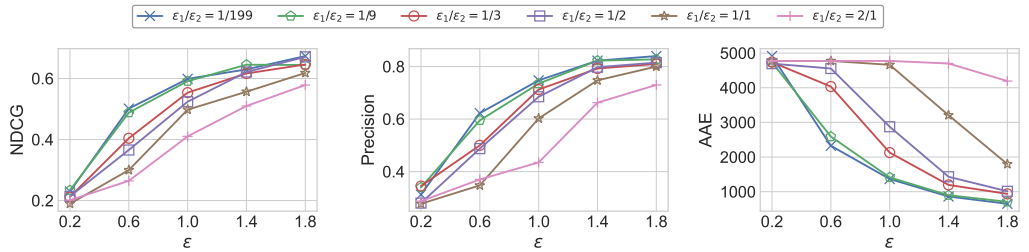

Figure 15: The impact of different allocations of privacy budget on accuracy of BDR. The evaluation is conducted on the Synthetic dataset (Normal distribution) with domain size 2000, while taking 1% data for warm-up stage.

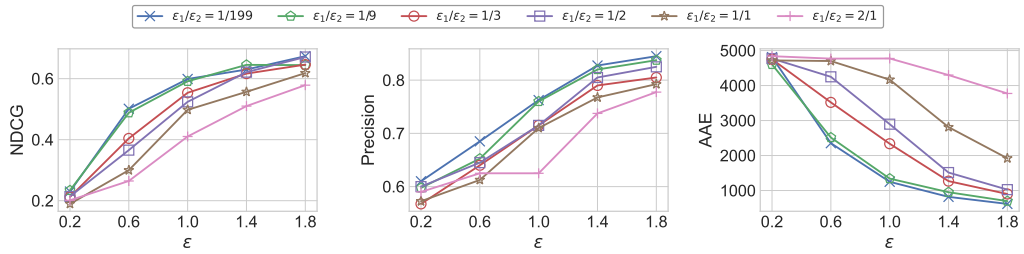

Figure 16: The impact of different allocations of privacy budget on accuracy of CNR. The evaluation is conducted on the Synthetic dataset (Normal distribution) with domain size 2000, while taking 1% data for warm-up stage.

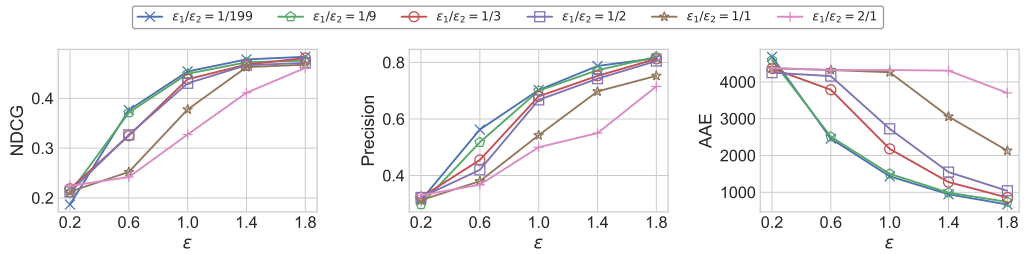

Figure 17: The impact of different allocations of privacy budget on accuracy of BDR. The evaluation is conducted on the Synthetic dataset (Exponential distribution) with domain size 500, while taking 1% data for warm-up stage.

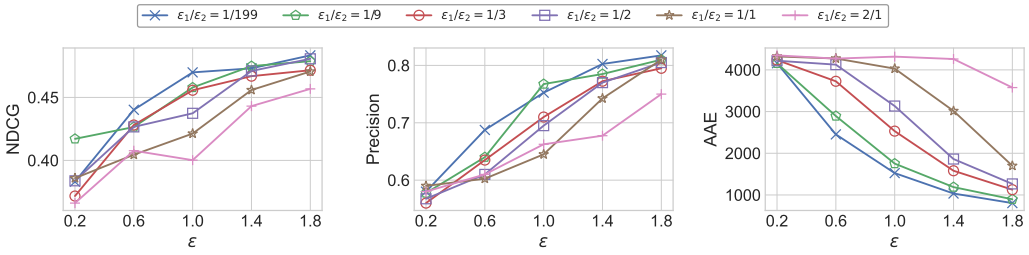

Figure 18: The impact of different allocations of privacy budget on accuracy of CNR. The evaluation is conducted on the Synthetic dataset (Exponential distribution) with domain size 500, while taking 1% data for warm-up stage.

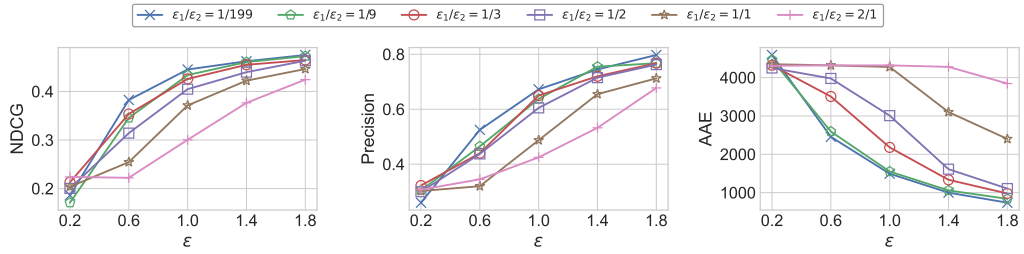

Figure 19: The impact of different allocations of privacy budget on accuracy of BDR. The evaluation is conducted on the Synthetic dataset (Exponential distribution) with domain size 1000, while taking 1% data for warm-up stage.

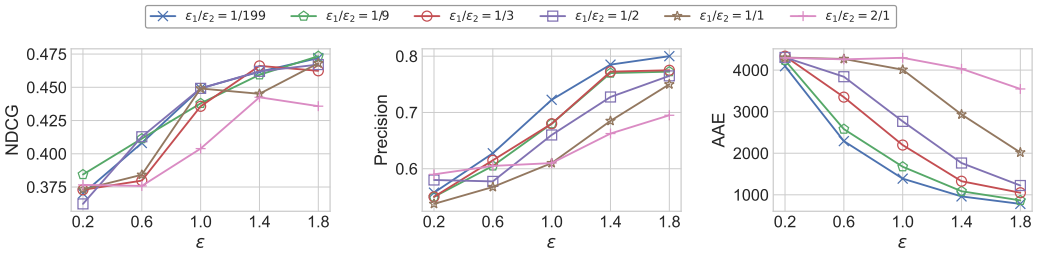

Figure 20: The impact of different allocations of privacy budget on accuracy of CNR. The evaluation is conducted on the Synthetic dataset (Exponential distribution) with domain size 1000, while taking 1% data for warm-up stage.

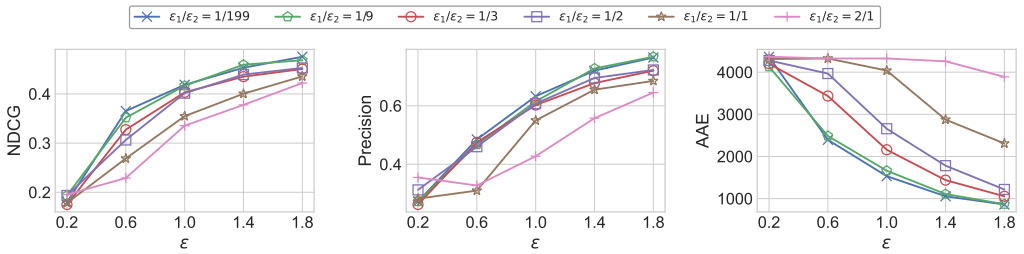

Figure 21: The impact of different allocations of privacy budget on accuracy of BDR. The evaluation is conducted on the Synthetic dataset (Exponential distribution) with domain size 2000, while taking 1% data for warm-up stage.

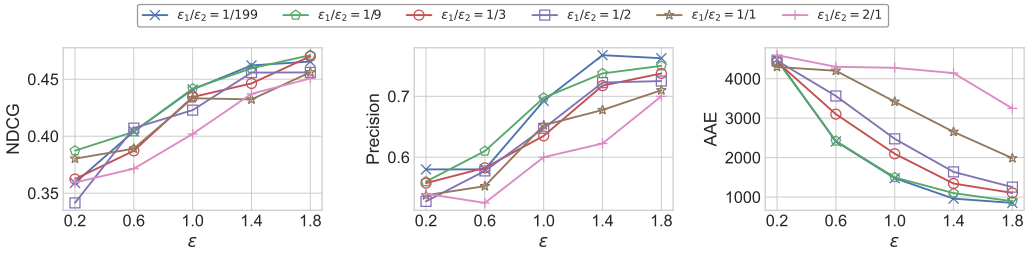

Figure 22: The impact of different allocations of privacy budget on accuracy of CNR. The evaluation is conducted on the Synthetic dataset (Exponential distribution) with domain size 2000, while taking 1% data for warm-up stage.

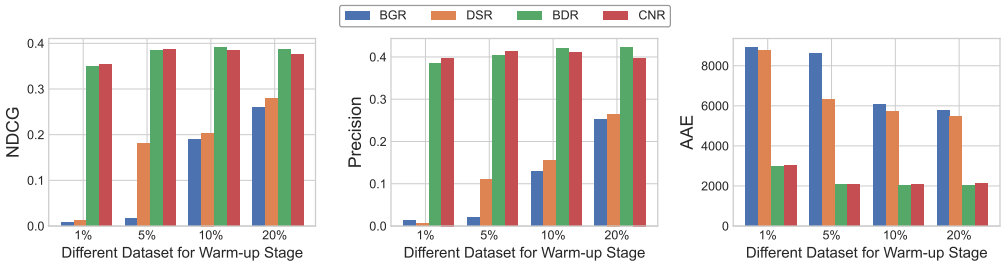

Figure 23: The impact of the warm-up stage on the accuracy of the proposed schemes for tracking Top-20 items on Retail dataset, where $\epsilon = 2$.

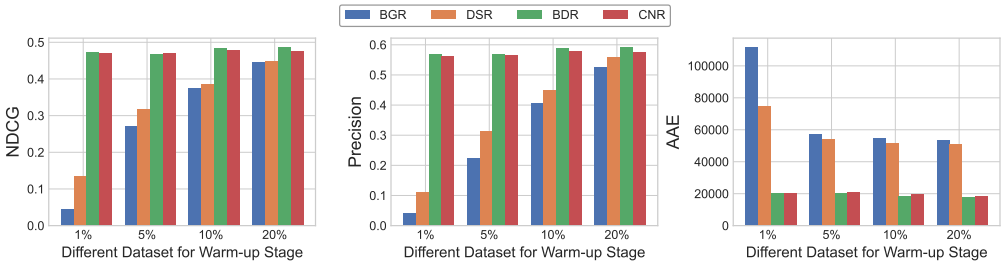

Figure 24: The impact of the warm-up stage on the accuracy of the proposed schemes for tracking Top-20 items on Kosarak dataset, where $\epsilon = 2$.

Table 2: Total communication overheads (MB) consumed by four schemes for deriving the Top-20 on each entire dataset. The left side of "/" is the uploading overheads for users, while the right side is the downloading overheads for users.

Scheme \ Dataset	Retail (3.97M)	Kosarak (30.5M)	Webdocs (1413.24M)
BGR	**3.64/0**	**35.09/0**	**1883.36/0**
DSR	**3.64/113.51**	**35.07/1062.79**	**1875.19/49891.47**
BDR	**3.24/98.82**	**26.67/709.04**	**1668.71/26295.89**
CNR	**3.17/92.03**	**26.49/698.95**	**1680.25/26861.62**

Table 3: Total server runtime (second) of four schemes for deriving the Top-20 on each entire dataset.

Scheme \ Dataset	Retail	Kosarak	Webdocs
BGR	**0.58**	**5.375**	**215.814**
DSR	**0.639**	**5.586**	**245.314**
BDR	**0.174**	**5.224**	**257.32**
CNR	**0.176**	**7.563**	**417.184**

Table 4: Total client runtime (second) of four schemes for deriving the Top-20 on each entire dataset.

Scheme \ Dataset	Retail	Kosarak	Webdocs
BGR	**0.22**	**1.88**	**184.481**
DSR	**0.336**	**2.929**	**226.501**
BDR	**0.637**	**3.549**	**197.065**
CNR	**0.859**	**3.653**	**196.88**