Improving Count-Mean Sketch as the Leading Locally Differentially Private Frequency Estimator for Large Dictionaries

Mingen Pan
Independent Researcher *
Email: mepan94@gmail.com

***Abstract*—This paper identifies that a group of latest locally-differentially-private (LDP) algorithms for frequency estimation, including all the Hadamard-matrix-based algorithms, are equivalent to the private Count-Mean Sketch (CMS) algorithm with different parameters. Therefore, we revisit the private CMS, correct errors in the original CMS paper regarding expectation and variance, modify the CMS implementation to eliminate existing bias, and optimize CMS using randomized response (RR) as the perturbation method. The optimized CMS with RR is shown to outperform CMS variants with other known perturbations in reducing the worst-case mean squared error (MSE), l_1 loss, and l_2 loss. Additionally, we prove that pairwise-independent hashing is sufficient for CMS, reducing its communication cost to the logarithm of the cardinality of all possible values (i.e., a dictionary). As a result, the optimized CMS with RR is proven theoretically and empirically as the leading algorithm for reducing the aforementioned loss functions when dealing with a very large dictionary. Furthermore, we demonstrate that randomness is necessary to ensure the correctness of CMS, and the communication cost of CMS, though low, is unavoidable despite the randomness being public or private.**

1. Introduction

Frequency is a fundamental metric to describe the properties of a dataset, counting the occurrences of specific values. Calculating the frequency of each value is straightforward if the original values are known to the analyst, as one can generate a hash map to record the occurrences of all values in the dataset. This method has linear time and space complexity relative to the dataset size. However, frequency calculation becomes challenging when data is protected, and their values are not directly accessible to the analyst.

One of the most popular approaches to protect data is differential privacy (DP) [1], which is considered the *de facto* standard for extracting statistics from a dataset while preserving individual data privacy. DP introduces sufficient noise to the statistical outputs such that the participation of any individual's data cannot be confidently inferred from the query results. However, DP still requires individuals to upload their data to a server before extracting statistics, posing risks of eavesdropping during data transmission or data breaches on the server side [2]. To address these concerns, Local Differential Privacy (LDP) [3] has been proposed, which requires enough perturbation on individual data on the client side before sending them to the server for statistical analysis, theoretically preventing eavesdropping or data breaches. However, LDP also adds more challenges to calculating statistics from a dataset, including frequency estimation.

A major challenge of LDP-preserving frequency estimation is that traditional mechanisms, such as randomized response [4], result in the precision of the estimated frequency increasing linearly with the number of all possible values in a dataset (referred to as a dictionary), making the estimation infeasible if the dictionary is too large. Several algorithms [5], [6], [7], [8], [9], [10], [11], [12], [13] have been developed to address this problem, and their precision is proven to be independent of the dictionary size, though some of them still have communication cost linear to the dictionary size. These algorithms will be further introduced in the Previous Work (Section 1.1). Many of these algorithms (e.g., [5], [9], [12], [14]) are found to be equivalent to the Count-Mean Sketch (CMS) algorithm (see Section 4 for more details). CMS hashes the original values through randomly selected hash functions and estimates the original frequencies based on the frequencies of the hashed values. The LDP version of CMS was originally developed by [7], but it had minor issues in calculating the expectation and variance (see Section 3.7) and did not explore the optimized parameters for CMS. Therefore, our work provides a corrected calculation of expectation and variance (Sections 3.1 and 3.2) and optimize CMS using randomized response as the perturbation method (referred to as OCMS+RR). OCMS+RR is shown to outperform CMS variants with other known perturbations in reducing the worst-case mean squared error ($\widehat{MSE}$), l_1 loss, and l_2 loss (Section 3.4). Moreover, our work proves that pairwise-independent hash functions are sufficient for CMS, and Section 3.5 proposes a practical approach to construct a pairwise-independent hashing family given any range of a hash function. The pairwise independence can reduce the communication cost to the logarithm of the dictionary size, allowing OCMS+RR to be applied to datasets with very large dictionaries. The pseudocode of OCMS+RR is presented in Section 3.6 (source code avail-

* *Currently working at Google LLC. This work was conducted independently and does not reflect the views or endorsement of Google.*

DOI 10.1109/CSF64896.2025.00024

able at https://github.com/mingen-pan/CountMeanSketch). When the dictionary size is much larger than the privacy factor ϵ defined by LDP, the aforementioned loss functions (defined in Section 2.2) are reduced by OCMS+RR as

$$\widehat{MSE} \leq \frac{e^{\epsilon/2}}{n(e^{\epsilon/2}-1)^2}$$
$$l_1 \xrightarrow{d \gg e^\epsilon} \frac{2de^{\epsilon/2}}{\sqrt{n}(e^\epsilon - 1)}$$
$$l_2 \xrightarrow{d \gg e^\epsilon} \frac{4de^\epsilon}{n(e^\epsilon-1)^2}$$

where n, and d are the dataset size and dictionary size, respectively. As a result, Section 5.1 compares existing algorithms with OCMS+RR and concludes that our OCMS+RR is the leading algorithm for reducing the worst-case MSE ($\widehat{MSE}$), l_1 loss, and l_2 loss, while its communication cost is only logarithmic relative to the dictionary size. Additionally, Section 5.2 proves the necessity of randomness in CMS and provides a counterexample to an attempt to de-randomize a CMS-equivalent algorithm by [14]. Section 5.3 shows that the communication cost of CMS is unavoidable, despite the randomness being public or private. Section 6 empirically demonstrates that OCMS+RR outperforms other algorithms.

1.1. Previous Work

RAPPOR [6] is the first algorithm to estimate frequency with precision independent of dictionary size. It represents the original value as a one-hot vector and applies LDP perturbation to every bit of the array. However, the communication cost is proportional to the dictionary size, as the one-hot vector is the same size as the dictionary. An alternative version of RAPPOR [15] hashes an original value into a smaller range than the dictionary size, resulting in a much smaller one-hot encoded vector. This version uses gradient descent to search for one possible solution for the original frequencies. However, this mapping is equivalent to dimension reduction in linear algebra, so the found solution is not guaranteed to be close to the original frequencies. To tackle the large dictionary, an algorithm was developed by [5], which transforms an original value to a column of a Hadamard matrix, uniformly samples one element from the column, and perturbs it. The original frequencies can be reconstructed from the transformed values by applying the Hadamard matrix again. The precision of this approach is independent of the dictionary size, and the communication cost is the logarithm of the dictionary size. The Apple Privacy team [7] modified the Count-mean Sketch (CMS) algorithm to be LDP. However, they failed to derive the expectation and variance correctly (see Section 3.7) and did not explore the optimized settings of CMS. In addition to CMS, other members of the Count Sketch family have also been adapted to LDP (e.g., [8], [13]). Wang et al. [9] summarized existing approaches and proposed a variant of RAPPOR and a local hashing method, which were claimed to be optimal in terms of MSE. However, [9] assumed the estimated frequencies to be always close to zero, so their optimality did not hold when the original frequency approaches one. Simultaneously, the Subset Selection algorithm [10] was proposed to estimate frequency with leading precision in l_1 / l_2 losses, but its communication cost is also proportional to the dictionary size when the privacy requirement is high. Acharya et al. [11] later integrated the Hadamard matrix into the subset selection and reduced the communication cost to the logarithm of the dictionary size. This method was further improved by [12], which applies the Hadamard transform twice. Though its performance can achieve the state-of-the-art order of precision in l_1 / l_2 losses, it may perform worse than the Hadamard Response in terms of MSE (as discussed in Section 4.3).

2. Background

2.1. Local Differential Privacy (LDP)

Local Differential Privacy (LDP) was initially proposed in [3] to ensure that querying an object does not significantly disclose its actual value. Consider a variable V that is perturbed by a mechanism M:

Definiton 2.1. *M is ϵ-LDP if and only if $\forall v, v' \in \mathcal{V}$, and $\forall u \in \mathcal{U}$,*

$$Pr(M(V) = u | V = v) \leq e^\epsilon Pr(M(V) = u | V = v'),$$

where $\mathcal{V}$ and $\mathcal{U}$ represent the dictionary (all possible values) of V and the range of M, respectively. This paper may abbreviate $(M(V)|V = v)$ as $M(v)$ when the variable is not the focus and its value is known.

Given the above definition, P is defined as a $|\mathcal{U}| \times |\mathcal{V}|$ matrix, where $P[u, v] = Pr(M(v) = u)$. Here u and v are represented by an index between $1..|\mathcal{U}|$ and $1..|\mathcal{V}|$, respectively. We consider only cases where $|\mathcal{U}| \geq |\mathcal{V}|$ and P is full rank, i.e., rank$(P) = |\mathcal{V}|$. This condition should hold for all meaningful LDP mechanisms, as the expected frequencies of the perturbed values correspond to a linear transformation of the original frequencies, and a rank-deficient transformation would prevent the reconstruction of the original frequencies from the perturbed values. Widely-used algorithms (e.g., [4], [6]) satisfy this condition. Subsequently, a matrix Q is constructed as $(P^TP)^{-1}P^T$, which satisfies $I = QP$ with I being an identity matrix. Given an output u from M, its decoding is defined to output $Q[:, u]$, i.e., the u-th column of Q. Thus, the reconstruction process (perturbation and decoding) of a variable V is defined as $R(V) = Q[:, M(V)]$. Similarly, $(R(V)|V = v)$ is abbreviated as $R(v)$ when the variable is not the focus and its value is known. An example of the reconstruction is presented in Appendix A. The randomness of $R(v)$ comes from $M(v)$, and each possible value of $R(v)$ is a vector of size $|\mathcal{V}|$ satisfying:

Property 2.1.

$$E[R(v)[v]] = 1$$
$$\forall v' \neq v : E[R(v)[v']] = 0$$

where E denotes expectation.

Proof: represent any u from $M(v)$ as a one-hot vector of size $|\mathcal{U}|$, where the u-th element is one. A random variable $\mathbf{u}$ represents all the possible outputs from $M(v)$ as one-hot vectors. Subsequently, $R(v) = Q\mathbf{u}$. Therefore, $E[R(v)] = QE[\mathbf{u}] = QP\mathbf{e}_v = \mathbf{e}_v$, where $\mathbf{e}_v$ is a one-hot vector with the v-th element being one. □

Meanwhile, we require the LDP mechanism M to be stateless, meaning that its output depends only on its input. This is formulated as

$$Pr(M(V) = u|V = v) = Pr(M(V) = u|V = v, \theta),$$

where θ represents any other information (except the randomness generator used by M). Considering two variables V and V', whose values are determined prior to perturbation, the stateless requirement ensures that their reconstruction process $R(V)$ and $R(V')$ are mutually independent random variables, formulated as

Property 2.2. *The covariance of $R(V)$ and $R(V')$ satisfies $\forall v, v', v_a, v_b \in \mathcal{V} : Cov(R(V)[v_a], R(V')[v_b]|V = v, V' = v') = 0$.*

Widely-used algorithms (e.g., [4], [6]) satisfy this property. Additionally, we observe that these LDP mechanisms also exhibit symmetry in variance, denoted as Var, formulated as:

Property 2.3. *Every element of $\{Var(R(v)[v'])|v, v' \in \mathcal{V}, v \neq v'\}$ has an identical value, denoted as $Var(R|\neq)$; every element of $\{Var(R(v)[v])|v \in \mathcal{V}\}$ has an identical value, denoted as $Var(R|=)$.*

2.2. Frequency Estimation

Let's define some symbols: $X^{(i)}$ is the i-th object of an arbitrary dataset of size n; the possible values of all the objects comprise a dictionary $[d]$, where d is its size, and $\forall i \in [n] : X^{(i)} \in [d]$. The frequency of value x is defined as $f(x) = \frac{\sum_{i\in[n]} \mathbf{1}\{X^{(i)}=x\}}{n}$. The LDP workflow of frequency estimation typically proceeds as follows:

1. Each object $X^{(i)}$ is perturbed by an LDP mechanism, which outputs $\tilde{X}^{(i)}$.

2. An analyst collects and aggregates the outputs as $\hat{f}(x) = A_x([\tilde{X}^{(i)}]_{i\in[n]})$, if the analyst is interested in $f(x)$, where A_x denotes the aggregation. The aggregation can be performed for every $x \in [d]$.

A potential scenario involves a group of clients sending their perturbed responses to a server, while their unperturbed responses constitute a dataset. The server then estimates the frequency of the unperturbed responses.

$\hat{f}(x)$ in Step 2 is an estimator of $f(x)$. Existing literature usually uses l_1 and l_2 losses to measure its overall precision, which are defined as follows:

$$l_1(\hat{f}) = \max_{\forall \mathbf{X}:|\mathbf{X}|=n} E[\sum_{x\in[d]} |\hat{f}(x) - f(x)|]$$

$$l_2(\hat{f}) = \max_{\forall \mathbf{X}:|\mathbf{X}|=n} E[\sum_{x\in[d]} (\hat{f}(x) - f(x))^2]$$

where $\mathbf{X} = [X^{(i)}]_{i\in[n]}$, and $\forall \mathbf{X}$ assumes the size n is constant. These two losses measure the overall precision of the estimation. However, in many cases, an analyst is also interested in the precision of the frequency estimation of an individual value. The mean squared error (MSE), formulated as $MSE(\hat{f}(x)) = E[(\hat{f}(x) - f(x))^2]$, is commonly used to measure the precision of an estimation, reflecting both the variance and bias of the estimation. Since an analyst is assumed to have no knowledge of the actual dataset, they need to assume the worst case when calculating the MSE, which is defined as

$$\widehat{MSE}(\hat{f}) = \max_{\forall \mathbf{X}:|\mathbf{X}|=n} \max_{x\in\mathbf{x}} MSE(\hat{f}(x)), \tag{1}$$

where $\widehat{MSE}$ denotes the upper bound of MSE, i.e., the worst case; $\mathbf{x}$ denotes the values of interest, which is $[d]$ by default. If the estimator $\hat{f}$ is unbiased, the worst-case MSE is the same as the worst-case variance, which is also positively correlated to the confidence interval (CI). Thus, an unbiased estimator optimized for $\widehat{MSE}$ is also optimized for CI.

3. Revisit Count-mean Sketch

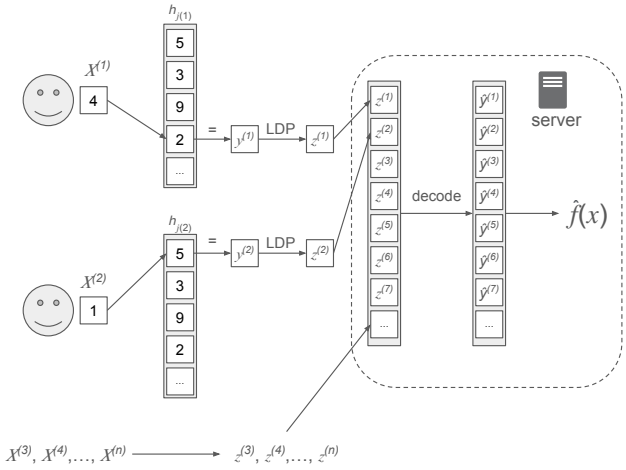


Figure 1: Visualization of Count-mean Sketch.

Count-Mean Sketch (CMS) is a probabilistic data structure for frequency estimation, which (1) uses multiple hash functions to map objects into different counters, and (2) calibrates the bias in each counter to estimate the frequency of the values of interest. Suppose there is a dataset $[X^{(i)}]_{i\in[n]}$, each with a value belonging to a dictionary $[d]$. When setting up the CMS program, a hashing family $\mathcal{H}$ with k hash functions mapping from $[d]$ to $[m]$ is provided. When uniformly sampling a hash function h from $\mathcal{H}$, we expect that $\forall(x_1, x_2) \in [d]^2 : Pr(h(x_1) = h(x_2)) \approx \frac{1}{m}$, which will be further discussed in Section 3.1. CMS operates with the following steps (Fig. 1):

1. Each object $X^{(i)}$ is assigned a hash function uniformly sampled from $\mathcal{H}$. The assigned hash function is

denoted as $h_{j^{(i)}}$, where $j^{(i)} \in [k]$ is the index of the hash function. The server and the client sync on the hash-function assignment.

2. Hash $X^{(i)}$ as $y^{(i)} = h_{j^{(i)}}(X^{(i)}) \in [m]$. Then, $y^{(i)}$ is perturbed by an ϵ-LDP mechanism as $z^{(i)}$.

3. Collect all the responses $(z^{(i)})_{i\in[n]}$, decode them following Section 2.1 as $(\hat{y}^{(i)})_{i\in[n]}$, where each $\hat{y}^{(i)}$ is an estimator of $y^{(i)}$, i.e., $\hat{y}^{(i)} = R(y^{(i)})$, with R defined in Section 2.1. Notice that each $\hat{y}^{(i)}$ is a vector of size m.

4. The frequency of any value $x \in [d]$ can be estimated as

$$\hat{f}(x) = \frac{m}{n(m-1)}\Big[\sum_{i\in[n]} \hat{y}^{(i)}[h_{j^{(i)}}(x)]\Big] - \frac{1}{m-1}. \quad (2)$$

Notably, $\hat{f}(x)$ may produce values outside the interval $[0, 1]$ due to perturbations in the collected data. One straightforward solution is to truncate $\hat{f}(x)$ to $[0, 1]$, i.e. $\min(1, \max(0, \hat{f}(x)))$, which also improves precision by eliminating deviations beyond the bounds. More advanced algorithms, such as projection onto a simplex, can also be used to enforce $\hat{f}(x) \in [0, 1]$ and improve precision, as discussed in [6].

Regarding privacy guarantee, the only output that depends on the value of $X^{(i)}$ is $z^{(i)}$, which has been perturbed by an ϵ-LDP mechanism. The postprocessing of the outputs from an LDP mechanism does not change its privacy guarantee [16], so we have proved:

Theorem 3.1. *The CMS frequency estimator $\hat{f}(x)$ is ϵ-LDP.*

3.1. Expectation and its Bias

This section derives the expectation of $\hat{f}(x)$. Here, we only consider the randomness of assigning hash functions to objects and the LDP perturbation, and regard $\mathcal{H}$ and $[X^{(i)}]_{i\in[n]}$ as constant. That is, $h_j(x)$ will not be considered a random variable; $\forall j \in [k], \forall (x, x') \in [d]^2$, whether $h_j(x') = h_j(x)$ is determined in advance as well, which will be denoted as $c_j(x, x')$.

The hashing family $\mathcal{H}$ is treated as constant for the following reasons: (1) $\mathcal{H}$ is visible prior to executing CMS, as the server must sync it with all clients in advance to ensure that the clients know which hashing family to sample; and (2) though the hashing family $\mathcal{H}$ can be regenerated before each run of the CMS, we will show later that it is sufficient to have a constant hashing family $\mathcal{H}$ at all times.

Some literature [14] also proposed a deterministic approach to assign hash functions, which will be discussed in Section 5.2. Here, we only focus on the CMS defined above and require the hash-function assignments to be random events. Now, the expectation of the frequency estimation by CMS is derived as:

Theorem 3.2. *The expectation of $\hat{f}(x)$ is*

$$E[\hat{f}(x)] = \frac{m}{m-1}[f(x) + \sum_{x'\in[d]\backslash x}\sum_{j\in[k]} \frac{c_j(x,x')}{k} f(x')] - \frac{1}{m-1} \quad (3)$$

where $f(x)$ is the frequency of value x; $[d]\backslash x$ represents the set of $[d]$ excluding x.

Corollary 3.1. *If and only if $\sum_{j\in[k]} \frac{c_j(x,x')}{k} = \frac{1}{m}$ is satisfied for all $x \neq x'$, then*

$$E[\hat{f}(x)] = f(x).$$

The proof of Theorem 3.2 and Corollary 3.1 is presented in Supplement S1. It is notable that $\forall x \neq x'$: $\sum_{j\in[k]} \frac{c_j(x,x')}{k} = \frac{1}{m}$ is equivalent to $\Pr_{h\in\mathcal{H}}(h(x) = h(x')) = \frac{1}{m}$, where $h \in \mathcal{H}$ represents uniform sampling, which is also the definition of a pairwise-independent hashing family. Thus, Corollary 3.1 can be rephrased as:

Corollary 3.2. *If and only if $\mathcal{H}$ is pairwise independent, i.e., $\Pr_{h\in\mathcal{H}}(h(x) = h(x')) = \frac{1}{m}$, we have $E[\hat{f}(x)] = f(x)$.*

3.2. Variance

Assuming that the LDP reconstruction process R in Step 3 of CMS satisfies Properties 2.1 and 2.2, we can derive:

Lemma 3.1. *If R satisfies Properties 2.1 and 2.2, we have*

$$Var(\hat{f}(x)) = \frac{m^2}{(m-1)^2 n}\sum_{x'\in[d]}\Big[\Big(\frac{1}{k}\sum_{j\in[k]} Var(R(h_j(x'))[h_j(x)])\Big) + \bar{c}(x', x) - \bar{c}(x', x)^2\Big] f(x'), \quad (4)$$

where $\bar{c}(x', x) = \sum_{j\in[k]} \frac{c_j(x',x)}{k}$.

Combining Lemma 3.1 and Property 2.3, we can derive:

Theorem 3.3. *If R satisfies Properties 2.1 - 2.3, we have*

$$Var(\hat{f}(x)) = \frac{m^2}{(m-1)^2 n}\Big[\sum_{x'\in[d]\backslash x} f(x') \Big(\bar{c}(x', x)Var(R|=) + (1 - \bar{c}(x', x))Var(R|\neq) + \bar{c}(x', x)(1 - \bar{c}(x', x))\Big) + f(x)Var(R|=)\Big], \quad (5)$$

where $Var(R|=)$ and $Var(R|\neq)$ are defined in Property 2.3.

If the hashing family $\mathcal{H}$ is pairwise independent, we further have:

Corollary 3.3. *If R satisfies Properties 2.1 - 2.3 and $\mathcal{H}$ is pairwise independent (i.e., $\forall x, x' : \bar{c}(x', x) = \frac{1}{m}$), we have*

$$Var(\hat{f}(x)) = \frac{m}{(m-1)^2 n}\Big[\big(1 - f(x)\big)\Big(Var(R|=) + (m-1)\, Var(R|\neq) + \frac{m-1}{m}\Big) + mf(x)Var(R|=)\Big]. \quad (6)$$

The lemma, theorem, and corollary of this section are proven in Supplement S2.

3.3. Accuracy and Precision Analysis

The mean squared error (MSE) measures the accuracy and precision of $\hat{f}(x)$, which is defined as

$$\begin{aligned} MSE(\hat{f}(x)) &= E[(\hat{f}(x) - f(x))^2] \\ &= E[\hat{f}(x) - E[\hat{f}(x)]]^2 + (E[\hat{f}(x)] - f(x))^2 \\ &= Var(\hat{f}(x)) + (E[\hat{f}(x)] - f(x))^2 \end{aligned} \quad (7)$$

When $\mathcal{H}$ is pairwise independent, we have $E[\hat{f}(x)] - f(x) = 0$, so $MSE(\hat{f}(x)) = Var(\hat{f}(x))$. If a hashing family is not pairwise independent but still satisfies $\forall (x, x') \in [d]^2 : \bar{c}(x, x')$ is identical, we can create an unbiased frequency estimator of $f(x)$ as

$$g(x) = \frac{m'}{n(m'-1)}\big[\sum_{i\in[n]} \hat{y}^{(i)}[h_{j^{(i)}}(x)]\big] - \frac{1}{m'-1}, \quad (8)$$

where $m' = \frac{1}{\bar{c}(x,x')}$. $g(x)$ has the following properties:

Theorem 3.4.

$$E[g(x)] = 0 \quad (9)$$

and

$$MSE(g(x)) = \frac{1}{n(1-\bar{c}(x,x'))^2}\Big[(1-f(x))\Big(\bar{c}(x',x)Var(R|=) + (1-\bar{c}(x',x))Var(R|\neq) + \bar{c}(x',x)(1-\bar{c}(x',x))\Big) + f(x)Var(R|=)\Big]. \quad (10)$$

The formal proof is similar to that of the original estimator $\hat{f}(x)$, so the proof is omitted here. The rationale is that the original estimator $\hat{f}(x)$ assumes $\bar{c}(x, x')$ being $\frac{1}{m}$, but here we integrate $\bar{c}(x, x')$ into the estimation and replace $\frac{1}{m}$ with $\bar{c}(x, x')$.

Besides MSE, one may also use the confidence interval (CI) of $\hat{f}(x)$ to measure precision. If $\hat{f}(x)$ is unbiased, its CI is the same as its concentration bound. Moreover, $\hat{f}(x)$ is equivalent to the sum of bounded random variables (i.e., $\sum \hat{y}$), so its bounds can be derived from the Chernoff bound. Below is the concentration bound for $\hat{f}(x)$ from the unbiased CMS using randomized response (referred to as CMS+RR):

Theorem 3.5. *CMS+RR with pairwise-independent hashing satisfies $\forall \alpha \in [0, \sqrt{\frac{e^\epsilon n}{m-1}}]$,*

$$Pr\big[|\hat{f}(x) - f(x)| \geq \alpha\sqrt{Var(\hat{f}(x))}\big] \leq 2\exp(-\frac{\alpha^2}{3}\cdot\frac{m-1}{e^\epsilon + m - 1}).$$

This theorem is proven in Supplement S3. Notably, this theorem focuses solely on CMS+RR, as Section 3.4 will demonstrate that it outperforms CMS implementations with other known perturbations.

3.4. Optimizing CMS

This section explores the parameters of CMS to reduce the l_1 loss, l_2 loss, and the worst-case MSE defined in Section 2.2. It is notable that l_2 loss is connected to MSE as follows:

$$\begin{aligned} l_2(\hat{f}) &= \max_{\forall \mathbf{X}:|\mathbf{X}|=n} \sum_{x\in[d]} E[(\hat{f}(x) - f(x))^2] \\ &= \max_{\forall \mathbf{X}:|\mathbf{X}|=n} \sum_{x\in[d]} MSE(\hat{f}(x)). \end{aligned} \quad (11)$$

Additionally, given the Cauchy-Schwarz inequality, we have $l_1(\hat{f}) \leq \sqrt{d\, l_2(\hat{f})}$. Thus, both losses are connected to MSE. Based on Eq. (7) and Corollary 3.1, if $\mathcal{H}$ is pairwise independent, MSE is equivalent to $Var(\hat{f}(x))$, which can be calculated using Eq. (6). Assuming that the privacy factor ϵ and the dictionary $[d]$ are given, the undetermined variables in this equation are only the LDP process R and the range of a hash function m.

We first consider randomized response (RR) as the LDP process, whose variance is as follows:

$$Var(RR(v)[v']) = \frac{1}{(e^\epsilon - 1)^2}\begin{cases} e^\epsilon(m-1) & \text{if } v = v' \\ e^\epsilon + m - 2 & \text{if } v \neq v'. \end{cases} \quad (12)$$

Now, we optimize m for CMS+RR, starting with the worst-case MSE. Here, we assume that an analyst has prior knowledge of the upper bound of $f(x)$ and derive:

Theorem 3.6. *If $\forall x \in \mathbf{x} : f(x) \leq f^*$ is guaranteed to be correct by prior knowledge, where $\mathbf{x}$ are the values of interest, the worst-case MSE among $\mathbf{x}$ for CMS+RR is optimized as*

$$\widehat{MSE}(\hat{f}) = \begin{cases} \frac{2(\Delta_{MSE} + e^\epsilon)}{n(e^\epsilon - 1)^2} & \text{if } f^* \leq \frac{1}{2} \\ \frac{e^{\epsilon/2}}{n(e^{\epsilon/2} - 1)^2} & \text{if } \frac{1}{2} < f^* \leq 1 \end{cases},$$

where

$$\Delta_{MSE} = e^{\epsilon/2}\sqrt{[(1-f^*)e^\epsilon + f^*][f^* e^\epsilon + (1-f^*)]},$$

and the hash-function range m is the closest integer to

$$\begin{cases} 1 + \frac{\Delta_{MSE}}{f^* e^\epsilon + 1 - f^*} & \text{if } f^* \le \frac{1}{2} \\ 1 + e^{\epsilon/2} & \text{if } \frac{1}{2} < f^* \le 1. \end{cases} \quad (13)$$

The theorem is proven in Appendix B. Notably, f^* is prior knowledge of an analyst and could be larger than $\max_x f(x)$, which is unknown to the analyst. When $f^* \ge \frac{1}{2}$, it has the same worst-case MSE as $f^* = 1$, which is guaranteed to be correct. On the other hand, if $f^* < \frac{1}{2}$ and the prior knowledge happens to be incorrect, i.e., $\max_x f(x) > f^*$, we demonstrate that

Theorem 3.7. *Given the optimized CMS+RR in Theorem 3.6, if $\delta = \max_x f(x) - f^* > 0$ and $f^* < \frac{1}{2}$,*

$$\widehat{MSE}(\hat{f}) = \widehat{MSE}(\hat{f}|f^*) + \frac{\delta(1-2f^*)e^\epsilon}{n\Delta_{MSE}} \le \widehat{MSE}(\hat{f}|f^*) + \frac{\delta(1-2f^*)}{n}. \quad (14)$$

where $\widehat{MSE}(\hat{f})$ and $\widehat{MSE}(\hat{f}|f^*)$ are the actual worst-case MSE and the worst-case MSE assuming $\max_x f(x) \le f^*$, respectively.

The theorem is proven in Appendix B. Since $\widehat{MSE}(\hat{f}|f^*) \ge \widehat{MSE}(\hat{f}|f^* = 0) = \frac{4e^\epsilon}{n(e^\epsilon-1)^2}$, as long as $\delta(1-2f^*) = O(\frac{4e^\epsilon}{n(e^\epsilon-1)^2})$, we still have $\widehat{MSE}(\hat{f}) = O(\widehat{MSE}(\hat{f}|f^*))$. Therefore, if the prior knowledge f^* is guaranteed to be correct or its error is bounded by the above condition, considering the prior knowledge f^* in the $\widehat{MSE}$ calculation is recommended. Section 6.2 provides an example of utilizing f^*. Otherwise, one can always assume $f^* = 1$ for simplicity. Hereinafter, we assume no prior knowledge of $f(x)$ unless explicitly mentioned, i.e., $f^* = 1$ by default.

Next, we optimize the l_1 and l_2 losses:

Theorem 3.8. *The l_2 loss and the upper bound of l_1 loss of CMS+RR are optimized as*

$$l_2^*(\hat{f}) = \frac{2(\Delta_l + de^\epsilon)}{n(e^\epsilon - 1)^2},$$

and

$$l_1^*(\hat{f}) \le \sqrt{\frac{2d(\Delta_l + de^\epsilon)}{n(e^\epsilon - 1)^2}},$$

respectively, where

$$\Delta_l = e^{\epsilon/2}\sqrt{(e^\epsilon + d - 1)(de^\epsilon - e^\epsilon + 1)},$$

and m is the closest integer to $1 + \frac{\Delta_l}{e^\epsilon + d - 1}$.

Corollary 3.4. *When $d \gg e^\epsilon$, the optimized l_2 loss and the upper bound of the optimized l_1 loss of CMS+RR approach*

$$l_2^*(\hat{f}) \to \frac{4de^\epsilon}{n(e^\epsilon - 1)^2},$$

and

$$\lceil l_1 \rceil^*(\hat{f}) \to \frac{2de^{\epsilon/2}}{\sqrt{n}(e^\epsilon - 1)},$$

respectively, where $\lceil l_1 \rceil$ represents the upper bound of the l_1 loss, and the corresponding $m \to 1 + e^\epsilon$.

The theorem and corollary above are proven in Appendix B. One caveat is that the CMS+RR optimized for l_1 / l_2 losses is not optimized for worst-case MSE. Assuming d is large enough, and $m = 1 + e^\epsilon$, we have

$$\widehat{MSE}(\hat{f}) = Var(\hat{f}(x)_{RR}|f(x) = 1) = \frac{(e^\epsilon + 1)^2}{n(e^\epsilon - 1)^2},$$

which is always larger than the optimized $\widehat{MSE}$ of CMS+RR, i.e., $\frac{e^{\epsilon/2}}{n(e^\epsilon-1)^2}$. What is worse, even when ϵ is very large, it only converges to $\frac{1}{n}$, not zero. Therefore, before an analyst estimates the frequencies of a dataset, they need to determine whether to focus on (1) the overall precision (i.e., l_1 / l_2 losses) or (2) the worst-case precision of the individual estimations (i.e., $\widehat{MSE}$).

Since optimizing $\widehat{MSE}$ and l_1 / l_2 losses requires different m, we will hereinafter refer to the CMS+RR optimized for $\widehat{MSE}$ and l_1 / l_2 losses as MSE-OCMS+RR and l-OCMS+RR, respectively. Additionally, OCMS+RR will be compared with the existing algorithms in Section 5.1.

Now, we evaluate the CMS using other existing LDP processes. A study [9] summarized the latest frequency estimation algorithms at that time. When considering different privacy regimes, they found that randomized response (RR), symmetric RAPPOR (s-RP), and asymmetric RAPPOR (a-RP) achieved state-of-the-art precision in some privacy regimes. Though more advanced algorithms exist, they are either as precise as these algorithms in their respective regimes, or they can be converted to CMS+RR (as shown in Section 4). Thus, in addition to RR, we will also evaluate CMS using s-RP and a-RP as the LDP process. Since the precision of these two mechanisms is independent of their input cardinality m, and $Var(\hat{f}(x))$ decreases with m when R is independent of m, the loss functions of CMS with RAPPOR mechanisms are optimized when $m \to \infty$. Supplement S4 derives their values as follows:

CMS+	RR	sRP	aRP
$\widehat{MSE}(\hat{f}\mid f^* < \frac{1}{2})$	$\frac{2(\Delta_{MSE}+e^\epsilon)}{n(e^\epsilon-1)^2}$	$\frac{e^{\epsilon/2}}{n(e^{\epsilon/2}-1)^2}$	$\frac{f^*(e^\epsilon-1)^2+4^\epsilon}{n(e^\epsilon-1)^2}$
$\widehat{MSE}(\hat{f}\mid f^* \ge \frac{1}{2})$	$\frac{e^{\epsilon/2}}{n(e^{\epsilon/2}-1)^2}$	$\frac{e^{\epsilon/2}}{n(e^{\epsilon/2}-1)^2}$	$\frac{f^*(e^\epsilon-1)^2+4^\epsilon}{n(e^\epsilon-1)^2}$
l_2	$\frac{4de^\epsilon}{n(e^\epsilon-1)^2}$	$\frac{de^{\epsilon/2}}{n(e^{\epsilon/2}-1)^2}$	$\frac{4de^\epsilon}{n(e^\epsilon-1)^2}$

CMS using RAPPOR mechanisms are less preferred for the following reasons: (1) their performance in reducing worst-case MSE or l_1 / l_2 losses is outperformed by CMS+RR in some conditions (highlighted in red in the table), and (2) they have very high communication costs (see Supplement S4 for details).

3.5. Unbiased CMS with Imperfect Hashing

OCMS+RR is built upon an unbiased CMS, which requires its hashing to be pairwise independent. However, the construction of a K-wise-independent hashing family requires the range of the hashed values (i.e., $[m]$) to comprise a finite field (Construction 3.32 of [17]). A finite field must have p^l elements [18], where p is a prime number and $l \in \mathbb{N}^+$. For instance, if a CMS has $m = 6$, there is no way to construct a pairwise independent hashing family for it. In practice, we can still construct an approximately pairwise-independent hashing family for any m. Consider a perfect pairwise independent hashing family with $p^l > d$:

$$\mathcal{H}_{pi} = \{h(x) = a_0 + a_1 x \mid a_0, a_1 \in [p^l]\}. \tag{15}$$

where a_0, a_1, and x are elements in a finite field $\mathbb{F}(p^l)$, so the addition and multiplication in Eq. (15) are all field operations. The range of any $h(x)$ is also the field $\mathbb{F}(p^l)$. One approach to construct p^l is to find a prime number $p' > d$ and set $l = 1$, i.e., $p^l = p'$. In this case, the field operations of $\mathbb{F}(p')$ simply require every operation to modulo p' [19]. However, finding a prime number when d is very large can be time-consuming. On the other hand, we can find any l such that $2^l > d$, and set $p = 2$, and the field operations of $\mathbb{F}(2^l)$ are introduced in [19].

Subsequently, we construct a new hashing family based on $\mathcal{H}_{pi}$:

$$\mathcal{H}_{api} = \{h'(x) = h(x) \bmod m \mid h(x) \in \mathcal{H}_{pi}\}.$$

where api represents approximately pairwise independent. Notably, the finite field size p^l should also be no less than m. Since $\forall h \in \mathcal{H}_{pi}$, $\forall y \in [m]$: $Pr(h(x) = y) = p^{-l}$, and assuming $p^l = qm + r$, where $q \in \mathbb{N}$ and $r \in \mathbb{N}$, we will have r hashed values with $Pr(h'(x) = y) = \frac{q+1}{p^l}$ and the remaining $m - r$ hashed values with $Pr(h'(x) = y) = \frac{q}{p^l}$. Thus, we have:

Lemma 3.2. *$\forall x_1, x_2 \in [d]^2$:*

$$Pr_{h' \in \mathcal{H}_{api}}(h'(x_1) = h'(x_2)) = r(\frac{q+1}{p^l})^2 + (m-r)(\frac{q}{p^l})^2 = \frac{(2q+1)r + mq^2}{(mq+r)^2},$$

where $h' \in \mathcal{H}_{api}$ represents uniform sampling.

Notice that $Pr_{h' \in \mathcal{H}_{api}}(h'(x_1) = h'(x_2)) = \bar{c}(x_1, x_2)$, which is a constant and can be substituted into the $g(x)$ in Theorem 3.4, where $m' = \frac{(mq+r)^2}{(2q+1)r+mq^2}$. As a result, $g(x)$ is an unbiased estimator of $f(x)$. We can further prove that

Theorem 3.9. *When*

$$\frac{2mVar(R|=) + m - 1}{n(m-1)^3} \frac{m}{\tau Var(\hat{f}(x))} \leq (2q+1)^2,$$

we have

$$Var(g(x|\mathcal{H}_{api})) < (1+\tau)Var(\hat{f}(x)),$$

where $g(x|\mathcal{H}_{api})$ and $\hat{f}(x)$ represent the $g(x)$ with $\mathcal{H}_{api}$ and a standard estimator of $f(x)$ with a pairwise-independent hashing family.

Applying the theorem to OCMS+RR, we can derive:

Corollary 3.5. *When $2q + 1 > \sqrt{\frac{1}{\tau}}$, we have*

$$MSE(g) < (1+\tau)\widehat{MSE}(\hat{f})$$
$$l_2(g) < (1+\tau)l_2^*(\hat{f})$$

for MSE-OCMS+RR and l-OCMS+RR, respectively.

The theorem and corollary are proven in Supplement S5. Notably, the condition of this corollary is independent of ϵ. If $\tau = 0.01$, we have $q > 4.5$ to satisfy the two corollaries above. Therefore, one could use $g(x)$ to approximate $\hat{f}(x)$ if $p^l \geq 5m$. Since a_0 and a_1 of $\mathcal{H}_{api}$ can be any numbers in $\mathbb{F}(p^l)$, we have $|\mathcal{H}_{api}| = (p^l)^2 = p^{2\lceil \log_p \max\{d+1, 5m\}\rceil}$, and $2\lceil \log_2 \max\{d+1, 5m\}\rceil$ bits are sufficient to represent a hash function in $\mathcal{H}_{api}$ if $p = 2$.

3.6. Implementation

OCMS+RR (Algorithm 1) is implemented to not only adapt the optimized hash range m proposed in Section 3.4, but also the approximately pairwise independent hashing family $\mathcal{H}_{api}$ and the corresponding estimator $g(x)$ in Section 3.5. The CLIENT function perturbs each object $X^{(i)}$, and their perturbed values, along with their hashing functions, are sent to the server. The server then calls SERVER to estimate the frequency of any x, i.e., $f(x)$. In particular, when the dictionary size d is small, i.e., $d < 2^{64} - 59$, the algorithm constructs the hashing family with p^l as a prime number. Otherwise, it computes the minimum $2^l > d$ as p^l. The source code of OCMS+RR is available at https://github.com/mingen-pan/CountMeanSketch).

3.7. Issues in the original CMS

Most frequency estimators ensure that the estimated frequency converges to the actual frequency as the dataset size increases, provided each object is queried once. However, this section demonstrates that the estimation from the original CMS exhibits a constant bias, even with an infinitely large dataset.

The original CMS randomly generates k hash functions, and each hash function $h_j(x)$ is generated by sampling d mutually independent random variables, each uniformly distributed among $[m]$. Subsequently, $\forall x, x' \in [d]$ with $x \neq x'$, $c_j(x, x')$ can be considered as a Bernoulli random variable with a probability of $\frac{1}{m}$ of being one. Since the k hash functions are generated independently, $\sum_{j \in [k]} \frac{c_j(x,x')}{k}$ is equivalent to a binomial random variable, whose expectation is $\frac{1}{m}$. However, the probability of the output of a binomial distribution exactly equaling its expectation is very low, so $\sum_{j \in [k]} \frac{c_j(x,x')}{k}$ will likely deviate from $\frac{1}{m}$ to some extent (see Slud's Inequality [20]). Given Corollary 3.1, $E[\hat{f}(x)]$

Algorithm 1 OCMS+RR (Part 1)

d: dictionary size
ϵ: privacy factor
Mode := either MSE-OCMS+RR or l-OCMS+RR
f^*: prior knowledge on the upper bound of $f(x)$. Default value is one.
$PRIME = 2^{64} - 59$ // Largest prime in uint64

function FINITEFIELDSIZE(d, m)
    $v \coloneqq \max\{d+1, 5m\}$ // see Corollary 3.5 for why $5m$
    if $v \leq PRIME$ **then**
        return $PRIME$
    end if
    return $2^{\lceil \log_2(v) \rceil}$
end function

function HASHRANGE(ϵ, d, f^*, Mode)
    if Mode is MSE-OCMS+RR **then**
        if $f^* \geq \frac{1}{2}$ **then**
            return round($e^{\epsilon/2} + 1$)
        end if
        $\Delta \coloneqq e^{\epsilon/2}\sqrt{[(1-f^*)e^{\epsilon} + f^*][f^*e^{\epsilon} + (1-f^*)]}$
        return round($\frac{\Delta}{f^*e^{\epsilon}+1-f^*} + 1$)
    else if Mode is l-OCMS+RR **then**
        $\Delta \coloneqq e^{\epsilon/2}\sqrt{(e^{\epsilon}+d-1)(de^{\epsilon}-e^{\epsilon}+1)}$
        return round($\frac{\Delta}{e^{\epsilon}+d-1} + 1$)
    end if
end function

$m \coloneqq$ HASHRANGE(ϵ, d, f^*, Mode)
$p_{\mathcal{H}} \coloneqq$ FINITEFIELDSIZE(d, m) // i.e., p^l in Section 3.5.

function HASH($a_0, a_1, x, m, p_{\mathcal{H}}$)
    if $p_{\mathcal{H}} = PRIME$ **then**
        $v \coloneqq (a_0 + a_1 x) \mod p_{\mathcal{H}}$
    else
        $op \coloneqq$ FiniteFieldOperation($p_{\mathcal{H}}$)
        $v \coloneqq op.add(a_0, op.mul(a_1, x))$
    end if
    return $v \mod m$
end function

// return $Q[v, u]$ in Section 2.1.
function DECODE(u: perturbed value, v: value of interest, m, ϵ)
    if $u = v$ **then**
        return $\frac{e^{\epsilon}+m-1}{e^{\epsilon}-1}$
    end if
    return $\frac{-1}{e^{\epsilon}-1}$
end function

function CLIENT(x: true value of a client, $X^{(i)}$)
    Uniformly Sample a_0 and a_1 from $[0, p_{\mathcal{H}} - 1]$.
    $y \coloneqq$ HASH($a_0, a_1, x, m, p_{\mathcal{H}}$) // i.e., $h_{j^{(i)}}(X^{(i)})$
    return RANDOMIZEDRESPONSE(y, m, ϵ), a_0, a_1
end function

Algorithm 1 OCMS+RR (Part 2)

function SERVER(x: value of interest, $\mathbf{C}$: all responses from the clients)
    $\hat{y}_{tot} \coloneqq 0$
    for z, a_0, a_1 in $\mathbf{C}$ **do**
        $h_x \coloneqq$ HASH($a_0, a_1, x, m, p_{\mathcal{H}}$) // i.e., $h_{j^{(i)}}(x)$
        $\hat{y} \coloneqq$ DECODE(z, h_x, m, ϵ) // i.e., $\hat{y}^{(i)}[h_{j^{(i)}}(x)]$
        $\hat{y}_{tot}$ += $\hat{y}$
    end for
    $n \coloneqq$ LENGTH($\mathbf{C}$)
    $q,\ r \coloneqq \lfloor p_{\mathcal{H}}/m \rfloor,\ \ p_{\mathcal{H}} \mod m$
    $m' \coloneqq \frac{(mq+r)^2}{(2q+1)r+mq^2}$
    return $\frac{m'}{n(m'-1)}\hat{y}_{tot} - \frac{1}{m'-1}$
end function

will likely deviate from $f(x)$ to some extent too, i.e., $\hat{f}(x)$ has bias.

To evaluate the degree of bias, when considering $\forall c_j(x, x')$ as random variables, we can derive:

Theorem 3.10.

$$\underset{\forall c_j(x,x')}{E}[E[\hat{f}(x)]] = f(x) \tag{16}$$

and

$$\underset{\forall c_j(x,x')}{Var}(E[\hat{f}(x)]) = \frac{1}{(m-1)k}\sum_{x' \in [d] \setminus x} f(x')^2. \tag{17}$$

The proof is presented in Supplement S6. The variance indicates that the bias is $O(\frac{1}{\sqrt{(m-1)k}})$. However, the original CMS paper [7] derived $E[\hat{f}(x)]$ to be unbiased. We found that they treated $c_j(x, x')$ as independent random variables for every pair of $X^{(i)}$ and $X^{(i')}$, which is incorrect because all pairs with the same values x and x' will always have the same $c_j(x, x')$.

The original paper also has an issue in its variance calculation, where they required the hashing family $\mathcal{H}$ to be 3-wise-independent because they assumed any $X^{(i_1)}$ and $X^{(i_2)}$ were correlated, which was problematic. In contrast, the non-private Count Sketch [21] only requires pairwise independence, which is consistent with our work. Notably, the LDP perturbation only affects the precision after the hashing, so it should not alter the pairwise-independence requirement for the hashing family.

4. Connection with Existing Frequency Estimators

This section demonstrates that various frequency estimators proposed in the previous literature are equivalent or closely connected to CMS. First, we define the Transformation-and-Sampling Framework (TnS Framework) as follows:

Definiton 4.1. *(TnS Framework)*

1. Transform the original value x to a vector of elements $M[x]$, i.e., a row of a matrix M, where the j-th element belongs to the finite set $\mathcal{V}_j$.

2. Uniformly sample an element from the vector $M[x]$, denoted as $M[x][j]$.

3. Perturb the sampled element $M[x][j]$ within $\mathcal{V}_j$ using some LDP mechanism.

The above process is equivalent to (1) uniformly sampling a column from M, denoted as $M[:, j]$, (2) mapping the original value x to $M[x, j]$, and (3) perturbing it. The column here serves as a hash function, and all the columns of M comprise a hashing family. Thus, we have proved:

Theorem 4.1. *An algorithm belonging to the TnS Framework is equivalent to CMS .*

4.1. Hadamard Encoding

Hadamard Encoding (HE) is an algorithm that (1) creates a Hadamard matrix with a size larger than the dictionary size d, (2) select a row from the Hadamard matrix with an index equal to $x+1$, and (3) uniformly samples an element from the selected row, and (4) perturbs the element using randomized response. It was originally proposed by [5] and has been widely used in the literature, e.g., [7], [12].

It is evident that HE belongs to the TnS framework, so it is equivalent to CMS. Additionally, Supplement S7 proves that the decoding process of HE is equivalent to Eq. (2). Moreover, all the rows of a Hadamard matrix, excluding the first element, comprise a 3-wise-independent hashing family [22], mapping to either -1 or 1. Therefore, HE is equivalent to the CMS+RR using pairwise-independent hashing with $m = 2$.

4.2. Hadamard Response

Hadamard Response was originally proposed by [11] and later developed in [14]. It has two equivalent versions, and the version in [14] is presented here: (1) publicly and randomly selects a row from a Hadamard matrix with size larger than the dictionary size d, (2) asks if the original value is mapped to one in the selected row (1 means yes, 0 means no), and (3) perturbs the answer using randomized response. It is straightforward to see that Step 1 is equivalent to sampling a hash function, and Step 2 is equivalent to hashing the original value. Additionally, its decoding process is equivalent to HE. Therefore, this algorithm is also equivalent to the CMS+RR using pairwise-independent hashing with $m = 2$.

4.3. Recursive Hadamard Response

Recursive Hadamard Response (RHR) [12] is an improvement of Hadamard Encoding, increasing precision by communicating more bits. It (1) splits the original value x into $x//2^{b-1}$ (floor division) and $x \bmod 2^{b-1}$; (2) creates a Hadamard matrix with a size larger than $d//2^{b-1}$, (3) select a row from the Hadamard matrix with an index equal to $(x//2^{b-1}) + 1$, and (4) uniformly samples an element from the selected row as s, and (5) perturbs $s \times (x \bmod 2^{b-1})$ using randomized response.

Steps 2 - 5 can be fitted in the TnS framework, mapping x to 2^b possible values. However, the collision probability of two inputs is not uniform. If two inputs x_1 and x_2 have the same mod (i.e., $x_1 \bmod 2^{b-1} = x_2 \bmod 2^{b-1}$), their collision probability is $1/2$. Otherwise, the probability is zero. In CMS, any pair of inputs have the same collision probability of $1/m$, which is 2^{-b} given 2^b hashed values. Thus, RHR is a skewed version of CMS+RR.

If the dataset is randomly generated, i.e., every pair of objects may have the same mod with a probability of 2^{b-1}, then the hash collision probability becomes 2^{-b}. Note that [12] sets $b = \epsilon$ if the dictionary is large enough. This is equivalent to $m = e^\epsilon$ in CMS, which is close but not identical to the optimized m used by OCMS+RR, so the precision of RHR at most has the same order as OCMS+RR.

However, we can always generate a dataset with all values having the same mod when divided by 2^{b-1}. This makes all these values have a probability of $1/2$ to collide with one another. Therefore, RHR in this case will perform worse than regular Hadamard Encoding, because they have the same collision probability but the randomized response of RHR considers more possible values, leading to higher variance. In the context of CMS, the MSE of RHR is equivalent to setting $\bar{c}(x, x') = \frac{1}{2}$ while setting R as a randomized response with $m > 2$ in Eq. (5). Section 6.3 demonstrates that even a real dataset will encounter this problem if it contains values with high frequency.

4.4. Local Hashing

Local Hashing [9] asks each client to (1) publicly and randomly generate a hashing function mapping $[d]$ to $[m]$, (2) hash the original value, and (3) perturb the hashed value using randomized response. It is straightforward to see that randomly generating a hashing function is equivalent to randomly selecting a hash function from the hashing family in CMS. Also, its decoding process is identical to Eq. (2). Therefore, this algorithm is equivalent to CMS. The original work [9] assumed $f(x) \approx 0$ when optimizing the MSE of their algorithm, but Section 3.4 demonstrates that both $f(x) = 0$ and $f(x) = 1$ could be the maximum points of the MSE. Moreover, they did not optimize the communication cost of sending a hash function to the server, which will be discussed in Section 5.1.

4.5. Recursive CMS

When the private CMS was initially proposed in [7], it suggested an algorithm to (1) map an original value to $[m]$ where m is large, and then (2) use Hadamard Encoding to perturb the hashed value. Section 4.1 has demonstrated that Hadamard Encoding is equivalent to CMS, so the CMS with Hadamard Encoding (CMS+HE) forms a recursive CMS. Generally speaking, the perturbation algorithm R in a CMS

can also be another CMS, and we will prove that a recursive CMS is equivalent to a regular CMS.

Here, we assume all the CMS involved are unbiased, i.e., the hashing family is pairwise independent. Suppose the first CMS hashes the original values to $[m_1]$, and its perturbation algorithm is another CMS hashing its input to $[m_2]$. Given pairwise independence, the collision probability $\bar{c}(x, x')$ of two distinct input values is $\frac{m_1+m_2-1}{m_1 m_2}$. Therefore, the recursive CMS is equivalent to a regular CMS with $m = \frac{m_1 m_2}{m_1+m_2-1}$.

Returning to the CMS+HE, it is equivalent to the CMS+RR with $m' = \frac{2m}{m+1}$, where m' and m are the hashing ranges of the equivalent and original CMS, respectively. If m is large enough, it is equivalent to the CMS+RR with $m' = 2$ (i.e., Hadamard Encoding).

4.6. RAPPOR

The encoding of RAPPOR [15] is equivalent to CMS if each object in RAPPOR belongs to its own cohort and there is only one hash function in each cohort. The hashing of RAPPOR and CMS can be considered a linear transformation from $\mathcal{R}^d$ (frequencies of original values) to $\mathcal{R}^{km}$ (frequencies of the hashed values). For an unbiased CMS (Corollary 3.1), we have $km = d^2 m > d$, so CMS can derive the original frequencies without bias. However, RAPPOR utilizes gradient descent to find the original frequencies and does not require $km \geq d$. The original RAPPOR [15] even chooses small km on purpose to reduce computation. Unfortunately, this is problematic because multiple values in a high-dimension simplex will map to the same value in a low-dimension simplex, and the value RAPPOR finds may not be the original value.

5. Discussion

5.1. Precision and Communication Cost

Tables 1 and 2 compare the precision and communication cost among various frequency estimators, respectively. We will demonstrate that OCMS+RR is the leading algorithm when considering both precision and communication cost.

Table 1 shows that the OCMS+RR is leading in reducing l_1 loss, l_2 loss, and the worst-case MSE, even considering the constant term. Since HE, OLH, and CMS+HE are equivalent to unbiased CMS+RR without the optimized parameters, it is expected that they are not leading in at least one dimension. Only OCMS+RR and s-RP are leading in the worst-case MSE, but Table 2 shows that s-RP requires d bits to communicate while OCMS+RR only requires $\max\{2\log d + \frac{\epsilon}{2}, \frac{3}{2}\epsilon + 6\}$ bits. If the communication cost is required to be $O(\log d)$, only RHR has the same order of precision as OCMS+RR in l_1 and l_2 losses. However, empirical study (Section 6) demonstrates that OCMS+RR outperforms RHR in all datasets, consistent with the theoretical analysis in Section 4.3. Therefore, OCMS+RR is the only leading algorithm for the worst-case MSE, l_1 loss, and l_2 loss while communicating efficiently.

	l_1	l_2	$\widehat{MSE}$
HE	$\frac{d(e^\epsilon+1)}{e^\epsilon-1}$	$\frac{d(e^\epsilon+1)^2}{(e^\epsilon-1)^2}$	$(\frac{e^\epsilon+1}{e^\epsilon-1})^2$
RHR	$\Theta(\frac{d}{\sqrt{\min\{\epsilon,\epsilon^2\}}})$	$\Theta(\frac{d}{\min\{\epsilon,\epsilon^2\}})$	$\Omega((\frac{e^\epsilon+1}{e^\epsilon-1})^2)$
OLH	$\frac{2de^{\epsilon/2}}{e^\epsilon-1}$	$\frac{4de^\epsilon}{(e^\epsilon-1)^2}$	$(\frac{e^\epsilon+1}{e^\epsilon-1})^2$
OCMS+RR	$\frac{2de^{\epsilon/2}}{e^\epsilon-1}$	$\frac{4de^\epsilon}{(e^\epsilon-1)^2}$	$\frac{e^{\epsilon/2}}{(e^{\epsilon/2}-1)^2}$
CMS+HE	$\frac{d(e^\epsilon+1)}{e^\epsilon-1}$	$\frac{d(e^\epsilon+1)^2}{(e^\epsilon-1)^2}$	$(\frac{e^\epsilon+1}{e^\epsilon-1})^2$
SS	$\frac{2de^{\epsilon/2}}{e^\epsilon-1}$ *	$\frac{4de^\epsilon}{(e^\epsilon-1)^2}$	$(\frac{e^\epsilon+1}{e^\epsilon-1})^2$
a-RP	$\frac{2de^{\epsilon/2}}{e^\epsilon-1}$	$\frac{4de^\epsilon}{(e^\epsilon-1)^2}$	$(\frac{e^\epsilon+1}{e^\epsilon-1})^2$
s-RP	$\frac{de^{\epsilon/4}}{e^{\epsilon/2}-1}$	$\frac{de^{\epsilon/2}}{(e^{\epsilon/2}-1)^2}$	$\frac{e^{\epsilon/2}}{(e^{\epsilon/2}-1)^2}$

TABLE 1: Precision comparison among frequency-estimation algorithms when $d \gg e^\epsilon$.

Note: HE, RHR, OLH, OCMS+RR, CMS+HE, SS, a-RP, and s-RP represent Hadamard Encoding, Recursive Hadamard Response, Optimized Local Hashing, Optimized CMS+RR, CMS with Hadamard Encoding, Subset Selection, asymmetric RAPPOR, and symmetric RAPPOR, respectively. Here, OCMS+RR represents MSE-OCMS+RR and l-OCMS+RR when comparing $\widehat{MSE}$ and l_1 / l_2 losses, respectively. The data source is presented in Appendix C.
*: the original value from [10] is problematic, and Appendix C recalculates the value as shown.

	Communication Cost (bits)
HE	$\log d$
RHR	$\log d + \epsilon$
OLH	$d\epsilon$
MSE-OCMS+RR	$\max\{2\log d + \frac{\epsilon}{2}, \frac{3}{2}\epsilon + 6\}$
l-OCMS+RR	$\max\{2\log d + \epsilon, 3\epsilon + 6\}$
original CMS	$\Omega(d \log m)$
SS	$\frac{d}{1+e^\epsilon}$
RAPPOR	d

TABLE 2: Communication Cost among frequency-estimation algorithms

Note: see Table 1 for the meaning of abbreviations. The communication cost only accounts for the cost between the server and one client. The data source is presented in Appendix C.

5.2. Necessity of Randomness

An algorithm was proposed by [11] to convert the Hadamard Response (a special case of CMS) into a deterministic algorithm, arguing that the randomness could be avoided, thereby decreasing the communication cost. However, we will show that one can always create a counterexample causing the frequency estimation to have a constant bias. Assume that we are interested in the frequency of value x, and $f(x)$ happens to be zero (no x in the dataset). Given any hash function (denoted as h_j for convenience), we denote x_j as the value colliding with x, i.e., $c_j(x, x_j) = 1$. Subsequently, we set the value of all objects assigned to the j-th hash function as x_j, and repeat this process for all the hash functions. As a result, we have:

$$\sum_{j\in[k]} \sum_{x'\in[d]\setminus x} c_j(X^{(i)}, x) f(x') = 1.$$

Substituting the above formula and $f(x) = 0$ into Eq. (3) results in $E[\hat{f}(x)] = 1$, which proves:

Theorem 5.1. *If the assignment of hash functions to the objects, $\{h_{j^{(i)}} | i \in n\}$, is constant in CMS, there exists a dataset resulting in*

$$E[\hat{f}(x)] - f(x) \neq 0.$$

The logic behind the theorem is intuitive: CMS utilizes randomness to ensure a stable probability of collision with other values. If one gives up the randomness, an attacker can always generate a dataset to maximize the collision. The reason why [11] believed a deterministic algorithm still worked was that they assumed that each $X^{(i)}$ was sampled from a distribution, so one cannot control the value of $X^{(i)}$. Their algorithm relies on the randomness in distribution sampling instead of the randomness in assigning hash functions.

The next question is how much randomness (in bits) is needed to ensure that CMS will work as expected. Here, we focus only on the randomness in assigning hash functions and assume that the LDP process R always uses its own randomness.

By examining the proof involving hash-function assignments in Supplements S1 and S2, we realize $Var(\hat{f}(x))$ is calculated based on $Cov(\hat{y}^{(i_1)}[h_{j^{(i_1)}}(x)], \hat{y}^{(i_2)}[h_{j^{(i_2)}}(x)]) = 0$ for any i_1 and i_2, which is further derived from the assumption that the hash-function assignments $\mathbf{1}(j^{(i_1)} = j)$ and $\mathbf{1}(j^{(i_2)} = j)$ are pairwise independent.

Recall that hash-function assignment uniformly samples from $\mathcal{H}$ for each object. This process maps n distinct inputs (object IDs) to n repeatable elements in $[k]$ (recall $k = |\mathcal{H}|$), which can be considered as hashing, denoted as s. We require s to be uniformly sampled from a pairwise-independent hashing family $\mathcal{S}$, which guarantees the pairwise independence of $\mathbf{1}(j^{(i_1)} = j)$ and $\mathbf{1}(j^{(i_2)} = j)$ for all i_1, i_2, and j. If we follow Section 3.5 to construct a hashing family mapping from $[n]$ to $[k]$, the parameter p^l should exceed $\max\{n, k\}$, which requires $2\lceil \log_2(\max\{n, k\} + 1)\rceil$ bits to represent one of its mappings. We have proved:

Theorem 5.2. $2\lceil \log_2(\max\{n, k\} + 1)\rceil$ *random bits are sufficient to generate all the hash-function assignments $[j^{(i)}]_{i\in[n]}$ while ensuring their pairwise independence, which is the prerequisite of Eq.* (4)*.*

5.3. Public or Private Randomness

Previous literature introduces the concept of public and private randomness as follows:

Public Randomness: the server generates a string of random bytes, which are accessible by the public (i.e., clients).

Private Randomness: the client generates a string of random bytes, which will be used in their algorithm. The client may or may not publish the random bytes.

Previous literature (e.g., [5], [12], [14]) had a strong interest in public randomness, because it was believed that public randomness could save communication costs. However, we argue that public and private randomness make no difference for an unbiased CMS, and no communication cost can be saved, for the following reasons: the server and a client (i.e., an object in CMS) have to sync on which hash function is assigned to the client, and they need to either sync (1) the identifier of a hash function, or (2) the assignment generator. There are at least $(d + 1)^2$ hash functions in a pairwise independent $\mathcal{H}_{pi}$, so identifying one of them requires $2\log_2 d$ bits. Meanwhile, Theorem 5.2 shows that at least $2\log_2 k$ bits are required to represent the assignment generator, where k is the hashing family size, so this is equivalent to requiring at least $4\log_2 d$ bits. Thus, syncing the identifier will be less expensive, and $2\log_2 d$ bits are needed to sync, regardless of which side generates the assignment. Additionally, $\log_2 n$ bits are required if the server and a client decide to sync on the assignment generator, because the client needs to send back the object ID (i.e., $i \in [n]$) to identify itself.

When reviewing the existing literature, the work claiming one-bit communication usually missed the communication cost of syncing the public randomness. For instance, [5] proposed a protocol to convert a private-randomness algorithm to a public one. Though the client only sends one bit as a response, it has to sync b bits from the server, where b is the number of bits the private-randomness algorithm would send. Moreover, the server also needs the client/object ID to know where the one bit comes from, which will contribute $\log n$ bits to the communication cost. The same problem also exists in the public-randomness algorithm in [14].

The only scenario where public randomness could save communication costs is when all the clients are co-located, referred to as a client group. The traffic inside a client group is assumed to be free. When utilizing the public randomness, the server will share the hash-function assignment generator with the client group, which costs $\lceil \max\{4\log_2 d, 2\log_2 n\}\rceil$ bits. Then, the client group shares the generator, calculates their assignment of the hash functions, and replies with their responses in sequence back to the server, which will cost $n \log m \approx n\epsilon$ bits. In this case, the total communication cost between the server and the client group is $\lceil \max\{4\log_2 d, 2\log_2 n\}\rceil + n\epsilon$. If the client group still uses private randomness, they will return $n(2\log_2 d + \epsilon)$ bits, which is larger than the cost from public randomness. However, the client-group assumption is contradictory to the motivation of LDP, which usually assumes the clients are distributed.

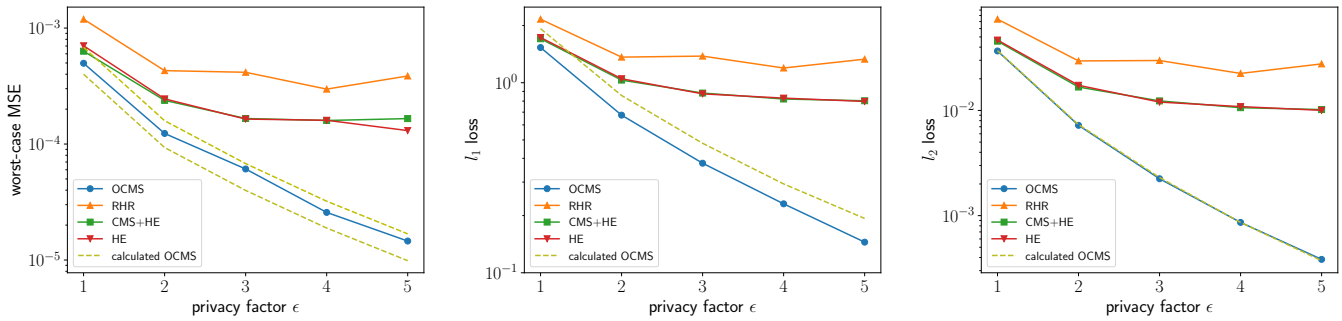
6. Experiment

This section empirically compares the OCMS+RR (implemented following Section 3.6) with the algorithms listed below, using both synthetic and real-world datasets:

1. Hadamard Encoding (HE): the first practical frequency estimation algorithm.

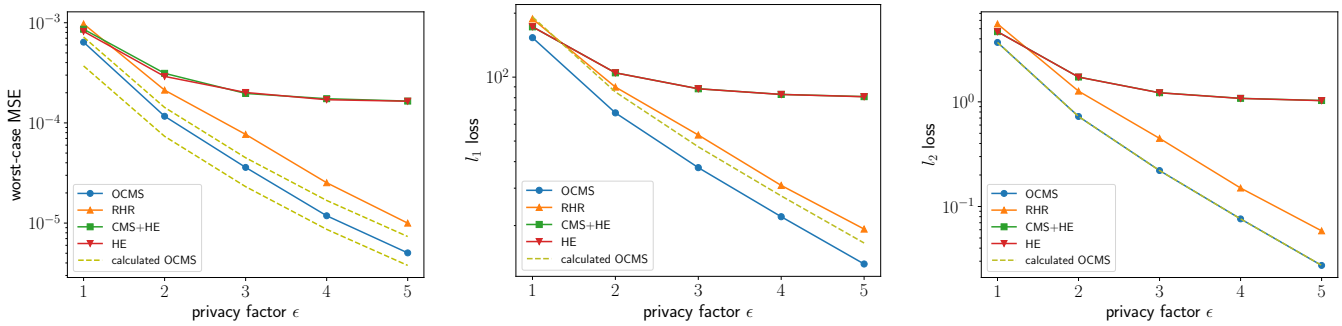
2. CMS with HE (CMS+HE): the original CMS algorithm with the same parameter as [7].

3. Recursive Hadamard Response (RHR): the state-of-the-art version of the Hadamard Response family.



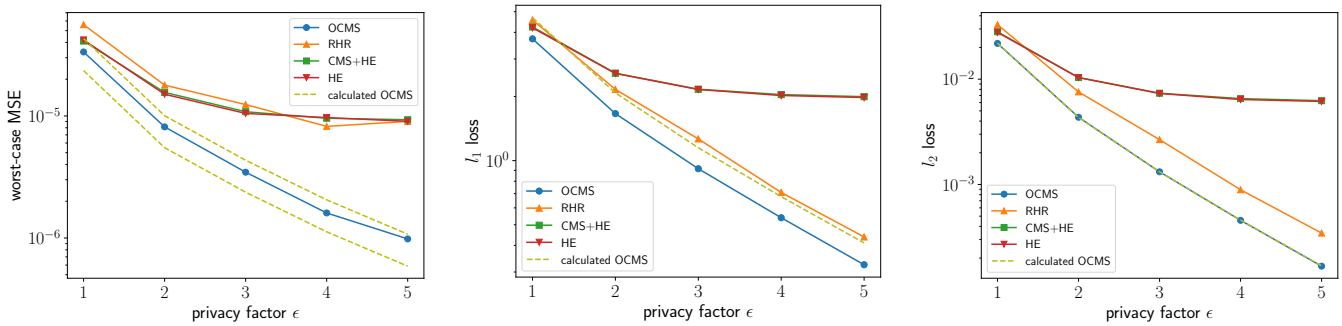
(a) (b) (c)

Figure 2: Worst-case MSE, l_1 loss, and l_2 loss vs. privacy factor ϵ given the Zipf dataset. See Section 6.1 for details.



(a) (b) (c)

Figure 3: Worst-case MSE, l_1 loss, and l_2 loss vs. privacy factor ϵ given the Gaussian dataset. See Section 6.2 for details.



(a) (b) (c)

Figure 4: Worst-case MSE, l_1 loss, and l_2 loss vs. privacy factor ϵ given the mini Kosarak dataset. See Section 6.3 for details.

Other algorithms listed in Section 5.1 are not included due to their very high communication cost, which is impractical for a very large dictionary.

In the following experiments, we use this estimator to estimate $\widehat{MSE}$:

$$\overline{MSE}(\hat{f}) = \max_{x \in \mathbf{x}} \frac{1}{t} \sum_{t} (\hat{f}(x) - f(x))^2, \tag{18}$$

where $\mathbf{x}$ denotes the values of interest and t denotes the experiment rounds. However, $\overline{MSE}(\hat{f})$ is biased because $\frac{1}{t}\sum_t(\hat{f}(x) - f(x))^2$ of each x is equivalent to a random variable fluctuating around $E[(\hat{f}(x) - f(x))^2]$, and the max operator will choose the largest deviated value. Supplement S8 provides an upper bound of $\overline{MSE}(\hat{f})$. As long as $\overline{MSE}(\hat{f})$ is between the upper bound and the calculated $\widehat{MSE}(\hat{f})$ (both bounds are plotted), we will consider the experimental $\overline{MSE}(\hat{f})$ consistent with the theoretical value from Theorem 3.6.

Moreover, we will compute l_1 / l_2 losses only from a subset of $[d]$, denoted as $\mathbf{x}$. By examining Eq. (21), one could prove that the l_1 / l_2 losses from Theorem 3.8 are still valid when replacing d with $|\mathbf{x}|$. Note that the calculated l_1 loss is an upper bound, so the experimental value is expected to be below it.

All the experiments below will be run 100 times for each privacy factor ϵ ranging from 1 to 5.

6.1. Verification of Precision and Equivalence

We use a synthetic dataset sampled from a specific Zipf's distribution, which has a frequency distribution as

$$Zipf(r) \propto \frac{1}{r^2},$$

where r is the order of a value ranked by its frequency. The dataset values are chosen deliberately, so all have the same mod for the Recursive Hadamard Response (RHR) (see Section 4.3). The dictionary size is set to 100k, and 10,000 objects are sampled from the distribution. The $\widehat{MSE}$, l_1 loss, and l_2 loss are evaluated based on the first 100 most frequent values (assuming they are known in advance). MSE-OCMS+RR with $f^* = 1$ (i.e., no prior knowledge on $f(x)$) and l-OCMS+RR are evaluated for $\widehat{MSE}$ and l_1 / l_2 losses, respectively, while RHR, HE, and CMS+HE are evaluated for all the metrics. Results are plotted in Fig. 2.

MSE-OCMS+RR and l-OCMS+RR outperform other algorithms in terms of $\widehat{MSE}$ and l_1 / l_2 losses, respectively, and their precision aligns with the calculated values. HE and CMS+HE have identical performance in all the metrics, supporting our theory that these two algorithms are equivalent. Consistent with our expectation, RHR performs the worst in all the metrics because we deliberately construct a dataset, though using only public information, to ensure every object has a 50% probability of colliding with another in RHR, resulting in poorer performance than HE, as analyzed in Section 4.3.

6.2. Synthetic Dataset with Prior Knowledge

A synthetic database is constructed with 10,000 objects sampled from a normal distribution with a standard deviation of 50. The mean of the distribution is an unknown integer picked from [1000, 9000]. The dictionary size is 10,000, so the values of all the objects are rounded to integers and truncated if they are outside of [0, 10,000]. Through several simulations of this distribution, we are confident that the probability of $\max_x f(x) \leq 0.01$ is high. Thus, we set $f^* = 0$ for MSE-OCMS+RR and its calculated $\widehat{MSE}$ is derived from Eq. (14) as:

$$\widehat{MSE}(\hat{f}) = \frac{1}{n}[\max_x f(x) + \frac{4e^\epsilon}{(e^\epsilon - 1)^2}].$$

Meanwhile, l-OCMS+RR is evaluated for l_1 / l_2 losses. RHR, HE, and CMS+HE are also evaluated for $\widehat{MSE}$, l_1 loss, and l_2 loss. Results are plotted in Fig. 3.

Similar to the above, MSE-OCMS+RR and l-OCMS+RR outperform other algorithms in terms of $\widehat{MSE}$ and l_1 / l_2 losses, respectively, and their precision aligns with the calculated values. HE and CMS+HE have identical precision in all the metrics, and they perform the worst. The precision of RHR lies between OCMS+RR and the other algorithms.

6.3. Real-world Dataset

Kosarak [23] is a dataset containing click-stream data from a Hungarian online news portal. The dataset is shrunk to 1% of its original size, denoted as mini Kosarak, which will be used in this section. It contains over 170,000 entries, all of which belong to a dictionary with a size of 26,000. MSE-OCMS+RR with $f^* = 1$ and l-OCMS+RR are evaluated for $\widehat{MSE}$ and l_1 / l_2 losses, respectively, while RHR, HE, and CMS+HE are evaluated for all the metrics. Results are plotted in Fig. 4.

MSE-OCMS+RR and l-OCMS+RR outperform other algorithms in terms of $\widehat{MSE}$ and l_1 / l_2 losses, respectively, and their precision aligns with the calculated values. HE and CMS+HE have identical precision in all the metrics, and they perform the worst in l_1 / l_2 losses. RHR has the same order of precision as l-OCMS+RR regarding l_1 / l_2 losses. However, it performs the worst in $\widehat{MSE}$. This is expected because Kosarak has one value whose frequency is above 0.4, which negatively impacts the MSE of other values sharing the same mod with it (see Section 4.3 for a detailed explanation).

7. Conclusion

This paper revisits the private CMS algorithm, corrects errors in the expectation and variance calculations in the original CMS paper, and optimizes CMS with randomized response to reduce all kinds of loss functions. The optimized CMS is proven theoretically and empirically to be the leading algorithm for reducing the worst-case MSE, l_1 loss, and l_2 loss, when requiring the communication cost to be logarithmic to the dictionary size. Moreover, this paper demonstrates that many existing algorithms are equivalent to CMS. Finally, we discuss the necessity of randomness in CMS and show that both public and private randomness have the same communication cost.

References

[1] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer, 2006, pp. 265–284.

[2] M. U. Hassan, M. H. Rehmani, and J. Chen, "Differential privacy techniques for cyber physical systems: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 746–789, 2019.

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

[4] Y. Wang, X. Wu, and D. Hu, "Using randomized response for differential privacy preserving data collection." in *EDBT/ICDT Workshops*, vol. 1558, 2016, pp. 0090–6778.

[5] R. Bassily and A. Smith, "Local, private, efficient protocols for succinct histograms," in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, 2015, pp. 127–135.

[6] P. Kairouz, K. Bonawitz, and D. Ramage, "Discrete distribution estimation under local privacy," in *International Conference on Machine Learning*. PMLR, 2016, pp. 2436–2444.

[7] A. Differential Privacy Team, "Learning with privacy at scale differential," 2017. [Online]. Available: https://docs-assets.developer.apple.com/ml-research/papers/learning-with-privacy-at-scale.pdf

[8] R. Bassily, K. Nissim, U. Stemmer, and A. Guha Thakurta, "Practical locally private heavy hitters," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[9] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 729–745.

[10] M. Ye and A. Barg, "Optimal schemes for discrete distribution estimation under locally differential privacy," *IEEE Transactions on Information Theory*, vol. 64, no. 8, pp. 5662–5676, 2018.

[11] J. Acharya, Z. Sun, and H. Zhang, "Hadamard response: Estimating distributions privately, efficiently, and with little communication," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1120–1129.

[12] W.-N. Chen, P. Kairouz, and A. Ozgur, "Breaking the communication-privacy-accuracy trilemma," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3312–3324, 2020.

[13] R. Pagh and M. Thorup, "Improved utility analysis of private countsketch," *Advances in Neural Information Processing Systems*, vol. 35, pp. 25 631–25 643, 2022.

[14] J. Acharya and Z. Sun, "Communication complexity in locally private distribution estimation and heavy hitters," in *International Conference on Machine Learning*. PMLR, 2019, pp. 51–60.

[15] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, 2014, pp. 1054–1067.

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

[17] S. P. Vadhan *et al.*, "Pseudorandomness," *Foundations and Trends® in Theoretical Computer Science*, vol. 7, no. 1–3, pp. 1–336, 2012.

[18] Wikipedia, "Finite field," http://en.wikipedia.org/w/index.php?title=Finite%20field&oldid=1220784972, 2024, [Online; accessed 15-May-2024].

[19] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 2018.

[20] E. V. Slud, "Distribution inequalities for the binomial law," *The Annals of Probability*, pp. 404–412, 1977.

[21] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2002, pp. 693–703.

[22] J. Håstad, "Lecture 3 - A theorist's toolkit," University Lecture. [Online]. Available: https://www.csc.kth.se/~johanh/verktyg/lecture3.pdf

[23] Kosarak. [Online]. Available: http://fimi.ua.ac.be/data/

Appendix A. Example on Reconstruction Process

Suppose a dictionary $\mathbb{V}$ has three possible values: 1, 2, and 3; and another dictionary $\mathbb{U}$ has five possible values: A, B, C, D, and E. A mechanism M perturbs a variable $V \in \mathbb{V}$ and outputs a value $U \in \mathbb{U}$. The probability that M outputs $U = u$ given $V = v$ is randomly assigned as

u \ v	1	2	3
A	0.133	0.143	0.192
B	0.031	0.212	0.658
C	0.287	0.155	0.039
D	0.161	0.254	0.060
E	0.389	0.236	0.051

which is equivalent to matrix P. Subsequently, the inverse matrix Q is derived as

v \ u	A	B	C	D	E
1	0.366	-0.088	2.289	-3.629	2.267
2	0.010	-0.393	-1.928	6.625	-1.294
3	0.332	1.560	0.476	-1.996	0.240

If M perturbs V and outputs $U = C$, the decoding process will return the column of Q corresponding to C, which is $[2.289, -1.928, 0.476]$. If $V = 1$, the probability that $M(V)$ outputs $U = C$ is 0.287. The same probability applies to the event that the reconstruction process $R(V)$, i.e., $Q[: M(V)]$, generates the corresponding column. Denote the column outputted by the reconstruction as $\tilde{V}$. One can verify that the expectation of $\tilde{V}$ equals $[1, 0, 0]$ when $V = 1$, which is consistent with Property 2.1.

Appendix B. Proof regarding Optimizing CMS with RR

We first explore the optimized m for CMS + RR. Substituting Eq. (12) into Eq. (6), we have

$$Var(\hat{f}(x)_{RR}) = \frac{1 - f(x)}{n(m-1)} + \frac{m[(1-f(x))(e^\epsilon - 1)(2-m) + me^\epsilon]}{n(m-1)(e^\epsilon - 1)^2} \quad (19)$$

Let us first study the worst-case MSE of CMS+RR, which is equivalent to the worst-case variance $Var(\hat{f}(x)_{RR})$. Observing that the variance is linear to $f(x)$, we have $\max_x MSE(\hat{f}(x)) = \max\{Var(\hat{f}(x)_{RR}|f(x) = 0), Var(\hat{f}(x)_{RR}|f(x) = 1)\}$. When $f(x) = 1$, we have

$$\frac{\partial Var(\hat{f}(x)_{RR}|f(x)=1)}{\partial f(x)} = \frac{m(m-2)e^\epsilon}{n(m-1)^2(e^\epsilon - 1)^2},$$

implying that $Var(\hat{f}(x)_{RR}|f(x) = 1)$ increases monotonically with m when $m \geq 2$. On the other hand, when $f(x) = 0$, we have

$$\frac{\partial Var(\hat{f}(x)_{RR}|f(x)=0)}{\partial f(x)} = \frac{(m - e^\epsilon - 1)(m + e^\epsilon - 1)}{n(m-1)(e^\epsilon - 1)^2},$$

which indicates a minimum at $m = e^\epsilon + 1$ given the range $m \geq 2$. However, when $m = e^\epsilon + 1$, $Var(\hat{f}(x)_{RR}|f(x) = 1)$ is always larger than $Var(\hat{f}(x)_{RR}|f(x) = 0)$. Given that $Var(\hat{f}(x)_{RR}|f(x) = 1)$ decreases as m decreases, while $Var(\hat{f}(x)_{RR}|f(x) = 0)$ increases when m decreases within

the range $[2, e^\epsilon + 1]$, there exists an m making them equal to each other. Solving

$$Var(\hat{f}(x)_{RR}|f(x) = 0) = Var(\hat{f}(x)_{RR}|f(x) = 1),$$

we have $m = e^{\epsilon/2} + 1$, and the corresponding variance is

$$Var(\hat{f}(x)_{RR}) = \frac{e^{\epsilon/2}}{n(e^{\epsilon/2} - 1)^2}.$$

When $m \in [2, e^{\epsilon/2} + 1]$, we have $Var(\hat{f}(x)_{RR}|f(x) = 1) < Var(\hat{f}(x)_{RR}|f(x) = 0)$. Also, $Var(\hat{f}(x)_{RR}|f(x) = 0)$ decreases with m in this range. Therefore,when $m \in [2, e^{\epsilon/2} + 1]$, we have

$$\begin{aligned}\max_{f(x)} Var(\hat{f}(x)_{RR}) &> \max_{f(x)} Var(\hat{f}(x)_{RR}|m = e^{\epsilon/2} + 1) \\ &= \frac{e^{\epsilon/2}}{n(e^{\epsilon/2} - 1)^2}.\end{aligned}$$

That is, m should never be smaller than $e^{\epsilon/2} + 1$.

Define f^* as the upper bound of $f(x)$, i.e., $\forall x : f(x) \leq f^*$. When $m \in [e^{\epsilon/2} + 1, e^\epsilon + 1]$, we have proven that $Var(\hat{f}(x)_{RR})$ increases with $f(x)$, so the goal becomes reducing $Var(\hat{f}(x)_{RR}|f(x) = f^*)$. The optimized m can be calculated by

$$\frac{\partial Var(\hat{f}(x)_{RR}|f(x) = f^*)}{\partial m} = 0,$$

which results in

$$m = 1 + \frac{\Delta_{MSE}}{f^* e^\epsilon + 1 - f^*}, \tag{20}$$

where

$$\Delta_{MSE} = e^{\epsilon/2}\sqrt{[(1 - f^*)e^\epsilon + f^*][f^* e^\epsilon + (1 - f^*)]}.$$

The corresponding variance is

$$Var^*(\hat{f}(x)_{RR}) = \frac{2(\Delta_{MSE} + e^\epsilon)}{n(e^\epsilon - 1)^2}$$

Notice that Eq. (20) is valid only when $m \in [e^{\epsilon/2} + 1, e^\epsilon + 1]$, which translates to $f^* \leq \frac{1}{2}$. When $f^* > \frac{1}{2}$, we have $m < 1 + e^{\epsilon/2}$, which has been proven to be worse than $m = 1 + e^{\epsilon/2}$. Therefore, we have

$$m = \begin{cases} 1 + \Delta_{MSE} & \text{if } f^* \leq \frac{1}{2} \\ 1 + e^{\epsilon/2} & \text{if } \frac{1}{2} < f^* \leq 1, \end{cases}$$

which corresponds to

$$Var^*(\hat{f}(x)_{RR}) = \begin{cases} \frac{2(\Delta_{MSE} + e^\epsilon)}{n(e^\epsilon - 1)^2} & \text{if } f^* \leq \frac{1}{2} \\ \frac{e^{\epsilon/2}}{n(e^{\epsilon/2} - 1)^2} & \text{if } \frac{1}{2} < f^* \leq 1 \end{cases}.$$

Regarding Theorem 3.7, we observe that $Var(\hat{f}(x)_{RR})$ in Eq. (19) is linear to $f(x)$, so

$$\begin{aligned}Var(\hat{f}(x)_{RR}|f(x) = f^* + \delta) - Var(\hat{f}(x)_{RR}|f(x) = f^*) \\ = \frac{(m-1)^2 - e^\epsilon}{(m-1)(e^\epsilon - 1)n}\delta.\end{aligned}$$

Substituting $m = 1 + \Delta_{MSE}$ into the above equation and simplifying it, we have:

$$\begin{aligned}Var(\hat{f}(x)_{RR}|f(x) = f^* + \delta) - Var(\hat{f}(x)_{RR}|f(x) = f^*) \\ = \frac{\delta(1 - 2f^*)e^\epsilon}{n\Delta_{MSE}}.\end{aligned}$$

Since $Var(\hat{f}(x)_{RR})$ increases with $f(x)$ given $m = 1 + \Delta_{MSE}$, we have

$$\begin{aligned}\widehat{MSE}(\hat{f}(x)) &= Var(\hat{f}(x)_{RR}|f(x) = f^* + \delta) \\ &= \widehat{MSE}(\hat{f}|f^*) + \frac{\delta(1 - 2f^*)e^\epsilon}{n\Delta_{MSE}},\end{aligned}$$

which proves the equality sign of Theorem 3.7. Given that $\frac{e^\epsilon}{\Delta_{MSE}}$ decreases when $f^* \in [0, \frac{1}{2}]$, we have $\frac{e^\epsilon}{\Delta_{MSE}} \leq 1$, which proves the "$\leq$" sign of Theorem 3.7. □

Now, we will explore the optimized parameters for l_2 loss. By considering only the pairwise independence of $\mathcal{H}$, we integrate Eq. (6) into Eq. (11):

$$\begin{aligned}l_2(\hat{f}) &= \sum_{x\in[d]} \frac{m}{n(m-1)^2}\big[\big(Var(R| =) + (m-1)Var(R| \neq) \\ &\quad + \frac{m-1}{m}\big)(1 - f(x)) + mVar(R| =)f(x)\big] \\ &= \frac{m}{n(m-1)^2}\big[\big(Var(R| =) + (m-1)Var(R| \neq) \\ &\quad + \frac{m-1}{m}\big)(d-1) + mVar(R| =)\big] \end{aligned} \tag{21}$$

Applying RR and Substituting Eq. (12) into Eq. (21), we obtain

$$l_2(\hat{f}_{RR}) = \frac{(d + e^\epsilon - 1)m^2}{n(m-1)(e^\epsilon - 1)^2} + \frac{(d-1)(2m + e^\epsilon - 1)}{n(m-1)(e^\epsilon - 1)}.$$

By studying its derivative with respect to m, we can find its minimum when

$$m = 1 + \frac{\Delta_l}{e^\epsilon + d - 1},$$

where

$$\Delta_l = \sqrt{(e^\epsilon + d - 1)(de^\epsilon - e^\epsilon + 1)}e^{\epsilon/2}.$$

In practice, we choose the closest integer to $1 + \frac{\Delta_l}{e^\epsilon + d - 1}$ as m. The corresponding optimized l_2 is the one in Theorem 3.8. Given the Cauchy-Schwarz inequality, we have $l_1 \leq \sqrt{d\, l_2}$, which derives the optimized l_1 in Theorem 3.8. Corollary 3.4 is proven accordingly when applying $d \gg e^\epsilon$.

Appendix C.
Data Source of Section 5.1

We start with the precision. Since HE is equivalent to unbiased CMS+RR with $m = 2$, we use Eq. (19) to compute its $\widehat{MSE}$, l_1 loss, and l_2 losses. The computed $\widehat{MSE}$ is the same as in [11]. The l_1 and l_2 losses of RHR were taken from [12], but it did not calculate the $\widehat{MSE}$. Section 4.3 shows that RHR is a skewed version of unbiased CMS, and the $\widehat{MSE}$ of RHR could be worse than that from the CMS+RR with $m = 2$ (i.e., HE), so the $\widehat{MSE}$ of RHR is $\Omega\left(\left(\frac{e^\epsilon+1}{e^\epsilon-1}\right)^2\right)$. OLH is equivalent to the unbiased CMS with $m = 1 + e^\epsilon$, so we use Eq. (19) and follow Section 3.4 to calculate its l_1 loss, l_2 loss, and $\widehat{MSE}$. Section 4.5 shows that CMS+HE is equivalent to the unbiased CMS+RR with $m = 2$ (i.e., HE), so its l_1 loss, l_2 loss, and $\widehat{MSE}$ are the same as HE. The l_1 and l_2 losses of SS are presented in [10], but their calculation of l_1 is incorrect: (1) they approximate $\sum_x \in [d]|\hat{f}(x)-f(x)|$ as a normal distribution, whereas the central limit theorem requires each $|\hat{f}(x)-f(x)|$ to be i.i.d, and (2) they derived the result based on the Mean Absolute Deviation of the approximated Gaussian distribution, which is equivalent to using $|\sum_x \hat{f}(x) - f(x)|$ to represent $\sum_x |\hat{f}(x) - f(x)|$. The correct way to compute l_1 is to use Cauchy–Schwarz inequality, i.e., $l_1 \leq \sqrt{d\, l_2}$, which provides a tight upper bound of l_1 as $\frac{2de^{\epsilon/2}}{e^\epsilon-1}$. The $\widehat{MSE}$ of SS is derived using Eq. (2) in [9] with

$$
\begin{aligned}
p &= \frac{ke^\epsilon}{ke^\epsilon + d - k} \\
q &= \frac{ke^\epsilon}{ke^\epsilon + d - k}\frac{k-1}{d-1} + \frac{d-k}{ke^\epsilon + d - k}\frac{k}{d-1} \\
k &= d/(e^\epsilon + 1) \\
f &= 1
\end{aligned}
$$

where k represents the size of a subset in SS. $f = 1$ because the $Var(\hat{f}(x))$ increases with $f(x)$ for SS.

The $\widehat{MSE}$ of a-RP and s-RP is calculated using Eq. (2) in [9] as well. Both cases where $f(x) = 0$ and $f(x) = 1$ are calculated, and the worst-case variance is the maximum between them.

Next, we present the data source regarding communication cost. We first compute the cost of OCMS+RR, which requires the server and a client to sync on the assigned hash function and the perturbed response $z^{(i)}$. Section 3.5 shows that $2\lceil \log_2 \max\{d+1, 5m\}\rceil$ bits are required to represent an approximately pairwise independent hash function. Also, we need another $\log_2 m$ bits to sync the perturbed response $z^{(i)}$. Given that OCMS+RR has either $m = 1 + e^{\epsilon/2}$ or $m = 1 + e^\epsilon$ depending on the optimization goal, the overall communication costs of MSE-OCMS+RR and l-OCMS+RR are approximately $\max\{2\log_2 d + \epsilon/2, \frac{3\epsilon}{2} + 6\}$ and $\max\{2\log_2 d + \epsilon/2, 3\epsilon + 6\}$, respectively.

The costs of HE and RHR are taken from their respective literature [11] and [12]. It is notable that we do not use the communication cost from their public-randomness scheme because they fail to consider the cost of syncing the public randomness, as pointed out in Section 5.3. For OLH, its hash function maps $[d]$ to $[1 + e^\epsilon]$, and each mapping is mutually independent, so it requires $d\log_2(1 + e^\epsilon) \approx d\epsilon$ bits to represent all the mappings. The original CMS [7] did not consider pairwise independence, so its hash function mapping from $[d]$ to $[m]$ requires $d\log_2 m$ bits to communicate. SS randomly selects $\frac{d}{e^\epsilon+1}$ values from $[d]$, and this subset selection requires $\frac{d}{e^\epsilon+1}$ bits to represent. RAPPOR generates a one-hot vector with size d, which requires d bits to transfer.

Appendix D.
Supplements

Supplement sections (prefixed with "S") can be found in the Supplementary Material of this article.