When Focus Enhances Utility: Target Range LDP Frequency Estimation and Unknown Item Discovery

Bo Jiang*†, Wanrong Zhang*†, Donghang Lu*†, Jian Du*, and Qiang Yan*.
* TikTok Inc., † Equal Contribution.
Email: {bojiang, wanrongzhang, donghang.lu, jian.du, yanqiang.mr}@tiktok.com

***Abstract*—Local Differential Privacy (LDP) protocols enable the collection of randomized client messages for data analysis, without the necessity of a trusted data curator. Such protocols have been successfully deployed in real-world scenarios by major tech companies like Google, Apple, and Microsoft. In this paper, we propose a Generalized Count Mean Sketch (GCMS) protocol that captures many existing frequency estimation protocols. Our method significantly improves the three-way trade-offs between communication, privacy, and accuracy. We also introduce a general utility analysis framework that enables optimizing parameter designs. Based on that, we propose an Optimal Count Mean Sketch (OCMS) framework that minimizes the variance for collecting items with targeted frequencies. Moreover, we present a novel protocol for collecting data within unknown domain, as our frequency estimation protocols only work effectively with known data domain. Leveraging the stability-based histogram technique alongside the Encryption-Shuffling-Analysis (ESA) framework, our approach employs an auxiliary server to construct histograms without accessing original data messages. This protocol achieves accuracy akin to the central DP model while offering local-like privacy guarantees and substantially lowering computational costs.**

I. Introduction

Differential Privacy (DP) is a mathematical definition of privacy that provides strong worst-case privacy guarantees for individuals within a dataset, while enabling data analysis. While numerous differentially private algorithms have been created for various analysis tasks, most high-profile real-world applications focus on the Local Differential Privacy (LDP) model. The LDP model is particularly attractive because it does not require trust in a central data curator; instead, it ensures that data is obfuscated at the source, before being collected. Some successful deployments in major tech companies include Google's RAPPOR [1], [2], a protocol integrated into Chrome, to collect web browsing behavior; Apple [3] collecting type history and emojis; and Microsoft [4] collecting telemetry across millions of devices.

Frequency estimation, where the goal is to estimate the occurrence of items within a dataset, is a common task in data analytics that has seen various implementations under the LDP model. The process for achieving this usually has several steps: clients first encode their responses (inputs) into a designated format, then perturb these encoded values to generate outputs. These outputs are sent to an aggregator, who collects and decodes the values to estimate the number of clients associated with each specific input. One notable framework is the "pure" Local Differential Privacy (LDP) framework [5], which includes many existing protocols (e.g., Basic RAPPOR) as special cases. This enables us to precisely analyze and compare the accuracy of different protocols and also offers optimal parameters in the randomization step. However, their framework has limitations in scenarios requiring extensive data collection, particularly within large domains. Some protocols have incorporated with hash functions to encode data into a more manageable domain size. The hash functions, on one hand, reduce communication cost and enable encoding data from unknown or unlimited domains; on the other hand, they introduce collisions during encoding, where multiple items are mapped to the same hash value. These collisions make the raw data values hard to distinguish, leading to reduced utility in terms of the accuracy of aggregation. To counter the utility reduction caused by collisions, RAPPOR adopts client cohorts and extra LASSO clustering in aggregation [6]. However, it is not straightforward to derive an error bound for utility. Comparably, Apple's Count Mean Sketch (CMS) is more favorable, as it utilizes a statistical data structure to average out other values in the collision and features a closed-form variance expression for the aggregation. However, Apple's CMS has deterministic perturbation/aggregation parameters which are not optimized to balance the accuracy and communication tradeoff. In this paper, we propose a Generalized Count-Mean-Sketch (GCMS) protocol that builds upon and extends the foundational ideas of Apple's CMS.

Our proposed mechanism aims to improve the existing framework by accounting for both the randomness inherent in truthful responses and the complications arising from hash collisions. By doing so, we enhance the accuracy of frequency estimation under the LDP model. One of the key advantages of our GCMS protocol is its ability to achieve a significantly smaller communication cost in terms of plaintext length while maintaining the same level of privacy protection as the original CMS. Furthermore, we provide a utility guarantee for our general CMS protocol, enabling the optimal design of parameters tailored to specific item frequency regimes. This ensures that our protocol is not only more efficient in terms of communication cost but also adaptable to various data

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.231397
www.ndss-symposium.org

collection scenarios to further boost the estimation accuracy.

To further amplify privacy, we present our protocol within the Encryption-Shuffling-Analysis (ESA) framework. Each client's encoded and randomized output is encrypted, and these encrypted outputs are then shuffled by a trusted shuffler, which involves randomly permuting the data to break the direct link between clients and their responses. This randomness in permutation significantly amplifies privacy by making it even harder to trace any response back to an individual client, enhancing an ϵ-LDP randomizer to $\mathcal{O}\left(1 - e^{-\epsilon_0}\sqrt{\frac{e^{\epsilon_0}(1/\delta)}{n}}\right)$-DP. The server then aggregates the shuffled and encrypted data and performs the analysis.

Note that these frequency estimation protocols only work effectively when the data domain is known. There are many scenarios where the server does not have full knowledge of the clients' input dictionary. For example, in word typing, new words are constantly being invented alongside the existing dictionary. Another example is URL collection: whenever new content is uploaded, a new URL is generated. Existing frequency estimation protocols for unknown domains primarily operate under the pure LDP model, but this comes at a cost. Fanti et al. [2] use RAPPOR to segment and reconstruct inputs via the EM algorithm, but the approach is computationally expensive. The private sequence fragment puzzle method [3], based on CMS, also segments and reconstructs data but struggles with trade-offs between fragment size and privacy leakage. PrivateTrie [7] constructs a trie structure for iterative data collection, but its communication overhead scales still quadratically with data length. These protocols all aim to ensure strong local privacy guarantees in the standard client-server setting but suffer from limited scalability or utility, which hinders their practical deployment.

To address these limitations, we introduce a protocol designed for unknown-domain data collection that offers a distinct trade-off. While our method improves performance and achieves accuracy comparable to central DP, it relies on a semi-trusted architecture. Specifically, we use the stability-based histogram technique [8], originally proposed in the central DP model, and implement it via the ESA framework to preserve privacy. An auxiliary server receives only encrypted and hashed messages, preventing it from tracing raw inputs. Messages are additionally encrypted with the auxiliary server's public key before passing through the shuffler. Our protocol avoids input segmentation and reconstruction, offering an efficient alternative to existing solutions.

However, unlike pure LDP protocols, our approach provides a quasi-local differential privacy guarantee, and it improves utility but requires stronger trust assumptions. Therefore, we recommend that practitioners select the protocol that best fits their deployment setting, as each method presents a unique trade-off between trust, privacy, and utility.

In summary, our paper makes the following contributions:

1. We propose a unified framework for LDP frequency estimation. By instantiating the corresponding parameters, our framework can be used to analyze many protocols that falls in the"pure" Local Differential Privacy framework, as well as Apple's CMS. Moreover, our design allows the length of the privatized message to be a tunable parameter, which offers flexibility for different communication considerations. By choosing the appropriate parameter, this approach significantly reduces communication costs while maintaining the same level of privacy protection.

2. We provide a general utility guarantee of our framework, beyond which, we propose a parameter optimization algorithm, tailored to specific item frequency regimes. In addition, we present a correction to the variance expression in Apple's original paper [3]. Our results also suggest that most of the LDP frequency estimation protocols, such as Apple's CMS, choose suboptimal parameters for the randomization step when there is a target frequency regime.

3. We propose a protocol for collecting data from an unknown domain that trades stronger trust assumptions for improved accuracy and lower computational overhead. By combining the stability-based histogram technique with the Encryption-Shuffling-Analysis (ESA) framework, and incorporating an auxiliary server that constructs the histogram without access to raw data, our protocol achieves accuracy comparable to the central DP model, while providing quasi-local privacy guarantees. It avoids the segmentation and reconstruction steps required by existing LDP methods, offering a more efficient and practical solution.

4. We visualize our theoretical analysis and conduct extensive experiments with real data. Our results suggest that: 1. GCMS constantly provides better utility than CMS under the same communication cost and under the same privacy leakage. 2. With optimized mechanism parameters, our OCMS achieves lower variance for specific target frequencies, enabling clients to obtain more precise results for query items within certain frequency ranges.

II. Preliminaries

In this section, we introduce several techniques relevant to this paper.

A. Privacy Models

Differential privacy [9] is a mathematical guarantee about database privacy.

Definition 1 ((ϵ, δ)-DP [9])**.** *A randomized algorithm $\mathcal{M} : \mathcal{X} \to \mathcal{R}$ is (ϵ, δ)-differentially private if for every pair of neighboring datasets $X, X' \in \mathcal{X}$, and for every event $S \subseteq \mathcal{R}$,*

$$\Pr(\mathcal{M}(X) \in S) \leq e^{\epsilon} \cdot \Pr(\mathcal{M}(X') \in S) + \delta. \tag{1}$$

When δ = 0, we say it is ϵ-DP (or pure DP).

In central DP model, there is a trusted server collecting raw data from clients and applies a randomized algorithm to the aggregated dataset to produce differentially private outputs. We define the neighboring datasets as the datasets that arbitrarily differ in the values at most one entry. Thus, (1) guarantees that the presence or absence of an individual's data in a dataset

does not significantly affect the outcome of queries, controlled by the privacy parameters ϵ and δ.

Local Differential Privacy (LDP) [10] removes the need for a trusted servers by ensuring that the privacy of each individual's data is protected before it is collected by the server. In LDP, each client applies a randomized algorithm to their own data locally, and only the perturbed data is sent to the server. Here, we define the neighboring datasets as any pair of values from the input support, and (1) guarantees that any input values are indistinguishable.

Shuffle Differential Privacy (Shuffle DP) [11] is an intermediate model between Central DP and LDP that enhances the privacy guarantees of LDP by adding a shuffling step. In this model, each client first perturbs their data locally (as in LDP) and then submits it to a shuffler, which randomly permutes the data before sending it to the server. Feldman et al. [12] have shown that the random permutation has a strong privacy amplification effect.

Theorem 1 (Privacy amplification by shuffling [12])**.** *For local randomizer with*

$$\epsilon \leq \log\left(\frac{n}{8\log(2/\delta)} - 1\right),$$

for any $\delta \in [0,1]$, shuffling can achieve (ϵ_c, δ)-DP with

$$\epsilon_c \leq \log\left(1 + (e^\epsilon - 1)\left(\frac{4\sqrt{2\log(4/\delta)}}{\sqrt{(e^\epsilon+1)n}} + \frac{4}{n}\right)\right), \quad (2)$$

where n denotes the number of data items.

The Encryption-Shuffling-Analysis (ESA) framework builds on the principles of Shuffle DP to further enhance security: clients encrypt their output with the server's public key. Therefore, only the server can decrypt and obtain the plaintext. This model has been widely adopted by Apple[3], Google[6], and Microsoft[4].

B. Pure LDP Protocol

The LDP mechanisms designed specifically for frequency estimation are called Frequency Oracles. Oracles vary in their construction, accuracy guarantees, and the size of the domain for which they are best suited. In terms of the design of the frequency oracle, the concept of a "pure" protocol is introduced in [13]. A pure protocol requires the probability that any value d_1 is mapped to its own support set to be the same for all values. We use p to denote this probability. It also requires a value $d_2 \neq d_1$ to be mapped to d_1's support set with probability q. This must be the same for all pairs of d_1, d_2. The pure protocol has the following two major advantages:

- The pure protocol simplifies the privacy analysis for LDP. A pure protocol satisfies ϵ-LDP as long as
$$p/q \leq e^\epsilon.$$
- The pure protocol provides a closed-form estimator for frequency estimation. Let v_i be the privatized data submitted by client i. The unbiased estimator for number of times that d occurs is
$$\hat{f}(d) = \frac{\sum_{i=1}^{N} \mathbb{1}_{\{d \in support(v_i)\}} - nq}{(p-q)^2}.$$
The corresponding variance is
$$\text{Var}[\hat{f}(d)] = \frac{f(d)p(1-p) + (n - f(d))q(1-q)}{(p-q)^2}.$$

III. Related Works

In this section, we present related works to this paper, including LDP mechanisms and real-world implementations for frequency estimation, as well as works that provide functions for data collection with unknown domain.

A. LDP Mechanisms for Frequency Estimation

Many existing protocols can be viewed as special cases of a pure protocol. Let us start with direct perturbation for a basic binary model. Binary Randomized Response (BRR), where $p = \frac{e^\epsilon}{e^\epsilon+1}$ and $q = \frac{1}{e^\epsilon+1}$, is proven to be optimal [14] for binary data collection. General Randomized Response (GRR), also known as Direct Encoding (DE) [13], where $p = \frac{e^\epsilon}{e^\epsilon+|\mathcal{D}|-1}$ and $q = \frac{1}{e^\epsilon+|\mathcal{D}|-1}$, can handle high-dimensional data for $d \geq 2$, but its utility decreases significantly for large d.

Unary encoding (UE) reduces the sensitivity of high-dimensional data by converting a value $d \in \mathcal{D}$ into a binary vector of size $|\mathcal{D}|$, where only the d-th bit is 1, and the rest are 0s. Symmetric Unary Encoding (SUE) [13] perturbs each bit of the unary encoded vector independently using $\epsilon/2$ BRR, ensuring a symmetric property $p + q = 1$. However, this symmetry does not minimize variance. Optimized Unary Encoding (OUE) [13] improves upon SUE by minimizing variance, specially under small ϵ and low frequencies. While UE-based methods enhance utility for direct encoding, they require each client to send at least $|\mathcal{D}|$ data to the server, increasing communication overhead.

To reduce communication cost, transformation-based approaches, such as Hadamard Randomized Response (HRR) [3] and S-Hist [15], compress high-dimensional data into a smaller domain. Subset Selection (SS) [16] reduces communication by randomly selecting s items from the domain $\mathcal{D}$. Among these, the hashing-based methods are most widely studied. Hash encoding maps high-dimensional data to a smaller domain [13], then uses either direct perturbation or unary encoding to release the hashed data.

Real-world LDP implementations, such as Google's RAPPOR [6], O-RAPPOR [17], and Apple's Count Mean Sketch [3], usually combine hash-based methods with UE. That is, the data is first hashed to a smaller domain, then released with UE. RAPPOR uses cohorts and LASSO regression to reduce information loss from hash collisions, while Apple's Count Mean Sketch uses a probabilistic data structure to average aggregated hashed items. With a large number of hash functions, the summation of the counts for the specific hashed values obtained by different hash functions converges to the

true count plus a calculable bias. However, the parameters used in this protocl are suboptimal, both in terms of accuracy and communication cost, as we will show later in this paper.

There are a line of work focusing on improving the three way tradeoff between accuracy, efficiency, and privacy of the LDP-based frequency estimation frameworks. Projective Geometry Response[18] uses geometric transformations for efficient, accurate estimations, while Adaptive Online Bayesian Estimation[19] applies Bayesian updates to adjust estimations in real-time, though both can be computationally complex. Wiener Filter-Based Deconvolution [20] enhances noise reduction using filter-based techniques, at the cost of higher computational overhead. Notably, PK-RAPPOR [21] incorporates prior knowledge of item frequency rankings, improving accuracy specifically for frequently occurring items. However, obtaining accurate item frequency rankings is challenging in practice, and these rankings can change over time. Nonetheless, the concept of leveraging prior information for improved mechanism design remains a valid and valuable approach, which also motivates our design.

There has been a growing body of work aimed at enhancing pure LDP protocols for specific data types and tasks. For example, Zheng et al. [22] introduce Joint Randomized Response, which leverages user group correlations to improve utility under LDP. Feng and Zhang [23] propose a multi-level personalized local differential privacy method, which allocates privacy budget for different attributes for different sensitivities; Vijayachandran [24] explores LDP mechanisms tailored for key-value pair data; Zhang et al. [25] design sketches-based techniques for join size estimation under LDP. These works complement our paper in that they focus on improving LDP utility for specialized tasks or specific data structures, while our work targets general-purpose frequency estimation.

We acknowledge a concurrent and independent work by Pan [26], which, like ours, revisits the Count-Mean Sketch (CMS) protocol. While both papers correct the bias in Apple's CMS, Pan focuses on optimizing parameter selection (e.g., hash width and depth) within the standard CMS framework to minimize worst-case MSE. In contrast, our work generalizes the CMS protocol to improve utility across a broader frequency regime and, importantly, propose a protocol to support unknown-domain data collection.

B. Data Collection with Unknown Domain

Most of the existing works on LDP for frequency estimation requires that the server knows the data domain $\mathcal{D}$. However, there are cases where the server does not fully know the input dictionary, especially for lengthy strings like URLs or full names. Few works address data collection with an unknown domain. Fanti et al. [2] propose an algorithm for RAPPOR that splits a message into disjoint segments, releases each segment with RAPPOR, and reconstructs the message using expectation maximization (EM). This process is highly costly due to the exhaustive search needed for EM. A private sequence fragment puzzle method based on CMS [3] also uses a segmentation-release-reconstruct approach. This method faces similar issues: small fragment sizes lead to numerous fragments and high reconstruction costs, while large fragment sizes increase privacy leakage. PrivateTrie [7] constructs a trie data structure to store string values iteratively from each client, reducing the reconstruction from a graph to a tree. However, it comes with quadratic communication cost relative to the data length.

IV. Generalized Count Mean Sketch

We first study the problem of privacy-preserving frequency estimation of clients' private data. We consider a set of n clients, each possessing private data $d_i \in \mathcal{D}$, where i denotes the client index. Each client aims to collaborate with a server to receive some service but does not fully trust the server and will only contribute their data if it is properly privatized and secured. To achieve this, each client submits a privatized and secured version v_i of d_i to the server. The server, requires the following functionalities:

- Upon receiving the set $\{v_i\}_{i=1}^n$, the server aims to estimate the number of clients who possess a specific data d, denoted as $f(d)$.
- The server may have complete, partial, or no knowledge of the domain $\mathcal{D}$. Therefore, the server requires some of the popular data strings collected to keep itself up-to-dated.

Utility Definition: When performing frequency estimation, the servers hopes that the estimator $\hat{f}(d)(\{v_i\}_{i=1}^n)$ has a small mean square error (MSE) with respect to the true frequency $f(d)$:

$$E[(\hat{f}(d) - f(d))^2].$$

Local mechanisms feature lightweight designs, and communication cost and computation cost should be considered as part of the system requirements. We define the communication cost as the number of bits needed to transmit the plaintext message of each client. We note that the actual bits transmitted end-to-end can increase after encryption. However, the length of the plaintext is a more accurate representation of the communication cost across various encryption protocols. Each client's input should not burden the server during aggregation. To this end, the framework should feature low computation cost, which is defined as the computation time at the server per client's message.

Given these considerations, we propose the Generalized Count Mean Sketch (GCMS) protocol. The term "generalized" refers to the increased flexibility in designing parameters such as p and q, compared to Apple's CMS. Building on this, we propose our optimal Count Mean Sketch (OCMS) that optimizes p and q for specific frequency regimes. We present our protocol within the ESA framework.

A. General Count Mean Sketch (GCMS) Framework

Preparation: Before data collection, the server generates k independent hash functions $\mathcal{H} \triangleq \{h_1, h_2, ..., h_k\}$, where each hash function deterministically maps any input to a discrete number in $[\mathbf{m}] = [1, ..., m]$, with m being the hashing range,

Algorithm 1 On-device LDP Algorithm

Input: $\mathcal{H}$: the hash universe; $[\mathbf{m}]$: extension domain; d: raw message; s: message size; p: inclusion probability; pk: server's public key.

Output: Encrypted privatized message v.

Randomly select $h_j \sim \mathcal{H}$;
Calculate the hashed value $r = h_j[d]$;
Initiate output vector $\mathbf{x}$ as an empty set;
Add r to $\mathbf{x}$ with probability of p;
if r is added to $\mathbf{x}$ **then**
  Randomly select $s-1$ elements from $[\mathbf{m}]/r$;
else
  Randomly select s elements from $[\mathbf{m}]/r$;
end if
Add selected elements to $\mathbf{x}$;
Encrypt with Server's public key

$$v = E_{pk}[\mathbf{x}, j];$$

return v

Algorithm 2 Constructing Sketch Matrix

Input: m: domain of hash function, k: size of the hash universe, $\langle \mathbf{x}_i, j_i \rangle_{i=1}^{n}$: decrypted clients' messages.

Output: Sketch matrix $\mathcal{M}$.

Initialized $\mathcal{M} = [0]^{[m \times k]}$;
for Each pair of $\langle \mathbf{x}, j \rangle$ **do**
  for Each value $x \in \mathbf{x}$ **do**
    $\mathcal{M}_j[x] \leftarrow \mathcal{M}_j[x] + 1$;
  end for
end for
return $\mathcal{M}$

Algorithm 3 Frequency Estimation

Input: d: item to check frequency, p: inclusion probability, s: message size, m: domain of hash function, $\mathcal{H}$: hash universe, $\mathcal{M}$: aggregated sketch matrix.

Output: Estimated frequency $\hat{f}(d)$.

Calculate q according to:

$$q = \frac{p(s-1) + (1-p)s}{m-1};$$

Initialized count $C(d) = 0$;
for $j \in [1, 2, ..., k]$ **do**
  $C(d) \leftarrow C(d) + \mathcal{M}_j[h_j[d]]$;
end for
Get $\hat{f}(d)$ according to:

$$\hat{f}(d) = \frac{C(d) - \frac{pn}{m} - qn\left(1 - \frac{1}{m}\right)}{(p-q)\left(1 - \frac{1}{m}\right)};$$

return $\hat{f}(d)$

$\mathcal{H}$ being the universe of hash functions. The server then sends $\mathcal{H}$ and its public key pk to each client.

The entire data collection pipeline is illustrated in Figure 1. It consists of three distinct phases, each operating on different platforms:

1) The initial phase involves all on-device algorithms, including hash encoding and privatization, and encryption with the server's public key pk, as described in Algorithm 1.
2) The encrypted privatized data is then transmitted through an end-to-end encrypted channel and received by the shuffler. The shuffler then forwards the data to the server after a random shuffling operation.
3) Finally, the server decrypts the input data, performs data aggregation, and obtains the Frequency Lookup Table - a sketch matrix $\mathcal{M}$, via Algorithm 2.

We now detail operations in each phase.

Phase 1: On-device operations. We present the process for a single data privatization and encryption in Algorithm 1, which consists of the following steps.

Hash Encoding: Each client first uniformly selects a hash function from $\{h_1, h_2, \ldots, h_k\}$ and calculates a hashed value $r = h_j(d)$ of their raw data d. By construction, r is an integer within $[1, m]$.

Probabilistic Inclusion: Each client initializes their privatized vector $\mathbf{x}$ as an empty set, then adds r to $\mathbf{x}$ with probability $p \in [0.5, 1]$.

Probabilistic Extension: Set the extension domain for each client as $[\mathbf{m}]/r = \{1, 2, \ldots, r-1, r+1, \ldots, m\}$. If r is added to $\mathbf{x}$, then uniformly select $s-1$ elements from $[\mathbf{m}]/r$ and append them to $\mathbf{x}$. If r is not added to $\mathbf{x}$, then uniformly select s elements from $[\mathbf{m}]/r$ and append them to $\mathbf{x}$. Return the privatized vector along with the selected hash index $\langle \mathbf{x}, j \rangle$.

Encryption and Release: Each client encrypts $\langle \mathbf{x}, j \rangle$ with the server's public key pk to obtain $v = E_{pk}[\mathbf{x}, j]$, then releases v to the shuffler.

Compared to the original CMS, our GCMS uses a novel method to constructing the privatized message (steps (b)&(c)): the length of our privatized message is s, with $s \leq m/2$, whereas in Apple's CMS, each client's input is a binary vector of size m. Our construction ensures that the probability of adding r is p, and the probability of adding all other hash values $[\mathbf{m}]/r$ is q, where

$$q = \frac{s-p}{m-1}. \tag{3}$$

This achieves the same randomness as in perturbing a binary vector, while significantly reduces the communication cost in terms of plain text length.

Phase 2: Shuffler's anonymization and shuffling. This phase, which is standard, involves *Anonymization* and *Shuffling* after receiving each client's input. The shuffler removes all id traceable information from the received message, e.g., the communication header. It then randomly shuffles the anonymized messages in batches and then releases the data to the server. The shuffled sequence is denoted as $\pi(v_1, v_2, \ldots, v_n)$, where π is a random permutation function. Additionally, the shuffler may block clients who have already

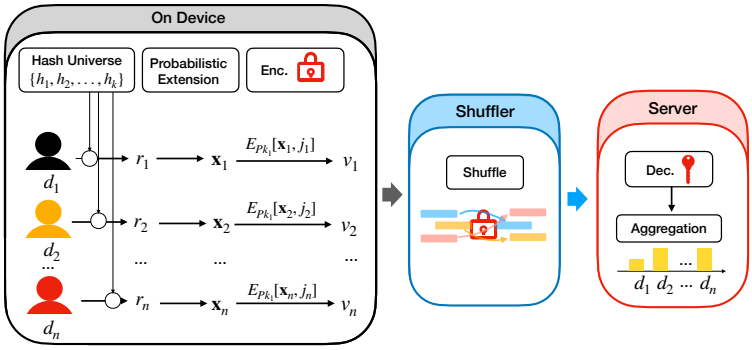


Fig. 1. Illustration of the optimal Count Mean Sketch with the ESA framework.

sent l messages to limit their contributions. This helps filter out heavy clients or counter data poisoning attacks, effectively limiting sensitivity for client-level DP analysis.

Phase 3: Server's frequency estimation.

The server first decrypts messages with the secret key and obtains:

$$\pi\left(\langle \mathbf{x}_1, j_1\rangle, \langle \mathbf{x}_2, j_2\rangle, \ldots, \langle \mathbf{x}_n, j_n\rangle\right).$$

Then, the server-side algorithm constructs a sketch matrix, in which the rows are indexed by hash functions, and each row j is the sum of the privatized vector of clients who selected the hash function h_j. We present this step in Algorithm 2.

To estimate the frequency of a message d, the server calculates the all hashed value of d: $\{h_j[d]\}_{j=1}^k$, and then aggregates the total count from the sketch matrix $\mathcal{M}$:

$$C(d) = \sum_{j=1}^{k} \mathcal{M}_j[h_j[d]].$$

An unbiased estimator for estimating the numbers of d occurring is

$$\hat{f}(d) = \frac{C(d) - \frac{pn}{m} - qn\left(1 - \frac{1}{m}\right)}{(p-q)\left(1 - \frac{1}{m}\right)}, \tag{4}$$

where q is given in (3).

In contrast to a fixed probabilities p, q in Apple's CMS, our GCMS allows for flexibility in designing p and s, thus enables optimal design tailored to specific item frequency regimes. We will discuss this in detail in Section IV-C.

Remark 1. *The communication cost, measured by the size of plaintext, is $\mathcal{O}(s \log m)$: there are s values to transfer, and each value requires $\log m$ bits. Comparably, We note that the communication cost for Apple's CMS is $\mathcal{O}(m)$ for each client. The computation cost at the server for aggregation is $\mathcal{O}(ns)$ for our GCMS, implying that the matrix $\mathcal{M}$ needs to be updated for n times, each contains s operations. In contrast, Apple's CMS requires a computation cost of $\mathcal{O}(nm)$, as each bit contained in the message needs to be de-biased and then aggregated.*

B. Theoretical Analysis of GCMS

In this section, we provide theoretical analysis of the privacy and utility of our GCMS. We first provide the LDP guarantee for each client's privatized vector.

Theorem 2 (Privacy of GCMS)**.** *The privatized vector $\mathbf{x}$ in GCMS is ϵ-locally differentially private, with*

$$\epsilon \leq \left| \log\left(\frac{p(m-s)}{(1-p)s}\right) \right|. \tag{5}$$

Since the hashing domain m is a constant, the privacy loss ϵ mainly depends on two parameters: p and s. For a large p and small s, the mechanism tends to release only the true hash value, resulting in a weak guarantee. Conversely, when p is small (close to 0.5) and s is large (close to $m/2$), the inclusion of the true hash value alongside other potential values becomes random, approximating a probability of 0.5. This randomness ensures that the true hash value cannot be distinguished, thus achieving a strong privacy guarantee.

We note that shuffling process can further amplify the above LDP guarantee by Theorem 1.

Next, we show the unbiased property of our estimator and give a general formulation for the MSE (variance) of the estimator in the following theorem.

Theorem 3 (Utility of GCMS)**.** *The estimator given in* (4) *is an unbiased estimator of $f(d)$:*

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

The variance of this estimator is

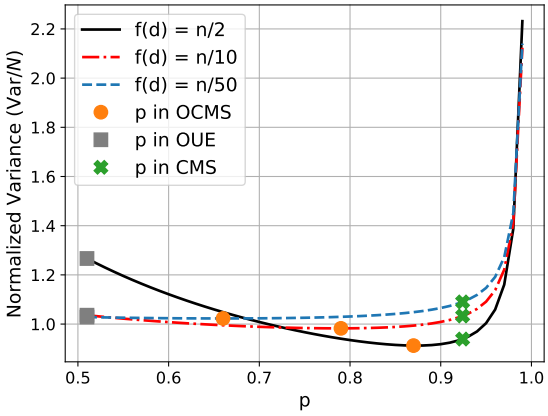
$$Var[\hat{f}(d)] = \frac{Var[C(d)]}{(p-q)^2(1-1/m)^2},$$

where

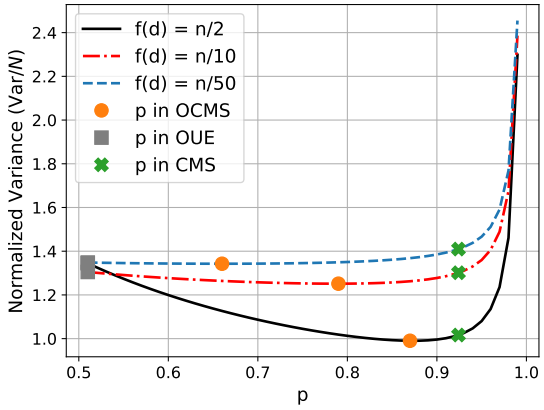
$$Var[C(d)] = f(d)\left(p - \frac{p^2}{k}\right) + (n - f(d))\left(\alpha_1 - \frac{1}{k}\alpha_1^2 - \alpha_2\right) + \alpha_2\left(\sum_{d' \neq d} f(d')^2\right),$$

and

$$\alpha_1 = \frac{p + (m-1)q}{m}, \ \alpha_2 = \frac{(p-q)^2}{km}\left(1 - \frac{1}{m}\right).$$



(a) $n = 100000$, $m = 1000$, $k = 500$.



(b) $n = 100000$, $m = 1500$, $k = 50$.

Fig. 2. Choices of p in OUE-LDP, Apple's CMS, and our OCMS. Optimal p depends on the ratio of $n/f(d)$ and k, and is independent of m.

Our variance formulation is general, and captures the original CMS and pure LDP protocols as special cases. The original CMS is with parameters $p = \frac{e^{\epsilon/2}}{e^{\epsilon/2}+1}$, $q = \frac{1}{e^{\epsilon/2}+1}$, and k hash functions with a hashing domain of $[\mathbf{m}]$. By instantiating these parameters within our framework, we can get the same utility guarantee as Theorem 4.2 in [3], with a correction to an error in the variance calculation on page 21. For a pure LDP protocol, setting $k = 1$ and $m = \infty$ within our framework yields the same utility guarantee as in [5].

Corollary 1 (Theorem 2 in [5])**.** *The variance of the pure LDP protocol is*

$$Var[\hat{f}(d)] = \frac{nq(1-q)}{(p-q)^2} + \frac{f(d)(1-p-q)}{p-q}. \tag{6}$$

Among all unbiased estimators, the one with the least variance is desired. Our general formulation characterizes the dependence of the variance with all the parameters of the protocols, thus provides guidance to optimize the designed protocol.

C. Optimal CMS (OCMS) for Targeted Frequency Estimation

In Wang et al. [5], Optimal Unary Encoding (OUE) was proposed to optimize the overall variance. However, OUE only considers optimizing the variance term with respect to n, based on the argument that most values appear infrequently and that low estimation variances for these infrequent values help avoid false positives when identifying more frequent ones. However, when analysts have a specific target frequency regime they aim to track, especially when $f(d)$ is at the same order as n, optimizing the entire variance including the term with respect to $f(d)$ can achieve more accurate estimation of these specific frequencies.

In many applications, there can be some specific frequency region that is of particular interest. For example, for online shopping, the data engineers might be interested in the frequency of the most popular items, to find the top-k heavy hitter, and use them for recommendation. For emoji dashboard editing, the least popular emojis are of more interest, as to be substituted in the next version. Another example is the popularity tracking of a news or video link, where the frequency could vary from time to time.

From the formulation of the utility of the algorithm, a natural question is, can we optimize the algorithm parameters p and q, to achieve the minimized variance, while making the algorithm achieve ϵ-LDP. We note that the total client number n is usually known as a priori; The variance monotonically decreases with the message size m and the number of hash functions k. On the other hand, large m increases the communication cost and server storage. Large k also enlarges the server storage and the complexity of hash function design. Therefore, the selection of m, k should be the maximum that allowed by the specific implementation requirement.

We also note that when ϵ is given, q can be expressed as a function of p and m: From (3) and (5).

$$q \geq \frac{m - e^{\epsilon} + e^{\epsilon}p - p}{(m-1)\left(\frac{e^{\epsilon}}{p} - e^{\epsilon} + 1\right)}. \tag{7}$$

Therefore, when n, m, k ϵ are considered as constant variables. The variance can be expressed as a function that determined by $f(d)$ and p. Then for any $f(d)$, the optimal mechanism parameter p can be derived from the optimization problem in the following lemma:

Lemma 1. *For the Optimal CMS mechanism, for a given ϵ, and for a targeted frequency $f(d)$, the optimal perturbation parameter p can be derived from the following optimization problem:*

$$p = argmin \left\{ \frac{kw - p + \lambda(w-1)(kw - p - wp)}{k(1-p)^2 p} \right\},$$

where

$$w \triangleq e^{\epsilon}(1-p) + p, \text{ and } \lambda = f(d)/n.$$

Remark 2. *Under a given privacy budget ϵ, the optimal selection of p depends on the ratio of $f(d)/n$ and k, and does not depend on m.*

Remark 3. *The optimal parameters p, q derived from Algorithm 4 and Eq. (7) can be adapted to enhance the "pure" protocols in [5], by setting $k = 1$ and $m = \infty$.*

Algorithm 4 Optimal CMS parameters

Input: ϵ: privacy parameter, λ: ratio of $f(d)/n$, k: size of the hash universe.

Output: Optimal perturbation parameter p, s.

Create function $f(p)$ that returns:

$$\frac{kw - p + \lambda(w-1)(kw - p - wp)}{k(1-p)^2 p};$$

Initialize $p_l = 0.5$ and $p_r = 1$;
while $p_l \leq p_r$ **do**
  $p_0 = p_l + \frac{p_r - p_l}{3}$;
  $p_1 = p_r - \frac{p_r - p_l}{3}$;
  if $f(p_0) \leq f(p_1)$ **then**
    $p_r = p_1$;
  else
    $p_l = p_0$;
  end if
end while
$p \leftarrow (p_l + p_r)/2$, $s \leftarrow \lceil \frac{m}{1+(1/p-1)e^{\epsilon}} \rceil$;
return p, s

In Algorithm 4, we present an efficient algorithm to derive the optimal p under the set of parameters of $\{\lambda, \epsilon, k\}$.

It is important to note that s is an integer, and the optimal s derived by Algorithm 4 involves a ceiling operation, which may result in a slightly smaller ϵ than the targeted one, yielding a stronger privacy guarantee.

We note that under the CMS or similar structure, such as unary encoding, RAPPOR, etc. the total privacy budget ϵ is allocated to p and q, and the proportion is approximately the following:

$$p \sim \frac{e^{\epsilon_p}}{e^{\epsilon_p} + 1}, \text{ and } q \sim \frac{1}{e^{(\epsilon - \epsilon_p)} + 1}.$$

The operational meaning of p is the probability of including the true item in the output, and q represents the probability to include a false item to provide confusion in the result. Therefore, a mechanism with a large probability of p and a small probability of q leads to accurate output and enhanced utility. The extent of the increase in p and the decrease in q is restricted by the privacy budget ϵ.

In terms of the allocation of the total budget, different mechanisms have different merits. In general, there are two different allocation principles. 1. Evenly divide ϵ to $\epsilon/2$ for p and $\epsilon/2$ for q. Therefore: $p = \frac{e^{\epsilon/2}}{e^{\epsilon/2}+1}$, $q = \frac{1}{e^{\epsilon/2}+1}$. Google's RAPPOR and Apple's CMS, etc., fall into this category. 2. grant all ϵ to q, and $\epsilon_p = 0$ to p, therefore, $p = 1/2$, $q = \frac{1}{e^{\epsilon}+1}$. The intuition behind this selection is that the true item is only included once with p, and all other items all have the probability of q being included. Therefore, with a limited budget, the mechanism should prioritize minimizing q. The Optimal Unary Encoding (OUE) falls into this category.

However, we argue that none of these solutions is optimal for any targeted frequency under the general CMS framework. In Fig. 2, we show two different cases regarding the values of m and k, for each case, we plot normalized variances derived in Theorem 3 with three different values of λ. Specifically, $\lambda = 1/2$, $\lambda = 1/10$, $\lambda = 1/50$, corresponding to three levels of popularity for items. Then we fix $\epsilon = 5$ and vary the value of p from 0.5 to 1 and showcase the corresponding variance. We highlight the optimal value of p corresponding to each case, along with the selection of p in CMS alike mechanisms and OUE alike mechanisms.

V. Privacy-preserving Data Collection with Unknown Domain

In the previous section, we introduced the GCMS algorithm for known item frequency estimation. However, this algorithm only functions effectively when the server knows the name of the item being queried.

When collecting unknown data strings, the server wants to recover and reconstruct as many data strings as possible as long as the privacy guarantee is not violated. Collecting data with an unknown domain in an LDP manner is intrinsically challenging due to:

- Encoding algorithms for preprocessing (e.g., UE or hashing) require agreement on $\mathcal{D}$ between the server and the clients.
- Segmenting the message increases the privacy loss to $k\epsilon$, where k denotes the number of pieces.
- Reconstructing the original input from segments incurs high computational costs, growing exponentially with the number of segments.

Our unknown-domain collection protocol introduces a distinct trade-off compared to existing methods. Prior approaches achieve local differential privacy with the standard client-server model but suffer from limited performance, which makes them impractical for real-world applications. In contrast, our protocol provides better performance; however, it relies upon stronger assumptions, which require a semi-trusted authority, and only ensures a quasi-local differential privacy guarantee. Consequently, we advise the users to choose the protocol most suitable for their specific use-case requirements, as no single protocol uniformly outperforms the others.

The key to avoid segmenting the original messages and the computational cost for the reconstruction is by encryption. Therefore, we take a different approach by using central DP techniques and leveraging cryptographic tools and the ESA framework to achieve local-like privacy guarantees.

A. The Protocol

Our protocol uses the stability-based histogram technique [8] for collecting new data. To safeguard against direct data collection and tracing by the server, we integrate the ESA framework. Beyond the ESA framework, we introduce an auxiliary server to construct the histogram, akin to a central curator under the central DP model, but with a crucial distinction: it does not access the original data messages. Instead, the auxiliary server only receives an encrypted version of the original message d by the server's public key, $E_{pk1}[d]$, along with its hashed value $H[d]$. This ensures the auxiliary

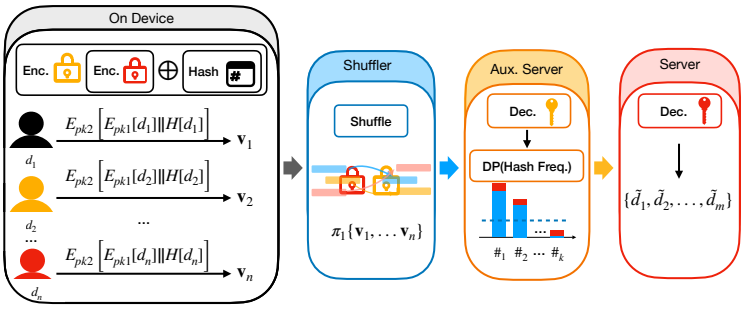


Fig. 3. Illustration of the framework of privacy-preserving data collection with unknown domain.

server gains no knowledge of the original messages but the hash, which is irreversible. To avoid common attacks against hash functions such as the brute-force attack, we replace the conventional hash with a keyed-hash function. For notational convenience, this keyed variant is denoted by $H[d]$ as well throughout the remainder of the paper. To construct the histogram the auxiliary server can count the number of message d occurring by counting the hashed value $H[d]$, and each bin is represented by a sampled $E_{pk1}[d]$ (two encrypted d with the same $pk1$ can different), which can only be decrypted by the server. To add additional layer of security protection, messages passing through the shuffler between the auxiliary server and clients are further encrypted using the auxiliary server's public key: $E_{pk2}[E_{pk1}[d]||H[d]]$. The overall framework contains four phases and is shown in Fig. 3. We next detail every step below.

Before the data collection, the server and the aux. server send their public keys, $pk1$ and $pk2$ to each client. The server and the aux. server agree on a set of privacy parameters (ϵ, δ) for the DP guarantee of the release.

Algorithm 5 On-device algorithm for unknown data string collection

Input: H: unique hash function, $pk1$: server's public key, $pk2$: aux. server's public key, d: raw data.

Output: Encrypted message v

Calculate the hashed value $H[d]$;
Encrypt d with $pk1 \leftarrow E_{pk1}[d]$;
Encrypt $H[d]$ and $E_{pk1}[d]$ with $pk2$:

$$v = E_{pk2}[E_{pk1}[d]||H[d]];$$

return v

Phase 1: On-device processing. In this phase, each client secures his data through the following process.

Data encryption with Server's public key: The private data is encrypted with the server's public key, the cyphertext is denoted as $E_{pk1}[d]$.

Data hashing: The private data is then hashed by an hash function, denoted as H (unique across clients). The hash function is identical for all clients. The hashed result is denoted as $H[d]$.

Algorithm 6 Aux. Server's DP protection

Input: $\pi\{v_1, .., v_n\}$: shuffled encrypted messages, T: Threshold for DP, b: scale of Laplacian noise.

Output: Encrypted message set.

Initiate hash map F, release set S;
Decrypt each message in $\pi\{v_1, .., v_n\}$ and obtain:
Arr = $\{E_{pk1}[d_1]||H[d_1], ..., E_{pk1}[d_n]||H[d_n]\}$;
for $E_{pk1}[d], H[d]$ in Arr **do**
  if $H[d]$ in F **then**
    $F[H[d]]$.append($E_{pk1}[d]$);
  else
    $F[H[d]]$ = $[E_{pk1}[d]]$;
  end if
end for
for key in F **do**
  if len(F[key]) + **Lap**(b) $\geq T$ **then**
    Randomly select $E_{pk1}[d]$ from F[key];
    Add $E_{pk1}[d]$ to S;
  end if
end for
return S

Encryption with the aux. server's public key The encrypted data and the hashed result are then encrypted with the aux. Server's public key. The encrypted message is denoted as $v = E_{pk2}[E_{pk1}[d]||H[d]]$ The on-device process is described in Algorithm 5. Then the encrypted message is passed on to the shuffler through an end to end encrypted channel.

Phase 2: Shuffler's anonymization and shuffling This phase is standard, and identical to the process described in Phase 2 of GCMS, we omit the details for simplicity.

Phase 3: Aux. server's DP protection In this phase, the aux. server provides DP protection for releasing the item names. This is achieved via the following process, and is detailed in Algorithm 6.

Decrypt message: The first step is to decrypt the messages received from the shuffler with the secret key. Then the aux. server observes $\{E_{pk1}[d_1]||H[d_1], ..., E_{pk1}[d_n]||H[d_n]\}$.

Hash frequency calculation: Since each client uses the same hash function, the hashed results from different clients with identical item must be identical. The aux. server calculates the

frequency of each hashed result and attaches the corresponding encrypted data to it.

Add DP noise: The aux. server adds Laplacian noise with scale b to frequency of each hashed result. For those hashed results with noisy frequency larger than a threshold T, randomly sample from its corresponding encrypted data and release to the server.

Phase 4: Server decrypts messages Finally, the server decrypts messages received from the aux. server and obtains the plaintext that contains the item names.

B. Privacy Analysis

The privacy is immediate from the stability-based histogram [8]. The relationship between the privacy parameters, the threshold T and the scale of the Laplacian mechanism is presented in the following Theorem.

Theorem 4 (Privacy of Algorithm 6)**.** *The released encrypted item set S is (ϵ, δ)-differentially private with*

$$\epsilon = \max\left\{\frac{1}{b}, \log\left(1 + \frac{1}{2e^{(T-1)/b} - 1}\right)\right\}, \tag{8}$$

and

$$\delta = \frac{1}{2}\exp(\epsilon(1-T)). \tag{9}$$

The steps for proving Theorem 4 follow directly from [8].

VI. Security Analysis

When each party in the framework is honest but curious, the security and privacy of each client's data are inherently guaranteed by LDP and encryption. Therefore we put more focus on potential risks if we extend our analysis to the malicious setting. Typically, the adversary aims to accomplish several objectives: (1) extract additional information about the honest clients and their messages, (2) downgrade the utility of the LDP results, and (3) disrupt the protocol, causing parties to abort. In a synchronized network setting, if some malicious parties decide to halt the protocol, it can be easily detected and the honest parties can simply exclude those parties from the next execution. So we mainly focus on the first two goals.

1) Analysis for GCMS/OCMS: In the context of our frequency estimation framework, three types of entities exist: clients, the shuffler, and the server. From an attacker's perspective, we assume all these parties could be malicious, what's more, different types of parties could collude.

Malicious Clients: For malicious clients, the worst attack they can do is similar to the data poisoning attack [27]: the adversary can send garbage messages as LDP reports, or register a huge amount of fake clients and let them send garbage messages as LDP reports. While truncating clients' maximum contribution can counter the former, the latter type, is considered outside the scope of this paper, as it cannot be mitigated by designing a better LDP protocol. Typically, strategies such as authentication or access control can be employed to counteract such Sybil attack.

Malicious Shuffler: Since all the client messages are encrypted with the server's public key, even a malicious shuffler cannot break the confidentiality of the messages. However, the malicious shuffler can still deviate from the protocol. For instance, the shuffler could launch a data poisoning attack by injecting false information as LDP reports or dropping LDP reports from honest clients. Since the shuffler represents a single point of failure, these attacks are feasible. Consequently, the shuffler must be enforced as a semi-honest party, and there exist several mechanisms to achieve it. For example, multiple servers can collaboratively perform shuffling using secure multi-party computation. This approach ensures that the shuffling process is executed correctly without revealing any additional information, thereby eliminating the potential for malicious behaviors. Alternatively, we can also put the shuffler functionality into trust execution environments (TEE) [28], which provide verifiable security. If all these requirement cannot be met in deployment, we note that GCMS can also operate in a pure local differential privacy (LDP) setting without a shuffler, just as Apple's CMS does. In this standalone mode, the privacy guarantee is fully provided by the local randomization mechanism. The use of a shuffler is optional and offers privacy amplification via shuffling. This is consistent with how shufflers are employed in other LDP deployments (e.g., Apple, Google).

Malicious Server: The malicious server has no motivation to downgrade the utility as it benefits from better LDP outcomes. Therefore, we only check if the malicious server can infer more information about the clients. However, since all messages accessed by the server are protected by LDP, it is assured that the server cannot learn additional information beyond what is allowed by the LDP protocol.

Collusion between parties: When the shuffler colludes with the server, the worst-case scenario is that the shuffler fails to shuffle the messages, thereby eliminating the privacy amplification effect. However, the results of the protocol are still protected by LDP, although at a reduced level of privacy. In cases where the server colludes with a group of malicious clients, the server still gets no extra information of the rest of the clients due to the LDP guarantee. When the shuffler colludes with malicious clients, the worst-case situation is that the shuffler consistently accepts inputs from the malicious clients while refusing to serve the honest ones. This issue cannot be addressed through LDP design alone. It can be mitigated by implementing measures such as enforced scheduling or adopting a decentralized shuffling solution, such as MPC.

2) Analysis for the aux. server: Our unknown item discovery framework introduces another entity: the aux. server. It is important to emphasize that this party must be enforced as honest, since a malicious aux. server could launch several powerful attacks. For instance, the aux. server can compute the hash of the specific value and compare it with the hash in each client message. In this way, the aux. server can check whether the client message is a specific value. What's more, the aux. server can also purposely decrease the frequency of a specific value. To do so, the aux. server simply computes the hash of the value and drops all messages with the same hash.

These attacks are unavoidable even when the message space is large, and can only be mitigated through techniques such as keyed hash functions, where the key is negotiated directly between the clients and the server. Finally, the aux. server can also obtain a list of frequencies, but without knowing which frequency corresponds to which item. This still enables targeted attacks, such as deleting all records associated with the most frequent item. Notably, such attacks are unavoidable even when keyed hash functions are used.

However, if we restrict the aux. server to be an honest-but-curious party—meaning it follows the protocol correctly but may attempt to infer additional information—our protocol maintains sufficient privacy guarantees. All plaintext messages are encrypted using the final server's public key, preventing the aux. server from learning their values. This also applies to hashed values when keyed hash functions are used, as the aux. server cannot compute corresponding hashes without the key. Moreover, all messages are shuffled before reaching the aux. server, preventing it from linking messages to their senders.

To conclude, the malicious aux. server could significantly change the result of the protocol, therefore, we suggest deploying the aux. server with a trusted authority, or implementing the aux. server in trust execution environments [28]. The requirement of a semi-trusted authority is what enables the performance improvements relative to existing methods; therefore, our protocol may not be suitable for scenarios in which such an authority cannot be assumed.

VII. Key Results Visualization and Experiments

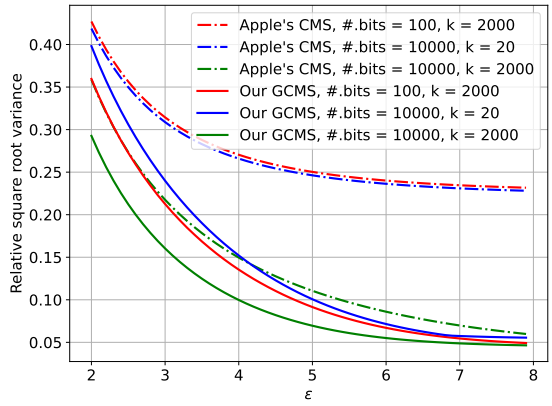
This section visualizes the key results and experimental evaluation for our GCMS, OCMS and related works. We start by visualizing the accuracy in terms of variance, demonstrating that our approach can achieve lower variance for items with specific frequencies. Additionally, we identify the optimal choices of p under various parameter settings. Finally, we conduct tests of our approach and related works using real-world databases, providing a comprehensive comparison of their performances.

A. Utility of GCMS

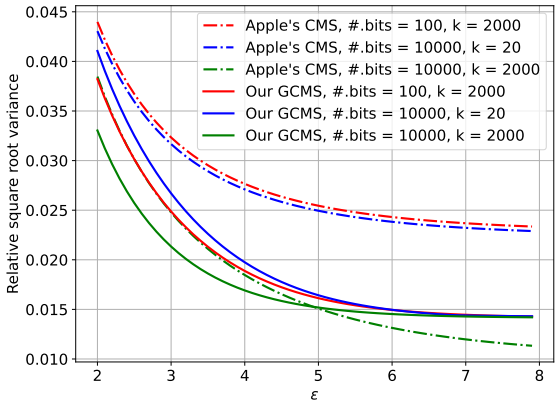
We use variance as the metric to measure accuracy, which can be directly translated to the mean square error of the estimation. To provide better visualization, we slightly modify original variance to relative square root variance, which is in the same order as the absolute error or relative error, and is defined as:

$$\sqrt{\mathrm{Var}(\hat{f}(d))}/f(d).$$

This modification does not affect the optimality of the mechanism. There are multiple factors jointly determining the variance, such as k (the number of hash function), m domain (hash domain size), $f(d)$ (true frequency of the queried item), and n (number of LDP reports). Figure 4 shows the accuracy of our approach and Apple's CMS under different parameter settings. We fix the number of LDP reports to be $100,000$, and the communication costs of different mechanisms at the same level. Then, we focus on two target query items with true frequencies $f(d)$ of 1000 and 10000. The results demonstrate that our approach achieves lower variance when the privacy budget ϵ is not excessively large. Apple's CMS outperforms our approach only when the $f(d)$ of the queried item is considerably large and the privacy budget is extremely high, which rarely happens in real-world scenarios.



(a) $n = 100000$, $f(d) = 1000$.

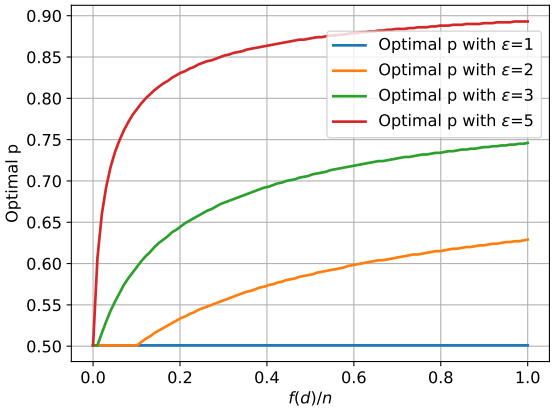


(b) $n = 100000$, $f(d) = 10000$.

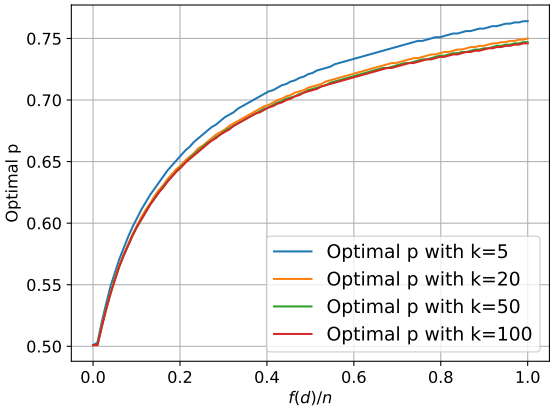
Fig. 4. Variance comparison between our approach and Apple's CMS under different epsilons. Lines with the same color indicate the same parameter setting. #.bits represents the bit length of the LDP reports, which is directly related to the hash function range m. k is the number of hash functions, n is the total number of LDP reports, and $f(d)$ is the true frequency of the queried item. The perturbation parameter p for our approach is 0.5.

B. Utility of OCMS

Lemma 1 suggests that the perturbation parameter p can be adjusted for a specific frequency $f(d)$ to minimize the variance for target frequency. We conduct experiments to compute optimal selection of p under various parameter settings, and the results are presented in Figure 5. The findings reveal an interesting observation: when ϵ is small, the optimal p remains 0.5 across a wide range of $f(d)$. This is consistent with the assertions in [13] that the optimal choice of p



(a) Optimal selection of p given different ϵ.



(b) Optimal selection of p given different k.

Fig. 5. Optimal selection of p in various parameter settings. $n = 10000, k = 200, m = 200$.

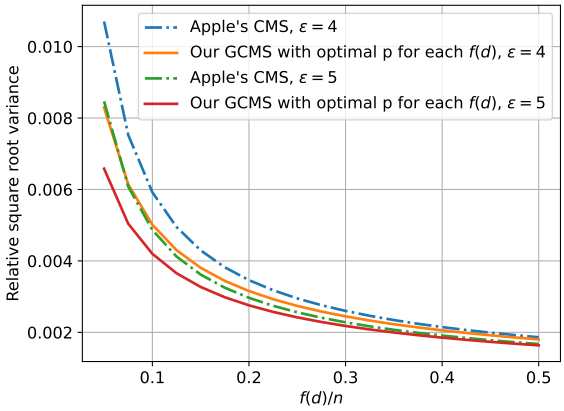


Fig. 6. Variance of our approach with optimized p for each target frequency $f(d)$. $n = 1000000, k = 65536, m = 1024$.

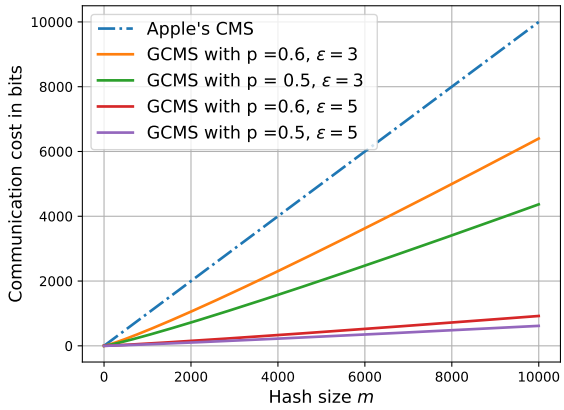


Fig. 7. Communication cost comparison between GCMS and Apple's CMS under fixed ϵs and ps.

is 0.5 when the privacy budget is small. However, as the privacy budget increases, the optimal p begins to diverge from 0.5, increasing with ϵ and $f(d)$ beyond a certain threshold. Therefore, our solution can also be viewed as generalization of the optimal unary encoding approach, offering improved performance when a larger ϵ is available. Figure 5(b) illustrates the relationship between the optimal p and the number of hash functions k. generally, a larger k corresponds to a higher optimal p. This trend indicates that our approach yields greater benefits when implemented with a more complex hashing mechanism.

With the optimal selection of p, our approach has more advantages when we are interested in querying items within a specific frequency range. By adjusting the parameter p, our approach can achieve better accuracy for a target frequency, resulting in lower variance for query items with similar frequencies. Figure 6 displays the best theoretical variance achievable for each target frequency, with the variance of Apple's CMS with a fixed p serving as the baseline. To ensure a fair comparison, we use the same parameter settings as in Apple's CMS experiments: $m = 1024$, $k = 65536$, $\epsilon = 4$. The results indicate that our approach significantly improves variance across all target frequencies.

C. Communication Costs

We compare the communication costs of GCMS against with that of Apple's CMS. As we described in Remark 1, the communication cost of the GCMS is $\mathcal{O}(s \log m)$, compared to $\mathcal{O}(m)$ for Apple's CMS. Note that while the communication cost for Apple's CMS only depends on the hash size m and is independent of the mechanism parameters, the communication cost of GCMS depends on s, which can be viewed as a function of ϵ and p.

In Fig. 7, we present the communication costs when varying p and ϵ. We fix the privacy parameter ϵ to be 3 and 5, respectively, and p to be 0.5 and 0.6, corresponding to the p in OUE-LDP and in OCMS for some targeted frequency. We then vary the hash size m from 0 to $10,000$ and obtain the corresponding communication costs. We can observe that our subset-selection feature improves the communication costs

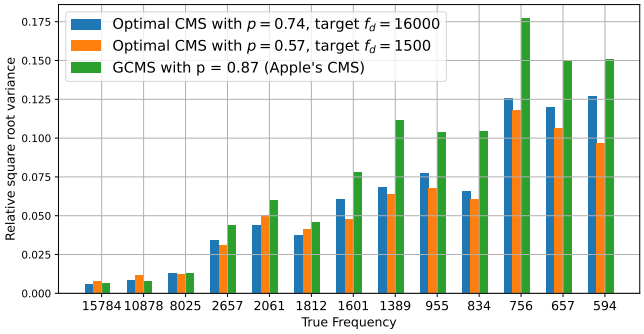


Fig. 8. Variance comparison with the "Adult Census Income" dataset with various p. Experiment setting: $n = 48000, m = 100, k = 100$. $\epsilon = 3.64$ for the $p = 0.74$ case, and $\epsilon = 3.75$ for the rest two cases.

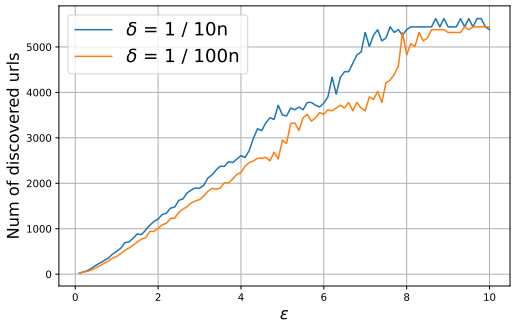


Fig. 9. Relationship between privacy budget and the amount of discovered url with our approach using Kaggle's URL dataset. Experiment setting: $n = 100000$. Each experiment is repeated 30 times, and the mean value is reported.

significantly. Additionally, a small p and large ϵ can further reduce the communication cost in GCMS.

D. Experiments with Real-world Datasets

We conduct experiments with the "Adult Census Income" dataset[29], which contains census information with $48,842$ records and 15 attributes. Our focus is to count the frequency of the "education" column, which contains 16 possible values with frequencies ranging from 80 to $16,000$. We use Python code to simulate the operations on both the client side and the server side, and ask the server to reconstruct the frequencies of all 16 possible values. We launch the code on a Mac Book Pro with M1 chips and 32GB RAM. We target the highest frequency value of $\frac{n}{3} = 16,000$ and apply Lemma 1 to determine an optimal p of 0.74. For mid-range frequency values, we pick $p = 0.57$ to achieve the best performance for items with a frequency around 1500. Additionally, we include $p = 0.87$ as the baseline because it corresponds to the parameter setting used in Apple's CMS, which is treated as a special case in our framework. The result is depicted in Figure 8. We observe that each specified case achieves lower variance at its target frequency, though the performance slightly decreases for non-target frequencies. Specifically, in the case of Apple's CMS, the resulting target frequency exceeds n due to the large p. Consequently, our approach demonstrates better performance almost the entire range of true frequencies.

E. Unknown Domain Data Collection

We conduct experiments for our unknown domain DP solution with Kaggle's URL dataset[1]. For convenience, we select the first $100,000$ data records and limited our analysis to the first 20 characters of each URL. This is because many URLs contain a client-ID-like string towards the end, making them unique and thus difficult to analyze effectively for frequency-based approaches. We observe that as the privacy budget ϵ decreases, the threshold T increases. Consequently, a smaller number of URLs is collected, reflecting a higher level of privacy protection at the expense of data availability. This trend is clearly illustrated in Figure 9. Additionally, as ϵ increases, the number of collected URLs tends to converge. The convergence number (approximately $5,500$ URLs) still significantly differs from the total number of distinct URLs (around $40,000$). This indicates that even at higher levels of ϵ, the method does not capture all unique URLs. The underlying reason is that, according to Theorem 4, T is approximately $1 - \log(2\delta)/\epsilon$, which is greater than 1 for most values of ϵ. Given that the majority of URLs in the dataset have a count of 1 and the Laplacian mechanism adds zero-mean noise, most of these URLs remain unrevealed. This feature ensures that user data with outliers is intractable.

It is worth noting that our protocol offers significant improvement in accuracy. We also conduct experiments on the Twitter dataset, following [7]. When compared to other existing LDP unknown domain data collection methods, PrivTrie is the only solution that obtains a practical level of accuracy (over 0.8 F-score) under $\epsilon = 2$. Under these same conditions, our unknown domain protocol achieves an F-score of 0.962, demonstrating a notable improvement in accuracy. However, we note that this is not a completely fair comparison, as all the methods mentioned are pure LDP protocols, whereas our approach is quasi-LDP, leveraging cryptographic techniques.

VIII. Conclusion

In this paper, we present a comprehensive framework for privacy-preserving data collection and frequency estimation, utilizing the Encryption-Shuffling-Analysis architecture. We introduce a Generalized Count Mean Sketch (GCMS) protocols that captures and optimizes various existing frequency estimation protocols. We provide its privacy and security analysis, as well as a general utility analysis that enables optimal parameter design, leading to the development of the Optimal Count Mean Sketch (OCMS) which minimizes variance for targeted frequency regimes. Additionally, we propose a protocol for data collection with unknown domains, addressing scenarios where the data domain is not predefined. Our approach achieves accuracy comparable to the central differential privacy (DP) model while providing local-like privacy guarantees and substantially lowering computational

[1]Link: https://www.kaggle.com/datasets/teseract/urldataset?resource=download

costs. We also visualize our theoretical analysis and provide extensive experimental results demonstrating the effectiveness of these protocols in terms of utility-privacy trade-offs and communication cost.

REFERENCES

[1] 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.

[2] G. Fanti, V. Pihur, and Úlfar Erlingsson, "Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries," Proceedings on Privacy Enhancing Technologies (PoPETS), vol. issue 3, 2016, 2016.

[3] A. Differential Privacy Team, "Learning with privacy at scale," https: //machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html, 2017.

[4] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in Proceedings of the 31st International Conference on Neural Information Processing Systems, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 3574–3583.

[5] 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.

[6] Úlfar Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in Proceedings of the 21st ACM CCS, 2014.

[7] N. Wang, X. Xiao, Y. Yang, T. D. Hoang, H. Shin, J. Shin, and G. Yu, "Privtrie: Effective frequent term discovery under local differential privacy," in 2018 IEEE 34th International Conference on Data Engineering (ICDE), 2018, pp. 821–832.

[8] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas, "Releasing search queries and clicks privately," in Proceedings of the 18th international conference on World wide web, 2009, pp. 171–180.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in Proceedings of the 3rd Conference on Theory of Cryptography, ser. TCC '06, 2006, pp. 265–284.

[10] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. D. Smith, "What can we learn privately?" 2008 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 531–540, 2008.

[11] U. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, "Amplification by shuffling: from local to central differential privacy via anonymity," in Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, ser. SODA '19. USA: Society for Industrial and Applied Mathematics, 2019, p. 2468–2479.

[12] V. Feldman, A. McMillan, and K. Talwar, "Stronger privacy amplification by shuffling for rényi and approximate differential privacy," in Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). SIAM, 2023, pp. 4966–4981.

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

[14] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," in Advances in Neural Information Processing Systems 27. Curran Associates, Inc., 2014, pp. 2879–2887.

[15] 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.

[16] S. Wang, L. Huang, P. Wang, Y. Nie, H. Xu, W. Yang, X.-Y. Li, and C. Qiao, "Mutual information optimally local private discrete distribution estimation," arXiv preprint arXiv:1607.08025, 2016.

[17] P. Kairouz, K. Bonawitz, and D. Ramage, "Discrete distribution estimation under local privacy," in Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ser. ICML'16. JMLR.org, 2016, p. 2436–2444.

[18] V. Feldman, J. Nelson, H. Nguyen, and K. Talwar, "Private frequency estimation via projective geometry," in International Conference on Machine Learning. PMLR, 2022, pp. 6418–6433.

[19] S. Aydin and S. Yildirim, "Adaptive online bayesian estimation of frequency distributions with local differential privacy," arXiv preprint arXiv:2405.07020, 2024.

[20] H. Fang, L. Chen, Y. Liu, and Y. Gao, "Locally differentially private frequency estimation based on convolution framework," in 2023 IEEE Symposium on Security and Privacy (SP), 2023, pp. 2208–2222.

[21] X. Chen, C. Wang, J. Cui, Q. Yang, T. Hu, and C. Jiang, "Incorporating prior knowledge in local differentially private data collection for frequency estimation," IEEE Transactions on Big Data, vol. 9, no. 2, pp. 499–511, 2023.

[22] Y. Zheng, S. R. Seeam, Y. Hu, R. Zhang, and Y. Zhang, "Locally differentially private frequency estimation via joint randomized response," arXiv preprint arXiv:2505.10349, 2025.

[23] X. Feng and C. Zhang, "Mpldp: Multi-level personalized local differential privacy method," IEEE Access, 2024.

[24] V. Vijayachandran and S. R, "Local differential privacy for data security in key value pair data," Journal of Computational Methods in Sciences and Engineering, vol. 24, no. 3, pp. 1955–1970, 2024.

[25] M. Zhang, X. Liu, and L. Yin, "Sketches-based join size estimation under local differential privacy," in 2024 IEEE 40th International Conference on Data Engineering (ICDE). IEEE, 2024, pp. 1726–1738.

[26] M. Pan, "Count-mean sketch as an optimized framework for frequency estimation with local differential privacy," arXiv preprint arXiv:2406.03785, 2024.

[27] Y. Wu, X. Cao, J. Jia, and N. Z. Gong, "Poisoning attacks to local differential privacy protocols for Key-Value data," in 31st USENIX Security Symposium (USENIX Security 22). Boston, MA: USENIX Association, Aug. 2022, pp. 519–536.

[28] J. Chen, Y. Tang, and H. Zhou, "Strongly secure and efficient data shuffle on hardware enclaves," in Proceedings of the 2nd Workshop on System Software for Trusted Execution, 2017, pp. 1–6.

[29] B. Becker and R. Kohavi, "Adult," UCI Machine Learning Repository, 1996, DOI: https://doi.org/10.24432/C5XW20.

APPENDIX A
PROOF OF THEOREM 2

Denote r and r' as the hashed results of two different input d and d'. Note that r and r' can be resulted from different hash functions. We next examine the likelihood ratio of r and r' resulting the same $\mathbf{x}$.

Case 1: $r \in \mathbf{x}$ and $r' \in \mathbf{x}$:

$$\frac{\Pr(X=\mathbf{x}|D=d)}{\Pr(X=\mathbf{x}|D=d')} = \frac{\Pr(X=\mathbf{x}|R=r)}{\Pr(X=\mathbf{x}|R=r')} = \frac{p/\binom{m-1}{s-1}}{p/\binom{m-1}{s-1}} = 1.$$

Case 2: $r \notin \mathbf{x}$ and $r' \notin \mathbf{x}$:

$$\frac{\Pr(X=\mathbf{x}|D=d)}{\Pr(X=\mathbf{x}|D=d')} = \frac{\Pr(X=\mathbf{x}|R=r)}{\Pr(X=\mathbf{x}|R=r')} = \frac{(1-p)/\binom{m-1}{s}}{(1-p)/\binom{m-1}{s}} = 1.$$

Case 3: $r \in \mathbf{x}$ and $r' \notin \mathbf{x}$:

$$\begin{aligned}\frac{\Pr(X=\mathbf{x}|D=d)}{\Pr(X=\mathbf{x}|D=d')} =& \frac{\Pr(X=\mathbf{x}|R=r)}{\Pr(X=\mathbf{x}|R=r')}\\ =& \frac{p\binom{m-1}{s-1}}{(1-p)\binom{m-1}{s}}\\ =& \frac{p}{1-p}\frac{\binom{m-1}{s}}{\binom{m-1}{s-1}}\\ =& \frac{p}{1-p}\frac{(m-1)!/s!(m-s-1)!}{(m-1)!/(s-1)!(m-s)!}\\ =& \frac{p}{1-p}\frac{(s-1)!(m-s)!}{s!(m-s-1)!}\\ =& \frac{p}{1-p}\frac{m-s}{s}.\end{aligned}$$

Case 4: $r \notin \mathbf{x}$ and $r' \in \mathbf{x}$:

Similar to case 3.

$$\begin{aligned}\frac{\Pr(X=\mathbf{x}|D=d)}{\Pr(X=\mathbf{x}|D=d')} =& \frac{\Pr(X=\mathbf{x}|R=r)}{\Pr(X=\mathbf{x}|R=r')}\\ =& \frac{(1-p)\binom{m-1}{s}}{p\binom{m-1}{s-1}}\\ =& \frac{1-p}{p}\frac{s}{m-s}.\end{aligned}$$

Therefore, when $p \in [0.5, 1]$ and $s \in [m/2, m]$

$$\log\left\{\frac{\Pr(X=\mathbf{x}|D=d)}{\Pr(X=\mathbf{x}|D=d')}\right\} \le \log\left\{\frac{p}{1-p}\frac{m-s}{s}\right\}.$$

This completes the proof of Theorem 1.

APPENDIX B
PROOF OF THEOREM 3

For simplicity, let $M_j[h_j[d]] = \sum_{i=1}^{n} Z_j^{(i)}(d)$. Here,

$$Z_j^{(i)}(d) = \mathbb{1}_{\{J_i=j\}}\Big\{B(\mathbb{1}_{\{d^{(i)}=d\}} + \mathbb{1}_{\{d^{(i)}\neq d\}}\mathbb{1}_{\{h_j(d^{(i)})=h_j(d)\}}) + Y\mathbb{1}_{\{d^{(i)}\neq d\}}\mathbb{1}_{\{h_j(d^{(i)})\neq h_j(d)\}}\Big\},$$

where B and Y are Bernoulli random variables representing the randomized response: $B = 1$ with probability p and $B = 0$ with probability $1-p$; $Y = 1$ with probability q and $Y = 0$ with probability $1-q$.

Before we prove Theorem 3, let us first prove several helping lemmas.

Lemma 2.

$$E[Z_j^{(i)}(d)] = \frac{p}{k}\mathbb{1}_{\{d^{(i)}=d\}} + \left(\frac{p}{mk} + (1-\frac{1}{m})\frac{q}{k}\right)\mathbb{1}_{\{d^{(i)}\neq d\}}.$$

Proof. By substituting into $\mathbb{1}_{\{h_j(d^{(i)})=h_j(d)\}}) = \frac{1}{m}$, we can observe that the lemma holds. □

Lemma 3.

$$E\left[\left(Z_j^{(i)}(d)\right)^2\right] = E[Z_j^{(i)}(d)].$$

Proof.

$$\begin{aligned}&E\left[\left(Z_j^{(i)}(d)\right)^2\right]\\ =&\frac{1}{k}E[B^2\mathbb{1}_{\{d^{(i)}=d\}} + \mathbb{1}_{\{d^{(i)}\neq d\}}(B\mathbb{1}_{\{h_j(d^{(i)})=h_j(d)\}}\\ &+ \mathbb{1}_{\{h_j(d^{(i)})\neq h_j(d)\}})Y)^2]\\ =&\frac{p}{k}\mathbb{1}_{\{d^{(i)}=d\}} + \left(\frac{p}{mk} + (1-\frac{1}{m})\frac{q}{k}\right)\mathbb{1}_{\{d^{(i)}\neq d\}}\\ =&E[Z_j^{(i)}(d)].\end{aligned}$$

□

Lemma 4. *For any indices $i_1 \neq i_2$,*

1) $Cov(Z_{j_1}^{(i_1)}(d), Z_{j_2}^{(i_2)}(d)) = 0$ *when* $j_1 \neq j_2$;
2) $Cov(Z_j^{(i_1)}(d), Z_j^{(i_2)}(d)) = 0$ *when* $d^{(i_1)} = d$ *or* $d^{(i_2)} = d$ *or* $d^{(i_1)} \neq d^{(i_2)}$;
3) $Cov(Z_j^{(i_1)}(d), Z_j^{(i_2)}(d)) = \left(\frac{p-q}{k}\right)^2\left(\frac{1}{m} - \frac{1}{m^2}\right)$ *when* $d^{(i_1)} = d^{(i_2)} = d'$ *and* $d' \neq d$.

Proof. **Case 1:** When $j_1 \neq j_2$, since the hash functions h_{j_1} and h_{j_2} are chosen independently, we have $Cov(Z_{j_1}^{(i_1)}(d), Z_{j_2}^{(i_2)}(d)) = 0$.

Case 2: When $j_1 = j_2 = j$, the calculation of covariance depends on the expectation of their product $E[Z_{j_1}^{(i_1)}(d)Z_{j_2}^{(i_2)}(d)]$. Let B_1, B_2 (and Y_1, Y_2) denote the randomized response probabilities for i_1, i_2, respectively. We first consider when $d^{(i_1)} = d$, and we have

$$\begin{aligned}&E[Z_{j_1}^{(i_1)}(d)Z_{j_2}^{(i_2)}(d)]\\ =&\frac{1}{k^2}E[B_1(B_2(\mathbb{1}_{\{d^{(i_2)}=d\}} + \mathbb{1}_{\{d^{(i_2)}\neq d\}}\mathbb{1}_{\{h_j(d^{(i_2)})=h_j(d)\}})\\ &+ Y_2\mathbb{1}_{\{d^{(i_2)}\neq d\}}\mathbb{1}_{\{h_j(d^{(i_2)})\neq h_j(d)\}})]\\ =&\frac{p}{k^2}\left(p\mathbb{1}_{\{d^{(i_2)}=d\}} + \left(\frac{p}{m} + (1-\frac{1}{m})q\right)\mathbb{1}_{\{d^{(i_2)}\neq d\}}\right)\\ =&\frac{p}{k}E[Z_j^{(i_2)}(d)]\\ =&E[Z_j^{(i_1)}(d)]E[Z_j^{(i_2)}(d)].\end{aligned}$$

Therefore,

$$\begin{aligned}&Cov(Z_j^{(i_1)}(d), Z_j^{(i_2)}(d))\\ =&E[Z_{j_1}^{(i_1)}(d)Z_{j_2}^{(i_2)}(d)] - E[Z_j^{(i_1)}(d)]E[Z_j^{(i_2)}(d)]\\ =&0.\end{aligned}$$

Similarly, we also have $Cov(Z_j^{(i_1)}(d), Z_j^{(i_2)}(d)) = 0$ when $d^{(i_2)} = d$. Then it remains to consider the case when they both are not equal to d: $d^{(i_1)} \neq d$ and $d^{(i_2)} \neq d$. We further divide it into two cases.

Case 3: When $d^{(i_1)} \neq d^{(i_2)}$, all random variables $B_1, B_2, Y_1, Y_2, \mathbb{1}_{\{d^{(i_2)}\neq d\}}, \mathbb{1}_{\{h_j(d^{(i_1)})\neq h_j(d)\}}, \mathbb{1}_{\{h_j(d^{(i_2)})\neq h_j(d)\}}$ are independent, so we have $Cov(Z_j^{(i_1)}(d), Z_j^{(i_2)}(d)) = 0$.

Case 4: When $d^{(i_1)} = d^{(i_2)} = d'$ and $d' \neq d$, we have

$$\begin{aligned}
&E[Z_{j_1}^{(i_1)}(d) Z_{j_2}^{(i_2)}(d)] \\
=&\frac{1}{k^2} E[B_1 B_2 \mathbb{1}^2_{\{h_j(d')=h_j(d)\}}) + Y_1 Y_2 \mathbb{1}^2_{\{h_j(d')\neq h_j(d)\}}] \\
=&\frac{1}{k^2}\left(\frac{p^2}{m} + q^2(1-\frac{1}{m})\right),
\end{aligned}$$

and by Lemma 2,

$$E[Z_j^{(i_1)}(d)]E[Z_j^{(i_2)}(d)] = \frac{1}{k^2}\left(\frac{p}{m} + q(1-\frac{1}{m})\right)^2.$$

Then,

$$\begin{aligned}
&Cov(Z_j^{(i_1)}(d), Z_j^{(i_2)}(d)) \\
=&E[Z_{j_1}^{(i_1)}(d) Z_{j_2}^{(i_2)}(d)] - E[Z_j^{(i_1)}(d)]E[Z_j^{(i_2)}(d)] \\
=&\left(\frac{p-q}{k}\right)^2 \frac{1}{m}(1-\frac{1}{m}).
\end{aligned}$$

□

We are now ready to prove Theorem 3.

Proof. By Lemma 2, we have

$$\begin{aligned}
&E[C(d)] \\
=&\sum_{j=1}^{k}\sum_{i=1}^{n} E\left[Z_j^{(i)}(d)\right] \\
=&f(d)\left(1-\frac{1}{m}\right)(p-q) + \frac{pn}{m} + \left(1-\frac{1}{m}\right)nq.
\end{aligned}$$

Therefore,

$$E[\hat{f}(d)] = \frac{E[C(d)] - \frac{pn}{m} - qn\left(1-\frac{1}{m}\right)}{(p-q)\left(1-\frac{1}{m}\right)} = f(d).$$

We can decompose the variance as follows.

$$\begin{aligned}
&Var(C(d)) \\
=&Var\left(\sum_{j=1}^{k}\sum_{i=1}^{n} Z_j^{(i)}(d)\right) \\
=&\sum_{i=1}^{n} Var(\sum_{j=1}^{k} Z_j^{(i)}(d)) + \sum_{i_1\neq i_2} Cov(Z_{j_1}^{(i_1)}(d), Z_{j_2}^{(i_2)}(d)) \\
=&\sum_{i=1}^{n}\left(\sum_{j=1}^{k} Var(Z_j^{(i)}(d)) + \sum_{j_1\neq j_2} Cov(Z_{j_1}^{(i)}(d), Z_{j_2}^{(i)}(d))\right) \\
&+ \sum_{i_1\neq i_2}\sum_{j=1}^{k} Cov(Z_j^{(i_1)}(d), Z_j^{(i_2)}(d)), \qquad (10)
\end{aligned}$$

where (10) follows from Lemma 4(1).

Again, by Lemma 4 (1), we need only consider $\sum_{i=1}^{n}\sum_{j=1}^{k} Var(Z_j^{(i)}(d))$ for the first term in (10):

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{k} Var(Z_j^{(i)}(d)) \\
=&\sum_{i=1}^{n}\sum_{j=1}^{k} E\left[\left(Z_j^{(i)}(d)\right)^2\right] - E^2[Z_j^{(i)}(d)] \\
=&pf(d) + \left(\frac{p}{m} + (1-\frac{1}{m})q\right)(n-f(d)) \\
&- \frac{p^2}{k}f(d) + \frac{1}{k}\left(\frac{p}{m} + (1-\frac{1}{m})q\right)^2 (n-f(d)) \qquad (11)
\end{aligned}$$

Then we turn to consider the second term in (10). By Lemma 4 (2) and (3), we need only consider the indices where $d^{(i_1)} = d^{(i_2)} = d^*$ and $d^* \neq d$.

$$\begin{aligned}
&\sum_{i_1\neq i_2}\sum_{j=1}^{k} Cov(Z_j^{(i_1)}(d), Z_j^{(i_2)}(d)) \\
=&k\left(\frac{p-q}{k}\right)^2 \frac{1}{m}(1-\frac{1}{m}) \sum_{d^*\neq d}\sum_{i_1:d^{(i_1)}=d^*}\sum_{i_2\neq i_1} \mathbb{1}_{\{d^{(i_2)}=d^*\}} \\
=&k\left(\frac{p-q}{k}\right)^2 \frac{1}{m}(1-\frac{1}{m}) \sum_{d^*\neq d}\sum_{i_1:d^{(i)}=d^*} (f(d^*)-1) \\
=&k\left(\frac{p-q}{k}\right)^2 \frac{1}{m}(1-\frac{1}{m}) \sum_{d^*\neq d} (f(d^*)-1)f(d^*) \\
=&k\left(\frac{p-q}{k}\right)^2 \frac{1}{m}(1-\frac{1}{m}) \left(\sum_{d^*\neq d} f(d^*)^2 - (n-f(d))\right) \qquad (12)
\end{aligned}$$

Combining (11) and (12), we have

$$\begin{aligned}
&Var(C(d)) \\
=&f(d)(p-\frac{p^2}{k}) + (n-f(d))\left(\frac{p^2}{mk} + \frac{q^2}{k}(1-\frac{1}{m})\right) \\
&+ \frac{(p-q)^2}{km}(1-\frac{1}{m})\left(\sum_{d^*\neq d} f(d^*)^2\right).
\end{aligned}$$

By Proposition 1, we have the following relation between Var$(\hat{f}(d))$ and $Var(C(d))$:

$$\text{Var}(\hat{f}(d)) = \frac{Var(C(d))}{(p-q)^2(1-1/m)^2},$$

completing the proof.

□

Appendix C
Proof of Lemma 1

In this section, we prove the result in lemma 1, which shows the optimal p can be derived from the following optimization problem:

$$p = \text{argmin}\left\{\frac{kw - p + \lambda(w-1)(kw-p-wp)}{k(1-p)^2 p}\right\},$$

We start from simplifying variables defined in the expression of the variance.

$$\alpha_2 = \frac{\left(p - \frac{p(m-e^\epsilon(1-p)-p)}{(e^\epsilon(1-p)+p)(m-1)}\right)^2}{km}\left(1-\frac{1}{m}\right)$$

$$= \frac{p^2(w-1)^2}{w^2k(m-1)},$$

and

$$\begin{aligned}\alpha_1 =& \frac{p}{m} + \left(1-\frac{1}{m}\right)\frac{p(m-e^\epsilon(1-p)-p)}{(e^\epsilon(1-p)+p)(m-1)}\\ =& \frac{p}{m} + \frac{p(m-e^\epsilon(1-p)-p)}{m(e^\epsilon(1-p)+p)}\\ =& \frac{pm}{m(e^\epsilon(1-p)+p)}\\ =& \frac{p}{e^\epsilon(1-p)+p} = \frac{p}{w}.\end{aligned}$$

Also

$$(p-q)^2(1-1/m) = \alpha_2 k(m-1) = \frac{p^2(w-1)^2}{w^2}.$$

Then minimizing $Var[\hat{f}(d)]$ is equivalent to minimize

$$\frac{w^2}{(w-1)^2p}\left[f(d)(1-\frac{p}{k}) + (n-f(d))(\frac{1}{w}-\frac{p}{w^2k})\right] \quad (13)$$

As $w-1 = (e^\epsilon-1)(1-p)$, and $e^\epsilon-1>0$, (13) is equivalent to

$$\begin{aligned}&\frac{w^2}{(1-p)^2p}\left[f(d)(1-\frac{p}{k}) + (n-f(d))(\frac{1}{w}-\frac{p}{w^2k})\right]\\ =&\frac{f(d)\left(kw^2-w^2p\right)+(n-f(d))\left(kw-p\right)}{k(1-p)^2p}\\ =&\frac{n\left(kw-p\right)}{k(1-p)^2p} + \frac{f(d)\left(kw^2-w^2p-kw+p\right)}{k(1-p)^2p}\\ =&\frac{n\left(kw-p\right)}{k(1-p)^2p} + \frac{f(d)\left(kw(w-1)+p(1-w)(1+w)\right)}{k(1-p)^2p}\\ =&n\left\{\frac{(kw-p)}{k(1-p)^2p} + \frac{\lambda(w-1)(kw-p-wp)}{k(1-p)^2p}\right\}.\end{aligned}$$

This completes the proof for Lemma 1.

Appendix D
Remarks for Implementation

Choice of threshold and the privacy-utility trade-off. The choice of T and b is critical in balancing the utility-privacy trade-off. The privacy guarantee becomes stronger with larger T and b, while T and b determine the utility from different perspectives: a larger T ensures that only popular data is revealed to the server, which is not ideal if the server's goal is to collect as much unknown data as possible. Conversely, a large b introduces more more randomness, reducing accuracy when the server aims to learn the most popular data strings. Therefore, the choice of T and b should align with the specific application scenarios.

Joint usage of the two frameworks. The two data collection frameworks proposed in this paper can be used jointly in a more on-demand manner in practice. For example, the server can collect popular data strings through the unknown domain collection framework and then check their frequency with the sketch matrix in the GCMS. The server can also track the popularity of targeted data by calculating an optimal p and s according to OCMS with previously estimated frequency, and send p and s to each client as their mechanism parameters in the next iteration. Another use-case example is the server maintaining a list of top-k popular items. The server can fine-tune p and s according to the frequency regime of the required popularity and send them to each client. Each time the server obtains a list of data strings from the unknown domain collection framework, the server checks the frequency and updates the list. The overall privacy guarantee is obtained by composition of the privacy of GCMS amplified by shuffler (Theorem 1) and the privacy of data string collection (Theorem 4).

Appendix E
Performance Benchmarks

We conduct experiments to evaluate the performance of our approach and Apple's CMS using the following metrics:

- **Client Runtime**: The time required for the client to generate a single LDP report. This excludes the communication time between the client and server, as it highly depends on network conditions, which are beyond the scope of this work.
- **Server Runtime**: The time needed for the server to construct the LDP matrix.
- **Client Communication Cost**: The size of each LDP report sent by one client.
- **Server Memory Cost**: The storage required on the server to store the LDP matrix.

It is important to note that our prototypes were implemented in Python without extensive optimization such as message compression; thus, actual performance may be better in a production environment. This experiment is conducted solely to demonstrate the comparative performance of the two approaches.

The experimental settings are identical to those described in Section VII-D, and the hardware setting is Mac Book Pro (M1 chip, 32GB RAM). Benchmark results are presented in Table I. Both client and server runtimes for GCMS outperform those of Apple CMS, primarily because GCMS uses a much smaller message vector. It also leads to smaller client communication cost. Additionally, the server-side computation in GCMS involves only simple counting operations, whereas Apple CMS requires floating-point arithmetic. As a result, GCMS achieves better server-side performance. In terms of memory usage, both approaches require the server to store matrices of the same size, resulting in equivalent storage costs.

Metric	OCMS	Apple CMS
Client Runtime (ms)	0.00659	0.01782
Server Runtime (s)	0.0346	0.2209
Server Memory Cost (bytes)	86752	86752
Client Communication Cost (bytes)	171.84	912.00

TABLE I
PERFORMANCE COMPARISON BETWEEN OCMS AND APPLE CMS