RM^2: Answer Counting Queries Efficiently under Shuffle Differential Privacy

QIYAO LUO, OceanBase, Ant Group, China
JIANZHE YU, Hong Kong University of Science and Technology, China
WEI DONG*, Nanyang Technology Univeristy, Singapore
QUANQING XU, OceanBase, Ant Group, China
CHUANHUI YANG, OceanBase, Ant Group, China
KE YI, Hong Kong University of Science and Technology, China

Differential privacy (DP) is a leading standard for protecting individual privacy in data collection and analysis. This paper explores the shuffle model of DP, which balances privacy and utility by allowing users to send messages to a trusted shuffler before reaching an untrusted analyzer anonymously. We focus on efficiently implementing the matrix mechanism in shuffle-DP, where efficiency is defined by the number of messages each user sends. Our contributions include a baseline shuffle-DP mechanism that naively adapts the matrix mechanism, followed by an improved mechanism that reduces message complexity while maintaining error levels comparable to central-DP. We demonstrate the versatility of our approach across common query workloads, such as range queries and data cubes, achieving significant improvements in message efficiency. Experimental results confirm that our method outperforms the baseline solution while closely matching the accuracy of central-DP mechanisms.

CCS Concepts: • **Security and privacy** → **Privacy-preserving protocols**; • **Information systems** → **Data management systems**.

Additional Key Words and Phrases: Differential privacy; Counting query; Shuffle model

ACM Reference Format:
Qiyao Luo, Jianzhe Yu, Wei Dong, Quanqing Xu, Chuanhui Yang, and Ke Yi. 2025. RM^2: Answer Counting Queries Efficiently under Shuffle Differential Privacy. *Proc. ACM Manag. Data* 3, 3 (SIGMOD), Article 210 (June 2025), 24 pages. https://doi.org/10.1145/3725415

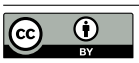
1 Introduction

Differential privacy (DP) [19, 20] has now become the de facto standard for provable, quantifiable privacy protection in personal data collection and analysis. Depending on the trust assumption, the following three models of DP have been studied in the literature:

- The *central model* of differential privacy (central-DP) assumes that there is a trusted data curator, who collects the data from the users and runs the required queries, before releasing the privatized results to the (untrusted) analyzer or public. This model offers the highest

*Wei Dong is the corresponding author.

Authors' Contact Information: Qiyao Luo, luoqiyao.lqy@antgroup.com, OceanBase, Ant Group, China; Jianzhe Yu, jyuca@cse.ust.hk, Hong Kong University of Science and Technology, China; Wei Dong, wei_dong@ntu.edu.sg, Nanyang Technology Univeristy, Singapore; Quanqing Xu, xuquanqing.xqq@oceanbase.com, OceanBase, Ant Group, China; Chuanhui Yang, rizhao.ych@oceanbase.com, OceanBase, Ant Group, China; Ke Yi, yike@cse.ust.hk, Hong Kong University of Science and Technology, China.



ACM 2836-6573/2025/6-ART210
https://doi.org/10.1145/3725415

utility (i.e., lowest errors in the privatized query results), but the trust assumption is the strongest. In real-world scenarios, it is hard to put full trust on any data curator; possibly, only certain government agencies (such as the US Census Bureau [2, 22]) might come close to being such a trusted data curator.

- The *local model* of differential privacy (local-DP), on the other hand, completely removes all trust assumptions. Under this model, each user privatizes their own data before sending it to the analyzer. This model is appealing to the end users, as they have complete control over how their data shall be privatized. As a result, many major companies have adopted local-DP when collecting sensitive user information [16, 21, 39, 40]. However, compared with central-DP, mechanisms in local-DP suffer a significant blowup in the error levels, from[1] $\tilde{O}(1)$ to at least $\tilde{O}(\sqrt{n})$, where n is the number of users, and this blowup is more serious for complicated query workloads. As a result, only some simple queries, such as bit counting and frequency estimation, have been actually deployed under local-DP.
- The *shuffle model* of differential privacy (shuffle-DP) [4, 5, 8, 10, 12, 23, 26, 33] has emerged in recent years, which for many problems has achieved a sweet spot between the two models above. This model assumes that messages from the users to the untrusted analyzer are anonymous, or equivalently, the users send their messages to a trusted shuffler, which randomly shuffles the messages before handing them to the analyzer. Although there is still some trust assumption in the shuffle-DP model, a trusted shuffler is much easier to implement than a trusted data curator who may perform arbitrary computations. A number of possible implementations of the shuffler exist, such as mix networks [9, 15], onion routing [18, 36], and trusted hardware [8, 37]. On the other hand, people have been able to design shuffle-DP mechanisms whose error levels match those of their central-DP counterparts up to constant factors [4, 5, 23, 26, 33], thus making shuffle-DP a more promising approach to private data collection and analysis.

A *counting query* returns the number of tuples in a dataset that satisfy a given predicate, such as the number of individuals with salary in a certain range. This is one of the most basic analytical queries that people are interested in answering on a dataset of personal information. Meanwhile, some errors in the query answers are tolerable, it thus has become a main test bed for DP techniques. A single counting query can be answered easily using classical DP mechanisms (in all the three DP models above), so research has focused more on the problem of how to answer a class of counting queries, often referred to as a query *workload*, which can be represented by a matrix $\boldsymbol{W}$ (please see Section 2 for the precise definition of $\boldsymbol{W}$).

In the DP literature, there are three main approaches to answering a given query workload $\boldsymbol{W}$:

(1) Use a DP algorithm to learn a privatized statistical model of the dataset, and then use the learned model to answer the queries in $\boldsymbol{W}$ [11, 27, 43]. However, as these learning algorithms need to make many iterations over the dataset, it is not clear how to adopt them in the local-DP or shuffle-DP model without numerous rounds of communication.
(2) Use the privacy composition theorem (Lemma 2.2) to divide the privacy budget to the queries in $\boldsymbol{W}$ and answer each with a single-query mechanism. The error of this approach is proportional to $\tilde{O}(\sqrt{\|\boldsymbol{W}\|_1})$, where $\|\boldsymbol{W}\|_1$, the ℓ_1-norm of $\boldsymbol{W}$, is equal to the maximum number of queries any user's data is involved in. Thus, it only works when $\|\boldsymbol{W}\|_1$ is small.
(3) The *matrix mechanism* [30, 32] designs another workload $\boldsymbol{A}$ (referred to as the *query strategy*) with a small $\|\boldsymbol{A}\|_1$, while all queries in $\boldsymbol{W}$ can be answered by combining a small number of queries in $\boldsymbol{A}$. When the workload $\boldsymbol{W}$ is well-structured, such as range queries and data

[1]The $\tilde{O}$ notation suppresses polylogarithmic factors.

cubes, good query strategies have been designed [17, 28, 35, 41], which have led to the state-of-the-art DP mechanisms for these workloads.

1.1 Our Contributions

In this paper, we study the problem of how to run the matrix mechanism in the shuffle-DP model efficiently, where efficiency is measured by the number of messages sent by each user to the shuffler. This is in line with all prior work on shuffle-DP, as it has been identified that the bottleneck of a shuffle-DP mechanism is the cost of shuffling all the messages. To this end, our contributions are summarized as follows.

(1) We first present a straw-man solution in Section 3 by directly replacing the components of the matrix mechanism with their shuffle-DP counterparts. This results in a shuffle-DP mechanism in which each users sends $\|\boldsymbol{A}\|_1 + \tilde{O}(Q/n)$ messages in expectation, where Q is the number of queries in the query strategy $\boldsymbol{A}$.
(2) In Section 4 we present an improved shuffle-DP mechanism RM^2 that works for any query strategy $\boldsymbol{A}$ whose queries form a DAG. RM^2 only sends $1+\tilde{O}(\text{ones}(\boldsymbol{C}^{\text{S}})/n)$ messages per user, where $\text{ones}(\boldsymbol{C}^{\text{S}})$ is equal to the number of nodes and edges in the corresponding DAG. We also show that RM^2 can achieve the same error level as that of the matrix mechanism in the central-DP model up to constant factors.
(3) In Section 5, we demonstrate the versatility of RM^2 by instantiating it for range queries (both one- and multi-dimensional) and data cubes, which are the most commonly studied workloads. For range queries, we show how RM^2 can use only $1 + o(1)$ messages per user, over both small and large domains. The experiments in Section 6 show that RM^2 significantly improves over the straw-man solution, while almost matching the errors of the central-DP matrix mechanism.

1.2 Related Work

The problem of answering a workload of counting queries has received considerable attention in central-DP, including the matrix mechanism [30, 32], the eigen-design algorithm [31] and differentially private data cube releasing [17]. For range queries, the hierarchical strategy [28, 35] and the Haar wavelet strategy [41] have achieved the optimal error. [35] studied how the branching factor of the tree in hierarchical strategy can improve the accuracy. [14, 42] translated central-DP algorithms to the local model and answered range queries in local-DP. Most work considers additive errors, as we do in this paper, but relative errors have also been used [24].

The shuffle model was first introduced in [29] which solves the secure summation problem in $O(\log n)$ messages. Combining this idea with differential privacy [8] has led to the development of the shuffle-DP model. [5] improved the message efficiency of shuffle-DP summation protocol to $O(1)$ messages and [26] further improved to $1 + o(1)$ messages, both achieving central-DP error. [10] studied the distinct count problem in shuffle-DP.

Some works have answered specific types of counting queries in shuffle-DP: [23, 33] developed shuffle-DP protocols for the frequency estimation problem that achieves central-DP error: [33] achieved $O(\log n)$ error with $1 + o(1)$ messages for small domain setting; and $O(\log n)$ error with constant messages or $\omega(1) \cdot O(\log n)$ error with $1 + o(1)$ messages for the large domain setting. [23] achieved $O(\log n)$ error with $O(\log n)$ messages and further combined the matrix mechanism with their frequency estimation protocol to tackle range queries. For 1D range queries, their protocol sends $O(\log^3 n)$ messages and achieves a maximum error of[2] $O(\log^{3.5} n)$.

[2]The results are from Table 3 in [23]; we assume $\varepsilon = \Theta(1)$ and $\log n = \Theta(\log(1/\delta)) = \Theta(\log B)$ for simplicity.

2 Preliminaries

2.1 Notation

We use boldface letters to denote vectors (in lower case) and matrices (in upper case), and subscripts to indicate the position of an element. Recall that the ℓ_1-norm of a vector $\boldsymbol{x}$ is $\|\boldsymbol{x}\|_1 \coloneqq \sum_i |\boldsymbol{x}_i|$ and that of a matrix $\boldsymbol{M}$ is $\|\boldsymbol{M}\|_1 \coloneqq \max_j \sum_i |\boldsymbol{M}_{i,j}|$. The ℓ_∞-norm of a vector $\boldsymbol{x}$ is $\|\boldsymbol{x}\|_\infty = \max_i |\boldsymbol{x}_i|$.

We consider a distributed setting with n users, where user i holds a value $\boldsymbol{v}_i \in [B] \coloneqq \{1, \ldots, B\}$ or nothing (then set $\boldsymbol{v}_i = \perp$). Thus the input dataset can be represented as a vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n) \in ([B] \cup \{\perp\})^n$. Let $\boldsymbol{x} \in \mathbb{N}^B$ be a column vector that represents the histogram of $\boldsymbol{v}$, i.e., $\boldsymbol{x}_i$ equals to the number of times that i appears in $\boldsymbol{v}$.

2.2 Differential Privacy

Given privacy parameters ε and δ, a randomized mechanism $\mathcal{M} : ([B] \cup \{\perp\})^n \to \mathcal{Y}$ is (ε, δ)-*differentially private (DP)*, if for any two neighboring datasets $\boldsymbol{v} \sim \boldsymbol{v}'$ (i.e., $\boldsymbol{v}$ and $\boldsymbol{v}'$ differ in one position), $\mathcal{M}(\boldsymbol{v})$ and $\mathcal{M}(\boldsymbol{v}')$ are (ε, δ)-indistinguishable. That is, for any subset of outputs $Y \subseteq \mathcal{Y}$,

$$\Pr[\mathcal{M}(\boldsymbol{v}) \in Y] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(\boldsymbol{v}') \in Y] + \delta. \tag{1}$$

Typically, the privacy parameter ε is a positive constant and δ is an inverse-polynomial of n (i.e., $0 \leq \delta < n^{-\Omega(1)}$).

The form of $\mathcal{M}(\boldsymbol{v})$ gives rise to different DP models: In the central-DP model, $\mathcal{M}(\boldsymbol{v})$ is allowed to be an arbitrary function of $\boldsymbol{v}$. In the local-DP model, $\mathcal{M}(\boldsymbol{v}) = (\mathcal{R}(\boldsymbol{v}_1), \ldots, \mathcal{R}(\boldsymbol{v}_n))$ where $\mathcal{R}$ is a local randomizer. Since $\boldsymbol{v}$ and $\boldsymbol{v}'$ differ in one position, only one of the $\mathcal{R}(\boldsymbol{v}_i)$'s would differ. So in the local model, (1) is equivalent to

$$\Pr[\mathcal{R}(\boldsymbol{v}_i) \in Y] \leq e^{\varepsilon} \cdot \Pr[\mathcal{R}(\boldsymbol{v}_i') \in Y] + \delta,$$

for any $\boldsymbol{v}_i \neq \boldsymbol{v}_i'$.

In the shuffle-DP model, the local randomizer $\mathcal{R}$ outputs a multiset of messages, and $\mathcal{M}(\boldsymbol{v})$ is the union of all the messages. This captures the situation where the analyzer receives all messages but has no knowledge on who sent which message.

Differential privacy has several nice properties:

Lemma 2.1 (Post-processing [20]). *If $\mathcal{M}$ satisfies (ε, δ)-DP and $\mathcal{M}'$ is any randomized mechanism, then $\mathcal{M}'(\mathcal{M}(\boldsymbol{v}))$ satisfies (ε, δ)-DP.*

Lemma 2.2 (Composition [20]). *Let $\mathcal{M}_1, \cdots, \mathcal{M}_m$ all be (ε, δ)-DP mechanisms. Consider a composed mechanism $\mathcal{M}(\boldsymbol{v}) = (\mathcal{M}_1(f_1(\boldsymbol{v})), \cdots, \mathcal{M}_m(f_m(\boldsymbol{v})))$ with functions $f_i : ([B] \cup \{\perp\})^n \to ([B] \cup \{\perp\})^n$. Suppose on any neighboring datasets $\boldsymbol{v} \sim \boldsymbol{v}'$, we have $f_i(\boldsymbol{v}) \sim f_i(\boldsymbol{v}')$ on at most k of the f_i's, while $f_i(\boldsymbol{v}) = f_i(\boldsymbol{v}')$ on the rest. Then $\mathcal{M}$ satisfies (ε', δ')-DP, where*

- *(basic composition) $\varepsilon' = k\varepsilon$ and $\delta' = k\delta$; or*
- *(advanced composition) $\varepsilon' = \sqrt{2k \log(1/\delta'')}\varepsilon + k\varepsilon(e^{\varepsilon} - 1)$ and $\delta' = \delta'' + k\delta$, for any $\delta'' > 0$.*

Below are some basic DP mechanisms:

Lemma 2.3 (Gaussian mechanism [20]). *Let $f : ([B] \cup \{\perp\})^n \to \mathbb{R}$ be a function with sensitivity 1, i.e., $|f(\boldsymbol{v}) - f(\boldsymbol{v}')| \leq 1$ for all neighbouring inputs $\boldsymbol{v} \sim \boldsymbol{v}'$. The mechanism $M(\boldsymbol{v}) = f(\boldsymbol{v}) + z$ satisfies (ε, δ)-DP, where z is sampled from the Gaussian distribution $\mathcal{N}(0, \frac{2\ln(1.25/\delta)}{\varepsilon^2})$. The error of the Gaussian mechanism is $O(\frac{1}{\varepsilon}\sqrt{\log \frac{1}{\delta}})$ with constant probability.*

Lemma 2.4 (Binomial mechanism [19, 23, 33]). *Let $f : ([B] \cup \{\perp\})^n \to \mathbb{Z}$ be a function with sensitivity 1. The mechanism $M(\boldsymbol{v}) = f(\boldsymbol{v}) + z - np$ or $M(\boldsymbol{v}) = f(\boldsymbol{v}) - z + np$ satisfies (ε, δ)-differential*

privacy, where z is sampled from the Binomial distribution $\mathrm{Bin}(n,p)$ *for*[3] $p = \frac{32\ln(2/\delta)}{\varepsilon^2 n}$. *The error of the Binomial mechanism is* $O(\frac{1}{\varepsilon}\sqrt{\log\frac{1}{\delta}})$ *with constant probability.*

2.3 Counting Queries and the Matrix Mechanism

	Original range queries	Query strategy: Identity	Query strategy: Hierarchy	Query strategy: Haar wavelet
Query matrix / Strategy matrices (e.g., $B=4$)	$\boldsymbol{W} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ & \cdots & & \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$	$\boldsymbol{A} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$
Error	$O\left(\frac{B}{\varepsilon}\log B\sqrt{\log\frac{1}{\delta}}\right)$	$O\left(\frac{\sqrt{B}}{\varepsilon}\log B\sqrt{\log\frac{1}{\delta}}\right)$	$O\left(\frac{\log^2 B}{\varepsilon}\sqrt{\log\frac{1}{\delta}}\right)$	$O\left(\frac{\log^2 B}{\varepsilon}\sqrt{\log\frac{1}{\delta}}\right)$

Table 1. Comparisons of different query strategies in 1D range queries under (ε,δ)-DP.

A *counting query* is defined by a 0-1 row vector $\boldsymbol{w} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_B] \in \{0,1\}^B$. The answer of the query is $\boldsymbol{wx} = \boldsymbol{w}_1\boldsymbol{x}_1 + \boldsymbol{w}_2\boldsymbol{x}_2 + \cdots + \boldsymbol{w}_B\boldsymbol{x}_B$, where $\boldsymbol{x}$ is the histogram. The counting query has sensitivity 1, so we can simply apply the Gaussian mechanism. The more interesting problem is how to answer a *workload* of W queries. It is convenient to represent such a workload as a matrix $\boldsymbol{W} \in \{0,1\}^{W\times B}$, so the answers to all the W queries can be denoted as $\boldsymbol{Wx}$.

Given a differentially private protocol $\mathcal{P} : ([B] \cup \{\perp\})^n \to \mathbb{N}^W$ that returns an estimate for each of the W queries, a common measure of utility is the maximum error (a.k.a., the ℓ_∞ error). More precisely, we say that $\mathcal{P}$ has error α with probability $1-\beta$, if

$$\Pr\left[\|\mathcal{P}(\boldsymbol{v}) - \boldsymbol{Wx}\|_\infty > \alpha\right] < \beta.$$

For simplicity, β is often taken as a constant when stating error bounds. In addition to the maximum error, other error measures such as the ℓ_2-error have also been adopted in the literature.

Most of the state-of-the-art central-DP mechanisms for counting queries can be considered as various instantiations of the *matrix mechanism* [32]. Instead of answering the original queries directly, it finds an alternative set of queries $\boldsymbol{A}$, known as the *query strategy*, and computes a differentially private estimate of $\boldsymbol{Ax}$, denoted $\widetilde{\boldsymbol{Ax}}$. This is done by running the Gaussian mechanism for each query in $\boldsymbol{A}$ together with the composition theorem (Lemma 2.2) with $k = \|\boldsymbol{A}\|_1$, since an element appears in at most $\|\boldsymbol{A}\|_1$ queries of $\boldsymbol{A}$. More precisely, we allocate to each query a privacy budget of $\varepsilon' = \varepsilon/\|\boldsymbol{A}\|_1, \delta' = \delta/\|\boldsymbol{A}\|_1$ with basic composition, or $\varepsilon' = \varepsilon/\sqrt{2\|\boldsymbol{A}\|_1\log(2/\delta)}, \delta' = \delta/(2\|\boldsymbol{A}\|_1)$ with advanced composition (setting $\delta'' = \delta/2$). Comparing the ε' values, one should use basic composition when $\|\boldsymbol{A}\|_1 < 4\log^2(2/\delta)$, and advanced composition otherwise. Finally, $\boldsymbol{Wx}$ can be estimated as $\boldsymbol{W}\boldsymbol{A}^+ \cdot \widetilde{\boldsymbol{Ax}}$, where $\boldsymbol{A}^+$ is the pseudoinverse of $\boldsymbol{A}$.

Remark. The traditional matrix mechanism under central-DP employs the vector form of the Gaussian mechanism based on ℓ_2 sensitivity [20]. Its error matches that of our proposed version when $\|A\|_1$ is small (i.e., using basic composition), but is improved by a factor of $O(\sqrt{\log(1/\delta)})$ when $\|A\|_1$ is large (i.e., using advanced composition). We present the matrix mechanism using

[3]For concrete values of ε, δ, and n, the value of p can be further optimized using numerical methods [33].

the Gaussian mechanism and composition theorem to facilitate understanding our extension to the matrix mechanism under shuffle-DP, which will be discussed in Section 3.2. To ensure a fair comparison, we provide both theoretical and empirical results for the traditional matrix mechanism in this paper.

The design of $\boldsymbol{A}$ depends on the given query workload $\boldsymbol{W}$, as illustrated in the following example:

Example 2.1. The one-dimensional range query workload consists of the $\binom{B}{2}$ queries in the form of $\boldsymbol{w} = [0, \ldots, 0, 1, \ldots, 1, 0, \ldots, 0]$. If we were to answer these queries directly using the Gaussian mechanism plus advanced composition (or using the vector form of the Gaussian mechanism), this would lead to a large error of $\tilde{O}(B)$. Table 1 presents three query strategies that achieve much smaller errors. In particular, the hierarchical strategy [28, 35] and the Haar wavelet strategy [41] have been demonstrated to achieve the optimal error for 1D range queries [32]. △

In addition to 1D range queries, good query strategies have also been identified for other workloads like 2D range queries [28, 31, 41], data cubes [17], k-way marginals [6, 17, 31], etc. In this paper, we focus on the problem of how to run the matrix mechanism in the shuffle-DP model for a given workload $\boldsymbol{W}$. Observe that the first step of identifying a good query strategy $\boldsymbol{A}$ is independent of the dataset, so we can use any of the previous methods in central-DP to find $\boldsymbol{A}$, while the third step of computing $\boldsymbol{W}\boldsymbol{A}^+ \cdot \widetilde{\boldsymbol{Ax}}$ is a simple post-processing step. Thus the main problem is how to compute $\widetilde{\boldsymbol{Ax}}$ for a given query strategy $\boldsymbol{A}$ in the shuffle-DP model. We will assume that all entries in $\boldsymbol{A}$ are 0's and 1's. Although some query strategies use -1 in $\boldsymbol{A}$, such as the Haar wavelet for 1D range queries, they do not provide more-than-constant-factor improvements, and these -1's pose difficulties in the shuffle-DP model.

3 A Straw-man Shuffle-DP Mechanism

Although the matrix mechanism has not been formally studied in the shuffle-DP model, it is not hard to find a solution by combining existing techniques. In this section, we describe this straw-man solution, before improving it in subsequent sections.

3.1 Answering One Counting Query

A single counting query $\boldsymbol{w}$ is equivalent to the *bit counting* problem: Each user holds a bit $\boldsymbol{b}_i$ and the objective is to count the total number of 1's. To reduce a counting query $\boldsymbol{w}$ to this problem, we simply set $\boldsymbol{b}_i = 1$ if $\boldsymbol{w}_{v_i} = 1$ (i.e., $\boldsymbol{v}_i$ is counted by $\boldsymbol{w}$) and 0 otherwise. A standard solution is to run the binomial mechanism in the shuffle-DP model, as follows [3, 19, 33]:

- **Local Randomizer**: User i sends a 1 to the shuffler if her $\boldsymbol{b}_i = 1$ (we call this a *real message*). Additionally, each user independently and randomly sends a 1 to the shuffler (called a *noise message*) with probability $p = O\left(\frac{\ln(1/\delta)}{\varepsilon^2 n}\right)$. Each user thus sends $\boldsymbol{b}_i + p$ messages in expectation.
- **Analyzer**: The analyzer counts the number of 1's received, which equals $\sum_i \boldsymbol{b}_i + \mathrm{Bin}(n, p)$, and subtracts np.

To prove that a mechanism satisfies shuffle-DP, we may draw an equivalence between the analyzer's view (i.e., the multiset of messages received) and the output of a central-DP mechanism, which in this case is the binomial mechanism (Lemma 2.4). Note that the binomial mechanism (in the central-DP model) outputs a single number $\sum_i \boldsymbol{b}_i + \mathrm{Bin}(n, p) - np$, while the analyzer (in the shuffle-DP model) receives $\sum_i \boldsymbol{b}_i + \mathrm{Bin}(n, p)$ messages of 1. Since all messages are identical after shuffling, these messages contain no more information than a single number $\sum_i \boldsymbol{b}_i + \mathrm{Bin}(n, p)$. Thus, the output distribution of the binomial mechanism is the same as that of the analyzer's view shifted by np. This output equivalence means that this shuffle-DP mechanism enjoys the same privacy guarantee as that of the binomial mechanism, i.e., (ε, δ)-DP.

3.2 Answering Many Counting Queries

All the queries in a query strategy $\boldsymbol{A} \in \{0,1\}^{Q\times B}$ can be answered by running Q instances of the shuffle-DP protocol above in parallel, one for each query (row) of $\boldsymbol{A}$, but with two modifications. First, since the messages from all these instances are shuffled together, there need to be a way for the analyzer to distinguish them. Thus, instead of sending all messages of 1, users should send messages (both real and noise) identified by j for the j-th query. Second, as we run many queries, we need to divide the privacy budget using the composition theorem with $k = \|\boldsymbol{A}\|_1$, as done in Section 2.3.

In order to better analyze the protocol, it is convenient to adopt a matrix representation. Let $\boldsymbol{S} = \begin{bmatrix}\boldsymbol{A} & \boldsymbol{I}_Q\end{bmatrix}$, where $\boldsymbol{I}_Q$ denotes a $Q \times Q$ identity matrix. Let $\boldsymbol{z}$ be a column vector of size Q where each $\boldsymbol{z}_j$ is drawn from $\mathrm{Bin}(n,p)$ for $j = 1,\ldots,Q$. Note that here, the value of p is computed from ε' and δ'. Then the analyzer's view can be represented by $\boldsymbol{S}\cdot\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{z}\end{bmatrix}$. The j-th entry of this vector is exactly the number of times message j is received.

Theorem 3.1. *The straw-man solution above satisfies (ε,δ)-shuffle-DP. It achieves the same error as that of the matrix mechanism within constant factors, while each user sends up to $\|\boldsymbol{A}\|_1 + pQ$ messages in expectation.*

Proof. Privacy follows from the composition theorem (Lemma 2.2), the privacy of the binomial mechanism (Lemma 2.4), and the equivalence argument between the binomial mechanism and the bit-counting protocol. The error is the same as that of the matrix mechanism because the only difference is that the Gaussian mechanism has been replaced by the binomial mechanism, while the errors of the two mechanisms only differ by constant factors.

To bound the message count, observe that each user sends up to $\|\boldsymbol{A}\|_1$ real messages, one for each query that includes the user's value $\boldsymbol{v}_i$. They then also send a noise message with probability p for each of the Q queries. □

4 Real Messages Reduction Mechanism

We first use a simple example to illustrate the basic idea behind our Real Messages Reduction Mechanism (RM^2).

Example 4.1. Consider a small domain of $[B] = \{1,2\}$ with three queries $\boldsymbol{q}_1 = \boldsymbol{x}_1$, $\boldsymbol{q}_2 = \boldsymbol{x}_2$, $\boldsymbol{q}_3 = \boldsymbol{x}_1 + \boldsymbol{x}_2$. In the straw-man solution, the analyzer receives

$$\begin{aligned}\widetilde{\boldsymbol{q}_1} &= \boldsymbol{x}_1 + \boldsymbol{z}_1,\\ \widetilde{\boldsymbol{q}_2} &= \boldsymbol{x}_2 + \boldsymbol{z}_2,\\ \widetilde{\boldsymbol{q}_3} &= \boldsymbol{x}_1 + \boldsymbol{x}_2 + \boldsymbol{z}_3,\end{aligned}$$

and the corresponding matrix $\boldsymbol{S}$ is

$$\boldsymbol{S} = \begin{bmatrix}1 & 0 & 1 & 0 & 0\\ 0 & 1 & 0 & 1 & 0\\ 1 & 1 & 0 & 0 & 1\end{bmatrix}.$$

The (expected) number of messages sent per user is $\|\boldsymbol{A}\|_1 + pQ = 2 + 3p$.

In RM^2, we first change $\widetilde{\boldsymbol{q}_3}$ to

$$\widetilde{\boldsymbol{q}_3} = \boldsymbol{x}_1 + \boldsymbol{x}_2 - \boldsymbol{z}_3,$$

namely, change the matrix S to

$$S^{\mathrm{C}} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & -1 \end{bmatrix}.$$

Note that this does not affect the privacy of $S^{\mathrm{C}} \cdot \begin{bmatrix} x \\ z \end{bmatrix}$ in the central-DP model (hence the superscript $^{\mathrm{C}}$), since either adding or subtracting a binomial noise satisfies DP (Lemma 2.4). Another minor change when using S^{C} is that one should add np to $\widetilde{q_3}$ to get an unbiased estimate for q_3.

However, we cannot compute this $\widetilde{q_3}$ in the shuffle-DP model. One may attempt to ask each user to send a real message identified by "3" if she holds 1 or 2, and with probability p send a noise message identified by "−3". Then the analyzer would count these two types of messages and take their difference as $\widetilde{q_3}$. However, this breaks DP since the number of "3" messages is equal to $x_1 + x_2$, which is revealed to the analyzer. So these noise messages did not serve their purpose of hiding the number of real messages. More fundamentally, this is because the analyzer's view is *not* equivalent to its central-DP counterpart: the former reveals both $x_1 + x_2$ and z_3, while the latter only $x_1 + x_2 - z_3$.

To avoid this problem, we can only use a matrix S with 0-1 entries. We can get such an S through a linear transformation (which flips the third row, and then adds the first two rows to it):

$$S^{\mathrm{C}} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & -1 \end{bmatrix} \xrightarrow[T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & -1 \end{bmatrix}]{T \cdot S^{\mathrm{C}}} S^{\mathrm{S}} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

We can compute $\widetilde{t} = S^{\mathrm{S}} \cdot \begin{bmatrix} x \\ z \end{bmatrix}$ in shuffle-DP, i.e.,

$$\begin{aligned} \widetilde{t_1} &= x_1 + z_1, \\ \widetilde{t_2} &= x_2 + z_2, \\ \widetilde{t_3} &= z_1 + z_2 + z_3. \end{aligned}$$

More precisely, the shuffle-DP mechanism using S^{S} runs 3 instances of the bit counting mechanism, one for each $\widetilde{t_i}$. The first two are the same as before. In the third instance that computes $\widetilde{t_3}$, the users send no real messages but only noise messages. In particular, the noise message counts z_1 and z_2 must be consistent with those in the first two instances. To do so, a user should first generate 3 independent Bernoulli random variables X_1, X_2, X_3 each equal to 1 with probability p. Then she sends X_1 noise message "1", X_2 noise message "2", and $X_1 + X_2 + X_3$ noise messages "3".

Critically, note that as long as T is invertible, there is a one-to-one correspondence between $[\widetilde{q_1}, \widetilde{q_2}, \widetilde{q_3}]$ and $[\widetilde{t_1}, \widetilde{t_2}, \widetilde{t_3}]$. This means that S^{C} in central-DP and S^{S} in shuffle-DP enjoy the same privacy and error. Now the interesting observation is that, while S^{S} maintains the same output distribution as S^{C}, it has a smaller message count: Each user sends only 1 real message and $5p$ noise messages, which is usually less than before since $p = \tilde{O}(1/n)$ is very close to 0. △

4.1 Mechanism Overview

Generalizing the example above, our RM^2 consists of the following steps (Figure 1 gives a schematic illustration):

(1) For a given workload $\boldsymbol{W}$, find a good query strategy $\boldsymbol{A}$ using any method from the matrix mechanism framework.
(2) Find noise coefficients $\boldsymbol{c} \in \{-1,+1\}^Q$ (e.g., $\boldsymbol{c} = [+1,+1,-1]$ in Example 4.1); set $\boldsymbol{S}^{\mathrm{C}} = \begin{bmatrix}\boldsymbol{A} & \mathrm{diag}(\boldsymbol{c})\end{bmatrix}$.
(3) Find an invertible $\boldsymbol{T}$ such that $\boldsymbol{S}^{\mathrm{S}} = \begin{bmatrix}\boldsymbol{A}^{\mathrm{S}} & \boldsymbol{C}^{\mathrm{S}}\end{bmatrix} = \boldsymbol{T} \cdot \boldsymbol{S}^{\mathrm{C}}$ is a 0-1 matrix.
(4) Compute $\boldsymbol{S}^{\mathrm{S}} \cdot \begin{bmatrix}\boldsymbol{x} \\ \boldsymbol{z}\end{bmatrix}$ in the shuffle-DP model by running multiple bit counting instances. The detailed algorithms for the local randomizer and analyzer are shown in Algorithm 1 and 2.
(5) The analyzer applies the inverse transform $\boldsymbol{W}\boldsymbol{A}^+ \cdot \boldsymbol{T}^{-1}$ to obtain the estimates for $\boldsymbol{W}\boldsymbol{x}$.

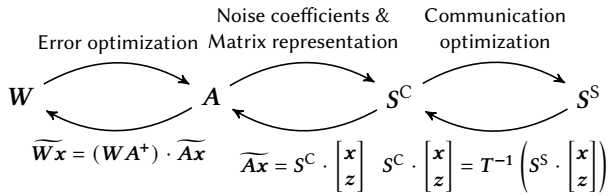


Fig. 1. Workflow of RM^2

Theorem 4.1. *For any query strategy $\boldsymbol{A} \in \{0,1\}^{Q\times B}$, any noise coefficient $\boldsymbol{c} = \{-1,+1\}^Q$, and any invertible $\boldsymbol{T}$ such that $\begin{bmatrix}\boldsymbol{A}^S & \boldsymbol{C}^S\end{bmatrix} = \boldsymbol{T} \cdot \begin{bmatrix}\boldsymbol{A} & \mathrm{diag}(\boldsymbol{c})\end{bmatrix}$ is a 0-1 matrix, RM^2 satisfies (ε,δ)-shuffle-DP. It achieves the same error as that of the matrix mechanism within constant factors, while each user sends $\|\boldsymbol{A}^S\|_1 + p \cdot \mathrm{ones}(\boldsymbol{C}^S)$ messages in expectation, where $\mathrm{ones}(\boldsymbol{C}^S)$ denotes the number of 1's in $\boldsymbol{C}^S$.*

Proof. Privacy and error follow from Theorem 3.1 and the equivalence between $\boldsymbol{S}^{\mathrm{C}}$ and $\boldsymbol{S}^{\mathrm{S}}$. The real message count follows from the same argument before, while the noise message count is because a user sends a noise message with probability p (although not independently) for each 1 in $\boldsymbol{C}^{\mathrm{S}}$. □

Remark. Although both the matrix mechanism and RM^2 use a linear transformation, there are some fundamental differences: (1) The matrix mechanism uses the transformation to reduce the error while our goal is to reduce the message count in shuffle-DP. In fact, in our transformation, the output distribution remains equivalent, which means that both the error and the privacy guarantee do not change. (2) The transformation in the matrix mechanism need not be invertible, while our transformation must be, in order to ensure the equivalence. (3) The matrix mechanism only transforms the query matrix from $\boldsymbol{W}$ to $\boldsymbol{A}$, while we transform the concatenated matrix $\boldsymbol{S}^{\mathrm{C}} = \begin{bmatrix}\boldsymbol{A} & \mathrm{diag}(\boldsymbol{c})\end{bmatrix}$.

4.2 Finding c and T

To instantiate RM^2 for a given query strategy $\boldsymbol{A}$, we need a way to find $\boldsymbol{c}$ in step (2) and $\boldsymbol{T}$ in step (3), so that $\|\boldsymbol{A}^{\mathrm{S}}\|_1 + p \cdot \mathrm{ones}(\boldsymbol{C}^{\mathrm{S}})$ is minimized. However, this is very complicated optimization problem. In this section, we present a method for all *DAG query strategies*, observing that all the known query strategies in the matrix mechanism fall into this category, or can be converted into.

We say that a query strategy $\boldsymbol{A}$ is a *DAG query strategy*, if there is a directed acyclic graph (DAG) such that

(1) there is a one-to-one correspondence between the queries in $\boldsymbol{A}$ and the vertices in the DAG;
(2) $\boldsymbol{A}$ contains all the *base queries*, i.e., queries that query each single element of $[B]$;

Algorithm 1: Local Randomizer $\mathcal{R}^{\mathrm{RM}^2}$

Public Parameters: $n, \varepsilon, \delta, \boldsymbol{T}, \boldsymbol{S}^{\mathrm{C}}$
Input: $v \in [B]$ held by the user
Output: A multiset $\mathcal{T}$ of messages to send out

$\boldsymbol{S}^{\mathrm{S}} \leftarrow \boldsymbol{T} \cdot \boldsymbol{S}^{\mathrm{C}}$;
$\varepsilon', \delta' \leftarrow$ the privacy budget using either basic or advanced composition with $k = \|\boldsymbol{A}\|_1$;
Compute the value of p in the binomial mechanism using ε', δ', n;
// Construct the response vector.
$\boldsymbol{u} \leftarrow [0]^{B+Q}$;
$\boldsymbol{u}_v \leftarrow 1$;
for $i \leftarrow B+1$ **to** $B+Q$ **do**
  $\boldsymbol{u}_i \leftarrow \mathrm{Ber}(p)$;
// Generate the response messages.
$\mathcal{T} \leftarrow \varnothing$;
for $i \leftarrow 1$ **to** Q **do**
  $\mathcal{T} \leftarrow \mathcal{T} \uplus \{(\boldsymbol{S}^{\mathrm{S}} \cdot \boldsymbol{u})_i$ number of $i\}$;
return $\mathcal{T}$

Algorithm 2: Analyzer $\mathcal{A}^{\mathrm{RM}^2}$

Public Parameters: $n, \varepsilon, \delta, \boldsymbol{T}, \boldsymbol{S}^{\mathrm{C}}$
Input: A multiset $\mathcal{T}$ of all messages received
Output: Query results $\mathcal{Y} \in \mathbb{N}^Q$

Compute p as in Algorithm 1;
for $i \leftarrow 1$ **to** Q **do**
  $\mathcal{Z}_i \leftarrow$ the frequency of i in $\mathcal{T}$;
// Reconstruct unbiased estimates of queries.
$\widetilde{\mathcal{Y}} \leftarrow \boldsymbol{T}^{-1} \cdot \mathcal{Z}$;
for $i \leftarrow 1$ **to** Q **do**
  $\mathcal{Y}_i \leftarrow \widetilde{\mathcal{Y}}_i - (\boldsymbol{S}^{\mathrm{C}})_{i,B+i} \cdot np$;
return $\mathcal{Y}$

(3) for each non-base query $\boldsymbol{p} \in \boldsymbol{A}$, its children form a disjoint union of $\boldsymbol{p}$ (the children of $\boldsymbol{p}$ are all the nodes that $\boldsymbol{p}$ has an edge to in the DAG); and
(4) the DAG is bipartite, i.e., the vertices can be two-colored so that no adjacent vertices have the same color, and all base queries have the same color.

An example of a DAG query strategy consisting of 8 queries on a domain of size $B = 4$ is shown in Figure 2. Actually, condition (4) above is not necessary; we will show how to convert any query strategy to a DAG query strategy in Section 4.2.1. In all the applications that we consider in Section 5, all the queries strategies induce a DAG that immediately fulfills all 4 conditions above.

Since $p = \tilde{O}(1/n)$ is very small, it is more important to minimize $\|\boldsymbol{A}^{\mathrm{S}}\|_1$. Below we show, for any given DAG query strategy $\boldsymbol{A}$, how to find $\boldsymbol{c}$ and $\boldsymbol{T}$ such that $\|\boldsymbol{A}^{\mathrm{S}}\|_1 = 1$.

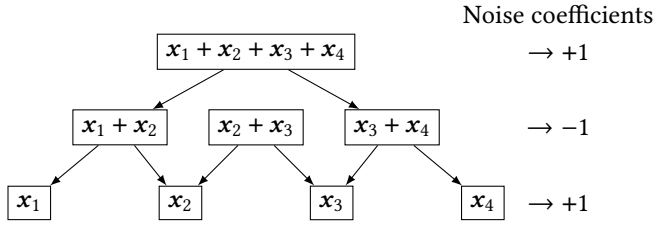


Fig. 2. An example of 8 queries, its underlying DAG and the noise coefficients assignment.

$$S^{\mathrm{C}} = \left[\begin{array}{cccc|cccccccc} & & & \boldsymbol{I}_4 & & \boldsymbol{I}_4 & & & & & & \\ 1 & 1 & & & & & -1 & & & & & \\ & 1 & 1 & & & & & -1 & & & & \\ & & 1 & 1 & & & & & -1 & & & \\ 1 & 1 & 1 & 1 & & & & & & 1 & & \end{array}\right] \xrightarrow{\text{Trans.}} S^{\mathrm{S}} = \left[\begin{array}{c|cccccccc} \boldsymbol{I}_4 & & & \boldsymbol{I}_4 & & & & & \\ & 1 & 1 & & & 1 & & & \\ & & 1 & 1 & & & 1 & & \\ & & & 1 & 1 & & & 1 & \\ & & & & & 1 & & 1 & 1 \end{array}\right]$$

Fig. 3. The RM^2 for the 8 queries (empty entries are 0)

(1) Assign $\boldsymbol{c}_i = +1$ to all base queries. Then assign -1 and $+1$ alternatively in a bottom-up fashion. Condition (4) above ensures that this can always be done consistently. Please see Figure 2 for an example.
(2) Generalizing Example 4.1, the idea is to eliminate all the non-base queries by performing row operations on $\boldsymbol{S}^{\mathrm{C}} = \begin{bmatrix}\boldsymbol{A} & \mathrm{diag}(\boldsymbol{c})\end{bmatrix}$, in a top-down fashion. Specifically, for each non-base query $\boldsymbol{q}_i$, if $\boldsymbol{c}_i = +1$, we subtract the rows of its children from it; if $\boldsymbol{c}_i = -1$, we flip it and add the rows of its children to it.
The linear transformation matrix $\boldsymbol{T}$ and its inverse $\boldsymbol{T^{-1}}$ are thus defined as follow:

$$\boldsymbol{T}_{i,j} = \begin{cases} \boldsymbol{c}_i & \text{if } j = i \\ -\boldsymbol{c}_i & \text{if } \boldsymbol{q}_j \text{ is a child of } \boldsymbol{q}_i \\ 0 & \text{otherwise} \end{cases};$$

$$(\boldsymbol{T}^{-1})_{i,j} = \begin{cases} \boldsymbol{c}_i & \text{if } j = i \\ 1 & \text{if } \boldsymbol{q}_j \text{ is a child of } \boldsymbol{q}_i \\ 0 & \text{otherwise} \end{cases}.$$

Example 4.2. Consider the 8 queries in Figure 2. The straw-man solution sends $4 + 8p$ messages per user. The transformation is shown in Figure 3. After the transformation, the message count is reduced to $1 + 16p$. △

Theorem 4.2. *For any DAG query strategy $\boldsymbol{A} \in \{0,1\}^{Q \times B}$, we can design $\boldsymbol{c} \in \{-1,+1\}^{Q}$ and an invertible $\boldsymbol{T}$ such that $\begin{bmatrix}\boldsymbol{A}^S & \boldsymbol{C}^S\end{bmatrix} = \boldsymbol{T} \cdot \begin{bmatrix}\boldsymbol{A} & \mathrm{diag}(\boldsymbol{c})\end{bmatrix}$ is a 0-1 matrix. Furthermore, it is guaranteed that $\|\boldsymbol{A}^S\|_1 = 1$ and $\mathrm{ones}(\boldsymbol{C}^S)$ is at most the total number of nodes (which is equal to Q) and edges in the DAG.*

Proof. We conclude the form of $\boldsymbol{S}^{\mathrm{S}} = \begin{bmatrix}\boldsymbol{A}^{\mathrm{S}} & \boldsymbol{C}^{\mathrm{S}}\end{bmatrix}$ after transformation. WLOG, we assume the first B queries in $\boldsymbol{A}$ are base queries querying from 1 to B.

- If $\boldsymbol{q}_i$ is a base query, no operation is performed, then $(\boldsymbol{S}^{\mathrm{S}})_{i,i} = (\boldsymbol{S}^{\mathrm{S}})_{i,i+B} = 1$, thus $(\boldsymbol{A}^{\mathrm{S}})_{i,i} = 1$ and $(\boldsymbol{C}^{\mathrm{S}})_{i,i} = 1$.

- If $\boldsymbol{q}_i$ is a non base query, then after the transformation,

$$(S^{\mathrm{S}})_{i,j+B} = \begin{cases} 1 & \text{if } j = i \\ 1 & \text{if } \boldsymbol{q}_j \text{ is a child of } \boldsymbol{q}_i \\ 0 & \text{otherwise} \end{cases},$$

  thus $(C^{\mathrm{S}})_{i,i} = 1$, $(C^{\mathrm{S}})_{i,j} = 1$ for all children $\boldsymbol{q}_j$.

So $\|A^{\mathrm{S}}\|_1 = 1$ and

$$\begin{aligned} \mathrm{ones}(C^{\mathrm{S}}) &= \sum_{\boldsymbol{q}_i} \left((C^{\mathrm{S}})_{i,i} + \sum_{\substack{\boldsymbol{q}_j \text{ is a} \\ \text{child of } \boldsymbol{q}_i}} (C^{\mathrm{S}})_{i,j} \right) \\ &= \sum_{\boldsymbol{q}_i} (C^{\mathrm{S}})_{i,i} + \sum_{\boldsymbol{q}_i} \sum_{\substack{\boldsymbol{q}_j \text{ is a} \\ \text{child of } \boldsymbol{q}_i}} (C^{\mathrm{S}})_{i,j}. \end{aligned}$$

The first sum equals the number of nodes in the DAG (i.e., Q) and the second equals the number of edges in the DAG. □

Remark. As will be seen in the applications in Section 5, the DAG in most cases are sparse, i.e., its number of edges is within a constant-factor Q. Thus, compared with the straw-man solution, RM^2 reduces the number of real messages (which is the more significant part) by a factor of $\|\boldsymbol{A}\|_1$, while increasing the number of noise messages by a constant factor.

4.2.1 DAG Query Strategy Conversion. Given a query strategy $\boldsymbol{A}$, there is a general method to convert it to a DAG query strategy with a constant increase in terms of error and communication.

First, identify all base queries and add the missing ones into $\boldsymbol{A}$, denoted as $\tilde{\boldsymbol{A}}$. $\tilde{\boldsymbol{A}}$ thus satisfies condition (2) and $||\tilde{\boldsymbol{A}}||_1 \leq ||\boldsymbol{A}||_1 + 1$. Since base queries are all present, every non-base query $\boldsymbol{p} \in \tilde{\boldsymbol{A}}$ can at least be represented as the disjoint union of base queries. One can also optimize the representation by finding an exact cover with minimum cardinality.

Then we make it "bipartite". For each query $\boldsymbol{p}$, we answer it twice: $\boldsymbol{p}^+$ and $\boldsymbol{p}^-$. The representation of $\boldsymbol{p}^+$ contains all $\boldsymbol{q}^-$'s where $\boldsymbol{q}$ is in the representation of $\boldsymbol{p}$, and the representation of $\boldsymbol{p}^-$ is simply $\boldsymbol{p}^+$. The corresponding DAG of the new query strategy $\boldsymbol{A}' \leftarrow \{\boldsymbol{p}^+, \boldsymbol{p}^- \mid \boldsymbol{p} \in \boldsymbol{A}\}$ is obviously bipartite (i.e., all $\boldsymbol{p}^+$'s have one color and all $\boldsymbol{p}^-$'s have the other color). Since $||\boldsymbol{A}'||_1 \leq 2||\tilde{\boldsymbol{A}}||_1 \leq 2(||\boldsymbol{A}||_1 + 1)$, the privacy budget allocated to each query in $\boldsymbol{A}'$ is roughly halved compared to the original privacy budget of each query in $\boldsymbol{A}$. This reduction in privacy budget leads to a proportional increase in the error by a factor of 2, but maintains the same asymptotic bound.

5 Applications

In this section, we show how RM^2 can be applied to a variety of workloads.

5.1 One-dimensional Range Queries

For 1D range queries, the hierarchical query strategy [28] (also known as the range tree [7]) consists of the $2B - 1$ queries shown in Figure 4 (assuming that B is a power of 2). Any range query can be answered by adding up $O(\log B)$ of these queries. It is already a bipartite DAG strategy (actually, a tree) with $O(B)$ nodes and edges, so we can apply RM^2 directly. Figure 5 shows the transformation for $B = 4$.

$$\boldsymbol{H}_{(B)} = \left[\begin{array}{c|c} 1 \quad 1 \quad \cdots \quad 1 & \\ \boldsymbol{H}_{(B/2)} & \boldsymbol{0} \\ \hline \boldsymbol{0} & \boldsymbol{H}_{(B/2)} \end{array}\right]$$

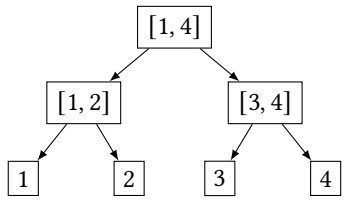


Fig. 4. Recursive construction of $\boldsymbol{H}_{(B)}$ (left) and tree structure of $\boldsymbol{H}_{(4)}$ (right).

$$\boldsymbol{S}^{\mathrm{C}} = \left[\begin{array}{cccc|cccc} & & \boldsymbol{I}_4 & & \boldsymbol{I}_4 & & & \\ 1 & 1 & & & & -1 & & \\ & & 1 & 1 & & & -1 & \\ 1 & 1 & 1 & 1 & & & & 1 \end{array}\right] \xrightarrow{\text{Trans.}} \boldsymbol{S}^{\mathrm{S}} = \left[\begin{array}{c|cccccccc} \boldsymbol{I}_4 & & & \boldsymbol{I}_4 & & & & & \\ & 1 & 1 & & & & 1 & & \\ & & & 1 & 1 & & & 1 & \\ & & & & & 1 & 1 & 1 & \end{array}\right]$$

Fig. 5. The RM^2 for 1D range queries of $B = 4$ (empty entries are 0)

Meanwhile, the query strategy has $\|\boldsymbol{A}\|_1 = \log B + 1$, so we use basic composition to divide the privacy budget. This results in $p = O\left(\frac{\log(1/\delta)\log^2 B}{\varepsilon^2 n}\right)$. Therefore, the performance of RM^2 can be summarized as follows.

Corollary 5.1. *For any $\varepsilon \geq 0$, $0 < \delta < 1$, and $n, B \in \mathbb{N}$, RM^2 satisfies (ε, δ)-shuffle-DP. It answers all 1D range queries with maximum error $O\left(\frac{\log^{1.5} B}{\varepsilon}\sqrt{\log B \log \frac{1}{\delta}}\right)$, and sends $1 + O\left(\frac{\log(1/\delta)\log^2 B}{\varepsilon^2} \cdot \frac{B}{n}\right)$ messages per user in expectation.*

In Table 2, we compare RM^2 with the previous DP mechanisms for 1D range queries.

Mechanism	Error	#Messages/user
Matrix	$O\left(\frac{\log^2 B}{\varepsilon}\sqrt{\log \frac{1}{\delta}}\right)$	central-DP
LWY [33]	$O\left(\frac{\sqrt{B}}{\varepsilon}\sqrt{\log B \log \frac{1}{\delta}}\right)$	$1 + O\left(\frac{\log(1/\delta)}{\varepsilon^2} \cdot \frac{B}{n}\right)$
GGKPV [23]	$O\left(\frac{\log^2 B}{\varepsilon}\sqrt{\log \frac{1}{\delta}}\right)$ †	$O\left(\frac{\log^2 B \log(1/\varepsilon\delta)}{\varepsilon^2}\right)$
Straw-man	$O\left(\frac{\log^2 B}{\varepsilon}\sqrt{\log \frac{1}{\delta}}\right)$	$\log B + 1 + O\left(\frac{\log(1/\delta)\log^2 B}{\varepsilon^2} \cdot \frac{B}{n}\right)$
RM^2	$O\left(\frac{\log^2 B}{\varepsilon}\sqrt{\log \frac{1}{\delta}}\right)$	$1 + O\left(\frac{\log(1/\delta)\log^2 B}{\varepsilon^2} \cdot \frac{B}{n}\right)$

† The bound given in [23] is $O\left(\frac{\log^{2.5} B \log(1/\delta)}{\varepsilon}\right)$, which is not tight.

Table 2. Comparisons of different DP mechanisms for the one-dimensional range queries

Remark. By default, the hierarchical strategy uses a binary tree. Past research [35] found that using a b-ary tree for some $b > 2$ can improve the performance. RM^2 can be easily applied to trees with different branching factors.

5.2 1D Range Queries in Large Domains

When the domain size $B < o(n)$, RM^2 is a $(1 + o(1))$-message shuffle-DP protocol. However, for a large B, e.g., the domain of all IP addresses, it incurs a prohibitive cost, and so does the straw-man solution. We observe that for the identity query strategy $\boldsymbol{A} = \boldsymbol{I}_B$, which only consists of the B base queries, computing $\boldsymbol{Ax} = \boldsymbol{x}$ degenerates into the *frequency estimation (FE)* problem. For the FE problem over a large domain, there is a $(1 + o(1))$-message shuffle-DP protocol [33] that can estimate all the base queries with error $\tilde{O}(1)$. However, as the identity query strategy is not a good strategy for range queries, directly using this FE protocol leads to errors as large as $\tilde{O}(\sqrt{B})$ (see Table 1).

In this subsection, we design a two-round shuffle-DP protocol that works with the hierarchical query strategy over a large domain. In the first round, our protocol reduces the domain size, as well as the number of queries in the query strategy, to less than n, while preserving all the range counts with at most $\tilde{O}(1)$ error. Then we apply RM^2 in the second round on this reduced domain. For simplicity we will assume that $\log n = \Theta(\log B) = \Theta(\log(1/\delta))$ and ε is a constant when stating the bounds in this subsection.

5.2.1 Domain Reduction. We say that a range is *heavy* if it contains at least ϕ elements, otherwise *light*, for some parameter ϕ to be determined later. The goal of domain reduction is to choose a small number of disjoint range queries in the hierarchical strategy, called the *atomic ranges*, such that their union is $[B]$. Meanwhile, we will ensure that each atomic range is either a base query or a light range. Suppose we have a way to decide if any range in the hierarchical query strategy is heavy or light. Then the domain reduction can be done in a top-down manner, as illustrated in the following example:

Example 5.1. Assume an initial domain of $[1, 32]$. The whole range $[1, 32]$ must be heavy. Then we check if its two children $[1, 16]$ and $[17, 32]$ are heavy, and recursively explore them if they are. As an example, suppose all the heavy ranges are shown in Figure 6. Note that they must form a connected subtree, since if a range is heavy, so must be its parent. Then we pick all the heavy base ranges, as well as every light range that has a heavy parent. In this example, this results in the $b = 9$ atomic ranges shown in the figure. △

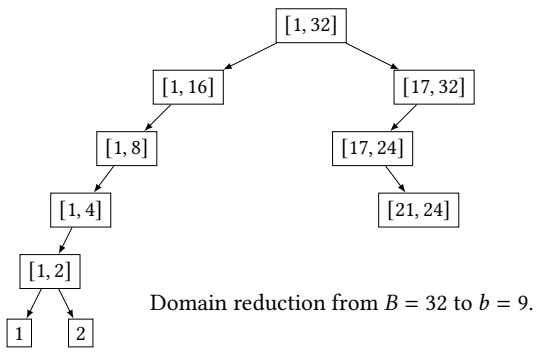


Fig. 6. An example of domain reduction

We realize that the shuffle-DP protocol for heavy hitter detection [33] can almost serve our purpose:

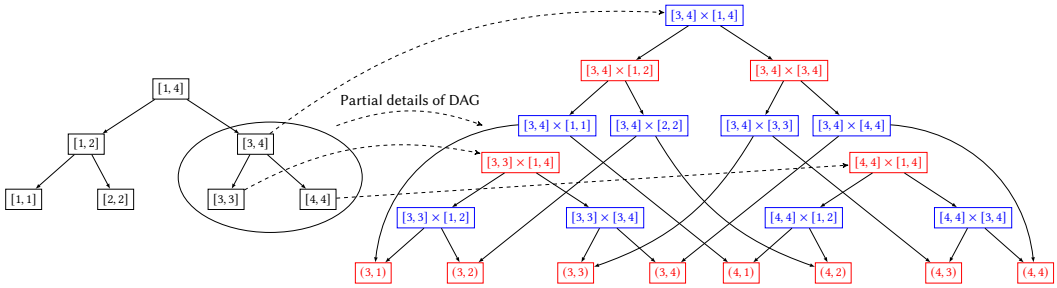


Fig. 7. First-level tree (left); part of second-level tree (right) and noise coefficients assignment (+1 in red / -1 in blue) for 2D range queries of $B = 4$.

Lemma 5.1 ([33]). *Take any $\phi = \Omega(\log^2 n)$. There is an (ε, δ)-shuffle DP protocol that returns $O(n \log n/\phi)$ ranges in the hierarchical strategy, which includes all the heavy ranges with high probability. The protocol sends $O(\log^2 n/\phi)$ messages in expectation per user.*

Note that this protocol may contain some false positives, i.e., light ranges reported as heavy, which means that the heavy subtree we find may be larger than it should be. However, this is not a problem for our domain reduction, as it still satisfies the condition that the atomic ranges are either light or base queries. Moreover, this protocol only returns $O(n \log n/\phi)$ potentially heavy ranges, which means that the reduced domain consists of at most $b = O(n \log n/\phi)$ atomic ranges.

5.2.2 Running RM^2 in the Reduced Domain. Next, we run RM^2 in the reduced domain, i.e., all elements in an atomic range will be considered as the same element. Given a range query in the original domain $[B]$, we approximate it by a range over the atomic ranges, where the approximation error is due to the original range query not aligning perfectly with the atomic ranges. The approximation error is at most 2ϕ, since the atomic ranges are either light or base queries (which must align with the query range).

Finally, we allocate half of the privacy budget to each round. The following corollary summarizes the performance guarantees of our range query shuffle-DP protocol over a large domain.

Corollary 5.2. *Take any $\phi = \Omega(\log^{2.5} n)$. Our range query protocol satisfies (ε, δ)-shuffle-DP. It sends $O(\log^2 n/\phi)$ messages in expectation per user in the first round, and $1 + O(\log^4 n/\phi)$ messages in the second round. The maximum error for all range queries is $O(\phi)$.*

Proof. The message counts follow from Lemma 5.1 and Corollary 5.1 by plugging in $b = O(n \log n/\phi)$. The error is 2ϕ plus the error in Corollary 5.1. Since, $\phi = \Omega(\log^{2.5} n)$, the latter is dominated by $O(\phi)$. □

Remark 1. The parameter ϕ can be used to control the error-message trade-off. In particular, by taking $\phi = \log^5 n$, we obtain an $(1 + o(1))$-message protocol still with a polylogarithmic error (although this can be too large in practice).

Remark 2. More precisely, the communication cost in the second round is actually $1 + O(\log^3 n \cdot b/n)$ where b is $O(n \log n/\phi)$ in the worst case. On a skewed dataset, however, b could be much smaller than this worst-case upper bound, as shown in the experiments.

5.3 Multi-dimensional Range Queries

A simple way to support d-dimensional range queries, i.e., ranges in the form of $[l_1, r_1] \times [l_2, r_2] \times \cdots \times [l_d, r_d]$, is to use the quad-tree (and its natural extension to higher dimensions). RM^2 supports

the quad-tree query strategy in the same way as 1D range trees, as it is just a 2^d-ary tree. In this subsection, we show how RM^2 can be applied on the d-dimensional range tree [7], whose query strategy actually induces a DAG. Furthermore, it has a better error guarantee than the quad-tree.

Assume the domain is $[B]^d$. The d-dimensional range tree [7] partitions the domain using d levels of range trees. We illustrate how this works in 2D; extension to higher dimensions is straightforward. The first level uses a binary range tree on the first dimension as shown on the left of Figure 7. Each node in this first-level tree is then turned into a second-level tree, which is another binary range tree built on the second dimension (Figure 7 shows how this is done on 3 first-level nodes). Each node in the second-level tree thus defines a range in this query strategy $\boldsymbol{A}$, by combining the two ranges in the two dimensions.

This query strategy induces the following DAG: For a query $[l_1, r_1] \times [l_2, r_2] \times \cdots \times [l_d, r_d]$, let j be the last index such that $l_j \neq r_j$. If j does not exist, it is a base query. Otherwise, we define its children in the DAG as $[l_1, r_1] \times \cdots \times [l_j, \frac{l_j+r_j}{2}] \times \cdots \times [l_d, r_d]$ and $[l_1, r_1] \times \cdots \times [\frac{l_j+r_j}{2} + 1, r_j] \times \cdots \times [l_d, r_d]$. We can show that this DAG is bipartite. More precisely, we assign noise coefficients:

$$\boldsymbol{c}_{[l_1,r_1]\times[l_2,r_2]\times\cdots\times[l_d,r_d]} \leftarrow (-1)^{\sum_{i=1}^{m} \log(r_i - l_i + 1)}.$$

To see that this always assigns opposite signs to adjacent nodes in the DAG, observe that for any non-base query $\boldsymbol{q}$, its children differ with it only in coordinate j and by a factor of 2. This DAG has $O(B^d)$ nodes and edges.

Corollary 5.3. *For any $\varepsilon \geq 0, 0 < \delta < 1$, and constant d, RM^2 is an (ε, δ)-shuffle-DP d-dimensional range counting protocol that sends $1 + O\left(\frac{\log(1/\delta)\log^{2d} B}{\varepsilon^2} \cdot \frac{B^d}{n}\right)$ messages per user in expectation. The maximum error is $O\left(\frac{\log^{1.5d} B}{\varepsilon}\sqrt{\log B \log \frac{1}{\delta}}\right)$.*

Proof. For a given range, each dimension is covered by $O(\log B)$ nodes in the tree of the i-th-level. And there are d dimensions, thus $k = O(\log^d B)$ total counters in the tree to cover the range. The ℓ_1 sensitivity is $h = O(\log^d B)$. Similar to 1D error analysis, the error guarantee (i.e., the sum of k Binomial noises with sensitivity h) is

$$2\sqrt{3\ln(2B^{2d}/\beta) \cdot \frac{32k\log(2h/\delta)h^2}{\varepsilon^2}} = O\left(\frac{\log^{1.5d} B}{\varepsilon}\sqrt{\log B \log \frac{1}{\delta}}\right).$$

□

Note that on this query strategy, the straw-man solution sends $(1 + \log B)^d + \tilde{O}(\frac{B^d}{\varepsilon^2 n})$ messages per user.

See Table 3 for detailed comparisons.

5.4 Data Cubes

Let $R(\boldsymbol{T}_1, \boldsymbol{T}_2, \cdots, \boldsymbol{T}_m)$ be a fact table with m attributes $\boldsymbol{T}_i$, $1 \leq i \leq m$. The domain of $\boldsymbol{T}_i$ is denoted as $\texttt{Dom}(\boldsymbol{T}_i)$, and the entire domain is thus $\texttt{Dom}(\boldsymbol{T}_1) \times \texttt{Dom}(\boldsymbol{T}_2) \times \cdots \times \texttt{Dom}(\boldsymbol{T}_m)$. Each user holds a tuple $r \in R$; we use r_i to denote the value of r on attribute $\boldsymbol{T}_i$. The queries in the data cube workload are commonly referred to as *cells*. A cell is specified by $(a_1, a_2, \cdots, a_m)$, where $a_i \in \texttt{Dom}(\boldsymbol{T}_i) \cup \{*\}$. A cell collects the tuples that match the given attributes, i.e.,

$$\text{Cell}(a_1, a_2, \cdots, a_m) = \{r \in R \mid r_i = a_i \vee a_i = *, 1 \leq i \leq m\}.$$

A cell is k-dimensional if exactly k of the a_i's are not $*$. These cells are organized into *cuboids*, where a k-dim cuboid is specified by k attributes $[C] \coloneqq \{\boldsymbol{T}_{i_1}, \boldsymbol{T}_{i_2}, \cdots, \boldsymbol{T}_{i_k}\}$. It consists of all the k-dim cells where these attributes are not $*$. Note that the cells in any cuboid are disjoint and they cover the

Mechanism	Error	#Messages/user
Matrix	$O\left(\frac{\log^{1.5d} B}{\varepsilon}\sqrt{\log B \log \frac{1}{\delta}}\right)$	central-DP
LWY [33]	$O\left(\frac{\sqrt{B^d}}{\varepsilon}\sqrt{d \log B \log \frac{1}{\delta}}\right)$	$1 + \tilde{O}\left(\frac{B^d}{\varepsilon^2 \cdot n}\right)$
GGKPV [23]	$O\left(\frac{\log^{1.5d} B}{\varepsilon}\sqrt{\log B \log \frac{1}{\delta}}\right)$	$\tilde{O}\left(\frac{d^3}{\varepsilon^2}\right)$
Straw-man	$O\left(\frac{\log^{1.5d} B}{\varepsilon}\sqrt{\log B \log \frac{1}{\delta}}\right)$	$(\log B + 1)^d + \tilde{O}\left(\frac{B^d}{\varepsilon^2 \cdot n}\right)$
RM^2	$O\left(\frac{\log^{1.5d} B}{\varepsilon}\sqrt{\log B \log \frac{1}{\delta}}\right)$	$1 + \tilde{O}\left(\frac{B^d}{\varepsilon^2 \cdot n}\right)$

Table 3. Comparisons of different DP mechanisms for the multi-dimensional range queries

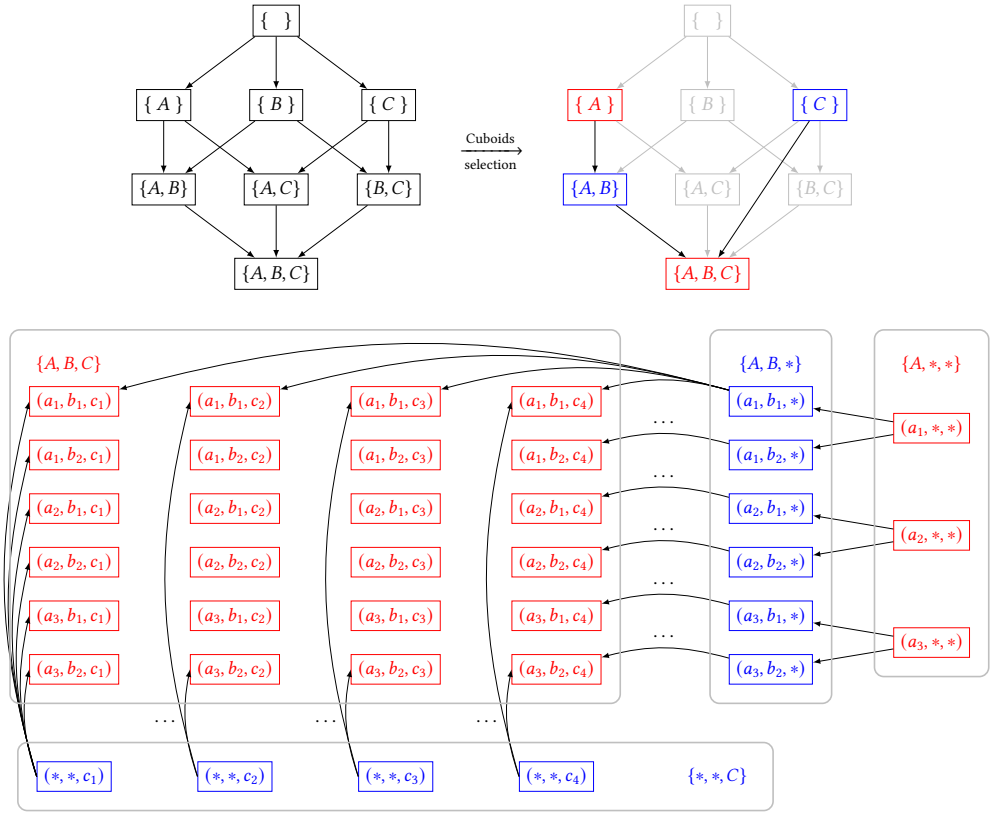


Fig. 8. An example of DP data cube mechanism that finds query strategies (top); the bipartite DAG and noise coefficients assignment (bottom; +1 in red / -1 in blue).

entire domain. There are $\binom{m}{k}$ k-dim cuboids. These cuboids form a lattice structure, as shown in Figure 8. In particular, the 0-dim cuboid consists of one cell that covers the entire domain, and the

m-dim cuboid consists of all the base queries. The k-way marginals are a special case of data cube queries, where each $\mathtt{Dom}(\boldsymbol{T}_i) = \{0, 1\}$, and they only query the cells in those cuboids whose dim is no more than k.

For a data cube workload, [17] has designed a query strategy $\boldsymbol{A}$ with a near-optimal error. Their query strategy chooses a subset of the cuboids, and takes all the cells in these cuboids into $\boldsymbol{A}$. The m-dim cuboid is always chosen, so all base cells are in $\boldsymbol{A}$. Note that since the cells in each cuboid completely cover the domain, $\|\boldsymbol{A}\|_1$ equals the number of chosen cuboids.

To apply RM^2, we need to construct a DAG corresponding to $\boldsymbol{A}$. We proceed in two steps. We first construct an inverse tree over the chosen cuboids. It is an inverse tree in the sense that every node has at most one child but possibly multiple parents, while the only node with no child is the m-dimensional cuboid. For each chosen cuboid $[C]$ (except the m-dimensional cuboid), we set its (only) child to be cuboid $[C']$, such that $[C']$ is a descendent of $[C]$ in the lattice, and each cell in $[C]$ can be decomposed into the minimum number of cells in $[C']$. Please see the middle figure in Figure 8 as an example, where the colored cuboids are the chosen ones and the solid lines indicate the inverse tree. For example, for the cuboid $\{A\}$, $\{A, B\}$ is always a better choice than $\{A, B, C\}$ as its child. If $\{A, C\}$ were also chosen, then we would pick it as the child of $\{A\}$ if $|\mathtt{Dom}(C)| < |\mathtt{Dom}(B)|$, and $\{A, B\}$ otherwise.

In the second step, we convert the inverse tree into a DAG. We first expand each chosen cuboid into the collection of cells it represents. Then between each parent-child cuboids in the inverse tree, we add edges to the DAG such that each cell is equal to the union of its children. Part of this DAG is shown in Figure 8, where we assume $\mathtt{Dom}(A) = \{a_1, a_2, a_3\}$, $\mathtt{Dom}(B) = \{b_1, b_2\}$, $\mathtt{Dom}(C) = \{c_1, c_2, c_3, c_4\}$. For example, the cell $(a_1, *, *)$ in the cuboid $(A, *, *)$ has two children $(a_1, b_1, *)$ and $(a_1, b_2, *)$ in the cuboid $(A, B, *)$.

We argue that this DAG must be bipartite. First, the inverse tree on the chosen cuboids is bipartite: We can color the bottom node red, and then alternatively color the rest of the nodes. Then the DAG can inherit the colors, which would not create any conflicts, since we only add edges in the DAG between cells in adjacent cuboids in the inverse tree.

By Theorem 4.1 and 4.2, RM^2 preserves the error of the query strategy for the data cube workload [17]. The number of real messages per user is reduced from $\|\boldsymbol{A}\|_1$ to 1. The number of noise messages increases from pQ to $p(Q + E)$, where Q denotes the number of cells in the query strategy and E the number of edges in the DAG. This is why we construct the inverse tree in a way that minimizes E. As verified by our experiments, E is not much larger than Q, and the increase in the noise messages is insignificant compared with the reduction in real messages.

6 Experiments

In this section, we compare our protocol RM^2 with some baseline protocols on four different counting query workloads, using both real and synthetic data. We also include the error of vector form **Gaussian** mechanism under central-DP for comparison. The codes and data are available at https://github.com/hkustDB/RM2.

All experiments are conducted on a Linux server with an Intel Xeon CPU @ 2.20GHz and 256 GB memory. We use the following metrics for the counting queries in each workload: (1) the 95%-percentile of the (additive) errors of all queries, and (2) the average number of messages sent per user. For each workload, we repeat the experiment 30 times and report the average after removing the 10% largest and 10% smallest values.

6.1 1D Range Queries in Small Domains

In the small domain setting, we used three synthetic datasets of size n with different distributions over domain $[B]$:

- Zipfian distribution (Zipf.): $f(x) \propto x^{-\alpha}$ with $\alpha = 1.5$;
- Gaussian distribution (Gaus.): $\mu = B/3, \sigma = 20$;
- Uniform distribution (Unif.).

We also evaluated two real datasets, using their date attribute:

- AOL-user-ct-collection (AOL) [34]: a collection of real-world website accesses with userID and clicking time;
- Netflix-Prize (Netf.) [1]: contains a set of rating scores with rating time given by users to movies.

We tried different ε, n and B and the results are shown in Table 4, 5 and 6 respectively. The experiments show that RM^2 is the only protocol that achieves "almost-a-single-message" communication; the other protocols require at least $\log B$ messages. The message count in RM^2 minus 1 is equal to the number of noise messages sent per user, which is proportional to $\tilde{O}(\frac{B}{\varepsilon^2 n})$. This becomes negligible when ε or n increases, or B decreases. On the other hand, the **straw-man** solution sends $\Omega(\log B)$ messages regardless of the other parameters. For **GGKPV** [23], we can only calculate its message number—this protocol could not finish in 1 day.

In terms of the error, the straw-man solution and RM^2 are similar, which is as expected since they have been shown to have the same output distribution. Both of them have errors 5∼6 times larger than that of the central-DP Gaussian mechanism, due to (1) the constant-factor difference between the Gaussian distribution and the binomial distribution; (2) and the optimality of the vector form of Gaussian mechanism. As with prior work [35], we also observe that choosing a query strategy using a b-ary tree for $b > 2$ lead to better errors and communication costs.

Finally, we see that RM^2 is a data-independent protocol: It performs consistently across different datasets under different distributions, in both error and communication cost.

Synthetic data ($n = 10^7$, $B = 1024$, $\delta = 10^{-14}$)													
Datasets	#Messages per user			95% Error									
	All			Unif.			Gaus.			Zipf.			
ε	2	4	8	2	4	8	2	4	8	2	4	8	
Gaussian	-			64.31	31.30	15.46	62.72	31.81	15.61	62.58	31.14	15.56	
GGKPV	31372	7695	1893	-			-			-			
Straw-man	10.65	10.16	10.04	312.82	162.25	85.24	309.53	159.11	82.47	313.41	159.46	81.28	
RM^2 ($b = 2$)	2.30	1.33	1.09	306.24	158.86	80.87	316.99	156.78	84.65	309.21	163.41	83.44	
RM^2 ($b = 4$)	1.23	1.05	1.02	202.21	98.64	58.54	196.10	99.38	56.84	196.38	98.08	57.92	

Real data												
Datasets	AOL ($n = 1230096$, $B = 720$, $\delta = n^{-2}$)						Netf. ($n = 99553886$, $B = 1798$, $\delta = n^{-2}$)					
	#Messages per user			95% Error			#Messages per user			95% Error		
ε	2	4	8	2	4	8	2	4	8	2	4	8
Gaussian	-			55.30	27.49	14.22	-			72.45	36.84	18.10
GGKPV	27600	6752	1657	-			43067	10587	2608	-		
Straw-man	14.59	11.18	10.32	278.54	142.13	74.47	11.19	11.04	11.01	414.98	199.75	100.93
RM^2 ($b = 2$)	7.47	2.66	1.45	281.94	141.18	75.04	1.34	1.08	1.02	400.38	201.01	101.08
RM^2 ($b = 4$)	2.08	1.29	1.09	173.71	85.56	53.42	1.06	1.02	1.01	259.08	131.76	75.03

Table 4. Comparisons among small domain 1D range counting protocols varying privacy parameter ε

Datasets n	#Messages per user All 10^6	All 10^7	All 10^8	95% Error Unif. 10^6	Unif. 10^7	Unif. 10^8	Gaus. 10^6	Gaus. 10^7	Gaus. 10^8	Zipf. 10^6	Zipf. 10^7	Zipf. 10^8
Gaussian	-			28.97	31.30	32.76	29.22	31.81	33.82	29.01	31.14	33.27
GGKPV	6659	7695	8731	-			-			-		
Straw-man	11.43	10.16	10.02	144.57	162.25	176.07	152.25	159.11	178.85	147.67	159.46	174.14
RM^2	3.86	1.33	1.04	143.57	158.86	173.65	147.81	156.78	175.49	147.05	163.41	179.36

Table 5. Comparisons among small domain 1D range counting protocols varying number of users n ($B = 1024$, $\varepsilon = 4$, $\delta = n^{-2}$)

Datasets B	#Messages per user All 512	All 1024	All 2048	95% Error Unif. 512	Unif. 1024	Unif. 2048	Gaus. 512	Gaus. 1024	Gaus. 2048	Zipf. 512	Zipf. 1024	Zipf. 2048
Gaussian	-			28.02	31.30	35.34	27.95	31.81	35.59	27.99	31.14	35.31
GGKPV	6215	7695	9335	-			-			-		
Straw-man	9.06	10.16	11.40	133.43	162.25	184.31	133.51	159.11	185.92	134.02	159.46	180.96
RM^2	1.13	1.33	1.80	134.56	158.86	183.83	130.92	156.78	185.32	133.11	163.41	186.35

Table 6. Comparisons among small domain 1D range counting protocols varying domain size B ($n = 10^7$, $\varepsilon = 4$, $\delta = 10^{-14}$)

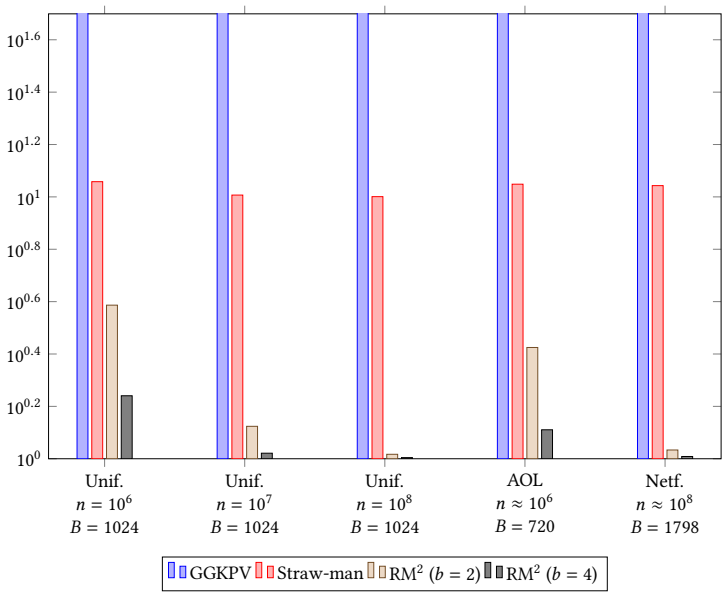


Fig. 9. Comparisons of average #messages/user among different datasets ($\varepsilon = 4, \delta = n^{-2}$)

Datasets	#Messages per user				95% Error			
	Unif.	Zipf.	SZipf.	IPNet.	Unif.	Zipf.	SZipf.	IPNet.
Gaussian	-				47.81	47.79	47.55	51.11
GGKPV	11921			13710	-			
Straw-man	32.13			34.54	1053.39	1053.74	1054.59	1129.67
DR + RM^2	6.05	2.86	3.96	6.39	1154.14	33.67	1146.46	952.39

Datasets	Unif.	Zipf.	SZipf.	IPNet.
#Msgs/user in DR	1.72			1.97
#Msgs/user in RM^2	4.33	1.14	2.24	4.42
Reduced domain b	11300	950	4856	14390
Domain reduction error	1126	0	1136	940
Estimation error	178.70	35.18	153.51	183.86

Table 7. Comparisons among large domain 1D range counting protocols and the details of our protocol ($\varepsilon = 10$, $\delta = n^{-2}$)

Datasets	2D range queries ($n = 10^8$, $B = 32$)						data cube queries: Unif. ($n = 10^7$, $\mathcal{X} = [8,8,4,4]$)						data cube queries: Census ($n \approx 2.5 * 10^6$, $\mathcal{X} = [10,7,4,3]$)					
	#Messages / user			95% Error			#Messages / user			95% Error			#Messages / user			95% Error		
ε	2	4	8	2	4	8	2	4	8	2	4	8	2	4	8	2	4	8
Gaussian	-			179.73	91.53	45.02	-			26.75	13.48	6.88	-			24.86	12.48	6.20
GGKPV	475131	116789	28719	-			Not supported											
Straw-man	38.11	36.52	36.13	1563.54	788.20	394.83	4.06	4.02	4.01	73.55	39.26	24.18	4.20	4.05	4.02	68.49	36.85	22.33
RM^2	6.25	2.31	1.32	1572.62	790.79	387.83	1.17	1.05	1.01	74.11	39.53	24.53	1.54	1.15	1.05	69.89	36.84	22.74

Table 8. Comparisons among data cube releasing protocols varying privacy parameter ε ($\delta = n^{-2}$)

6.2 1D Range Queries in Large Domains

In the large domain, our protocol contains two sub-protocols: domain reduction protocol (DR), where we set the truncation threshold $\phi = 6000$, and 1D range counting protocol (RM^2). We set their privacy budgets to $\varepsilon_1 = 4$ and $\varepsilon_2 = 6$ respectively. Straw-man consists of $(1 + \log B)$ FE protocols [33] which sends $1 + o(1)$ messages per user and gets $O(\log^2 n)$ error in each level. We evaluated three synthetic dataset and one real dataset:

- Unif.: $n = 10^7$ numbers sampled from $B = 2^{30}$ uniformly;
- Zipf.: $n = 10^7$ numbers sampled from $B = 2^{30}$ following Zipfian distribution with $\alpha = 1.5$, which is a skewed dataset;
- Shuffled Zipfian (SZipf.): apply a bijection mapping on the Zipf. dataset, which is a sparse dataset;
- IP-Network-Traffic-Flows (IPNet.) [38]: a collection of IP flows generated by network devices, including source and destination IP addresses. We collect all IP addresses and gets $n = 12564270$, $B = 2^{32}$. This is also a sparse dataset.

The experimental results are shown in Table 7. The theoretical protocol GGKPV takes more than 10000 messages per user and the straw-man solution takes roughly $\log B$ messages. Our protocol reduces the messages to no more than 7 in total, where DR is data-independent and RM^2 relies on the reduced domain size b. The reduced domain size is roughly $n \log B/\phi$ for uniform and sparse datasets, but is much smaller for a skewed dataset.

Since straw-man chooses a $1 + o(1)$ protocol in each level, resulting in a $\omega(1)$ error gap with the optimal central-DP, its error is roughly 22 times as large as Gaussian mechanism. The error of our protocol consists of two parts: domain reduction, which has error $O(\phi)$, and that of RM^2, which is roughly 3 times as that of the Gaussian mechanism. In Table 7 we show them separately. We see that most of the error comes from the domain reduction, which is necessary to reduce the communication cost.

6.3 2D Range Queries and Data Cube Queries

We also evaluate 2D range queries and data cube queries in Table 8. In 2D range queries, we try a uniformly generated synthetic dataset of size $n = 10^8$ and input domain $\mathcal{X} = [32]^2$. In data cube queries, we use a uniformly generated synthetic dataset of size $n = 10^7$ and input domain $\mathcal{X} = [8, 8, 4, 4]$. We also try a real U.S. Census dataset [13] (Census). It contain $n = 2458285$ records with 4 categorical dimensions: Class of Worker (domain size 10), Place of Birth (7), School Enrollment (4) and Years of Military Service (3), $\mathcal{X} = [10, 7, 4, 3]$.

The results in 2D range queries and data cube queries are similar to the 1D small domain range queries: Our protocol improves the communication costs such that each user only sends $1 + o(1)$ messages for all queries, while other protocols necessitate sending at least $(1 + \log B)^2$ messages in 2D range queries and $\|\boldsymbol{A}\|_1$ messages in data cube queries. Our error is the same as that in straw-man solution and 3~9 times as large as the Gaussian mechanism's error, which matches our expectation.

7 Conclusions

In this paper, we studied the problem of answering counting queries accurately and efficiently in the shuffle-DP model. We have designed RM^2 to reduce the communication cost while preserving the accuracy of the mechanism, and have demonstrated its versatility through a number of common query workload

There are some interesting open questions for future research:

(1) Is RM^2 applicable to other mechanisms in shuffle-DP, such as the (discrete) Laplace mechanism [24, 25] which has a better accuracy in the bit counting problem?
(2) For large domain 1D range queries, our two-round protocol is not ideal. Improving to a one-round protocol in constant or $1 + o(1)$ messages and $\tilde{O}(1)$ error could be interesting.
(3) Currently RM^2 only supports 0-1 entries in the query strategy. Supporting both positive and negative entries would make it more widely applicable.

Acknowledgments

This work has been supported by HKRGC under grants 16205422, 16204223, and 16203924; and by NTU-NAP startup grant 024584-00001. We also thank the anonymous reviewers for valuable suggestions on improving the paper.

References

[1] 2016. Netflix Prize dataset. https://archive.org/download/nf_prize_dataset.tar. Accessed: 2016-02-04.
[2] John M. Abowd. 2018. The U.S. Census Bureau Adopts Differential Privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 2867. https://doi.org/10.1145/3219819.3226070
[3] Naman Agarwal, Ananda Theertha Suresh, Felix Yu, Sanjiv Kumar, and H. Brendan McMahan. 2018. cpSGD: communication-efficient and differentially-private distributed SGD. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 7575–7586.
[4] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. 2019. The Privacy Blanket of the Shuffle Model. In *CRYPTO*.
[5] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. 2020. Private Summation in the Multi-Message Shuffle Model. In *ACM SIGSAC Conference on Computer and Communications Security*.
[6] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. 2007. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the Twenty-Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (Beijing, China) *(PODS '07)*. Association for Computing Machinery, New York, NY, USA, 273–282. https://doi.org/10.1145/1265530.1265569
[7] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. 2008. *Computational Geometry: Algorithms and Applications* (3rd ed. ed.). Springer-Verlag TELOS, Santa Clara, CA, USA.

[8] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. 2017. Prochlo: Strong Privacy for Analytics in the Crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 441–459.

[9] David L. Chaum. 1981. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM* 24 (1981), 84–90.

[10] Lijie Chen, Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. 2021. On Distributed Differential Privacy and Counting Distinct Elements. In *ITCS*. 56:1–56:18.

[11] Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. 2015. Differentially Private High-Dimensional Data Publication via Sampling-Based Inference. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Sydney, NSW, Australia) *(KDD '15)*. Association for Computing Machinery, New York, NY, USA, 129–138. https://doi.org/10.1145/2783258.2783379

[12] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. 2019. Distributed Differential Privacy via Shuffling. In *Advances in Cryptology – EUROCRYPT 2019*, Yuval Ishai and Vincent Rijmen (Eds.). Springer International Publishing, Cham, 375–403.

[13] Meek Chris, Thiesson Bo, and Heckerman David. 2001. US Census Data (1990). UCI Machine Learning Repository. https://doi.org/10.24432/C5VP42

[14] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. 2019. Answering range queries under local differential privacy. *Proc. VLDB Endow.* 12, 10 (jun 2019), 1126–1138. https://doi.org/10.14778/3339490.3339496

[15] G. Danezis, R. Dingledine, and N. Mathewson. 2003. Mixminion: design of a type III anonymous remailer protocol. In *Symposium on Security and Privacy*. 2–15.

[16] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting telemetry data privately. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 3574–3583.

[17] Bolin Ding, Marianne Winslett, Jiawei Han, and Zhenhui Li. 2011. Differentially private data cubes: optimizing noise sources and consistency. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data* (Athens, Greece) *(SIGMOD '11)*. Association for Computing Machinery, New York, NY, USA, 217–228. https://doi.org/10.1145/1989323.1989347

[18] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium (USENIX Security 04)*.

[19] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *Advances in Cryptology - EUROCRYPT 2006*, Serge Vaudenay (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 486–503.

[20] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[21] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (Scottsdale, Arizona, USA) *(CCS '14)*. Association for Computing Machinery, New York, NY, USA, 1054–1067. https://doi.org/10.1145/2660267.2660348

[22] Simson Garfinkel. 2022. Differential Privacy and the 2020 US Census. *MIT Case Studies in Social and Ethical Responsibilities of Computing* Winter 2022 (jan 24 2022). https://mit-serc.pubpub.org/pub/differential-privacy-2020-us-census.

[23] Badih Ghazi, Noah Golowich, Ravi Kumar, Rasmus Pagh, and Ameya Velingker. 2021. On the Power of Multiple Anonymous Messages: Frequency Estimation and ;Selection in the Shuffle Model of ;Differential Privacy. In *Advances in Cryptology – EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part III* (Zagreb, Croatia). Springer-Verlag, Berlin, Heidelberg, 463–488. https://doi.org/10.1007/978-3-030-77883-5_16

[24] Badih Ghazi, Pritish Kamath, Ravi Kumar, Pasin Manurangsi, and Kewen Wu. 2022. On differentially private counting on trees. *arXiv preprint arXiv:2212.11967* (2022).

[25] Badih Ghazi, Ravi Kumar, Pasin Manurangsi, and Rasmus Pagh. 2020. Private counting from anonymous messages: near-optimal accuracy with vanishing communication overhead. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 328, 10 pages.

[26] Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Rasmus Pagh, and Amer Sinha. 2021. Differentially Private Aggregation in the Shuffle Model: Almost Central Accuracy in Almost a Single Message. In *International Conference on Machine Learning*.

[27] Moritz Hardt and Guy N. Rothblum. 2010. A Multiplicative Weights Mechanism for Privacy-Preserving Data Analysis. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. 61–70. https://doi.org/10.1109/FOCS.2010.85

[28] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2010. Boosting the accuracy of differentially private histograms through consistency. *Proc. VLDB Endow.* 3, 1–2 (sep 2010), 1021–1032. https://doi.org/10.14778/1920841.

1920970

[29] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. 2006. Cryptography from Anonymity. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. 239–248. https://doi.org/10.1109/FOCS.2006.25

[30] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. 2010. Optimizing linear counting queries under differential privacy. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '10)*. Association for Computing Machinery, New York, NY, USA, 123–134. https://doi.org/10.1145/1807085.1807104

[31] Chao Li and Gerome Miklau. 2012. An adaptive mechanism for accurate query answering under differential privacy. *Proc. VLDB Endow.* 5, 6 (feb 2012), 514–525. https://doi.org/10.14778/2168651.2168653

[32] Chao Li, Gerome Miklau, Michael Hay, Andrew Mcgregor, and Vibhor Rastogi. 2015. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB Journal* 24, 6 (dec 2015), 757–781. https://doi.org/10.1007/s00778-015-0398-x

[33] Qiyao Luo, Yilei Wang, and Ke Yi. 2022. Frequency Estimation in the Shuffle Model with Almost a Single Message. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 2219–2232. https://doi.org/10.1145/3548606.3560608

[34] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *Proceedings of the 1st international conference on Scalable information systems*.

[35] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Understanding hierarchical methods for differentially private histograms. *Proc. VLDB Endow.* 6, 14 (sep 2013), 1954–1965. https://doi.org/10.14778/2556549.2556576

[36] M.G. Reed, P.F. Syverson, and D.M. Goldschlag. 1998. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications* (1998), 482–494.

[37] Michael K. Reiter and Aviel D. Rubin. 1998. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security* (1998), 66–92.

[38] Juan Rojas Meléndez, Adrian Pekar, Alvaro Rendón Gallón, and Juan Corrales. 2020. Smart User Consumption Profiling: Incremental Learning-Based OTT Service Degradation. *IEEE Access* 8 (11 2020). https://doi.org/10.1109/ACCESS.2020.3037971

[39] Abhradeep Guha Thakurta, Andrew H Vyrros, Umesh S Vaishampayan, Gaurav Kapoor, Julien Freudiger, Vivek Rangarajan Sridhar, and Doug Davidson. 2017. Learning new words. US Patent 9,594,741.

[40] Abhradeep Guha Thakurta, Andrew H Vyrros, Umesh S Vaishampayan, Gaurav Kapoor, Julien Freudinger, Vipul Ved Prakash, Arnaud Legendre, and Steven Duplinsky. 2017. Emoji frequency detection and deep link frequency. US Patent 9,705,908.

[41] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. 2011. Differential Privacy via Wavelet Transforms. *IEEE Trans. on Knowl. and Data Eng.* 23, 8 (aug 2011), 1200–1214. https://doi.org/10.1109/TKDE.2010.247

[42] Jianyu Yang, Tianhao Wang, Ninghui Li, Xiang Cheng, and Sen Su. 2020. Answering multi-dimensional range queries under local differential privacy. *Proc. VLDB Endow.* 14, 3 (nov 2020), 378–390. https://doi.org/10.14778/3430915.3430927

[43] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. PrivBayes: Private Data Release via Bayesian Networks. *ACM Trans. Database Syst.* 42, 4, Article 25 (oct 2017), 41 pages. https://doi.org/10.1145/3134428

Received October 2024; revised January 2025; accepted February 2025