Orthogonal Mechanism for Answering Batch Queries with Differential Privacy

Dong Huang
Institute for Infocomm Research, Singapore
huangd@i2r.a-star.edu.sg

Shuguo Han
Institute for Infocomm Research, Singapore
shan@i2r.a-star.edu.sg

Xiaoli Li
Institute for Infocomm Research, Singapore
xlli@i2r.a-star.edu.sg

Philip S. Yu
University of Illinois at Chicago, USA
psyu@cs.uic.edu

ABSTRACT

Differential privacy has recently become very promising in achieving data privacy guarantee. Typically, one can achieve ϵ-differential privacy by adding noise based on Laplace distribution to a query result. To reduce the noise magnitude for higher accuracy, various techniques have been proposed. They generally require high computational complexity, making them inapplicable to large-scale datasets. In this paper, we propose a novel orthogonal mechanism (OM) to represent a query set Q with a linear combination of a new query set $\tilde{Q}$, where $\tilde{Q}$ consists of orthogonal query sets and is derived by exploiting the correlations between queries in Q. As a result of orthogonality of the derived queries, the proposed technique not only greatly reduces computational complexity, but also achieves better accuracy than the existing mechanisms. Extensive experimental results demonstrate the effectiveness and efficiency of the proposed technique.

Categories and Subject Descriptors

H.2.0 [**Database Management**]: General: Security, Privacy Protection; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithm, Security

Keywords

Differential Privacy, Data Mining, Algorithms

1. INTRODUCTION

Privacy has been identified as a crucial issue in many data mining applications [7, 23, 26]. While various privacy-preserving techniques have been proposed to achieve privacy protection for data publishing and data analysis [1, 2, 18, 28], they have been proved to become vulnerable under certain conditions and thus cannot provide the privacy guarantee as claimed. For instance, the well-known *k*-anonymity [28] does not fully protect data privacy, even assuming adversary's knowledge is only limited to quasi-identifiers such as age and ZIP code [4]. On the other hand, there has been a tremendous growth in data collection of personal demographic information for real-world applications as they generally contain very useful information for knowledge discovery. For example, Amazon and YouTube collect users' viewing and buying records for their product recommendations. However, those data contain sensitive personal information, which is a threat to privacy [19].

Recently, differential privacy (DP) [10] has been proposed to achieve privacy guarantee for sensitive data and is robust to the change of individual data records. Due its distinct advantages, it has been widely applied in various applications [5, 12, 21, 24]. To achieve differential privacy, Dwork [11] proposed the Laplace Mechanism (LM) to add noise to query results based on Laplace distribution. If LM processes each query independently to answer a batch of queries, then it produces the noise variance $\Theta(m)$, where m is the number of queries. As such, when query sets become very large, the method fails to provide useful results as the noise added could make the real data negligible. Recent results have shown that smaller variances can be obtained if one exploits the correlations between different queries [8, 17, 31], but these algorithms have very high computational complexity by applying expensive matrix operations, making them inapplicable to large-scale datasets.

In this paper, we focus on how to efficiently and accurately answer a batch of counting queries – a very common task in data mining [12, 20] and statistical analysis [14], while preserving data privacy. Specifically, given a query set $Q = \{q_1, \ldots, q_m\}$, we compute a refined query set $\tilde{Q} = \{\tilde{q}_1, \ldots, \tilde{q}_r\}$ such that $\forall q_i \in Q$ $(i = 1, 2, \ldots, m)$ can be linearly represented by $\tilde{q}_1, \ldots, \tilde{q}_r$, where the error under $\tilde{Q}$ is less than that under Q.

We now use the following example to illustrate the idea. Consider census record data about individuals shown in Table 1. Suppose we receive a query set $Q = \{q_1, \ldots, q_6\}$ from users, where
q_1: the count of {Race:*White*, Salary:$>$*50K*},
q_2: the count of {Race:*White*, Salary:$\leqslant$*50K*},
q_3: the count of {Race:*Black*, Salary:$>$*50K*},
q_4: the count of {Race:*Black*, Salary:$\leqslant$*50K*},

Table 1: Census Record Data.

Sex	Race	State	Salary
M	*White*	*NY*	$\leqslant$ *50K*
M	*Other*	*NY*	>*50K*
F	*White*	*WA*	>*50K*
F	*Black*	*NY*	>*50K*
..	..	..	..

q_5: the count of {Race:*White* and *Black*, Salary:>*50K*},
q_6: the count of {Race:*White* and *Black*, Salary:$\leqslant$*50K*}.
We decompose each of the queries in Q as follows:
$q_1 = C_{MWH} + C_{FWH}, q_2 = C_{MWL} + C_{FWL}, q_3 = C_{MBH} + C_{FBH}$,
$q_4 = C_{MBL} + C_{FBL}$, $q_5 = C_{MWH} + C_{MBH} + C_{FWH} + C_{FBH}$ and,
$q_6 = C_{MWL} + C_{MBL} + C_{FWL} + C_{FBL}$, where C_{MWH}, C_{MWL}, C_{MBH}, C_{MBL}, C_{FWH}, C_{FWL},C_{FBH}, C_{FBL} are the count of {Sex:*M*, Race:*White*, Salary:>*50K*}, {Sex:*M*, Race:*White*, Salary:$\leqslant$*50K*}, {Sex:*M*, Race:*Black*, Salary:>*50K*}, {Sex:*M*, Race:*Black*, Salary:$\leqslant$*50K*}, {Sex:*F*, Race:*White*, Salary:>*50K*}, {Sex:*F*, Race:*White*, Salary:$\leqslant$*50K*}, {Sex:*F*, Race:*Black*, Salary:>*50K*}, {Sex:*F*, Race:*Black*, Salary:$\leqslant$*50K*}, respectively.

Note that these queries are highly correlated. For instance, $q_5 = q_1 + q_3$ and $q_6 = q_2 + q_4$. Let $S(Q)$ be the sensitivity of the query set Q and ϵ be the privacy budget assigned to protect the dataset. When we directly process Q using LM, the noise variance of one query is $2S^2(Q)/\epsilon^2$. As the sensitivity of a query in Q in the above example is 2 according to Definition 1 described in Section 3, processing Q directly incurs a noise variance of $2 * 2^2/\epsilon^2 = 8/\epsilon^2$ for each query, leading to a bigger noise variance given a small ϵ. In addition, such method also results in another concern: inconsistency. The noisy answers computed by applying LM independently to each query may violate the constraints $q_5 = q_1 + q_3$ and $q_6 = q_2 + q_4$, which further deteriorates the accuracy of data analysis [14, 31].

In contrast, if we process a subset of Q, e.g., $\tilde{Q} = \{q_1, \ldots, q_4\}$ and subsequently use $\tilde{Q}$ to derive other queries , namely q_5 and q_6, then the noise variance can be largely reduced. For example, the noise variance for each query in $\tilde{Q}$ is $2/\epsilon^2$ as one count change only affects one query in $\tilde{Q}$. Clearly, exploring the correlation of the queries in $\tilde{Q}$ produces much less noise variance. Inspired by this, we design a novel algorithm to achieve less noise variance instead of independently adding noise to each query in Q.

In summary, we propose a novel orthogonal mechanism (OM) for effectively answering a batch of queries while achieving ϵ-differential privacy. To highly reduce the correlation of queries, the proposed mechanism first decomposes the original query set Q into *orthogonal query subsets*. Then, we derive an independent query set $\tilde{Q}$ by combining corresponding queries from the orthogonal query subsets. Using the orthogonal property, it requires less noise to achieve ϵ-differential privacy and obtains significantly better efficiency without employing matrix operations. The contributions of this paper are summarized as follows:

- **Reduce the errors**. The example mentioned above has shown that the sensitivity has a significant impact on the error. The proposed OM constructs a new query set $\tilde{Q}$ to represent the original query set Q by exploring the correlation of Q. Experimental results demonstrate that OM outperforms existing works in terms of accuracy.

- **Reduce the computational complexity**. The computational complexity of the proposed mechanism is significantly lower than existing mechanisms. We exploited the correlation between original queries by constructing independent and orthogonal queries, where the number of independent queries is minimized and we do not apply any expensive matrix operations as existing work does [17, 31].

2. RELATED WORK

Differential privacy [10], as a rigorous privacy concept, can provide very strong privacy guarantee to protect sensitive data. Following the idea, many researchers have proposed algorithms to protect privacy during data analysis [12, 19, 20, 30, 32]. To answer a large number of highly correlated queries, the noise magnitude introduced by Laplace Mechanism is linear to the number of queries, which fails to provide accurate results due to large noise magnitude.

To address this issue, Hay *et al.* [14] proposed a mechanism to improve the accuracy of histogram queries by considering consistency. The proposed mechanism first uses the Laplace Mechanism to inject noise to the queries, and then explores the consistency of histogram by reducing and rearranging the noisy answers to maintain the original consistency. Although this mechanism addresses the consistency issue effectively, it does not reduce the noise magnitude to answer a batch of queries due to the error complexity of $O(l \log^3 m)$, where l and m represent the number of distinct values of queries results and the number of queries respectively.

A K-norm mechanism by Hardt and Talwar [13] achieves l_2-error $\Theta(\min\{m\sqrt{m}/\epsilon, m\sqrt{\log(m/n)/\epsilon}\})$ for m random linear queries with sensitivity 1. It is an instantiation of the exponential mechanism [21] with the score function defined by the norm $\|\cdot\|_K$. Rastogi and Nath [27] proposed mechanism for distributed time-series data based on Fourier Perturbation Algorithm (FPA_k), which offers good practical utility without any trusted server. To answer m queries, FPA_k improves the expected errors from $\Theta(m)$ to $\Theta(k)$, where $k \ll m$ is the number of Fourier coefficients that is used to reconstruct all the m query answers. However, the value of k affects the approximation accuracy. A careful selection of k is required in order to achieve a good trade-off between noise magnitude and the approximation accuracy.

Li *et al.* [17] proposed the matrix mechanism (MM) to reduce noise magnitude for linear counting queries. An algorithm to approximate optimal query strategies for the matrix mechanism was proposed with the complexity of at least $O(\lambda^3 n^2)$, where λ and n are the number of new queries to be derived and the domain size of query set respectively. The optimal solution is relatively slow to be found due to its high computational complexity. In [29], a mechanism for answering range-count queries was proposed based on wavelet transforms. Instead of injecting noise directly into the queries, the mechanism first applies a wavelet transform on the frequency matrix to derive a new matrix and then adds polylogarithmic noise to the new matrix to achieve ϵ-differential privacy. Finally, the noisy queries can be derived by inverting wavelet transform. The mechanism obtains the variance of the noise in the query result bounded by a polylogarithm of m.

Table 2: List of common notations.

Notation	Description
$\mathbf{D}$, $\mathbf{d}_i$	Database instance, and a tuple (data record) of database
d, t	The number of attributes and the number of tuples
$\mathcal{A}$	The set of all features in $\mathbf{D}$
$\mathcal{B}$	The selected subset of features for query
$\mathcal{D}$	New representative database of $\mathbf{D}$ based on $\mathcal{B}$
n	The domain size of the chosen subset $\mathcal{B}$, i.e., $\|dom(\mathcal{B})\|$
Q, m, W	Query set, its cardinality and its workload matrix
$Q(\mathcal{D})$	Query set on the database instance $\mathcal{D}$
$q(\mathcal{D})$	A query on the database instance $\mathcal{D}$
$S(Q)$	Sensitivity of query set Q

Yuan *et al.* [31] further proposed a low-rank mechanism based on a low rank approximation of the workload matrix. The basic idea is to randomly transform the original queries into new queries and then applies the low rank approximation technique. However, the selection of new queries has significant impact on the accuracy [31], hence a careful construction of the new queries is required.

3. BACKGROUND

In this section, we briefly introduce differential privacy, Laplace Mechanism (LM) and two important theorems to be used in the paper.

We consider a database $\mathbf{D}$ with d attributes $\mathcal{A} = \{A_1, \ldots, A_d\}$ and t tuples. Let $dom(\mathcal{A})$ be the cross-product of the domains of attributes in $\mathcal{A}$ [17, 31]. One chooses a subset of attributes $\mathcal{B} \subseteq \mathcal{A}$ for queries where query set is $Q = \{q_1, \ldots, q_m\}$. For simplicity, we assume $dom(\mathcal{B}) = \{x_1, x_2, \ldots, x_i, \ldots, x_n\}$ where x_i represents one element in the chosen query domain of $\mathcal{B}$. For instance, x_i could represent {Sex:*M*, Race:*White*, Salary:*>50K*} in our running example. Let c_i be the number of individual records which are equivalent to x_i in $\mathbf{D}$. Following existing work [31], we use x_i to replace c_i as well in the below equations where necessary. In such way, we can represent a relational database $\mathbf{D}$ as $\mathcal{D} = \{x_1, \ldots x_i, \ldots, x_n\}$. Query set $Q = \{q_1, \ldots, q_m\}$ of cardinality m is a mapping from the database domain to real numbers, i.e., $Q : \mathbb{D} \rightarrow \mathbb{R}^m$. Without loss of generality, we assume $m \leq n$. Table 2 provides a list of frequent notations used in the paper.

Different from traditional privacy algorithms, differential privacy ensures the privacy guarantee by comparing the aggregate information of two neighbouring databases that are used in the unbounded differential privacy [10, 16]:

DEFINITION 1 ([10]). *Neighbouring Database: Two databases $\mathbf{D}_1$ and $\mathbf{D}_2$ are neighbouring databases if they differ on exactly one data record with or without the data record, e.g.,*

$$\begin{aligned} \mathbf{D}_1 &= \{\mathbf{d}_1, \ldots, \mathbf{d}_{i-1}, \mathbf{d}_i, \mathbf{d}_{i+1}, \ldots, \mathbf{d}_t\} \text{ and} \\ \mathbf{D}_2 &= \{\mathbf{d}_1, \ldots, \mathbf{d}_{i-1}, \quad \mathbf{d}_{i+1}, \ldots, \mathbf{d}_t\} \end{aligned}$$

where $\mathbf{d}_i$ represents a data record or tuple in database $\mathbf{D}_1$. When two neighboring databases $\mathbf{D}_1$ and $\mathbf{D}_2$ differ in exactly one individual record, the corresponding derived count databases $\mathcal{D}_1$ and $\mathcal{D}_2$ differ in one component by exactly one, meaning

$$|x_i - x_i'| = 1 \text{ where } x_i \in \mathcal{D}_1, x_i' \in \mathcal{D}_2$$

Before discussing the Laplace Mechanism, we first present the definition of ϵ-differential privacy proposed by Dwork [11].

DEFINITION 2 ([11]). *ϵ-Differential Privacy: Given $\epsilon > 0$, a randomized algorithm $\mathcal{K} : \mathbb{D} \rightarrow \mathbb{R}^l$ is said to satisfy ϵ-differential privacy, if for any two neighbouring databases $\mathbf{D}_1$ and $\mathbf{D}_2$ and for any subset of outputs $S \subseteq Range(\mathcal{K})$, the following condition holds:*

$$\frac{Pr(\mathcal{K}(\mathbf{D}_1) \in S)}{Pr(\mathcal{K}(\mathbf{D}_2) \in S)} \leq \exp(\epsilon) \tag{1}$$

where the probability is taken over the randomness of $\mathcal{K}$ and ϵ is the privacy budget used to protect data privacy.

From Eq. (1), when ϵ increases, we get a higher accuracy on the statistics of a database but less privacy is protected. In general, ϵ is set to a small value (e.g., $\epsilon \leq 1$) for strong privacy protection.

Similar to traditional perturbation methods, we can achieve differential privacy by adding noise to the result of a query, where its noise magnitude depends on the sensitivity of the query function which is defined as follows [31]:

DEFINITION 3 ([17]). *Query Matrix Sensitivity: Given a query matrix $Q = \{q_1, \ldots, q_m\} : \mathbb{D} \rightarrow \mathbb{R}^m$, the sensitivity of Q is defined as*

$$S(Q) = \max_{\|\mathcal{D}_1 - \mathcal{D}_2\|_1 = 1} \|W\mathcal{D}_1 - W\mathcal{D}_2\|_1 = \max_j \sum_{i=1}^{n} |\omega_{ij}|, \tag{2}$$

where W is the workload of query set Q and ω_{ij} is the element in the ith row and jth column of the matrix W, representing the jth coefficient for query q_i on element x_j. Thus the sensitivity of a query matrix is the maximum L_1 norm of a column.

PROPOSITION 1 ([17, 31]). *Laplace Mechanism: Given a query set $Q : \mathbb{D} \rightarrow \mathbb{R}^m$ over domain $\mathbb{D}$ and assuming its sensitivity is $S(Q)$, a randomized mechanism $\mathcal{K}$ is said to provide ϵ-differential privacy if*

$$\mathcal{K}(Q, \mathcal{D}) = W\mathcal{D} + Lap(S(Q)/\epsilon)^m$$

where $Lap(a)^m \in \mathbb{R}^m$ denotes a column vector consisting of independent samples from Laplace distribution with scale a.

Dinur and Nissim [9] proved that the privacy may be compromised if the adversary is allowed to execute all possible queries where the noise magnitude is $o(t)$, where t is the number of individuals in the database. If the adversary is only allowed to execute polynomial-bounded queries, it is required to inject the noise of magnitude $\Omega(\sqrt{t})$ to protect the privacy. Compared with traditional algorithms, ϵ-differential privacy requires less noise, as it is independent with the cardinality of the database to achieve privacy guarantee. For multiple queries, we have the following two important theorems.

THEOREM 1 ([22]). *Sequential Composition: Given a sequence of queries $q_1, q_2, \ldots, q_m$, the randomized mechanism $\mathcal{K}$ with noise distribution $Lap(\sum_{i=1}^{m} S(q_i)/\epsilon)$ on each query satisfies ϵ-differential privacy.*

As the noise added to each query is proportional to the sensitivity summation of all queries, the mechanism developed by Theorem 1 may fail to provide useful results if m is big.

THEOREM 2 ([22]). *Parallel Composition: Let q_i each provide ϵ-differential privacy in database $\mathcal{D}_i$, where $\mathcal{D}_i$ are disjoint subsets of the input domain $\mathbb{D}$. The sequence of q_i provides ϵ-differential privacy.*

We note that when the query domain is divided into disjoint subsets for queries, the noise added to each query can be independent.

4. THE PROPOSED TECHNIQUE

This section presents the proposed orthogonal mechanism for answering multiple counting queries. Here, we first give the problem definition in Section 4.1, followed by the description and analysis of our protocol in Sections 4.2 and 4.3 respectively.

4.1 Problem Definition

In this work, our objective is to investigate the efficient method to answer multiple queries on the count of individuals in multiple subsets under ϵ-differential privacy framework. Suppose there are m queries (e.g.,$Q = \{q_1, \ldots, q_m\}$) and the m corresponding subsets are $\Omega_1, \ldots, \Omega_m$. Without loss of generality, we assume all related attributes are categorical. Let $dom(\mathcal{B})$ be the cross-product of the domain attributes in $\Omega = \bigcup \Omega_i$. In this case, the multiple queries can be further decomposed as a number of linear counting queries. Specifically, a linear query computes a linear combination of the counts in $\mathcal{D}$, which is derived from original database $\mathbf{D}$.

DEFINITION 4. *Linear Query: Given the count database $\mathcal{D}$, a linear query q is the dot product between the weight vector $w = [\omega_1, \ldots, \omega_n]$ and count database $\mathcal{D}$, i.e.,*

$$q(\mathcal{D}) = w\mathcal{D} = \sum_{i=1}^{n} \omega_i x_i$$

When m number of linear queries $Q = \{q_1, \ldots, q_m\}$ are conducted simultaneously, we can represent Q by a workload matrix $W \in \mathbb{R}^{m \times n}$. Hence, we have the following definition:

DEFINITION 5. *Query Set: Given a count database $\mathcal{D}$ and m linear queries of Q, query set Q is the product of the query matrix $W \in \mathbb{R}^{m \times n}$ and count database $\mathcal{D}$, i.e.,*

$$Q(\mathcal{D}) = W\mathcal{D} = [\sum_{j=1}^{n} \omega_{1j} x_j, \ldots, \sum_{j=1}^{n} \omega_{mj} x_j]^T$$

Given the database $\mathcal{D}$ and a query set $Q = \{q_1, \ldots, q_m\}$ with workload $W \in \mathbb{R}^{m \times n}$, we wish to find the query results $Q(\mathcal{D}) = W\mathcal{D}$. Note that the decomposition of Q has significant impact on the workload matrix W. In this paper, we consider the decomposition such that $dom(x_i) \bigcap dom(x_j) = \phi$ where $\forall i \neq j$. From Proposition 1, we use the Laplace Mechanism to achieve ϵ-differential privacy. It is worthy to mention that the noisy answers derived by this mechanism is unbiased, i.e.,

$$\mathrm{E}\{\mathcal{K}(Q, \mathcal{D})\} = Q(\mathcal{D}) + \mathrm{E}\{Lap(S(Q)/\epsilon)\} = Q(\mathcal{D})$$

as the expectation of Laplace distribution used here is zero. For each query, we have the following property

$$Var(\mathcal{K}(q_i(\mathcal{D}))) = \mathrm{E}\{(\mathcal{K}(q_i(\mathcal{D})) - q_i(\mathcal{D}))^2\},$$

which means the variance of a noisy answer is equivalent to the expected squared error. Therefore, we use the variance of the noisy answers to evaluate the utility of algorithms in this paper.

In addition, the noise magnitude of applying Laplace Mechanism is $\Theta(m)$ according to Theorem 1, leading $Var(\mathcal{K}(Q, \mathcal{D}))$ to be $\Theta(m^2)$. In this case, the noisy answers may fail to provide useful results. For example, a database with d binary features can provide 2^d number of possible different queries, and thus the corresponding variance becomes $Var(\mathcal{K}(Q, \mathcal{D}))$ is $\Theta(2^{2d})$. To address this issue, we will reduce the noise magnitude by transforming the query set Q into a number of orthogonal query sets.

4.2 Orthogonal Mechanism

The proposed Orthogonal Mechanism is achieved by the construction of orthogonal queries from the query set. Before providing details of the algorithm, we formally define orthogonal queries and orthogonal query sets first.

DEFINITION 6. *Orthogonal Queries: For any two queries $q_1(\mathcal{D}_1)$ and $q_2(\mathcal{D}_2)$ on count database $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively, we call q_1 and q_2 are orthogonal queries if $\mathcal{D}_1$ and $\mathcal{D}_2$ are disjoint sets, denoted as $q_1 \perp q_2$.*

For the relationship between two query sets Q_1 and Q_2, we have the following definition:

DEFINITION 7. *Orthogonal Query Sets: Given two query sets Q_1 and Q_2, they are orthogonal if $q_i \perp q_j$ for every $q_i \in Q_1$ and $q_j \in Q_2$, denoted as $Q_1 \perp Q_2$.*

With the above definitions, we have an important proposition as follows:

PROPOSITION 2. *Orthogonal Mechanism: If every pair Q_i and Q_j in a set of the query sets are orthogonal query sets (i.e., $Q_i \perp Q_j$) while each achieves ϵ-differential privacy, then the entire set provides ϵ-differential privacy.*

PROOF. Based on the definition of orthogonal queries, the datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ are disjoint sets. By applying Theorem 2, the proposition can be then approved. □

That is, if we represent the original query set by orthogonal queries, we can decrease the required noise effectively by making them uncorrelated. Inspired by this, we propose the orthogonal mechanism where the details are shown in Algorithm 1.

In Algorithm 1, we first construct a new query set $\tilde{Q}$ with workload $\tilde{W}$ to represent the original query set Q, where the original

Algorithm 1 Orthogonal Mechanism

Input: Database $\mathcal{D}$ and workload W of query set Q
Output: Approximated query result $\mathcal{K}(Q, \mathcal{D})$

1: Construct a new query set $\tilde{Q}$ together with $\tilde{W}$ based on Algorithm 2 and a matrix B according to Algorithm 3, where $W = B\tilde{W}$
2: Generate a column vector $\boldsymbol{\beta} = Lap(S(\tilde{Q})/\epsilon)^s = [\beta_1, \ldots, \beta_s]^T$, where $\beta_i = Lap(S(\tilde{Q})/\epsilon)$
3: **return** $\mathcal{F}(Q, \mathcal{D}) = B(\tilde{W}\mathcal{D} + \boldsymbol{\beta})$

query domain is divided into disjoint subsets and the new query $\tilde{Q}$ is constructed based on the disjoint subsets. More details of the construction are illustrated in Algorithm 2. Then we construct a matrix B such that $W = B\tilde{W}$ as shown in Algorithm 3. Finally, we obtain the noisy answer $\mathcal{F}(Q, \mathcal{D})$ (at line 3 of Algorithm 1).

It is important to note that LRM in [31] also constructed B and $\tilde{W}$ such that $W = B\tilde{W}$. However, the construction of matrices B and $\tilde{W}$ in OM is total different from the method used in LRM. Please refer to [31] for more details.

Algorithm 2 Algorithm for the construction of new query set $\tilde{Q}$

Input: Query set $Q = \{q_1, \ldots, q_m\}$ and its workload W
Output: New query set $\tilde{Q}$ with $\tilde{W}$

1: Let $rank(W) = r$ and randomly select r independent queries $\hat{q}_1, \ldots, \hat{q}_r$ from Q
2: Let D be the query domain of query set Q, which is $dom(\mathcal{B}) = \{x_1, \ldots, x_i, \ldots, x_n\}$) introduced in Section 3.
3: Let D_i be the query domain of $\hat{q}_i$, $\forall i = 1, \ldots, r$
4: Set $\tilde{Q} = \phi$
5: Initialize $j = 0$
6: **while** D is not empty **do**
7: Update $j = j + 1$
8: Count the number of occurrence of each x_i in $D_1, \ldots, D_r$, denoted by $\kappa(x_i)$, $i = 1, \ldots, n$
9: Set $l = \arg\max_i \kappa(x_i)$
10: Find the index set $S_l = \{i | x_l \in D_i, \forall i = 1, 2, \ldots, r\}$
11: Set $\tilde{D}_j = \bigcap_{i \in S_l} D_i$
12: Let τ be the cardinality of set S_l
13: **for** $i = 1$ to τ **do**
14: Construct a query $\tilde{q}_i$ such that it can represent $\tilde{D}_j$ within $D_{S_l(i)}$
15: **end for**
16: Let τ_j be the number of independent query set among the query set $C = \{\tilde{q}_1, \ldots, \tilde{q}_\tau\}$
17: Construct τ_j independent and normalized queries $\tilde{q}_{j1}, \ldots, \tilde{q}_{j\tau_j}$ from C
18: Set $\tilde{Q}_j = \{\tilde{q}_{j1}, \ldots, \tilde{q}_{j\tau_j}\}$
19: Update $\tilde{Q} = \tilde{Q} \bigcup \tilde{Q}_j$
20: Update $D_i = D_i - \tilde{D}_j \; \forall i = 1, \ldots, r$
21: Update $D = D - \tilde{D}_j$
22: **end while**
23: **return** $\tilde{Q}$ with its workload $\tilde{W}$

In Algorithm 2, the query domain D of query set Q is first divided into n disjoint subsets and the workload matrix W is derived. We then randomly select r independent queries $\hat{q}_i$ out of m queries (Line 1), where r is the rank of W. For each query $\hat{q}_i$, we compute its query domain D_i (Line 3). Then, we count the number of occurrence of x_i in $D_1, \ldots, D_r$, denoted by $\kappa(x_i)$, and find the index l such that $\kappa(x_l) = \max_i \kappa(x_i)$. Based on l, we find the index set S_l which contains the element x_l (Line 10). Then we find the common subset $\tilde{D}_j$ of query domains D_i whose indices belong to the index set (Line 11). For the common subset $\tilde{D}_j$, we derive a new query $\tilde{q}_i$ for every query domains whose indices belong to the index set (Lines 13-15). After that, we find independent queries from $\tilde{q}_i$s and normalize them (Line 17). Finally, we remove the common subset from each query domain (Line 20) and the entire query domain (Line 21). In this way, for every two queries $\tilde{q}_1, \tilde{q}_2 \in \tilde{Q}_j$, $\tilde{q}_1$ and $\tilde{q}_2$ are independent. While for every two queries $\tilde{q}_1 \in \tilde{Q}_i$ and $\tilde{q}_2 \in \tilde{Q}_j$ where $i \neq j$, $\tilde{q}_1 \perp \tilde{q}_2$, then $\tilde{Q}_i \perp \tilde{Q}_j$.

In Algorithm 3, we derive matrix B by using the orthogonality of $\tilde{Q}_j$ (computed from Line 18 of Algorithm 2) and the independence of every two queries in a query subset $\tilde{Q}_j$. For the query domain of each query in Q, we first check whether the new query domain of $\tilde{q}_j$ belongs to it (Line 6). If the condition is true, we then further search a query in $\tilde{q}_j$ which is linear to the query in Q and compute the coefficient (Line 9).

Algorithm 3 Algorithm for the construction of matrix B

Input: Original workload $W \in \mathbb{R}^{m \times n}$ and new workload $\tilde{W} \in \mathbb{R}^{s \times n}$
Output: Matrix $B \in \mathbb{R}^{m \times s}$ such that $W = B\tilde{W}$.

1: Let D_i be the query domain of q_i, $\forall i = 1, \ldots, m$
2: Let $\tilde{D}_j$ be the query domain of $\tilde{q}_j$, $\forall j = 1, \ldots, s$
3: Generate zero matrix $B \in \mathbb{R}^{m \times s}$
4: **for** $i = 1$ to m **do**
5: **for** $j = 1$ to s **do**
6: **if** $\tilde{D}_j \subseteq D_i$ **then**
7: $rank\_ij = rank(q_i(\tilde{D}_j), \tilde{q}_j)$
8: **if** $rank\_ij == 1$ **then**
9: Find k such that $q_i(\tilde{D}_j) = k\tilde{q}_j$
10: Set $B(i, j) = k$
11: **end if**
12: **end if**
13: **end for**
14: **end for**
15: **return** B

We use an example to illustrate how the new query is derived in order to make the procedure of Algorithm more clear.
Example 1: Suppose we are interested in the statistics of the dataset shown in Table 1. Particularly, consider the size of the query domain as 4, and the four corresponding elements are the four counting results: $\{C_{WH}, C_{WL}, C_{BH}, C_{BL}\}$. We are interested in answering a query set $Q = \{q_1, q_2, \ldots, q_6\}$ with workload matrix expressed as

$$W = \begin{bmatrix} 0.3657 & 0 & 0.9812 & 0 \\ 0 & 0.0645 & 0 & 0 \\ 0 & 0.5879 & 0.7602 & 0 \\ 0 & 0 & 0 & 0.7310 \\ 0 & 0.7313 & 0 & 0 \\ 0 & 0 & 0.7122 & 0.9053 \end{bmatrix}.$$

where it is randomly generated.

According to Algorithm 2, we first check the rank of W, which is 4. We need to find four independent rows from W and construct a new query set $\tilde{Q} = \{\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_4\}$ with workload

$$\tilde{W} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

For the construction of matrix B, we first define the query domain by identifying those columns with non-zero elements in each query. Then we construct $B(i, j)$ by comparing the query domain between W and $\tilde{W}$. Finally, we get

$$B = \begin{bmatrix} 0 & 0.9812 & 0.3657 & 0 \\ 0.0645 & 0 & 0 & 0 \\ 0.5879 & 0.7602 & 0 & 0 \\ 0 & 0 & 0 & 0.7310 \\ 0.7313 & 0 & 0 & 0 \\ 0 & 0.7122 & 0 & 0.9053 \end{bmatrix}$$

according to Algorithm 3.

Note that the noise variance of the entire query set Q is around $75/\epsilon^2$ since $S(Q) \approx 2.5$ and there are 6 queries, while the noise variance based on query set $\tilde{Q}$ is less than $18/\epsilon^2$ due to $S(\tilde{Q}) = 1$. This demonstrates that the OM algorithm helps further reduce the noise variance.

4.3 Protocol Analysis

We present privacy analysis, complexity analysis and accuracy analysis in the section. We theoretically show that our algorithms satisfy the requirements of differential privacy, have less complexity and more accurate.

THEOREM 3. *Algorithm 1 is ϵ-differentially private.*

PROOF. The proof is clear since $\mathcal{K}(\tilde{Q}, D) = \tilde{W}D + \beta$ is the Laplace Mechanism and $\mathcal{F}(Q, D) = B \cdot \mathcal{K}(\tilde{Q}, D)$ is a post-processing of the output of the ϵ-differential privacy $\mathcal{K}(\tilde{Q}, D)$. □

In the proposed mechanism, we do not derive the matrix B where $W = B\tilde{W}$ by the above method as we take full advantage of the orthogonal property, which has been elaborated in Algorithm 3. The computational complexity of applying traditional matrix decomposition method is $O(n^3)$ while the computational complexity of the proposed Algorithm 3 is only $O(msn)$. The total complexity of Algorithms 2 and 3 is $O(Nrn + msn + nm^{\eta-1})$, where N is the number of iterations in Algorithm 2 and $\eta < 2.38$ is the matrix multiplication exponent [6] used to compute matrix rank. Furthermore, the proposed mechanism does not approximate the solutions [17, 31] and we achieve the exact solution from Algorithms 2 and 3 as we have the following theorems.

THEOREM 4. *Let $\tilde{Q}$ be a new query set and $\tilde{W} \in \mathbb{R}^{s \times n}$ derived from Algorithm 2, the original workload W of query set Q can be always represented as a linear combination of the new query set $\tilde{Q}$.*

PROOF. As shown in Algorithm 2. for each query q_i, its query domain can be divided into τ_i subsets. For each subset, there always exists a query $\tilde{q}'_j$ such that $q_i = \sum_{j=1}^{\tau_i} \alpha_{ij} \tilde{q}'_j$, where $\tilde{q}'_j \in \tilde{Q}$. □

COROLLARY 1. *Given the original workload matrix W and a new workload matrix $\tilde{W}$ derived from Algorithm 2, there is one and only one solution for $W = B\tilde{W}$ and it is derived from Algorithm 3.*

PROOF. According to Theorem 4, given matrices W and $\tilde{W}$, there always exists a matrix B_1 such that $W = \tilde{W}$. Suppose there exists another matrix $B_2 \neq B_1$ such that $W = B_2\tilde{W}$. Then we get $(B_1 - B_2)\tilde{W} = 0$. This demonstrates that the row vectors of $\tilde{W}$ are linear dependent, which contradicts the independence of $\tilde{W}$ in Algorithm 2. □

Algorithm 2 and Algorithm 3 decompose the original query set Q with $W = B\tilde{W}$. From Algorithm 2, $\tilde{Q}$ consists of orthogonal query subsets $\{\tilde{Q}_1, \tilde{Q}_2, \ldots, \}$. That is, $\tilde{Q} = \bigcup_j \tilde{Q}_j$. We get $S(\tilde{Q}) = \max_j S(\tilde{Q}_j)$. Thus, this property helps reduce the required noise magnitude. Based on Algorithms 1, 2 and 3, we get the following result:

THEOREM 5. *The expected squared error of $\mathcal{F}(Q, \mathcal{D})$ obtained in Algorithm 1 with respect to the decomposition $W = B\tilde{W}$ is*

$$2 \cdot trace(B \cdot B^T) S(\tilde{Q})^2 / \epsilon^2$$

PROOF. Since $\mathrm{E}\{\mathcal{F}(Q, \mathcal{D})\} = Q(\mathcal{D})$, the expected square error of $\mathcal{F}(Q, \mathcal{D})$ is equal to $Var(B\boldsymbol{\beta})$ where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_s]^T$. We get $Var(B\boldsymbol{\beta}) = \sum_j \sum_i B_{ij}^2 Var(\beta_j)$. From Algorithm 1, we get $Var(\beta_j) = 2S(\tilde{Q})^2/\epsilon^2$. Thus, $Var(B\boldsymbol{\beta}) = 2 \cdot \frac{S(\tilde{Q})^2}{\epsilon^2} \sum_{ij} B_{ij}^2$. Note that $trace(B \cdot B^T) = \sum_{ij} B_{ij}^2$. Therefore, the expected squared error of the orthogonal mechanism is equal to $2 \cdot trace(B \cdot B^T)(\frac{S(\tilde{Q})}{\epsilon})^2$. □

Theorem 5 shows that the expected squared error of the orthogonal mechanism not only depends on the sensitivity of the new query set, but also depends on the matrix B. It is important to note that B is uniquely determined by $\tilde{W}$ as shown in the proposed algorithm from Corollary 1. Thus, the structure of $\tilde{W}$ completely determines the expected squared error.

THEOREM 6. *Given the workload W, $W = B\tilde{W}$ is the optimal workload decomposition to minimize expected squared error if $(B, \tilde{W})$ is the optimal solution to the following program:*

$$\textit{Minimize: } S(\tilde{W}) \tag{3}$$

$$\textit{s.t. } W = B\tilde{W} \tag{4}$$

$$trace(B^T B) \leq 1 \tag{5}$$

PROOF. Let $(B_*, \tilde{W}_*)$ be the optimal solution in Eqs. (3)-(5). If it is not the optimal decomposition to minimize expected squared error, there must exist another decomposition $(B_1, \tilde{W}_1)$ such that

$$trace(B_1^T B_1) S^2(\tilde{W}_1) \leq trace(B_*^T B_*) S^2(\tilde{W}_*) \tag{6}$$

Clearly, we further construct $B_2 = B_1 / \sqrt{trace(B_1^T B_1)}$ and $\tilde{W}_2 = \sqrt{trace(B_1^T B_1)} \cdot \tilde{W}_1$. Clearly, $B_1\tilde{W}_1 = B_2\tilde{W}_2$. As $trace(B_2^T B_2) = 1$ and $trace(B_1^T B_1) S^2(\tilde{W}_1) = trace(B_2^T B_2) S^2(\tilde{W}_2)$, we get

$$S^2(\tilde{W}_2) = trace(B_2^T B_2) S^2(\tilde{W}_2) \leq trace(B_*^T B_*) S^2(\tilde{W}_*) \leq S^2(\tilde{W}_*).$$

However, this contradicts to the definition of $(B_*, \tilde{W}_*)$. □

Theorem 6 further shows that the $\tilde{W}$ completely determines the expected squared errors. The aim of the proposed decomposition algorithm is to decrease the correlation of $\tilde{W}$ and thus reduces the expected squared errors.

5. EXPERIMENTAL EVALUATION

This section experimentally studies the effectiveness and efficiency of our proposed OM mechanism for answering a number of correlated count queries. Particularly, we compare the proposed mechanism with the state-of-the-art mechanisms, including the Laplace Mechanism (LM) [11], Low-Rank Mechanism (LRM) [31] and Matrix Mechanism (MM) [17], using one real benchmark dataset from UCI machine learning repository [1]. In order to provide a fair comparison, we use the algorithms described in [31] for LRM and MM.

The dataset we are using is the *Adult* dataset, which has been widely used to benchmark the performance among various proposed mechanisms [3, 15, 25]. It contains 32,561 individuals with 14 attributes, among which there are 8 categorical attributes and 6 numerical attributes. The class attribute is a binary category representing income levels, i.e., $> 50K$ or $\leq 50K$.

To perform a fair comparison, three different cases of workload matrices are considered. For the generation of W, we first randomly generate an $m \times n$ matrix H, where each element of the matrix is drawn from the standard uniform distribution. Then we set $w_{ij} = rand$ if $h_{ij} \leq \tau$, and $w_{ij} = 0$ otherwise. Clearly, τ is related to $S(W)$, which increases with increasing τ according to the definition of $S(W)$. Thus, higher value for τ means higher sensitivity of W. Furthermore, higher value for τ causes higher correlation among queries. Here τ is set as $\tau = [0.2, 0.4, 0.8]$ for the generation of W_τ, i.e. $W_{0.2}, W_{0.4}, W_{0.8}$. All experiments were conducted on an Intel Xeon E7 2.00GHz PC with 64GB RAM. We evaluate the performance improvement based on accuracy (or squared errors) and efficiency (or execution time). Note that we have performed our experiments for 20 times and average results are reported. Scenarios are also generated to simulate the real-world applications by varying three parameters, namely, query size m, query domain size n and privacy budget ϵ, under the three workload matrices.

5.1 Accuracy Evaluation

We are now ready to compare the experimental results for three techniques: OM, LM, MM and LRM, in terms of accuracy in the section.

Varying privacy budget ϵ. According to the definition of ϵ-differential privacy, smaller ϵ implies stronger privacy and thus results in larger noise variances. We study the impact of ϵ on the expected squared errors by varying ϵ from 0.2 to 1.6.

Fig. 1 shows the average squared errors versus the variation of ϵ with parameters $m = 200$ and $n = 1000$. It shows that the average squared errors achieved by all the four mechanisms decrease when we increase privacy budget ϵ. More importantly, OM achieves much smaller average squared errors compared with existing LRM, MM and LM methods for all the three workload matrices with different τ values. When τ increases, the sensitivity of the original query set also increases. It can be seen that the average squared errors achieved by the four mechanisms also increase, although LRM is not sensitive to the changes of the workload matrix. Note that the errors between LM and LRM increase with the increasing τ, and the errors between LRM and OM decrease with increasing τ. The average squared error achieved by MM is the highest. Nevertheless, the proposed OM mechanism achieves significantly better accuracies across the benchmark dataset compared with the three state-of-the-art approaches.

Varying query size m. $m = [50, 100, 150, 200, 250, 300]$ and $n = 1000$ are set for *Adult* dataset. Fig. 2 shows the comparison of average squared errors achieved by the four mechanisms with three workload matrices. We observe that the average squared errors achieved by the four mechanisms increase with the increasing m. However, OM is not so sensitive to the changes of m and τ and consistently better than the other three mechanisms.

Varying domain size n. $n = [500, 1000, 1500, 2000, 2500]$ and $m = 200$ are set for the *Adult* dataset. Fig. 3 shows the comparison of average squared errors versus domain size n. When τ increases, the average squared errors achieved by LM and OM also increase. MM is not so sensitive to change of τ. But the average squared error achieved by MM is the highest. When n increases, OM achieves the least squared errors consistently under the different scenarios across the benchmark dataset.

5.2 Execution Time Evaluation

In this section, we further compare the execution time among LRM, MM and OM, where the scenarios are set as the same in the Accuracy Evaluation subsection.

Varying query size m. From Fig. 4, we observe that OM is the fastest among the three mechanisms. Particularly, the execution time of the three mechanisms does not show strong correlation with query size m when $\tau = 0.2$. But when $\tau = 0.4$ or $\tau = 0.8$, execution time of both LRM and OM increases with increasing query size m. Furthermore, the execution time of OM is not so sensitive to the change of τ and m.

Varying domain size n. From the comparisons shown in Fig. 5, the execution time of all mechanisms increases with increasing query size n. The relationship between the execution time of LRM and OM and the query domain size is approximately linear, but the execution time of OM under different scenarios is still much smaller than that of LRM. For example, the execution time of OM is around 300s when $n = 1000$, while that of LRM is around 2000s for different workload matrices in *Adult* dataset, demonstrating that the proposed OM is more efficient.

6. CONCLUSIONS

In this paper, we proposed a novel mechanism, orthogonal mechanism, for answering query set while achieving ϵ-differential privacy. It significantly reduces the noise magnitude by removing the correlation between queries as much as possible. The procedure of the decomposition does not depend on the expensive matrix operation. The computational complexity of the proposed work is thus much lower than existing mechanisms due to its orthogonal properties. Experimental results demonstrated that the proposed mechanism is very accurate and efficient, and hence scalable enough for handling large-scale datasets.

7. REFERENCES

[1] N. R. Adam and J. C. Worthmann. Security-control methods for statistical databases: a comparative study. *ACM*

[1] http://archive.ics.uci.edu/ml/

Computing Surveys (CSUR), 21(4):515–556, 1989.

[2] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu. Achieving anonymity via clustering. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART*, pages 153–162, 2006.

[3] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proceedings. 21st International Conference on Data Engineering*, pages 217–228. IEEE, 2005.

[4] J. Brickell and V. Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *Proceedings of the 14th ACM SIGKDD*, pages 70–78, 2008.

[5] K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*, pages 289–296, 2009.

[6] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of Computing*, pages 1–6, 1987.

[7] T. Dalenius. The invasion of privacy problem and statistics productionâĂŤan overview. *Statistik Tidskrift*, 12:213–225, 1974.

[8] B. Ding, M. Winslett, J. Han, and Z. Li. Differentially private data cubes: optimizing noise sources and consistency. In *Proceedings of the ACM SIGMOD*, pages 217–228, 2011.

[9] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART*, pages 202–210, 2003.

[10] C. Dwork. Differential privacy. In *Proceedings of the 33rd international conference on Automata, Languages and Programming*, pages 1–12. Springer-Verlag, 2006.

[11] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography*, pages 265–284. Springer, 2006.

[12] A. Friedman and A. Schuster. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD*, pages 493–502, 2010.

[13] M. Hardt and K. Talwar. On the geometry of differential privacy. In *Proceedings of the forty-second ACM symposium on Theory of Computing*, pages 705–714, 2010.

[14] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment*, 3(1-2):1021–1032, 2010.

[15] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the eighth ACM SIGKDD*, pages 279–288, 2002.

[16] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *Proceedings of the ACM SIGMOD*, pages 193–204, 2011.

[17] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART*, pages 123–134, 2010.

[18] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *IEEE 23rd International Conference on Data Engineering*, pages 106–115, 2007.

[19] F. McSherry and R. Mahajan. Differentially-private network trace analysis. *ACM SIGCOMM Computer Communication Review*, 41(4):123–134, 2011.

[20] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. In *Proceedings of the 15th ACM SIGKDD*, pages 627–636, 2009.

[21] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science*, pages 94–103, 2007.

[22] F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD*, pages 19–30, 2009.

[23] A. R. Miller. *The assault on privacy: computers, data banks, and dossiers*. University of Michigan Press, 1971.

[24] N. Mohammed, R. Chen, B. Fung, and P. S. Yu. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD*, pages 493–501, 2011.

[25] N. Mohammed, B. Fung, P. C. Hung, and C.-k. Lee. Anonymizing healthcare data: a case study on the blood transfusion service. In *Proceedings of the 15th ACM SIGKDD*, pages 1285–1294, 2009.

[26] M. A. Palley and J. S. Simonoff. The use of regression methodology for the compromise of confidential information in statistical databases. *ACM Transactions on Database Systems (TODS)*, 12(4):593–608, 1987.

[27] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the ACM SIGMOD*, pages 735–746, 2010.

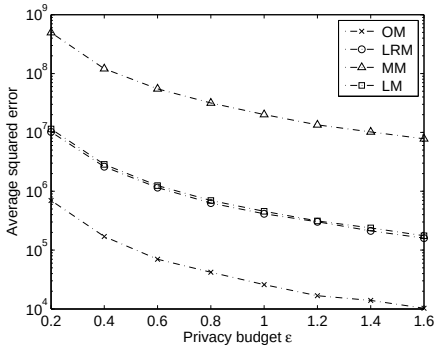
[28] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[29] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1200–1214, 2011.

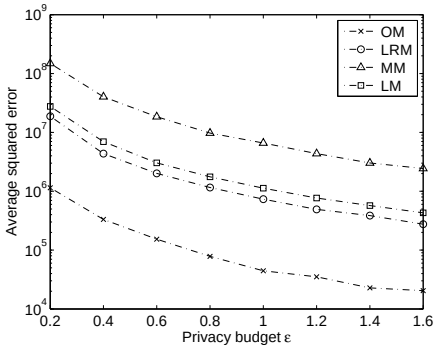
[30] Y. Xiao, L. Xiong, and C. Yuan. Differentially private data release through multidimensional partitioning. In *Secure Data Management*, pages 150–168. Springer, 2010.

[31] G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, and Z. Hao. Low-rank mechanism: optimizing batch queries under differential privacy. *Proceedings of the VLDB Endowment*, 5(11):1352–1363, 2012.

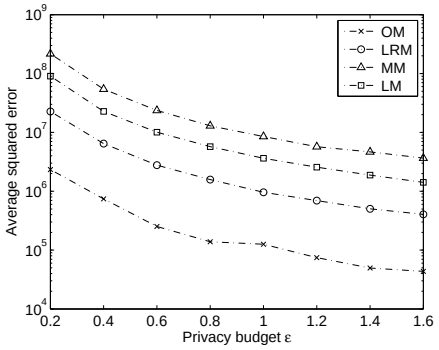
[32] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett. Functional mechanism: regression analysis under differential privacy. *Proceedings of the VLDB Endowment*, 5(11):1364–1375, 2012.



(a) W with $\tau = 0.2$

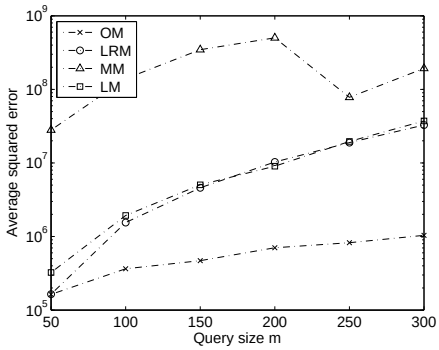


(b) W with $\tau = 0.4$

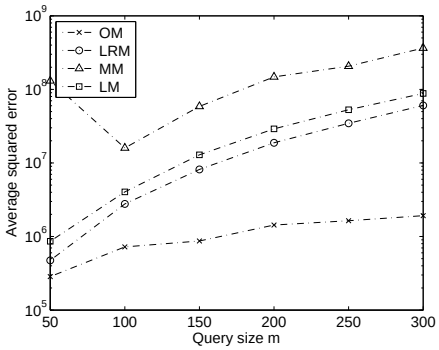


(c) W with $\tau = 0.8$

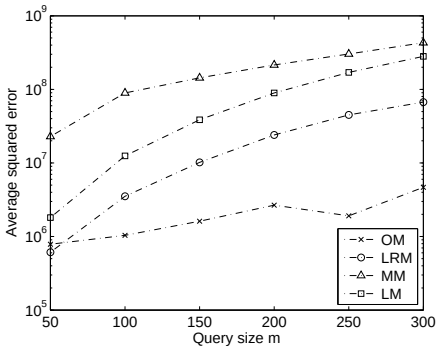
Figure 1: Accuracy comparison by varying ϵ.



(a) W with $\tau = 0.2$

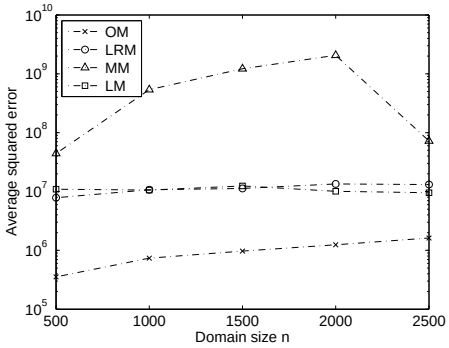


(b) W with $\tau = 0.4$

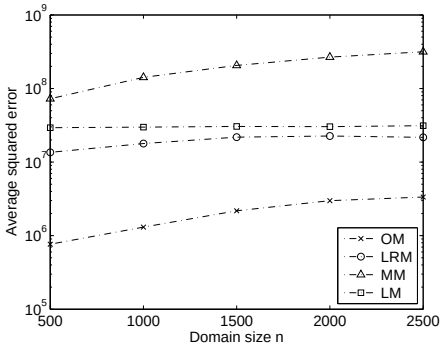


(c) W with $\tau = 0.8$

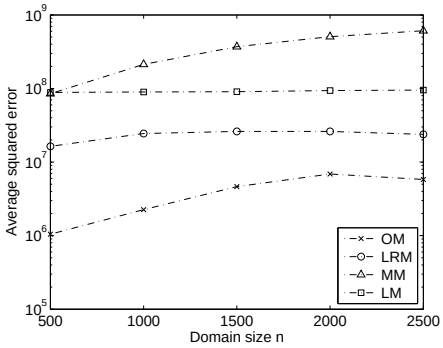
Figure 2: Accuracy comparison by varying m.



(a) W with $\tau = 0.2$

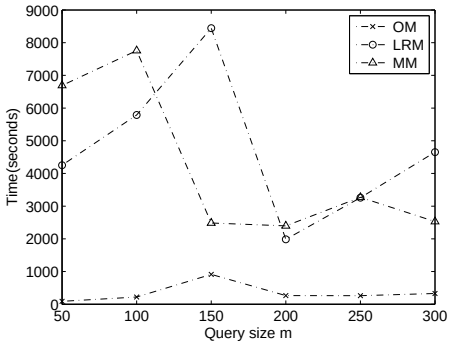


(b) W with $\tau = 0.4$

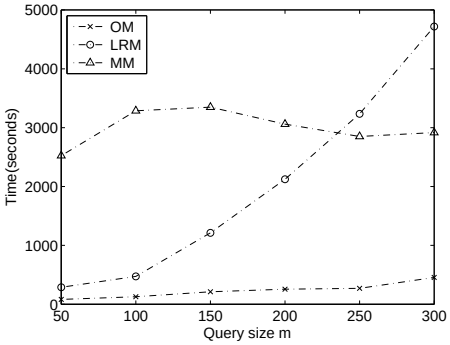


(c) W with $\tau = 0.8$

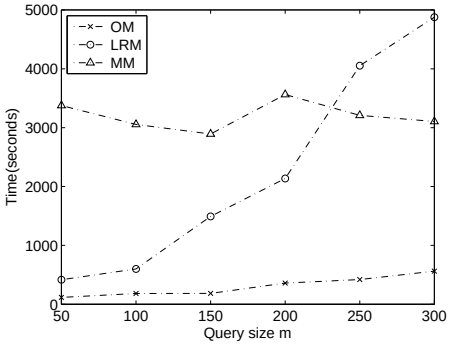
Figure 3: Accuracy comparison by varying n.



(a) W with $\tau = 0.2$

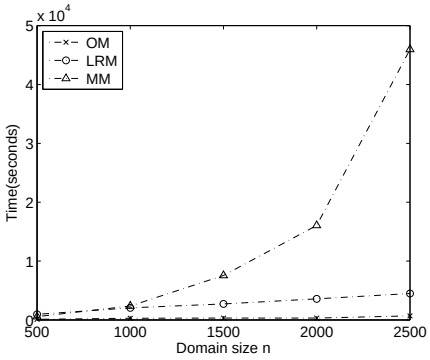


(b) W with $\tau = 0.4$

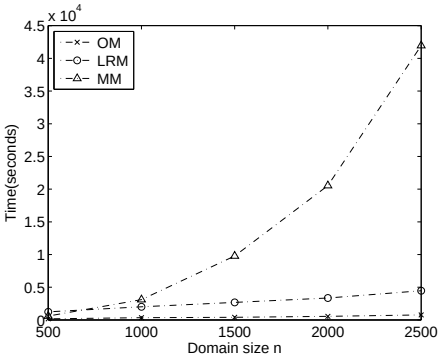


(c) W with $\tau = 0.8$

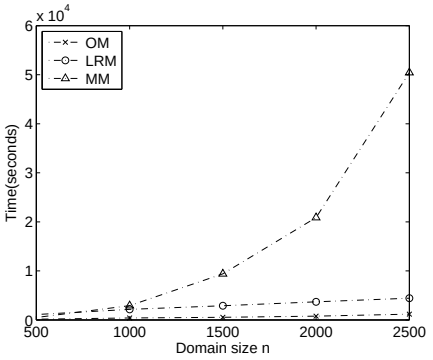
Figure 4: Execution time comparison by varying query size m.



(a) W with $\tau = 0.2$



(b) W with $\tau = 0.4$



(c) W with $\tau = 0.8$

Figure 5: Execution time comparison by varying domain size n.