Relation Mining Under Local Differential Privacy

Kai Dong
Southeast University

Zheng Zhang
Southeast University

Chuang Jia
Southeast University

Zhen Ling*
Southeast University

Ming Yang
Southeast University

Junzhou Luo
Southeast University

Xinwen Fu
University of Massachusetts Lowell

Abstract

Existing local differential privacy (LDP) techniques enable untrustworthy aggregators to perform only very simple data mining tasks on distributed private data, including statistical estimation and frequent item mining. There is currently no general LDP method that discovers relations between items. The main challenge lies in the curse of dimensionality, as the quantity of values to be estimated in mining relations is the square of the quantity of values to be estimated in mining item-level knowledge, leading to a considerable decrease in the final estimation accuracy. We propose LDP-RM, the first relation mining method under LDP. It represents items and relations in a matrix and utilizes singular value decomposition and low rank approximation to reduce the number of values to estimate from $O(k^2)$ to $O(r)$, where k is the number of all considered items, and $r < k$ is a parameter determined by the aggregator, signifying the rank of the approximation. LDP-RM serves as a fundamental privacy-preserving method for enabling various complex data mining tasks.

1 Introduction

Benefiting from local differential privacy (LDP) techniques [20], an untrustworthy aggregator is able to discover global knowledge from private data owned by distributed users without compromising privacy. Typically, an LDP technique comprises a **perturbation** algorithm and an **aggregation** algorithm. The perturbation algorithm perturbs private data to ensure privacy, and is employed by each user locally when responding to the aggregator's queries. The aggregation algorithm extracts global knowledge from all the perturbed responses provided by users, and is employed by the aggregator. However, existing LDP techniques are only capable of discovering simple global knowledge, such as statistical information [12,13,42], or popular items (e.g., emojis) [3,43].

Currently, there is no general method that discovers relations between or among items under LDP while some very recent LDP techniques can be applied to extract specific types of relations. Take the LDP technique that discovers frequent item-sets as an example [43]. An item-set represents co-occurrence relation of items. The estimated frequent item-sets are obtained through item-level perturbation and aggregation. However, this approach relies on a strong assumption that an item-set comprising two frequent items is also frequent. This assumption is problematic as shown by a counter example in Fig. 1 of Sec. 3.2.

To the best of our knowledge, we are the first to introduce the generic problem of relation mining under LDP. In this problem, we assume that each user possesses some private items and relations among those items are private information about the user. The untrustworthy aggregator is interested in globally useful statistics of relations, measured by two criteria: **support** and **confidence**. Support represents the relation's popularity and is defined as its frequency, while confidence denotes its reliability and is calculated as the ratio of the relation's support to the support of the items within the relation. Therefore, the goal of the aggregator is to uncover high-support and high-confidence relations.

The primary challenge in addressing this problem lies in the "Curse of Dimensionality". To discover relations with high support and high confidence, the aggregator needs to estimate the support and confidence of all potential relations. The quantity of values to be estimated in mining relation-level knowledge is the square of the quantity of values to be estimated in mining item-level knowledge [9], leading to a tremendous increase in the extent to which each user disturbs her data to ensure privacy. In LDP, a parameter named *privacy budget* (typically denoted as ε) is used to determine the privacy level. Its reciprocal determines the amount of noise that should be added to a response, and it must be split among multiple responses

[*] Corresponding author: Prof. Zhen Ling of Southeast University, China.

discovering high-support and high-confidence relations from distributed private data. This technique serves as a fundamental building block for data mining under LDP. With LDP-RM, the aggregator iteratively estimates the support of all relations from scratch, using a global matrix (referred to as the *aggregator matrix*) to save and update the estimations. Each row/column in this matrix corresponds to an item, and each element represents the support of a relation. To update the matrix, the aggregator collects perturbed reports constructed by distributed users. Each report can be abstracted as a user matrix that captures the user's private items and their relations. Directly protecting the privacy of the entire user matrix would inevitably split the privacy budget, leading to excessive noise and insufficient estimation accuracy. To address this issue, the **basic idea** of LDP-RM is to employ singular value decomposition [17] to compress the high-dimensional user matrix into a low-dimensional vector. By perturbing and reporting this vector instead of the user matrix, the amount of noise needed for privacy protection is significantly reduced.

LDP-RM works as follows. Without loss of generality, let k represent the number of all considered items. The aggregator begins by guessing an initial global matrix of size $k \times k$. Instead of directly aggregating and estimating the true global matrix of size $k \times k$, the aggregator estimates its projection onto the initial global matrix. This projection is represented as the *singular vector* (calculated using singular value decomposition), with a length of k. As a result, the number of values to estimate is reduced from k^2 to k. The aggregator then recovers the global matrix from the singular vector averaged over all users' data. If the aggregator estimates only the best rank-r approximation of the global matrix instead of the entire matrix, the number of values to estimate is further reduced to $r < k$. The **primary challenge** lies in performing singular value decomposition without possessing knowledge of the true global matrix. To address this challenge, LDP-RM employs an iterative scheme which updates the global matrix in each iteration. After several iterations, the aggregator obtains the final estimated global matrix and discovers high-support and high-confidence relations from it. Our experimental evaluation demonstrates the effectiveness of LDP-RM, and the ineffectiveness of the state-of-the-art LDP techniques — including SVIM [43], SVSM [43], CALM [50], and PCKV [18] — in mining relations.

Our major contributions can be summarized as follows:

- We are the first to introduce and investigate the problem of relation mining under LDP. This is a fundamental problem for carrying out complex data mining tasks on private data owned by distributed users.
- We propose LDP-RM, the first relation mining method under LDP. This method can be used to discover various kinds of relations of high support and high confidence, including but not limited to the co-occurrence of items.
- We conduct extensive experiments to validate the effectiveness of LDP-RM.

2 Background

In this section, we introduce local differential privacy and data mining under LDP. The notation used throughout this paper is summarized in Tab. 1.

2.1 Local Differential Privacy

Local differential privacy (LDP) is a model of differential privacy (DP) which offers robust privacy guarantees for distributed users [12]. Consider each user has a private value x from a given domain $\mathbf{X}$. In the local setting, each user applies an algorithm Ψ to perturb her individual input value $x \in \mathbf{X}$. The perturbed value $\Psi(x)$ is then sent to the aggregator. To ensure privacy, the algorithm Ψ should satisfy:

Definition 1 (ε-Local Differential Privacy)**.** *An algorithm $\Psi(\cdot)$ satisfies ε-local differential privacy (ε-LDP), where $\varepsilon \geq 0$, if and only if for any two values $x_1, x_2 \in \mathbf{X}$, we have:*

$$\forall T \in Range(\Psi) : \Pr(\Psi(x_1) = T) \leq e^{\varepsilon} \cdot \Pr(\Psi(x_2) = T),$$

where $Range(\Psi)$ denotes the range of Ψ.

The parameter ε is commonly known as the *privacy budget*, and a smaller ε guarantees a higher level of privacy. When the aggregator needs to estimate multiple values, it must be split among multiple reports from a user, in accordance with the *sequential composition theorem* [12]. Splitting the privacy budget can result in over-perturbed values provided by users, leading to a significant decrease in estimation accuracy. This problem can be mitigated by dividing users into groups, with the aggregator estimating only one value from each group to ensure that each user reports only once [4, 12, 32, 37, 43–45].

2.2 Data Mining under LDP

A data mining task discovers useful knowledge from large datasets. A data mining task under LDP operates within the confines of LDP constraints, which are imposed on distributed user data. Existing LDP techniques facilitate two primary types of data mining tasks: statistical estimation [12, 13, 42] and item mining [3, 43]. Taking the movie recommendation scenario as an example. Statistical-estimation can quantify the frequency of specific movie views. Item mining can identify the most popular movies.

In the most general scenario, consider n users, each possessing a private value x from a given domain $\mathbf{X}$. The aggregator aims to extract useful knowledge from users' private data without compromising privacy. A data mining task under LDP can be formalized as an LDP protocol $\mathcal{T}$, consisting of a pair of algorithms $\langle \Psi, \Phi \rangle$, defined as follows:

$$\mathcal{T}(\varepsilon) \triangleq \langle \Psi, \Phi \rangle. \tag{1}$$

Here Ψ denotes the perturbation algorithm employed by users to perturb local data to satisfy ε-LDP, and Φ denotes the

Table 1: Summary of Notations

Symbol	**Meaning**
x, $\mathbf{x}$, X, d	Item, user's set of items, set of all items, domain size of X
w, $\mathbf{w}$, $\mathbf{I}$	Relation, user's set of relations, user matrix
ε, $\Psi(\cdot)$, $\Phi(\cdot)$,	Privacy budget, perturbation algorithm, aggregation algorithm
$s(\cdot)$, $c(\cdot)$	Support of an item or a relation, confidence of a relation
k, k_s, k_c	Number of high-support items, number of high-support relations, number of high-confidence relations
n, n_1, n_2, n_3, T	Number of total users, number of users in the 1^{st}, 2^{nd}, 3^{rd} group, number of sub-groups/iterations
$\mathbf{Y}$, $\mathbb{Y}$	Candidate set of high-support items, that of high-support relations
$\mathbb{W}$	Candidate set of high-support and high-confidence relations
$\mathbf{M}_{agg}$, $\mathbf{M}_{gt}$, $\mathbf{M}'_{agg}$	Aggregator matrix, true matrix, updated aggregator matrix
θ, r, $\mathbf{M}_r$	Bias threshold, low rank, best rank-r approximation of the aggregator matrix
$\mathbf{u}$, $\mathbf{v}^\top$, $\mathbf{U}$, $\mathbf{V}^\top$, σ, Σ	Left-singular vector, transpose of right-singular vector, matrix of $\mathbf{u}$, $\mathbf{v}^\top$, singular value, matrix of σ
$\mathbf{P}$, $\widehat{\mathbf{P}}$, $\mathbf{S}$, $D_{\mathbf{S}}$, $\widehat{\mathbf{S}}$	Projection matrix, aggregation of $\mathbf{P}$, user report vector, domain of $\mathbf{S}$, estimated mean of $\mathbf{S}$
l, s_l, $\overrightarrow{s_l}$, $(\overrightarrow{s_l})^*$	Index of singular value in singular vector, pseudo singular value of index l, mapped s_l, perturbed $\overrightarrow{s_l}$
G, $C(\cdot)$, $C_H(\cdot)$	Hashed domain, count function, hash count function
ℓ, L, $u'(L)$	Padding length, padding length ℓ at the 90 percentile, update factor

aggregation algorithm employed by the aggregator to extract useful knowledge. Existing LDP protocols can be applied to perform statistical estimation tasks and item-level data mining tasks. However, a general method for mining relations between items under LDP remains elusive.

2.2.1 Statistical Estimation under LDP. Numerous LDP protocols have been proposed for various statistical estimation tasks. Our work pertains to two specific estimation tasks: mean estimation and frequency estimation.

In a mean estimation task, the aggregator aims to discover the averaged value $\bar{x}$ from all the n users:

$$\bar{x} \triangleq \frac{1}{n}\sum_{j=1}^{n} x^j, \tag{2}$$

where $x^j \in \mathbf{X} = [-1,1]$ denotes the normalized numeric value held by the j^{th} user. Prominent LDP protocols for mean estimation include Duchi [10], PM [39], HM [39], and others. This paper employs the HM protocol.

HM. HM stands for Hybrid Mechanism. A HM protocol takes as inputs a value $x \in \mathbf{X}$ and a privacy budget ε. It adaptively invokes the perturbation algorithm of PM and Duchi (denoted as Ψ_{PM} and Ψ_{Duchi} correspondingly), depending on the value of ε:

$$\Psi_{\mathrm{HM}(\varepsilon)}(x) \triangleq \begin{cases} \Psi_{\mathrm{PM}(\varepsilon)}(x), & \text{if } \varepsilon > 0.61; \\ \Psi_{\mathrm{Duchi}(\varepsilon)}(x), & \text{otherwise .} \end{cases} \tag{3}$$

$$\Phi_{\mathrm{HM}}(x) \triangleq \frac{1}{n}\sum \Psi_{\mathrm{HM}(\varepsilon)}(x). \tag{4}$$

Appendix A gives a concise overview of PM and Duchi.

In a frequency estimation task, the aggregator aims to discover the number of users possessing a given value x. Let $\mathbf{x}^j \subseteq \mathbf{X} = \{1,2,...,d\}$ represent the set of categorical values held by the j^{th} user. The aggregator estimates the frequency of each item $x_i \in \mathbf{X}$, denoted as $s(x_i)$, by computing:

$$s(x_i) = \frac{1}{n}\sum_{j=1}^{n} \mathbb{1}(x_i, \mathbf{x}^j), 1 \le i \le d, \tag{5}$$

where $\mathbb{1}(x, \mathbf{x})$ is an indicator function, defined as:

$$\mathbb{1}(x, \mathbf{x}) \triangleq \begin{cases} 1 & \text{if } x \in \mathbf{x} \\ 0 & \text{if } x \notin \mathbf{x} \end{cases} \tag{6}$$

An LDP protocol employed for frequency estimation is also referred to as a Frequency Oracle (FO). Basic FOs, such as GRR [42], Rappor [13], OLH [42], OUE [42], JLRR [13], HRR [1], are applicable only in scenarios where each user possesses a single value. Our work is partially based on an item-level data mining LDP protocol (see Section 2.2.2) and, as a consequence, indirectly utilizes these protocols. A conof each item in the value domain $\mathbf{X}$, and finally outputs the top-k frequent items (denoted as $\mathbf{Y}$) along with their estimated frequencies. Φ_{SVIM} is defined as:

$$\Phi_{\mathsf{SVIM}(\mathbf{X},\varepsilon,n,k)} \triangleq \{(x, \Phi_{\mathsf{PSFO}(\ell,\varepsilon)}(x)) | x \in \mathbf{Y}\}, \quad (7)$$

where $\Phi_{\mathsf{PSFO}(\ell,\varepsilon)}(x)$ is the aggregation algorithm of PSFO which outputs the support of item x. We provide a concise overview of PSFO in Appendix C, and some more details of SVIM in Appendix D.

3 Relation Mining under LDP

In this section, we define the problem of relation mining under LDP, and discuss the challenges.

3.1 Problem Definition

While existing LDP techniques are unable to discover relation-level knowledge, such knowledge holds significant importance, particularly in domains such as social network analysis, healthcare informatics, finance, and marketing [30]. For instance, association rule mining, which uncovers co-occurrence relations among items, benefits offline retailers like Walmart in optimizing product placement [27]. Beyond association rules, other relations, including temporal relations, are harnessed by popular platforms like Amazon for product recommendations [31].

We consider an untrustworthy aggregator aiming to discover useful relations from private data owned by distributed users. The usefulness of a relation is often measured from two aspects, namely, support and confidence. The **support** of a relation w indicates its popularity and is defined as the proportion of users that have this relation among all n users:

$$s(w) \triangleq \frac{1}{n}\sum_{j=1}^{n} \mathbb{1}(w, \mathbf{w}^j), \quad (8)$$

where $\mathbf{w}^j$ denotes the set of relations possessed by the j^{th} user, and $\mathbb{1}(\cdot,\cdot)$ is the indicator function defined as in Equation (6). The **confidence** of a relation indicates its reliability and is defined as the ratio of the relation's support to the first item's support. Let (x_a, x_b) denote a relation between two items x_a and x_b. Its confidence can be calculated as:

$$c((x_a, x_b)) \triangleq s((x_a, x_b))/s(x_a). \quad (9)$$

In data mining under LDP, the criteria for determining high support or confidence differ from traditional data mining. In traditional data mining, predefined minimum thresholds can be employed to assess high support or confidence. As confidence computation inherently depends on support computation, a logical approach is to first identify high-support relations and subsequently select high-confidence ones. However, in the local context with an untrustworthy aggregator, they remain unknown beforehand. To tackle this challenge, existing LDP methods replace the support threshold with a parameter, denoted as k, and focus on discovering the top-k items or itemsets with high support [4, 6, 24, 32, 37, 43–45]. Similarly, in the context of relation mining under LDP, the aggregator's objective is to initially identify the top-k_s relations with high support, from which they can derive the final top-k_c relations among all possible relations.

Mathematically, consider a set of users $\mathbf{U} = \{u_1, u_2, ..., u_n\}$ and a set of items $\mathbf{X} = \{x_1, x_2, ..., x_d\}$, both known to the aggregator. Each user possesses a private set of items $\mathbf{x} \subseteq \mathbf{X}$ and a private set of relations $\mathbf{w} \subseteq \mathbf{X}^2$. The objective of relation mining under LDP is to first identify the top-k_s relations in terms of support, and then, from these k_s relations, find the top-k_c relations in confidence.

3.2 Challenges

To the best of our knowledge, no existing LDP methods can be applied to discover relation-level knowledge in general scenarios. A line of approaches aims to discover high-support item-sets [32, 43]. However, these are essentially item mining techniques, as they treat specific relations (i.e., co-occurrence relationships of items) as items and rely on basic FO protocols for statistical estimation. Furthermore, they are limited to statistics on high-support items or relations, neglecting the equally important criterion of confidence.

The primary challenge in addressing the problem of relation mining under LDP lies in the "Curse of Dimensionality". Given that both the set of individual items and that of individual relations are considered private, the aggregator must estimate the support for each item and that for each relation under the constraint of LDP. For estimating the support of each item, numerous item-level mining LDP protocols, such as SVIM, can be directly employed. However, estimating the support of relations by directly utilizing existing methods necessitates converting each 2-dimensional relation into a new 1-dimensional item (e.g., by using one-hot encoding) [43]. As a result, the quantity of values to be estimated becomes extremely large, leading to insufficient estimation accuracy.

An intuitive solution is to guess out high-support relations based on the support of items. Following this intuition, an existing approach called Set-Value itemSet Mining (SVSM) [43], which finds high-support co-occurrence relations, assumes that items are roughly independent of each other, so that the support of only those relations composed of high-support items needs to be estimated. However, this technique is not applicable to general relation mining tasks, as it can only discover certain co-occurrence relations when relatively low-support items are not strongly related to each other. In contrast, in a general relation scenario, the degrees of independence between any two items can vary significantly.

Furthermore, this intuitive approach presents challenges even in scenarios of association rule mining under LDP. This

$$\begin{array}{c|cccccc|} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ x_1 & 40 & 20 & 20 & 0 & 0 & 0 \\ x_2 & 20 & 30 & 10 & 0 & 0 & 0 \\ x_3 & 20 & 10 & 30 & 0 & 0 & 0 \\ x_4 & 0 & 0 & 0 & 25 & 25 & 25 \\ x_5 & 0 & 0 & 0 & 25 & 25 & 25 \\ x_6 & 0 & 0 & 0 & 25 & 25 & 25 \end{array}$$

Figure 1: Number of users who possess x_{row} and x_{col}.

is because there is a conflict between mining high-support relations and mining high-confidence relations. Referring back to Equation (9), it is clear that, given the support of a relation, as the support of the item increases, the confidence of that relation decreases. The aggregator inevitably relies on high-support items to select high-support relations, this process tends to favor low-confidence relations, thus contradicting the goal of high-confidence relation mining.

We present an example to clarify this conflict. An association rule mining task is conducted under LDP with 6 items (denoted as x_1 to x_6), and 75 users. A 6×6 matrix representing the support of each item and relation is as illustrated in Fig. 1. Diagonal elements (in gray) signify the support of items, while other elements indicate the support of co-occurrence relations between their row- and column-labels. The aggregator's objective is to first identify the top-12 relations in support and then determine the top-6 relations in confidence as the final results. Correct results are marked in blue. However, if the aggregator utilizes the intuitive approach to select candidate relations, he will choose 12 relations (in orange) composed of high-support items, ultimately failing to discover any correct relations.

4 Method

In this section, we present LDP-RM and describe how it addresses the problem of relation mining under LDP.

4.1 Basic Idea

To address the "Curse of Dimensionality" in relation mining, LDP-RM must reduce the quantity of data that needs to be estimated. When dealing with binary relations defined among d items, there is a potential existence of $d \times (d-1)$ distinct relations. In such instances, one must recover a quantity of $O(d^2)$ support values, to accurately discover the relations with the highest support. If the mining task also extends to ternary relations or higher-order relations, the data recovery requirement escalates to $O(d^3)$ or beyond. This results in less accurate mining results. To facilitate understanding LDP-RM, this section delves into a simplified scenario that concentrates on binary relations. Later, in Sec. 6.1, we generalize LDP-RM to enable the mining of cascading relations involving more than two items.

By employing a widely adopted pre-estimation technique [43], we can significantly diminish the number of items that need to be considered. The aggregator initially estimates the support of all d items initially and subsequently identifies the top-k items in support (where $k < d$). By considering the relations only between top-k items instead of all items, the quantity of values to be estimated can be reduced from $O(d^2)$ to $O(k^2)$. Many existing LDP techniques [15, 37, 44] incorporate this approach into their mechanism design to confine the scope of the problem. Nevertheless, this pre-estimation procedure alone is insufficient to tackle the curse of dimensionality in relation mining under LDP.

Basic Idea. In what follows, we describe how LDP-RM further reduces the quantity of values to be estimated from $O(k^2)$ to below $O(k)$. LDP-RM organizes all support values into a matrix, and uses only a few singular values obtained through singular value decomposition (SVD) and low-rank approximation to represent the entire matrix. Each user puts her information into a **user matrix** of size $k \times k$, where rows/columns correspond to items. Off-diagonal/diagonal elements are binary (1 or 0), indicating possession of a relation/item by this user. We use **true matrix** to term the aggregation of all user matrices, and **aggregator matrix** to denote the aggregator's estimate on it. By using a variant of SVD on both the aggregator matrix, and each user matrix, a local vector of length k (i.e., the **singular vector**) can be obtained to indicate the projection from the user matrix onto the aggregator matrix. Consequently, the aggregator only needs to aggregate this vector of size k from each user to recover the $k \times k$ true matrix which is the aggregation of all the user matrices. Additionally, LDP-RM uses the best rank-r approximation of the aggregator matrix to decrease the size of users' private vectors from k to r, resulting in a further reduction in the number of values to be estimated. This reduces the problem of estimating the support of relations with a domain size of k^2 to estimating the mean value of a vector with length r (where $r < k$).

Technical Challenge. However, performing SVD in this context is challenging since neither the aggregator nor the user has prior knowledge of the true matrix. This creates a "catch-22" situation since neither party has information about the true matrix, which prevents distributed users from performing SVD and, consequently, prevents the aggregator from estimating the matrix.

Solution. To address this challenge, LDP-RM utilizes an iterative scheme, updating the estimation of the aggregator matrix to ultimately converge to the true matrix. Initially, the aggregator makes an educated guess regarding a global matrix, and sends it to a group of users. Each user in this group computes the compression of their private data onto the guessed global matrix, samples a singular value, and then introduces noise to this value before reporting it to the aggregator. The noised singular values from users are aggregated to reconstruct an estimated singular vector, utilizing LDP protocols

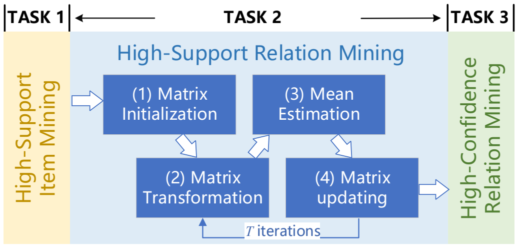

TASK 1
TASK 2
TASK 3
High-Support Item Mining
High-Support Relation Mining
(1) Matrix Initialization
(3) Mean Estimation
(2) Matrix Transformation
(4) Matrix updating
T iterations
High-Confidence Relation Mining

Figure 2: Workflow of LDP-RM

for mean estimation. This process enables the reconstruction of an improved estimated global matrix. With this matrix, the aggregator selects a new group of users to initiate a new iteration to finally obtain an accurate estimation of the true global matrix.

4.2 Details

We now detail the workflow of LDP-RM. A flowchart illustrating LDP-RM can be found in Fig. 2. In LDP-RM, the problem of RM under LDP is decomposed into the following three distinct tasks:
Task 1. "High-Support Item Mining", finds the top-k items in support and estimate their support.
Task 2. "High-Support Relation Mining" finds the top-k_s relations in support which constitute a candidate set.
Task 3. "High-Confidence Relation Mining" finds the top-k_c relations in confidence from the candidate set.

To save privacy budget, LDP-RM randomly divides all users into three groups corresponding to three tasks, with each user queried only once in each task. We denote the number of users in each group as n_1, n_2, and n_3, respectively.

4.2.1 TASK 1. High-Support Item Mining. In the first task, LDP-RM identifies the top-k items in support and estimates their support values. To achieve this, the aggregator interacts with the first group of users and employs the SVIM protocol. Each participating user perturbs their private items and randomly reports an item. Subsequently, the aggregator collects the reports and estimates the support of all items in the item domain $\mathbf{X}$ using the aggregate algorithm defined in Equation (7). The aggregator then selects the top-k items in support and adds them to a set denoted as $\mathbf{Y}$. This step aims to reduce the item domain's size from d to k and reduce the size of the relation domain from d^2 to k^2.

4.2.2 TASK 2. High-Support Relation Mining. In the second task, LDP-RM finds top-k_s relations in support from $\mathbf{Y}^2$, and estimates their support. To accomplish this task, the aggregator interacts with the second group of users through the following 4 stages: (1) Matrix Initialization. The aggregator initializes an aggregator matrix based on the top-k items in support. Each user then initializes their local user matrix. (2) Matrix Transformation. The aggregator performs singular value decomposition of the aggregator matrix. Each user subsequently transforms her private user matrix into a private singular vector. (3) Mean Estimation. Each user perturbs an element sampled from her private singular vector. The aggregator collects all perturbed reports and estimates the mean vector. (4) Matrix updating. The aggregator updates his aggregator matrix based on the mean vector. To accomplish the second task, stages 2 to 4 are performed for several rounds. The candidate set is constructed based on the final aggregator matrix. We now detail the four stages.

STAGE 1. Matrix Initialization. In this stage, the aggregator matrix and user matrices are initialized. The aggregator matrix represents the support of all item in $\mathbf{Y}$ and that of all relations in $\mathbf{Y}^2$, and a user matrix represents the user's possession of all items in $\mathbf{Y}$ and all relations in $\mathbf{Y}^2$. This stage is composed of two steps.

Step 1. Aggregator Matrix Initialization. This step is performed at the aggregator side. The aggregator constructs an aggregator matrix denoted as $\mathbf{M}_{agg}$. The diagonal elements represent the support of items, while the off-diagonal elements represent the support of relations. The initialization of the aggregator matrix is based on the assumption that items are independent of each other:

$$\widetilde{s}(x_a,x_b) = \begin{cases} \widehat{s}(x_a), & \text{if } x_a = x_b, \\ \widehat{s}(x_a)\widehat{s}(x_b), & \text{otherwise.} \end{cases} \tag{10}$$

where x_a and x_b are items in $\mathbf{Y}$, $\widehat{s}(\cdot)$ denotes the estimated support of an item and is obtained in Task 1, and $\widetilde{s}(\cdot,\cdot)$ represents the guessed support of a relation.

Step 2. User Matrix Initialization. In this step, the aggregator first sends $\mathbf{Y}$ (the set of top-k items in support) to the second group of users. Each user in this group constructs her user matrix of order $k \times k$ (denoted as $\mathbf{I}$) to indicate her possession of items in $\mathbf{Y}$ and that of relations in $\mathbf{Y}^2$. Each element in $\mathbf{I}$ is a binary value and is based on the user's private itemset $\mathbf{x}$, private relation set $\mathbf{w}$, and the global public set $\mathbf{Y}$:

$$\mathbf{I}_{\mathrm{Idx}(x_a),\mathrm{Idx}(x_b)} = \begin{cases} 1, & \text{if } (x_a,x_b) \in \mathbf{w} \text{ or} \\ & (x_a = x_b \text{ and } x_a \in \mathbf{x}) \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

where x_a and x_b are two items in $\mathbf{Y}$, $\mathrm{Idx}(\cdot)$ calculates a unique index number in [1,k] for each item in $\mathbf{Y}$.

STAGE 2. Matrix Transformation. In this stage, the aggregator performs singular value decomposition (SVD) [17] on the aggregator matrix $\mathbf{M}_{agg}$ and calculates a low rank approximation $\mathbf{M}_r$. Each user then calculates a singular vector $\mathbf{S}$, which is transformed from her user matrix $\mathbf{I}$ based on a variant of SVD. This stage comprises three steps.

Step 1. Singular Value Decomposition. In this step, the aggregator performs SVD on $\mathbf{M}_{agg}$ as follows:

$$\mathbf{M}_{agg} = \mathbf{U}_k \mathbf{\Sigma}_k (\mathbf{V}_k)^\top = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i (\mathbf{v}_i)^\top . \tag{12}$$

Where $\mathbf{U}_k$ and $\mathbf{V}_k$ are unitary matrices of order $k \times k$, $(\mathbf{V}_k)^\top$ is the conjugate transpose of $\mathbf{V}_k$. $\mathbf{u}_i$ is a column of $\mathbf{U}_k$, $(\mathbf{v}_i)^\top$ is a row of $(\mathbf{V}_k)^\top$, and they are called left- and right-singular vectors. $\mathbf{\Sigma}_k$ is a $k \times k$ diagonal matrix whose element σ_i is called the singular value of the decomposed matrix. All the singular values are non-negative real numbers, and are arranged in descending order from top-left to bottom-right.

Step 2. Low Rank Approximation. In this step, the aggregator specifies a positive integer r and calculates the best rank-r approximation of $\mathbf{M}_{agg}$, denoted as $\mathbf{M}_r$, defined as:

$$\mathbf{M}_r \equiv \arg\min_{\mathbf{M}} \ \|\mathbf{M}_{agg} - \mathbf{M}\|, \tag{13}$$

$$\text{s.t.: } \mathrm{Rank}(\mathbf{M}) \leq \mathrm{Min}\left(r, \mathrm{Rank}(\mathbf{M}_{agg})\right). \tag{14}$$

Where $\|\cdot\|$ calculates the Frobenius norm of a matrix, $\mathrm{Rank}(\cdot)$ calculates the rank of a matrix, $\mathrm{Min}(\cdot,\cdot)$ identifies the smaller of two integers.

According to the Eckart-Young-Mirsky theorem [28] (described in Appendix E), the optimal rank-r approximation of $\mathbf{M}_{agg}$ under the Frobenius norm is given by:

$$\mathbf{M}_r = \mathbf{U}_r \Sigma_r \mathbf{V}_r^\top = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top . \tag{15}$$

Where $\mathbf{U}_r$ is a matrix of order $k \times r$, identical to $\mathbf{U}_k[1:r]$. And $\mathbf{V}_r^\top$ is a matrix of order $r \times k$, identical to $\mathbf{V}_k^\top[1:r]$.

Step 3. Singular Vector Generation. In this step, the aggregator first sends $\mathbf{U}_r$ and $\mathbf{V}_r$ to the users. Then, each user generates her singular vector, denoted as $\mathbf{S}$ of size r.

The calculation of the singular vector is as follows. Since $\mathbf{U}_r$ is not square, its inverse does not exist. However, the left-singular vectors $\mathbf{u}_1, \ldots, \mathbf{u}_r$ are all orthogonal vectors, and $\mathbf{U}_r$ has full column rank. Consequently, we utilize the Moore-Penrose generalized inverse of $\mathbf{U}_r$, given by:

$$(\mathbf{U}_r)^+ = ((\mathbf{U}_r)^\top \mathbf{U}_r)^{-1} (\mathbf{U}_r)^\top = (\mathbf{U}_r)^\top . \tag{16}$$

Similarly, $(\mathbf{V}_r)^\top$ has full row rank, and its Moore-Penrose generalized inverse is given by:

$$(\mathbf{V}_r)^{\top +} = \mathbf{V}_r ((\mathbf{V}_r)^\top \mathbf{V}_r)^{-1} = \mathbf{V}_r . \tag{17}$$

Based on the generalized inverse vectors, each user calculates her singular vector by:

$$\mathbf{S} = \mathrm{Diagonal}(\mathbf{U}_r^\top \mathbf{I} \mathbf{V}_r) = [s_l]_r = \left[\mathbf{u}_l^\top \cdot \mathbf{I} \cdot \mathbf{v}_l\right]_r , \tag{18}$$

where $\mathrm{Diagonal}(\cdot)$ calculates the diagonal of a matrix, $\mathbf{S}$ is a vector of size r. We name each element of this vector as the pseudo singular value, denoted as s_l, where $1 \leq l \leq r$.

STAGE 3. Mean Estimation. In this stage, the mean singular vector is estimated using a sampling mechanism proposed in [43] and a HM protocol proposed in [39]. Each user perturbs one pseudo singular value s_l sampled from her singular vector $\mathbf{S}$ and reports it to the aggregator. The aggregator then collects all perturbed reports and estimates the mean vector. This stage consists of four steps.

Step 1. Query Value Sampling. In this step, the aggregator samples a positive integer l from the range [1,r] for each user, and queries this user the l^{th} value (i.e., s_l) in her singular vector. The calculation of s_l is only related to $\mathbf{u}_l$ and $\mathbf{v}_l$ according to Equation (18), so the aggregator only needs to send l, $\mathbf{u}_l$ and $\mathbf{v}_l$ to the user, rather than the entire matrices $\mathbf{U}_r$ and $\mathbf{V}_r$.

Step 2. Value Domain Calculation. This step is performed at the user side. After receiving a query on a pseudo singular value s_l, the user calculates the domain of this value. This allows her to perturb the value later to meet the constraint of LDP. The domain of s_l is only related to $\mathbf{u}_l^\top$ and $\mathbf{v}_l$ and is independent of any value in the user matrix $\mathbf{I}$. If we represent $\mathbf{u}_l^\top$ as $\{u_{l1}, u_{l2}, \cdots, u_{lk}\}$, and $\mathbf{v}_l$ as $\{v_{l1}, v_{l2}, \cdots, v_{lk}\}^\top$, a sampled value s_l can be expanded as:

$$s_l = (\mathbf{u}_l)^\top \mathbf{I} \mathbf{v}_l = \sum_{a=1}^{k} \sum_{b=1}^{k} u_{la} \cdot \mathbf{I}_{a,b} \cdot v_{lb} . \tag{19}$$

Since each value in $\mathbf{I}$ is either 0 or 1, it is evident that the minimum possible value of a given pseudo singular value can be achieved when the following conditions are met:

$$\mathbf{I}_{a,b} = \begin{cases} 1, & \text{if } u_{la} \cdot v_{lb} \leq 0; \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

Thus we have:

$$\min(s_l) = \sum_{a=1}^{k} \sum_{b=1}^{k} \frac{u_{la} \cdot v_{lb} - |u_{la} \cdot v_{lb}|}{2} , \tag{21}$$

where $|\cdot|$ calculates the absolute value of a real number. Similarly, the maximum possible value is:

$$\max(s_l) = \sum_{a=1}^{k} \sum_{b=1}^{k} \frac{u_{la} \cdot v_{lb} + |u_{la} \cdot v_{lb}|}{2} . \tag{22}$$

Let $\mathrm{D}_{\mathbf{S}}(s_l)$ denote the domain of the s_l, it is given by:

$$\mathrm{D}_{\mathbf{S}}(s_l) = [\min(s_l), \max(s_l)] . \tag{23}$$

Step 3. Singular Value Perturbation. This step is performed at the user side. Firstly, each user calculates the mapped pseudo singular value $\overrightarrow{s_l}$, which is the mapping of s_l from $\mathrm{D}_{\mathbf{S}}(s_l)$ to the range [-1,1]:

$$\overrightarrow{s_l} = 2 \times \frac{s_l - \min(s_l)}{\max(s_l) - \min(s_l)} - 1 . \tag{24}$$

Afterward, to meet the constraint of LDP, the user applies the perturbation algorithm of the HM protocol to perturb $\overrightarrow{s_l}$ as:

$$(\overrightarrow{s_l})^* = \Psi_{HM(\varepsilon)}(\overrightarrow{s_l}), \tag{25}$$

and then reports the perturbed value $(\overrightarrow{s_l})^*$ to the aggregator.

Step 4. Mean Vector Estimation. This step is performed at the aggregator side. The aggregator collects all reported values sent from users in Step 3 and estimates a mean value for each pseudo singular value using the aggregation algorithm of the HM protocol:

$$\widehat{(\overrightarrow{s_l})^*} = r * \Phi_{HM}(\overrightarrow{s_l}) = \frac{r}{\eta}\sum_{j=1}^{\eta}(\overrightarrow{s_l}^{\,j})^*, \qquad (26)$$

where η is the number of users participating in the mean estimation of the same query value (i.e., received the same l in Step 1). The size of the singular vector (also the rank of approximation) is denoted by r, which is used to correct the sampling error. $(\overrightarrow{s_l}^{\,j})^*$ represents the perturbed pseudo singular value uploaded by the j^{th} user. Subsequently, the aggregator remaps each estimated mean value from [-1,1] back to the domain $D_{\mathbf{S}}(s_l)$:

$$\widehat{s_l} = \min(s_l) + \left(\widehat{(\overrightarrow{s_l})^*} + 1\right)(\max(s_l) - \min(s_l))/2. \qquad (27)$$

Finally, the aggregator obtains the mean singular vector:

$$\widehat{\mathbf{S}} = [\widehat{s_1}, \widehat{s_1}, ..., \widehat{s_r}] \qquad (28)$$

STAGE 4. Matrix Updating. This stage is performed at the aggregator side. After estimating the mean singular vector $\widehat{\mathbf{S}}$, the aggregator estimates the aggregator matrix as follows:

$$\mathbf{M}'_{agg} = \mathbf{U}_r \,\mathrm{Diag}(\widehat{\mathbf{S}})\mathbf{V}_r^{\top} = \sum_{l=1}^{r} \widehat{s_l} \cdot \mathbf{u}_l \cdot \mathbf{v}_l^{\top}. \qquad (29)$$

Where $\mathrm{Diag}(\widehat{\mathbf{S}})$ diagonalizes $\widehat{\mathbf{S}}$ into a diagonal matrix.

Note that $\mathbf{M}'_{agg}$ may not be an accurate estimation since the initial matrix $\mathbf{M}_{agg}$ (in Stage 1) is not precise. To address this issue, LDP-RM performs Stage 2 to Stage 4 iteratively, using $\mathbf{M}'_{agg}$ in the current iteration as the new initial matrix $\mathbf{M}_{agg}$ in the next iteration. If T iterations are employed, the n_2 users participating in the second task should be divided into T subgroups, each with n_2/T users. All n_2 users in the second group participate in Stage 1, while only one subgroup of users takes part in Stage 2 to 4 in each iteration. In the final (i.e., T^{th}) iteration, the final estimated aggregator matrix is obtained, and the aggregator selects the top-k_s relations in support and moves them to a candidate set, denoted as $\mathbb{Y}$.

4.2.3 TASK 3. High-Confidence Relation Mining.

In the third task, LDP-RM identifies top-k_c relations in confidence from the candidate set $\mathbb{Y}$. The aggregator interacts with the third group of users and initiates an SVIM protocol. Each participating user randomly selects a private relation, perturbs it, and reports the perturbed value. The aggregator then estimates the support of all relations in $\mathbb{Y}$:

$$\Phi_{\mathrm{SVIM}(\mathbb{Y},\varepsilon,n_3,k_s)} = \{(\mathbf{w}, \Phi_{\mathrm{PSFO}(\ell,\varepsilon)}(\mathbf{w}))|\mathbf{w} \in \mathbb{Y}\}, \qquad (30)$$

Based on the support of candidate relations in $\mathbb{Y}$ and the support of related items in $\mathbf{Y}$ (obtained in Task 1), the aggregator computes the confidence of each candidate relation. Finally, the aggregator selects the top-k_c relations in confidence from $\mathbb{Y}$ as the output of LDP-RM.

5 Analysis

In this section, we start with a proof that LDP-RM satisfies ε-LDP, then analyze the estimation error and bias of it.

5.1 Privacy

LDP-RM satisfies ε-LDP defined on users' items and relations. This is because, each task in LDP-RM involves the use of protocols that satisfy ε-LDP, namely, either a SVIM protocol or a HM protocol. Thus, LDP-RM satisfies ε-LDP. A rigorous proof is provided in Appendix F.

5.2 Utility

The utility of the results discovered by LDP-RM is primarily influenced by two factors: estimation error and bias.

Estimation Error. Estimation error is the difference between the true value and the estimated value. LDP-RM employs the HM protocol (which is built on PM and Duchi) to estimate the mean singular vector, and employs the SVIM protocol (which is built on PSFO, GRR and OLH) to estimate the support of items/relations. The L_∞ error of HM is bounded by $O_1 = O(\sqrt{r\log(r/\beta)}/\varepsilon\sqrt{n})$ with probability at least $1-\beta$, according to [39]. The L_∞ error of SVIM is bounded by $O_2 = O(\ell\sqrt{\log(\ell/\beta)}/\varepsilon\sqrt{n})$ with probability at least $1-\beta$. The proof can be found in Appendix H.

Estimation Bias. Estimation bias is the deviation from the true value during estimation. In LDP-RM, the aggregator is tasked with estimating the best rank-r approximation as opposed to the complete full-rank matrix. This inherently introduces a degree of bias in the process of recovering the true matrix. To mitigate this bias, one may contemplate the use of a larger value for the parameter r. However, it's important to note that opting for a larger value of r also implies an increased number of pseudo singular values that need to be estimated. Consequently, there exists a trade-off between diminishing the bias and minimizing the error. Given a bias bound denoted by θ provided by the aggregator, LDP-RM calculates the smallest possible value of r that satisfies this bound according to the Eckart-Young-Mirsky Theorem (described in Appendix E), as follows:

$$\min r, \text{ subject to: } \sum_{i=1}^{r}\sigma_i \Big/ \sum_{i=1}^{k}\sigma_i \geq \theta. \qquad (31)$$

Compared to the estimation error, the bias has a minimal impact on the final results. LDP-RM strikes a balance by selecting a relatively small r. The value of r can also be dynamically adjusted in different iterations, e.g., a smaller value in early iterations and a larger value in later iterations.

Biased Estimation Vs. Unbiased Estimation. We now explain why we employ a biased estimation method rather than an unbiased one. Consider an algorithm called HM-RM, which differs from LDP-RM in that each user in HM-RM generates a matrix $\mathbf{P} = \mathbf{U}_k^\top \mathbf{I} \mathbf{V}_k$ in Stage 2, rather than just a vector $\mathbf{S}$. If HM-RM is directly used, each user must perturb and report k^2 elements in $\mathbf{P}$, and the aggregator estimates a mean matrix $\widehat{\mathbf{P}}$ by using the HM protocol. This results in L_∞ error bounded by $O_3 = O(k\sqrt{\log(k^2/\beta)}/\varepsilon\sqrt{n})$ with probability at least $1-\beta$. We prove that HM-RM is unbiased in Appendix G. Our experiments in Sec. 7.2.1 and 7.2.2 demonstrate that the biased LDP-RM significantly outperforms the unbiased HM-RM.

5.3 Computational Overhead

The computational overhead of LDP-RM comprises three distinct components. This is because LDP-RM divides users into three groups.Moreover, within the second group (Task 2 participants), users are further subdivided into T subgroups across T iterations. Importantly, each user undergoes only one query by the aggregator, rendering the user-side computational overhead independent of task or iteration round.

Specifically, for **Task 1** and **Task 3**, LDP-RM employs two SVIM protocols with n_1 and n_3 users participating in these tasks, respectively. The user-side computational overhead for these two tasks amounts to $O(\log d)$ and $O(\log k_s)$, while the aggregator-side computational overhead is characterized by $O(n_1 \log d)$ and $O(n_3 \log k_s)$, respectively. For **Task 2**, n_2 users are involved, and each user computes a pseudo singular value s_l, resulting in a per-user computational overhead of $O(k^2)$. Subsequently, users employ the HM protocol to perturb and report s_l, incurring a computational overhead of $O(\log r)$. Consequently, the total computational overhead for each user in the second group is $O(k^2 + \log r)$. As the aggregator needs to conduct SVD and update the matrix in each iteration, incurring a computational overhead of $2k^2$, the aggregator-side computational overhead for Task 2 reaches $O(n_2(k^2+\log r) + 2Tk^2)$. Thus, the cumulative aggregator-side computational overhead is given by:

$$O(n_1 \log d + n_2(k^2 + \log r) + 2Tk^2 + n_3 \log k_s). \tag{32}$$

We summarize the above analysis in Tab. 2. A recommended setting on all the parameters can be found in Tab. 4 of Appendix I. Typically, d and k_s are often large, whereas r, T and k are often small. With this setting, our proposed LDP-RM is slightly slower than SVSM [43] and PCKV [18], and superior to CALM [49], in computational overhead while LDP-RM is the only approach that can mine all types of relations.

6 Generalizing LDP-RM

In this section, we first propose an improvement to LDP-RM that enables the discovery of relations among more than two items. We then introduce an advanced frequency oracle on large domains that utilizes LDP-RM.

6.1 Mining Relations Comprising More Items

We now improve LDP-RM to identify cascading relations comprising more than two items, e.g., $((x_a, x_b), x_c)$. Modifications are made in Stage 1 and Stage 4 of Task 2. In Stage 1, after the aggregator identifies the top-k frequent items, he guesses the support of each possible relations by multiplying the support of any number of items. If the guessed support of a relation surpasses that of an item in $\mathbf{Y}$, the relation is treated as a new item and moved into $\mathbf{Y}$, while the item with the lowest estimated support is moved out from $\mathbf{Y}$. In Stage 4, before the aggregator updates the aggregator matrix $\mathbf{M}_{agg}$, he verifies whether a relation and an item exist such that the estimated support of the relation is larger than that of the item. If so, the set of high-support items $\mathbf{Y}$ is reconstructed by selecting the top-k frequent items/relations. After T iterations, the final top-k_c relations can consist of two items or more items.

6.2 Frequency Oracle on Large Domains

Consider a frequency estimation scenario under LDP where there is a large number of values to be estimated. In this context, LDP-RM can serve as a fundamental FO, which we refer to as SVD-FO. SVD-FO encodes a unidimensional item domain into a bidimensional one and utilizes LDP-RM for frequency estimation. Given an original item domain $\mathbf{X} = \{x_1, x_2, \cdots, x_d\}$, SVD-FO examines a virtualized domain $\mathbf{V}$, where $|\mathbf{V}| = \lceil d^{1/2} \rceil$. An one-to-one mapping $e(\cdot)$ from $\mathbf{X}$ to $\mathbf{V}^2$ is then employed to encode any item $x_i \in \mathbf{X}$ into a tuple $e(x_i) = (a,b) \in \mathbf{V}^2$. Subsequently, LDP-RM is applied to estimate the matrix $(\mathbf{M}_{a,b}) \in \mathbb{R}^{|\mathbf{V}| \times |\mathbf{V}|}$, in which each element $\mathbf{M}_{a,b}$ represents the frequency of an item $x_i = e^{-1}(a,b)$.

SVD-FO surpasses the state-of-the-art FO, specifically PSFO, in estimating item frequencies within high-dimensional contexts. This performance improvement stems from a reduction in the number of values to be estimated, decreasing from d to $r \times T < \lceil d^{1/2} \rceil \times T < d$, where r represents the approximation rank and T denotes number of iterations. However, in cases with a small domain size, SVD-FO may not necessarily outperform PSFO.

7 Evaluation

In this section, we first introduce the experiment setup including computing environment, compared methods, datasets and evaluation metrics, and then present results of experiments on LDP-RM in various data mining tasks.

Table 2: Computational Overhead of LDP-RM and Existing Techniques.

Methods	**User Side**			**Aggregator Side**
	Group $\sharp 1$ (n_1)	**Group** $\sharp 2$ (n_2)	**Group** $\sharp 3$ (n_3)	
LDP-RM	$O(\log d)$	$O(k^2 + \log r)$	$O(\log k_s)$	$O(n_1 \log d + n_2(k^2 + \log r) + 2Tk^2 + n_3 \log k_s)$
SVSM [43]	$O(\log d)$	$O(\log k_s)$		$O(n_1 \log d + (n_2 + n_3) \log k_s)$
CALM [50]	$O(\log d)$	$O(2^l)$		$O(n_1 \log d + (n_2 + n_3) 2^l)$
PCKV [18]	$O(\log d)$	$O(\log k_s)$		$O(n_1 \log d + (n_2 + n_3) \log k_s)$

7.1 Experiment Setup

Environment. All experiments are conducted on a desktop with an Intel i7-1165G7 2.80GHz CPU and 16GB memory. Each experiment is performed 20 times, with mean and standard deviation reported.

Compared Methods. LDP-RM is the first method which addresses the problem of relation mining under LDP. We consider four compared methods, including **(1) SVIM** [43] which mines high-support items, **(2) SVSM** [43] which mines high-support itemsets, **(3) CALM** [50] which estimates marginal probabilities, **(4) PCKV** [18] which estimates support of a key and the mean value of that key, and **(5) HM-RM**, an unbiased relation mining method discussed in Sec. 5.2. All algorithms are implemented using Python 3.9.

Datasets. We conduct experiments on four public datasets.

(1) The IFTTT Dataset: We construct a binary relational dataset containing 354 non-repetitive items extracted from top-200 applets in IFTTT in 2023 [19]. One applet can be regarded as a relation which consists of two items. We consider 300,000 users in this dataset, and the items and relations possessed by each user is generated based on the popularity of each applet according to official statistics. This dataset is used to simulate a general scenario of relation mining.

(2) The Movie Dataset [29]: A movie viewing dataset containing 5,000 items and 400,000 users. Each user possesses temporal relations in the form of (x_a, x_b), signifying that after viewing movie x_a, the user subsequently watches movie x_b. This dataset is used to simulate a scenario of relation mining.

(3) The Modified IFTTT Dataset: A dataset modified from the IFTTT dataset which contains 354 items and 800,000 users. Within this dataset, relations are among items and are in the form of $((x_a, x_b), x_c)$. All relations are generated based on the relationship between IFTTT applets. This dataset is utilized to simulate the discovery of relations among items.

(4) The Retail Dataset [7]: A retail market basket dataset containing 2,603 items and 300,000 users. Each user possess several items. This dataset is used to simulate a scenario of association rule mining.

(5) The Kosarak Dataset [16]: A website click dataset containing 41,270 items and 990,002 users. This dataset is used to simulate a scenario of item mining on a large domain.

Evaluation Metrics. To measure estimation accuracy, we employ 3 different metrics. We denote the set of true relations with top-k_s support and top-k_c confidence as $\mathbb{W}_t$, and the set of estimated ones as $\mathbb{W}_e$. Consequently, $\mathbb{W}_t \cap \mathbb{W}_e$ represents the true positive instances (TP).

(1) F-Measure (F1) is a widely used metric to measure accuracy, which is defined as the harmonic mean of precision (denoted as P) and recall (denoted as R):

$$\text{F1} \triangleq \frac{2}{1/P + 1/R} = \frac{2PR}{P + R}, \tag{33}$$

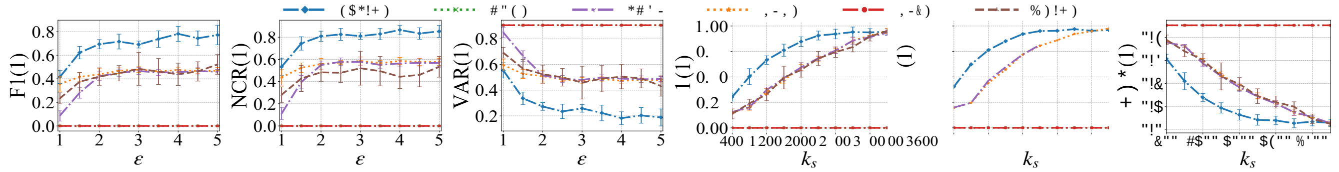
$$\text{where } P \triangleq \frac{|\mathbb{W}_t \cap \mathbb{W}_e|}{|\mathbb{W}_e|}, R \triangleq \frac{|\mathbb{W}_t \cap \mathbb{W}_e|}{|\mathbb{W}_t|}. \tag{34}$$

(2) Normalized Cumulative Ranks (NCR) is a metric used to evaluate the level of accuracy achieved, which takes into consideration that the benefit of mining relations with higher confidence ranks should be greater than those with lower ranks. It is defined as:

$$\text{NCR} = \sum_{\mathbf{w} \in \mathbb{W}_e \cap \mathbb{W}_t} q(\mathbf{w}) \Bigg/ \sum_{\mathbf{w} \in \mathbb{W}_t} q(\mathbf{w}), \tag{35}$$

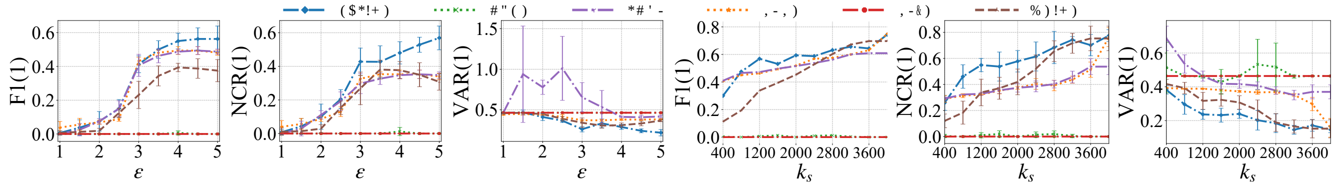
where $q(\cdot)$ is a quality function defined as follows. There are

e ywi$99.68.97j /T1_3 7.372 Tf 8.118 -1.494 Td (t)Tj /T1_



(a) F1 on ε (b) NCR on ε (c) VAR on ε (d) F1 on k_s (e) NCR on k_s (f) VAR on k_s

Figure 3: Performance of LDP-RM and existing techniques in mining relations comprising 2 items from IFTTT dataset. Results of Experiment E1 are in (a), (b) and (c), that of E2 are in (d), (e) and (f). We set $k = 64, k_c = 32, \theta = 0.5, T = 5$ for both E1 and E2, with $k_s = 1600$ in E1, and $\varepsilon = 4$ in E2.



(a) F1 on ε (b) NCR on ε (c) VAR on ε (d) F1 on k_s (e) NCR on k_s (f) VAR on k_s

Figure 4: Performance of LDP-RM and existing techniques in mining relations comprising 2 items from Movie dataset. Results of Experiment E3 are in (a), (b) and (c), that of E4 are in (d), (e) and (f). We set $k = 64, k_c = 32, \theta = 0.5, T = 5$ for both E3 and E4, with $k_s = 1600$ in E3, and $\varepsilon = 4$ in E4.

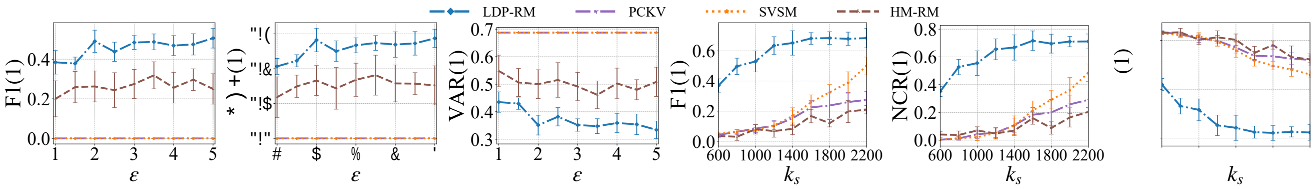
7.2.1 Experiments in mining relations between items.

We conduct four experiments denoted as **E1** to **E4** on two datasets to validate the performance of LDP-RM in mining relations. The IFTTT dataset is used in **E1** and **E2**. The Movie dataset is used in **E3** and **E4**. Since relation mining require separate estimation of support for items and that for relations, we make the following settings to achieve a fair comparison among different methods. For LDP-RM and HM-RM, it accomplishes its Task 1, Task 2, and Task 3 by communicating with n_1, n_2, and n_3 users, respectively. For SVIM, SVSM, CALM, and PCKV, they communicate with n_1 users to estimate support of items, and with $n_2 + n_3$ users to estimate support of relations. For all methods, the ratio of n_1, n_2, n_3 is set to 30%, 35%, 35% respectively. The parameter setting in the following experiments are discussed in Appendix I.

(E1) The results on varying ε are illustrated in Figs. 3a, 3b, 3c. Among all methods, LDP-RM always performs the best. SVSM is better than the PCKV when $\varepsilon < 2$ because PCKV requires allocating an additional privacy budget for mean estimation. SVIM and CALM struggle to identify target relations due to the excessive value estimation requirements, resulting in insufficient privacy budget and user participation. The unbiased HM-RM is inferior to the biased LDP-RM, which validates our analysis in Sec. 5.2.

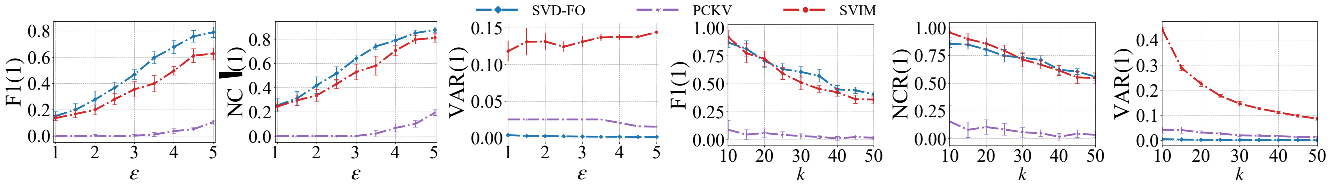
(E2) The results on varying k_s are illustrated in Figs. 3d, 3e, 3f. As k_s increases, F1 and NCR also increases, while VAR decreases. This is because a larger k_s means a smaller support threshold which decreases the difficulty in discovering high-support and high-confidence relations. Among all methods, LDP-RM always performs the best.

(E3) The results on varying ε are illustrated in Figs. 4a, 4b, 4c.4c



(a) F1 on ε (b) NCR on ε (c) VAR on ε (d) F1 on k_s (e) NCR on k_s (f) VAR on k_s

Figure 5: Performance of LDP-RM and existing techniques in mining cascading relations (3 items) from modified IFTTT dataset. Results of Experiment E5 are in (a), (b) and (c), that of E6 are in (d), (e) and (f). We set $k = 64, k_c = 32, \theta = 0.5, T = 5$ for both E5 and E6, with $k_s = 1600$ in E5, and $\varepsilon = 4$ in E6.



(a) F1 on ε (b) NCR on ε (c) VAR on ε (d) F1 on k (e) NCR on k (f) VAR on k

Figure 6: Performance of LDP-RM and existing techniques in mining frequent items from Kosarak dataset. Results of Experiment E9 are in (a), (b) and (c), that of E10 are in (d), (e) and (f). We set $\theta = 0.5, T = 5$ for both E9 and E10, with $k = 32$ in E9, and $\varepsilon = 4$ in E10.

Table 3: Performance of LDP-RM and existing techniques in mining association rules from Retail and Retail* dataset. We set $\varepsilon = 4, k = 64, k_s = 1600, k_c = 32, \theta = 0.5, T = 5$.

Method	**Retail**			**Retail***		
	F1	NCR	VAR	F1	NCR	VAR
LDP-RM	0.558	0.640	0.246	0.546	0.654	0.180
SVSM	0.554	0.435	0.317	0.463	0.396	0.308
CALM	0.183	0.195	0.568	0.108	0.159	0.461
SVIM	0	0	0.603	0	0	0.491
PCKV	0	0	0.603	0	0	0.491
HM-RM	0.325	0.329	0.414	0.379	0.437	0.285

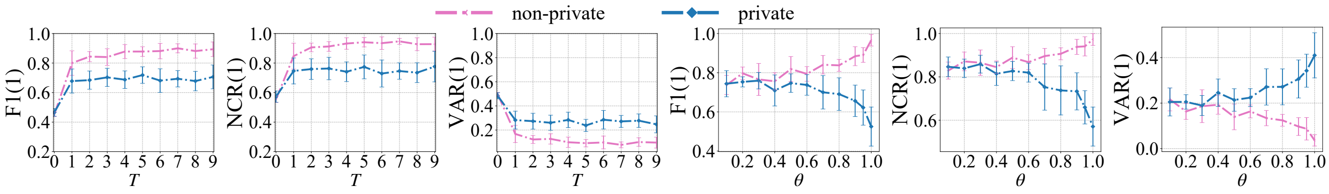
of F1, while exhibiting superiority in NCR and VAR. The reason for SVSM's commendable performance lies in the fact that the assumption of high-support relations being composed of high-support items remains valid within the original Retail dataset. CALM, SVIM, PCKV and HM-RM perform poorly.

(E8) This experiment investigates the performance of different methods in association rule mining when the previous assumption in **E7** is not held. To relax this assumption, we slightly modify the Retail dataset by excluding data related to the top-8 items in support, since well-known relations comprising high-support items may not be the targets of the mining task. We denote the modified dataset as **Retail***. All parameters are set same as in **E7**, and the results is presented in Tab. 3 (right part). There is a significant decline in the accuracy achieved by SVSM, while the accuracy achieved by LDP-RM remains at an adequate level.

7.2.4 Experiments in item mining.

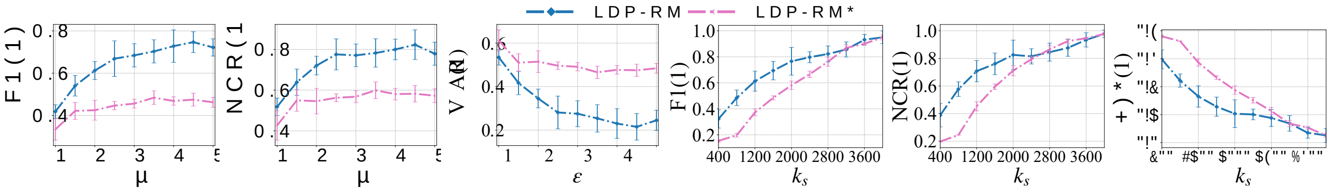
The Kosarak dataset is used to evaluate the performance of SVD-FO (described in Sec. 6.2). To simulate the scenario of item mining in a large domain with a small number of users, we randomly select 50,000 users from the dataset. The number of users is comparable to the domain size $d = 41,270$. SVIM and PCKV, both employing PSFO, are selected for comparison.

(E9) The results on varying ε are illustrated in Figs. 6a, 6b, 6c.



(a) F1 on T (b) NCR on T (c) VAR on T (d) F1 on θ (e) NCR on θ (f) VAR on θ

Figure 7: Ablation Experiments: Performance of private and non-private versions of LDP-RM in mining relations from IFTTT dataset. Results of Experiment E11 are in (a), (b) and (c), that of E12 are in (d), (e) and (f). We set $\varepsilon = 4$, $k = 64$, $k_s = 1600$, $k_c = 32$ for both E11 and E12 with $\theta = 0.5$ in E11, and $T = 5$ in E12.



(a) F1 on ε (b) NCR on ε (c) VAR on ε (d) F1 on k_s (e) NCR on k_s (f) VAR on k_s

Figure 8: Ablation Experiments: Performance of LDP-RM and its variant LDP-RM* that without iterative process in mining relations from IFTTT dataset. Results of Experiment E13 are in (a), (b) and (c), that of E14 are in (d), (e) and (f). We set $k = 64, k_c = 32, \theta = 0.5, T = 5$ for both E13 and E14, with $k_s = 1600$ in E13, and $\varepsilon = 4$ in E14.

(E12) The results on varying bias bound θ are illustrated in Figs. 7d, 7e, 7f. The rank of approximation r increases as θ increases, according to Equation (31). For non-private LDP-RM, F1 and NCR is nearly perfect when $\theta = 1$ (i.e., $r = k = 64$). For private LDP-RM, the accuracy starts to decrease when $\theta > 0.5$. This is because the larger the θ is, the more elements are needed to be estimated.

7.2.6 Comparison of iterative and non-iterative algorithms.

The IFTTT dataset is used to illustrate the effectiveness of the iterative process. We denote the variant LDP-RM without iteration as LDP-RM*.

(E13) The results on varying ε are illustrated in Figs. 8a, 8b, 8c. LDP-RM significantly outperforms LDP-RM* owing to the fact that LDP-RM* relies on an imprecise global matrix that is entirely based on guesswork to produce results. In contrast, LDP-RM continually aggregates user information to update the matrix through T rounds of iterations.

(E14) The results on varying k_s are illustrated in Figs. 8d, 8e, 8f. It underscores the importance of the iterative process.

8 Related Work

We are the first to introduce and investigate the problem of relation mining under LDP. Consequently, there has been no prior research focused on this problem. Nonetheless, four lines of research are closely related to our work.

Frequent Item-set Mining under LDP. This line of research focuses on discovering high-support item-sets, with the state-of-the-art technique SVSM [43]. The limitations of using SVSM for relation mining are analyzed in Sec. 3.2 and verified in Sec. 7.2. In addition to SVSM, several other approaches have been proposed. Zhang et al. [49] mine frequent item-sets from perturbed items. Wang et al. [40] estimate the size distribution of frequent item-sets. Afrose et al. [2] mine frequent item-sets by splitting the privacy budget rather than grouping users. Ma et al. [24] utilize FP-tree in combination with Hadamard response to improve the accuracy of the results.

Marginal Release under LDP. This line of research concentrates on estimating marginal probabilities. The state-of-the-art technique, CALM [50], optimizes the trade-off between the number and length of marginals, taking into account the number of users, attributes, and privacy budget. Other techniques skillfully balance privacy and estimation accuracy, employing various techniques, such as Fourier Transformation [9] and Expectation Maximization [34].

Key-Value Collecting under LDP. This line of research focuses on estimating the support and mean value of a key. Ye et al. put forth three methods, namely PrivKV, PrivKVM, and PrivKVM+ [47]. Gu et al. propose a state-of-the-art technique called PCKV [18], which combines a sampling mechanism with GRR [42] and OUE [42], to improve estimation accuracy.

Relation Mining under DP. This line of research is under the constraint of DP in the centralized setting, not LDP. Traditional DP techniques perturb individual data by using the

Laplace mechanism [11], exponential mechanism [26], or some other mechanisms [14, 23]. These mechanisms destroy the relationship among the data values, rendering the perturbed data unsuitable for complex data mining tasks. Several researches have been proposed for relation mining. Li et al. [22] uses a trusted publisher to perturb the support of numerous high-support item-sets, and extracts the top-k subsets in support [5] from the perturbed data. To speed up computation in perturbation and extraction, some techniques such as transaction truncating [8, 36, 48], Markov Chain Monte Carlo [35], and binary estimation [46] have been proposed. Other techniques, such as frequent pattern tree [21] and sequence exponential mechanism [38], aim to save privacy budget. In addition to the techniques that focus on high-support relation mining under DP, Maruseac et al. [25] introduce a method that enables high-confidence relation mining.

9 Conclusion

It is a fundamental problem in performing complex data mining tasks on private data owned by distributed users. To address this problem, we propose LDP-RM, the first general relation mining method under LDP. By employing singular value decomposition and low rank approximation, it reduces the number of values to estimate, thereby improving the final estimation accuracy. Experimental evaluation demonstrates the effectiveness of LDP-RM in mining relations.

Acknowledgement

This research is supported by National Key R&D Program of China Grant No. 2023YFC3605804, National Natural Science Foundation of China (62072098, 62232004, 62072103), US National Science Foundation Awards (1931871, 2325451), Jiangsu Key R&D Program (BE2022065-5, BE2022680), Jiangsu Provincial Key Laboratory of Network and Information Security (BM2003201), Key Laboratory of Computer Network and Information Integration of Ministry of Education of China (93K-9), and Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] Jayadev Acharya, Ziteng Sun, and Huanyu Zhang. Hadamard response: Estimating distributions privately, efficiently, and with little communication. In *The International Conference on Artificial Intelligence and Statistics*, 2019.

[2] Sharmin Afrose, Tanzima Hashem, and Mohammed Eunus Ali. Frequent itemsets mining with a guaranteed local differential privacy in small datasets. In *International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 232–236, 2021.

[3] Apple. Learning with privacy at scale. [online], https://docs-assets.developer.apple.com/ml-research/papers/learning-with-privacy-at-scale.pdf, 2017.

[4] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Thakurta. Practical locally private heavy hitters. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 2288–2296, 2017.

[5] Raghav Bhaskar, Srivatsan Laxman, Adam Smith, and Abhradeep Thakurta. Discovering frequent patterns in sensitive data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 503–512, 2010.

[6] Jonas Böhler and Florian Kerschbaum. Secure multiparty computation of differentially private heavy hitters. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2361–2377, 2021.

[7] Tom Brijs. Retail market basket data set. In *Frequent Itemset Mining Implementations (FIMI)*, 2003.

[8] Xiang Cheng, Sen Su, Shengzhi Xu, Peng Tang, and Zhengyi Li. Differentially private maximal frequent sequence mining. *Computers & Security*, 55:175–192, 2015.

[9] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. Marginal release under local differential privacy. In *International Conference on Management of Data (COMAD)*, pages 131–146, 2018.

[10] John C Duchi, Michael I Jordan, and Martin J Wainwright. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201, 2018.

[11] Cynthia Dwork. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12, 2006.

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

[13] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving

[15] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *arXiv:1503.01214*, 2015.

[16] Bart Goethals. Frequent itemset mining dataset repository. *Frequent Itemset Mining Implementations (FIMI)*, 2003.

[17] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Linear algebra*, 2:134–151, 1971.

[18] Xiaolan Gu, Ming Li, Yueqiang Cheng, Li Xiong, and Yang Cao. PCKV: locally differentially private correlated key-value data collection with optimized utility. In *USENIX Security Symposium*, pages 967–984, 2020.

[19] IFTTT. Top (pageaKtnpImptsop (pageaKt[(of)(pageaKt (23 (.)-358.998[onErle],ng)]TJJJ.7 rgTJ/T2_1 9.963 Tf1 0 0 243.3.019 510.62

privacy. In *IEEE International Conference on Data Engineering (ICDE)*, pages 821–832, 2018.

[38] Ning Wang, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, Yu Gu, and Ge Yu. Privsuper: A superset-first approach to frequent itemset mining under differential privacy. In *IEEE International Conference on Data Engineering (ICDE)*, pages 809–820, 2017.

[39] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. Collecting and analyzing multidimensional data with local

Optimized Local Hashing (OLH). An OLH protocol [42] is a FO protocol that utilizes GRR and is composed of a pair of algorithms: $\langle \Psi_{\text{OLH}}, \Phi_{\text{OLH}} \rangle$. The reporting protocol Ψ_{OLH} maps an input value from a large domain $\mathbf{X}$ into a smaller domain G by first randomly selecting a hash function H from a predefined family of hash functions, then computing the hashed value $H(x)$. Next, a GRR protocol is used to perturb the hashed value. It has been proven in [42] that minimal variance is achieved when $|G| = \lceil e^{\varepsilon} + 1 \rceil$. The definitions of Ψ_{OLH} and Φ_{OLH} are as follows:

$$\Psi_{\text{OLH}(\varepsilon)}(\cdot) \triangleq \langle H, \Psi_{\text{GRR}(\varepsilon)}(H(\cdot)) \rangle,$$
$$\Phi_{\text{OLH}(\varepsilon)}(\cdot) \triangleq \frac{C_H(\cdot) - n/|G|}{e^{\varepsilon}/(e^{\varepsilon} + |G| - 1) - 1/|G|}.$$

Where $\Psi_{\text{GRR}(\varepsilon)}$ is a GRR protocol, $C_H(\cdot)$ counts the number of possible input values that share the same hash result with the user's input value $x \in \mathbf{X}$, n is the total number of users.

C The PSFO Protocol

A Padding-and-Sampling-based Frequency Oracle (PSFO) protocol [43] is specified by the following parameters: an item set $\mathbf{x} \in \mathbf{X}$, a padding length ℓ, and a privacy budget ε. The PSFO adaptively selects a GRR protocol $\langle \Psi_{\text{GRR}}, \Phi_{\text{GRR}} \rangle$
h

F Proof that LDP-RM Satisfies ε-LDP

LDP-RM is proven to satisfy ε-LDP, as follows.

For the first task, the SVIM protocol is utilized, which has already been proven to satisfy ε-LDP [43]. For the second task, the rank of the approximation matrix is determined by the bounded bias threshold θ, and each user reports a pseudo singular value only once. The reported value is perturbed using the HM mechanism, which has also been proven to satisfy ε-LDP [39]. For the third task, the SVIM protocol, which satisfies ε-LDP, is used again. Benefiting from the property of post-processing of LDP [12], the calculation performed by the aggregator does not affect whether ε-LDP is satisfied. Overall, the sub-procedure in LDP-RM for each task satisfies ε-LDP, and since each user only participates in one task, the entire procedure of LDP-RM also satisfies ε-LDP.

G Proof that HM-RM is Unbiased

To prove that HM-RM is an unbiased method, it is only necessary to show that the aggregator matrix $\mathbf{M}'_{agg}$ can be estimated in an unbiased manner in Task 2, since the first and third task are accomplished using the unbiased SVIM protocol.

We assume that the user item-sets in different groups are identically distributed to that of all users. In HM-RM, the j^{th} user generates a matrix $\mathbf{P}^j = \mathbf{U}_k^\top \mathbf{I}^j \mathbf{V}_k$, and she needs to sample one element in the matrix to perturb and report. Since the HM protocol is unbiased, the aggregated mean estimation of $\mathbf{P}$, represented as $\widehat{\mathbf{P}}$, satisfies:

$$\mathbf{E}[\widehat{\mathbf{P}}] = \sum_j \mathbf{P}^j/n = \sum_j \mathbf{U}_k^\top \mathbf{I}^j \mathbf{V}_k/n$$

Let the true matrix be denoted as $\mathbf{M}_{gt}$. Because $\mathbf{U}_k$ and $\mathbf{V}_k$ are both orthogonal matrices, it satisfies $\mathbf{U}_k(\mathbf{U}_k)^\top = \mathbf{E}$. Similar with Equation (29), we have:

$$\begin{aligned}\mathbf{E}[\mathbf{M}'_{agg}] &= \mathbf{E}[\mathbf{U}_k \widehat{\mathbf{P}} \mathbf{V}_k^\top] = \mathbf{U}_k \mathbf{E}[\widehat{\mathbf{P}}](\mathbf{V}_k)^\top \\ &= \mathbf{U}_k(\mathbf{U}_k)^\top \sum_{j=1}^{n} \mathbf{I}^j \mathbf{V}_k (\mathbf{V}_k)^\top/n = \sum_{j=1}^{n} \mathbf{I}^j/n = \mathbf{M}_{gt}\end{aligned}$$

where $\widehat{\mathbf{P}}$ is the aggregated result of $\mathbf{P}$. In HM-RM, users can calculate and report the complete singular value matrix to the aggregator, allowing for support of relations to be obtained in only one iteration.

H L_∞ Error of SVIM

The L_∞ error of SVIM depends on that of PSFO, since PSFO is invoked many times in SVIM. Let P_x be the true support of x and P_x^* be the estimated support of x. We have $P_x^* = \Phi_{\mathrm{PSFO}(\ell,\varepsilon)}(x)$. Since P_x^* is an unbiased estimation of P_x and it invokes GRR or OLH adaptively according to whether the condition of $d < \ell(4\ell-1)e^\varepsilon + 1$ is met, the variance of P_x^* is:

Table 4: Parameter Setting Guideline.

Param.	Recommended setting
ε	4 (default), 2(high privacy protection restrictions)
k	64 (default), 128(large domain)
k_s	1600 (default), recommended range $(1000, 3000)$
k_c	32 (default), recommended range $(32, 64)$
T	5 (default)
θ	0.5 (default), flexible setting based on the dataset

$$\mathrm{Var}(P_x^*) = \begin{cases} n\frac{e^\varepsilon * \ell + d - 1}{(e^\varepsilon - 1)^2} & \text{if } d < \ell(4\ell-1)e^\varepsilon + 1; \\ n\frac{4\ell^2 e^\varepsilon}{(e^\varepsilon-1)^2} & \text{otherwise.} \end{cases}$$

For any given positive integer t, the probability that the L_∞-norm of the difference between P_x^* and P_x (denoted as $||P_x^* - P_x||_\infty$) is greater than t, is calculated as:

$$\begin{aligned}&||P_x^* - P_x||_\infty = Pr[||\frac{1}{n}\sum_{j=1}^{n}(\mathbb{1}(x)^* - \mathbb{1}(x))||_\infty \geq t] \\ &(\text{Let } y^* \triangleq 2\mathbb{1}(x)^* - 1 \in [-1,1], y \triangleq 2\mathbb{1}(x) - 1 \in [-1,1]) \\ &= Pr[||\frac{1}{n}\sum_{j=1}^{n}(y^* - y)||_\infty \geq 2t] \leq 2\exp(\frac{-(2nt)^2}{\frac{1}{n}\sum_{j=1}^{n} Var(y^*) + \frac{2nt}{3}}) \\ &(\text{according to Bernstein's inequality [33]}) \\ &= 2\exp(-\frac{4(nt)^2}{4n\frac{4\ell^2 e^\varepsilon}{(e^\varepsilon-1)^2} + \frac{2nt}{3}}) = O(exp(-nt^2(\varepsilon/l)^2)).\end{aligned}$$

Taking $t = O\left(l\sqrt{\log(l/\beta)}/(\varepsilon\sqrt{n})\right)$, we have that the L_∞ error is $O\left(l\sqrt{\log(l/\beta)}/(\varepsilon\sqrt{n})\right)$ with probability at least $1-\beta$.

I Parameter setting

The primary parameters of LDP-RM include ε, k, k_s, k_c, θ, T. ε determines the balance between privacy and utility, where a smaller ε implies higher privacy. k, k_s, and k_c continuously prune the relation domain, reducing it from d^2 to k^2 to k_s to k_c, aiding in identifying target relations. These parameters indirectly affect the trade-off between privacy and utility. If set too small, they risk overlooking relations with infrequent items. Conversely, if set too large, it requires estimating numerous relation frequency values, reducing accuracy. θ limits r-rank approximation, balancing estimation error and bias. T influences the number of iterations and user groups in Task 2, indirectly affecting the trade-off between error and bias. In our experiments, we set these parameters reasonably: $k = 64$, $k_s = 1600$, $k_c = 32$, $\theta = 0.5$, and $T = 5$. Based on our parameter tuning experience, we provide a recommended parameter configuration guideline, summarized in Tab. 4, as a reference for future LDP-RM applications.