# Data Dependent Itemset Mining Under Local Differential Privacy

**Renan R. Oliveira**[1], **José S. Costa**[1], **José M. Monteiro**[1], **Javam C. Machado**[1]

[1] Laboratório de Sistemas e Bancos de Dados (LSBD)
Departamento de Computação / UFC – Fortaleza – CE – Brazil

{renan.oliveira, serafim.costa, jose.monteiro, javam.machado}@lsbd.ufc.br

***Abstract.*** *Frequent itemset mining with Apriori-based methods is widely used to uncover patterns in large datasets. However, under the private setting of Local Differential Privacy (LDP), this task poses key challenges, such as domain scalability and user set-valued data. Prior works address these problems through fixed-size transaction reporting protocols but often rely on hard-coded parameters, overlooking complexities from differing dataset profiles. To address this limitation, we propose a data-dependent framework for identifying the top-$k$ most frequent itemsets under LDP. Through initial metadata queries and the integration of an optimization problem, we significantly improve over prior work on both synthetic and real-world datasets.*

## 1. Introduction

In recent years, Differential Privacy (DP) [Dwork et al. 2006] has become the standard for ensuring individual privacy in data analysis [Abadi et al. 2016, Dwork 2011]. Historically, this has been implemented in a centralized model, where a trusted curator gathers raw data and applies privacy-preserving transformations before publishing aggregate insights. However, growing concerns over data breaches and misuse [Zhu et al. 2017, K et al. 2021, Frenkel and Isaac 2018, Satariano et al. 2018] have led to the development of Local Differential Privacy (LDP) frameworks [Erlingsson et al. 2014]. LDP shifts the privacy mechanism to the data source, removing the reliance on a trusted curator while still enabling statistical analysis with strong privacy guarantees.

A primary application of LDP is the private estimation of feature frequencies [Neto et al. 2024]. As research in DP advanced, issues derived from frequency estimation were addressed under the LDP model. Among these, frequent itemset mining remains a core data mining task with applications in market basket analysis, user behavior modeling, and association rule mining. Identifying frequently co-occurring itemsets uncovers patterns in data and supports data-driven decision-making. Beyond traditional domains, it has been applied in large-scale systems such as hardware diagnostics, industrial monitoring, and IT infrastructure management, where it integrates with AI techniques to detect recurring faults, correlated anomalies, and co-occurring error patterns, enabling root cause analysis, failure prediction, and automated maintenance [Lin et al. 2020].

However, itemset mining raises significant privacy concerns, as publishing frequent itemsets can inadvertently reveal sensitive user information. Even with traditional anonymization, patterns may be linked to individuals, reinforcing the need for stronger guarantees such as LDP. Yet, discovering frequent itemsets under LDP remains challenging: the exponential number of possible itemsets dilutes counts, making them difficult to distinguish from noise added by Frequency Oracles (FOs). Specialized protocols like Padding and Sampling Frequency Oracle (PSFO) [Qin et al. 2016] address this issue but often rely on hard-coded, unoptimized parameters. Real-world datasets exhibit diverse distribution profiles, including differences in skewness and sparsity, aspects often overlooked in existing approaches. Effectively addressing this problem requires a data-dependent solution capable of adapting to each dataset's underlying profile.

In this paper, we leverage metadata and formulate an optimization problem to introduce a novel framework for identifying the top-$k$ most frequent itemsets under LDP. Unlike existing approaches that rely on predefined parameters, our framework dynamically optimizes both the PSFO reporting parameters and the framework's candidates search space to adapt to different data profiles. It remains data-dependent in both critical stages: identifying the top-$k$ most frequent items and discovering the $k$ most relevant itemsets. This design enhances robustness across different datasets, addressing limitations of prior frameworks and improving accuracy in privacy-preserving itemset mining.

**Contributions.** In summary, this paper makes the following contributions:

- We propose a novel data-dependent framework for identifying top-$k$ items and itemsets under LDP
- We formulate and solve an optimization problem involving the LDP protocol used for reporting user data.
- We introduce a dynamic search space for Apriori-based candidate estimation.
- We conduct a comprehensive experimental evaluation on real-world and synthetic datasets, benchmarking our framework against the leading prior work in apriori-based itemset mining under LDP.

## 2. Background And Definitions

### 2.1. Local Differential Privacy (LDP)

Under LDP, sensitive information $v$ from each user is encoded by a randomized algorithm $\Psi$, and its output $\Psi(v)$ is sent to the aggregator, which is responsible for collecting all users' reports. Intuitively, LDP guarantees that, no matter what the value of $\Psi(v)$ is, it is approximately equally as likely to be a result of perturbing $v$ as any other $v'$ differing from $v$. Therefore, if $\Psi(v)$, instead of $v$ , is collected, the users never reveal their private value $v$. The user's degree of privacy is controlled by the privacy budget $\epsilon$. More formally,

**Definition 1 (Local Differential Privacy [Erlingsson et al. 2014])** . *An algorithm $\Psi(\cdot)$ satisfies $\epsilon$-local differential privacy ($\epsilon$-LDP), where $\epsilon \geq 0$, if and only if for any pair of inputs $(v, v')$, and any set $R$ of possible outputs of $\Psi$, we have*

$$Pr[\Psi(v) \in R] \leq e^{\epsilon} Pr[\Psi(v') \in R] \tag{1}$$

### 2.2. Frequency Oracle (FO)

Under LDP, frequency estimation is tackled with FO protocols, which estimate the frequency of sufficiently recurring values $v \in D$ in a domain $D$, while ensuring $\epsilon$-LDP guarantees. Users apply noise locally with $\Psi$, and the aggregator estimates frequencies by applying $\Phi$ to the reports. These protocols are widely applicable to LDP tasks, as many problems reduce to frequency estimation. Below, we discuss two FOs used in this work.

#### 2.2.1. Generalized Randomized Response (GRR)

Randomized Response [Warner 1965], originally designed for binary domains, can be readily extended to larger domains [Kairouz et al. 2016]. Each user reports their private value $v \in D$ with probability $p$, effectively reporting it's true value. Otherwise, with probability $(1 - p)$, the user sends a randomly chosen value $v' \in D$, effectively reporting noise. Formally, the user-end algorithm $\Psi$ is defined as follows:

$$\forall_{x \in D} Pr\left[\Psi_{GRR_{(\epsilon)}}(v) = x\right] = \begin{cases} p = \frac{e^\epsilon}{e^\epsilon + |D| - 1} & \text{if } x = v \\ q = \frac{1-p}{|D|-1} = \frac{1}{e^\epsilon + |D| - 1} & \text{if } x \neq v \end{cases}$$

GRR satisfies $\epsilon$-LDP since $p/q = e^\epsilon$. After receiving the vector of the $n$ reports$\mathbf{r} = \langle x_1, x_2, \cdots, x_n \rangle$, where $x_i \in D$ is the reported value of the i-th user, the aggregator can estimate the frequency of any value $v \in D$ with $\Phi$. Considering $C(n)$ as the number of times $v$ appears in the vector $\mathbf{r}$, the unbiased [Wang et al. 2017] estimator for the frequency of $v \in D$ is $\Phi_{GRR_{(\epsilon)}}(v) := (C(v)/n - q)/(p - q)$ with variance $Var\left[\Phi_{GRR_{(\epsilon)}}(v)\right] = \frac{e^\epsilon + |D| - 2}{n(e^\epsilon - 1)^2}$. Since the variance is linear in $|D|$,the protocol's accuracy decreases as the domain size increases.

#### 2.2.2. Optimized Local Hashing (OLH)

The Optimized Local Hashing protocol [Wang et al. 2017] tackles the limitation of large domains in GRR by applying a hash function that maps each input value to a smaller domain of size $g$. GRR is then applied to the hashed output. To minimize variance, $g$ is set to $\lceil e^\epsilon + 1 \rceil$. The hashing process uses a universal hash function family called $\mathbb{H}$, such that each $H \in \mathbb{H}$ maps a value from $D$ to $\{1...g\}$. The variance of estimations using OLH is $Var\left[\Phi_{OLH_{(\epsilon)}}(v)\right] = \frac{4e^\epsilon}{n(e^\epsilon - 1)}$.

### 2.3. Padding and Sampling Frequency Oracle (PSFO)

One approach in the literature [Qin et al. 2016] for handling set-valued inputs is to apply padding and sampling to the user's transaction before reporting their data with a standard FO. Conceptually, PSFO combines a padding and sampling (PS) function to process set-valued inputs and an FO for reporting data. The PS function works by filling transactions with dummy items until they reach a parameterized length. It then samples a single element from the resulting transaction.

**Definition 2 (Padding and Sampling Function [Qin et al. 2016])** *The padding and sampling function PS is parameterized by a positive integer $\ell$ and takes a set $v \subseteq I$ as input. It assumes the existence of $\ell$ dummy items $\bot_1, \bot_2, \ldots, \bot_\ell \notin I$. If $|v| < \ell$, $PS_\ell(v)$ adds $\ell - |v|$ unique random dummy items to the transaction $v$, then samples an element $x$ at uniform random from it.*

An standard FO is used to report the element $x$ sampled by PS. In this context, the FO's domain includes both the data domain and the dummy elements introduced during padding: $I \cup \{\bot_1, \bot_2, \ldots, \bot_\ell\}$. The aggregator obtains each item frequency estimate from the FO protocol and multiplies it by $\ell$. PSFO standardizes transactions by padding them to a fixed length $\ell$, ensuring each item $x$ has an equal chance $1/\ell$ of being sampled and reported. This scaling of the estimates corrects the underestimation originated from the sampling step. While PS does not explicitly truncate transactions longer than $\ell$, the scaling done still assumes a transaction of length $\ell$, effectively causing information loss of frequently occurring items in transactions beyond that length. Scaling also amplifies the FO's noise by a factor of $\ell$.

**Definition 3 (Padding and Sampling Frequency Oracle [Qin et al. 2016])** *A Padding and Sampling Frequency Oracle (PSFO) protocol is parameterized by a positive integer $\ell$, a frequency oracle (FO), and a privacy budget $\epsilon$. It consists of a pair of algorithms $\langle \Psi, \Phi \rangle$, defined as:*

$$PSFO(\ell, FO, \epsilon) := \left\langle \Psi_{FO(\epsilon)}(PS_\ell(\cdot)),\ \Phi_{FO(\epsilon)}(\cdot) \cdot \ell \right\rangle$$

### 2.4. Adaptive FO

When applying an FO in conjunction with a PS function, [Wang et al. 2018] demonstrates that GRR benefits from privacy amplification, while OLH benefits less significantly. To address this, the authors propose a new criterion for selecting the FO protocol, introducing an adaptive protocol, Adap, which dynamically chooses to use GRR with privacy budget $(\ln(\ell(e^\epsilon - 1) + 1)$ if $d < e^\epsilon \ell(4\ell - 1) + 1$ and OLH with privacy budget $\epsilon$ otherwise.

## 3. Related Works

LDP-FPMiner [Chen and Wang 2022] introduces an approach based on frequent pattern trees (FP-trees). Their method constructs and optimizes a noisy FP-tree under LDP constraints. Hardamard Encoding is utilized in [Zhao et al. 2023] along side of PSFO to propose a new framework. PrivMiner [Li et al. 2025] proposes a novel technique for estimating frequent itemsets under LDP. LDPMiner [Qin et al. 2016], a pioneering method for querying relevant items under LDP, introduces a novel protocol for reporting set-valued data. It adopts a two-phase approach where all users participate in both steps, each satisfying $(\epsilon/2)$-LDP, ensuring $\epsilon$-LDP by sequential composition.

### 3.1. Set Value Itemset Mining (SVSM)

SVSM [Wang et al. 2018] is an apriori-based approach; it improves upon LDPminer by addressing the itemset mining problem left open in its work [Qin et al. 2016], while also outperforming its item mining performance. It introduces SVIM for frequent item mining and SVSM for itemset mining. Rather than splitting the privacy budget $\epsilon$, like LDPMiner, SVSM partitions the data for each query, each satisfying $\epsilon$-LDP. The framework consists of a multi-step query process utilizing PSFO to identify frequent items, which are then used to generate candidate itemsets for their frequency estimation.

While SVSM improves upon LDPMiner by introducing additional steps and optimizing noise, it lacks data dependability, especially evident in it's first step. Although the lower value of $\ell$ used reduces variance, it introduces a significant amount of bias into the reports. While this aims to reduce noise introduced by padding in PSFO when users have transaction sizes smaller than $\ell$, it fails to exploit greater data utility in datasets where the minimum transaction size exceeds $1$, as it continues to enforce a fixed and suboptimal $\ell = 1$. In this context, we refer to the estimates accuracy as utility, reflecting the practical value of theprivately reported data to the aggregator.

## 4. Defining the Problem

We consider a problem where each user holds a private subset of items. The data aggregator seeks to find the most relevant itemsets while ensuring each user's report satisfies $\epsilon$-LDP. Let $D = \{v_1, v_2, \cdots, v_n\}$ be the dataset of the $n$ users' transactions. Each transaction $v_i$ is a subset of items, i.e, $v \subseteq I$, where $I$ denotes the domain of all possible items in $D$. In this context, we define the relevance of an item or itemset in terms of its count, that is, the frequency with which it appears across the dataset. Hence, a lower count indicates less relevancy and a high count higher relevancy. Ultimately, the aggregator's goal is to find the set $IS$ of the top-$k$ most relevant itemsets $\mathbf{x}$, defined as:

$$IS = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \mid count(\mathbf{x}_1) \geq \cdots \geq count(\mathbf{x}_k) \geq count(\mathbf{x}) \; \forall \; \mathbf{x} \notin IS$$

While existing FO protocols are effective for querying attribute counts in tabular datasets, where each user reports a single attribute value, set-valued data introduces additional challenges. Standard FO protocols struggle to scale well in this context, as they rely on binary encoding of the users data to apply noise and report the data. In this setting, encoding each transaction as a single value binary array over the domain $\mathcal{P}(I)$ (the power set of items in $I$) quickly becomes infeasible due to the exponential number of possible combinations. Additionally, FO protocols are only capable of identifying values that occur frequently within the dataset, since the magnitude of the added noise is proportional to the square root of the population size [Chan et al. 2012]. In set-valued datasets, although certain items and itemsets occur frequently, the transactions themselves are typically unique or relatively infrequent. Thus, directly encoding and querying full transactions under LDP would effectively yield only noise when no single transaction pattern appears often enough to stand out.

## 5. DDSM: Data Dependent itemSet Mining

In this section, we present the proposed framework in detail and outline the approach taken to ensure its data dependability. We begin by formalizing the transaction length, *i.e.*, the size of the report, optimization process, then show how this foundation is leveraged to guide and enhance each step of our framework.

### 5.1. Optimal Limit Size

The PS function, defined in Definition 2, is capable of adding a significant amount of noise to the user's response. Depending on the parameter $\ell$, some users may end up reporting way more noise than others. Overall, the amount of noise reported due to padding is data dependent on the number of user sets smaller than $\ell$. For a transaction $v$ and a transaction size limit $\ell$ used in the PSFO protocol, when $|v| < \ell$, there's a probability $P_{Sdummy} = \frac{\ell - |v|}{\ell}$ of the protocol feeding the added noise to the FO and a probability $P_{Sreal} = \frac{|v|}{\ell}$ of the protocol reporting a real value to the FO, strengthening the utility of the estimation. Although transactions of size $|v| > \ell$ have zero probability of sampling noise, since they will not be padded by the PS function, they can still impact the protocol's utility due to underestimation, as their items will be sampled with a probability bigger than $1/|v|$ while being scaled only by a factor of $\ell$. For instance, in Figure 1, PS is fed a set $v$ of size 3 and $\ell$ is set to size 5. After padding, PS has a probability $P_{Sdummy} = 40\%$ of sampling a dummy item, and $P_{Sreal} = 60\%$ of sampling a real item.

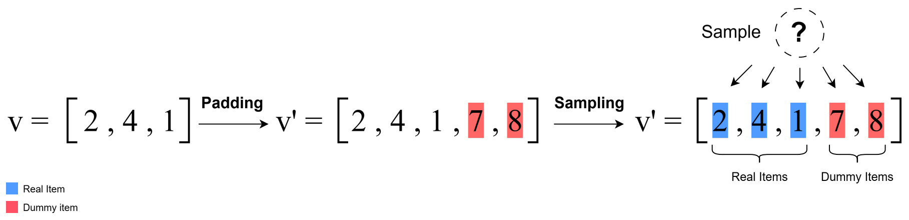

**Figure 1. Illustration of the sampling done by the PS function**

While larger transactions carry more utility in their reports for larger values of $\ell$, smaller transactions, once padded, add to much noise into the user's private set. Similarly, when $\ell$ is small, larger transaction are underestimated in their reports, whereas smaller transactions have less noise padded into them. This dynamic underlies the choice of $\ell$. Underestimation introduces bias, while padding increases variance through the addition of dummy items. Therefore, before applying a PSFO protocol, we propose to select an $\ell$ that balances this trade-off. This optimal value is closely tied to the distribution of transaction sizes in the dataset that the aggregator operates over.

We introduce an objective function to compute the optimal value $L_{opt}$ for the PSFO protocol. It models the trade-off between the utility loss and gains that result from adjusting the transaction size limit. By formalizing this relationship, the objective function allows the identification of the value of $\ell$ that offers the best balance between noise and data utility. Loss is defined as $P_{\text{Sdummy}} f_l L$, the expected number of times PSFO samples a dummy item from a set of transactions of size $l$, multiplied by $L$, a limit size set for the PSFO protocol and also the times which the sample will be scaled. Likewise, Gain is defined as Gain $= P_{\text{Sreal}} f_l L$, the expected number of times PSFO samples a real item from a set of transactions of size $l$, multiplied by $L$. The objective function is computed as the sum of the differences between Gain and Loss for each transaction size $l$ in the dataset's size distribution, denoted by $f$, where $f_l$ indicates the count of transactions of size $l$ within the $f$ distribution. The optimization problem is then configured to find the value of $L$ for PSFO that maximizes the total utility across all possible lengths $l \in [1, 2, \ldots, d]$:

$$L_{\text{opt}} = \arg\max_{L} \left( L \sum_{l=1}^{d} (P_{\text{Sreal}} f_l - P_{\text{Sdummy}} f_l) \right) \tag{2}$$

We refer to Equation (2) as the function Opt$(\cdot)$, which computes the optimal transaction limit size to be used with PSFO in order to maximize its reporting utility for a given transaction length distribution: $L_{opt} = \text{Opt}([\Phi_{\text{OLH}(\epsilon)}(1), \ldots, \Phi_{\text{OLH}(\epsilon)}(d)])$.

## 5.2. Our Framework

We propose a new framework for identifying top-$k$ items and itemsets under LDP that includes: (1) Data Dependable Item Mining (DDIM) for mining frequent items and (2) Data Dependable itemSet Mining (DDSM) for finding frequent itemsets. The framework begins by identifying frequent items with DDIM, then proceeds to estimate frequent itemsets with DDSM through a multi-step process. To answer each query, we partition the data into user groups, each tasked with reporting a specific query at each step. As demonstrated in [Filho and Machado 2023], for both the OLH and GRR protocols, partitioning users rather than the privacy budget results in improved utility. Specifically, we employ a 6-group setup: 4 groups are dedicated to acquiring frequent items, while the remaining 2 are used to estimate the frequent itemsets. Each step can be viewed as one of two core operations executed by the data aggregator: (i) querying the distribution of users' private set sizes, and (ii) estimating frequencies using the queried data from previous steps. A high-level overview of the framework is shown in Figure 2 and Figure 3.

### 5.2.1. Acquiring Frequent Items (DDIM)

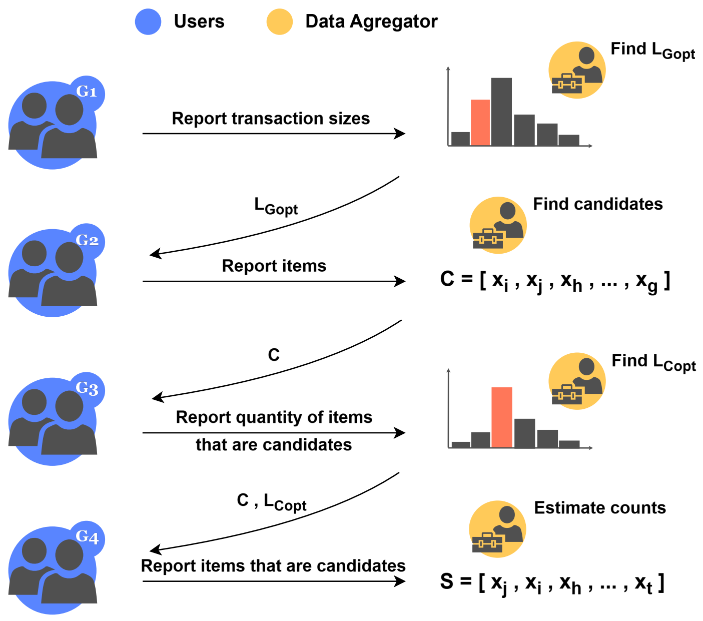
Users
Data Agregator
G1
Report transaction sizes
Find $L_{Gopt}$
G2
$L_{Gopt}$
Report items
Find candidates
C = [ $x_i$ , $x_j$ , $x_h$ , ... , $x_g$ ]
G3
C
Report quantity of items that are candidates
Find $L_{Copt}$
G4
C , $L_{Copt}$
Report items that are candidates
Estimate counts
S = [ $x_j$ , $x_i$ , $x_h$ , ... , $x_t$ ]

**Figure 2. Flux Chart Illustrating the Steps for Item Mining Task (DDIM)**

**Step 1: Fetching Distribution.** Prior to estimating item statistics, the aggregator queries metadata to gain preliminary insight about the transactions. By understanding how transaction sizes are distributed within the group $G_1$, the aggregator can generalize this distribution to the entire dataset, improving candidate estimation in the subsequent steps. Specifically, each user in Group $G_1$ reports the length $l$ of their transaction $v$ using a privacy budget $\epsilon$ through a standard FO protocol. As each user reports a single value from an attribute of fixed domain (*i.e* the set of possible sizes $I$), OLH is used: $\Psi_{\text{OLH}(\epsilon)}(|v|)$. The aggregator then estimates the size distribution of the group transactions by computing $\Phi_{\text{OLH}(\epsilon)}(l)$ for all values of $l \in [1, \ldots, d_w]$. We use $d_w = |I|/10$ to reduce noise while still capturing a relevant amount of transaction lengths. Lastly, Equation 2 is applied to the array of estimated counts to compute the optimal transaction size limit for the general population of items, denoted by $L_{Gopt} = \text{Opt}([\Phi_{\text{OLH}(\epsilon)}(1), \ldots, \Phi_{\text{OLH}(\epsilon)}(dw)])$.

**Step 2: Finding Candidates.** The data aggregator now shifts focus from analyzing metadata to querying item frequencies. The optimal length $L_{Gopt}$ from Step 1 is provided to user group $G_2$, whose users report a sample from their private set $v$ using PSFO with this length as the limit: $\Psi_{\text{PSFO}(\text{L}_{\text{Gopt}},\text{Adap},\epsilon)}(v)$. Using $\Phi_{\text{PSFO}(\text{L}_{\text{Gopt}},\text{Adap},\epsilon)}(x)$, the aggregator estimates the items' frequencies and constructs the candidate set $C$, consisting of the $zk$ most frequent items. We propose the usage of $z = \lceil \log_{10} |I| \rceil$, to extend the top-$k$ range, accounting for possible ranking shifts of candidates that may be outside the top-$k$ due to estimation noise. While $C$ identifies a reliable set of heavy hitters their frequencies and relative rankings remain highly inaccurate due to the variance introduced by the high value of $L$ used on PSFO. In the next step, we calculate a new value for $L$ from the distribution of candidate items present in the users' transactions.

**Step 3: Candidate Set Size Estimation.** The data aggregator now seeks to solely estimate the $zk$ candidates in $C$. Due to the difference in size distribution between the sets of general transactions $\{|v_i \cap C| \mid i = 1, 2, \cdots, n\}$ and the candidate-containing subsets$\{|v_i| \mid i = 1, 2, \cdots, n\}$, a new value for $\ell$ must be determined to optimize the reports estimation. To infer the size distribution, each user in group $G_3$, provided with $C$, reports the number of candidates in their private sets, i.e, $\Psi_{\text{OLH}(\epsilon)}(|v \cap C|)$, the aggregator then computes $\Phi_{\text{OLH}(\epsilon)}(l)$ for all $l \in [1, 2, \ldots, zk]$, and subsequently obtains $L_{Copt} = \text{Opt}([\Phi_{\text{OLH}(\epsilon)}(1), \ldots, \Phi_{\text{OLH}(\epsilon)}(zk)])$ to be used in Step 4.

**Step 4: Estimation And Adjustment.** Lastly, provided $L_{Copt}$ and $C$, group G4 reports from their private sets the items that are candidates by locally computing $\Psi_{\text{PSFO}(\text{L}_{\text{Copt}},\text{Adap},\epsilon)}(v \cap C)$. The aggregator then estimates the list of counts of size $zk$ by executing $\Phi_{\text{PSFO}(\text{L}_{\text{Copt}},\text{Adap},\epsilon)}(x)$ for all $x \in C$. In this process, users whose transaction lengths exceed $L_{Copt}$ may cause underestimation of the counts. To attempt correcting the estimates, [Wang et al. 2018] proposes a correction factor based on the assumption that unreported counts follow a distribution similar to the reported ones. Accordingly, each estimate is scaled by a factor of $u(L)$, defined as:

$$u(L) := \frac{\sum_{\ell=1}^{zk} \Phi_{\text{OLH}(\epsilon)}(\ell)(\ell)}{\sum_{\ell=1}^{zk} \Phi_{\text{OLH}(\epsilon)}(\ell)(\ell) - \sum_{\ell=L+1}^{zk} \Phi_{\text{OLH}(\epsilon)}(\ell)(\ell - L)} \tag{3}$$

In Equation 3, the numerator denotes the total number of items. The denominator accounts for the total number of items, excluding those from sets exceeding $L$, the transaction size limit set for PSFO. The excluded sets are the total number of items missed by the PSFO correction. Since the update of the counts is simply post-processing arithmetic and uses previous information obtained from Step 3 and Step 4, there are no privacy concerns in this operation [Dwork et al. 2014]. Ultimately, the aggregator obtains a set $S$ containing the $k$ most relevant items and their corresponding counts.

### 5.2.2. Acquiring Frequent Itemsets (DDSM)

Upon obtaining the set of frequent items and their estimated counts, the combinatorial complexity of the exponential number of potential itemsets is addressed by constructing a candidate itemset collection, denoted as $IC$. Let $S$ be the $k$ most relevant items returned by DDIM. For any given itemset $\mathbf{x}$, its frequency is approximated as $\tilde{f}_x = \prod_{x \in \mathbf{x}} \Phi'(x)$, where $\Phi'(x) = 0.9 \frac{\Phi(x)}{\max_{x \in S} \Phi(x)}$ represents the normalized estimate of $\mathbf{x}$, and $\Phi(x)$ is the count of the item $x$ estimated by DDIM. The factor of $0.9$ is introduced to attenuate the normalized estimates for the most relevant item, preventing the guessed frequency of any set without it from being unrealistically close to that of a set containing it.

$$\text{IC} := \left\{ \mathbf{x} : \mathbf{x} \subseteq S,\ 1 < |\mathbf{x}| < \log_z k,\ \prod_{x \in \mathbf{x}} \Phi'(x) \right\} \tag{4}$$

The top $zk$ itemsets with the highest estimated counts are selected to form the itemset candidates $IC$. This approach is justified by the observation that frequent itemsets often contain frequent items, whereas the contrary, where frequent itemsets are composed predominantly from infrequent items, is unlikely. Formally, the set $IC$ is defined in Equation 4 and then truncated to satisfy $|IC| = zk$.

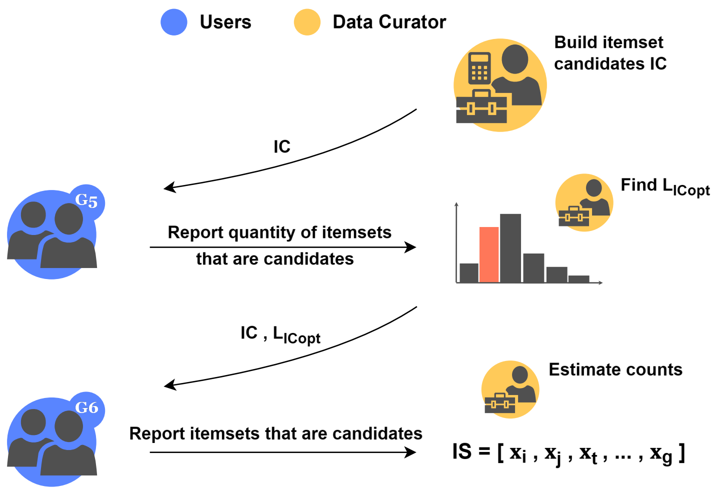
Users
Data Curator
Build itemset candidates IC
IC
G5
Report quantity of itemsets that are candidates
Find L_ICopt
IC , L_ICopt
Estimate counts
G6
Report itemsets that are candidates
IS = [ x_i , x_j , x_t , ... , x_g ]

**Figure 3. Flux Chart Illustrating the Steps for Itemset Mining task (DDSM)**

**Step 5: Candidate Set Size Estimation.** In this Step, group $G_5$ is provided with the list of candidate itemsets $IC$. Each user is tasked with reporting the number of itemsets from their private set $v$ that are present in the candidates $IC$ i.e., $\Psi_{\mathrm{OLH}(\epsilon)}(|vs|)$, where $vs = \{\mathbf{x} \mid \mathbf{x} \in IC,\ \mathbf{x} \subseteq v\}$. The aggregator estimates the size distribution of the reported sets of itemsets by computing $\Phi_{\mathrm{OLH}(\epsilon)}(l)$ for all $l \in [1, \ldots, zk]$, and finds an optimal value $L_{ICopt} = \mathrm{Opt}([\Phi_{\mathrm{OLH}(\epsilon)}(1), \ldots, \Phi_{\mathrm{OLH}(\epsilon)}(zk)])$ to be used in Step 6.

**Step 6: Estimation And Adjustment.** At last, the aggregator provides $L_{ICopt}$ and the candidate itemsets' counts $IC$ to group $G_6$, which is tasked with reporting the itemsets from their private set $v$ that are in $IC$ using PSFO, with length limit set to $L_{ICopt}$: $\Psi_{\mathrm{PSFO}(\mathrm{L_{ICopt}},\mathrm{Adap},\epsilon)}(vs)$. The frequency estimates are obtained by computing $\Phi_{\mathrm{PSFO}(\mathrm{L_{ICopt}},\mathrm{Adap},\epsilon)}(\mathbf{x}) \cdot u(L_{ICopt})$ for all itemsets $\mathbf{x} \in IC$, producing the set $IS$ of the estimated $k$ most relevant itemsets and their corresponding counts.

## 6. Experimental Evaluation

To evaluate the performance of our framework, we compare it against the state-of-the-art Apriori-based itemset mining approach under LDP. We assess both item and itemset estimates, since the accuracy of item estimates directly influences itemset estimation. In the experimental setup, both solutions are evaluated over ten independent runs on three real-world and one synthetic dataset. All implementations use Python 3.12.5. SVSM and SVIM, serving as our baseline, are implemented as described in the literature [Wang et al. 2018], while DDSM and DDIM follow the methodology outlined in Section 5. We allocate $50\%$ of the data to item mining queries done by DDIM, and the other $50\%$ to itemset mining queries done by DDSM. For DDIM, this $50\%$ is partitioned into four groups: $G_1$, $G_2$, $G_3$, and $G_4$, with $10\%$, $70\%$, $8\%$, and $12\%$ of the data respectively. DDSM divides the remaining $50\%$ into two groups, $G_5$ and $G_6$, with $10\%$ and $90\%$.

**Datasets.** The data workloads and their characteristics are detailed below:

- **Accidents:** Contains traffic accident data from the National Institute of Statistics for Flanders (Belgium) [Geurts et al. 2003], covering the period from 1991 to 2000. Composed of 340,183 records and 468 unique types of accidents, each record can have up to 51 and as low as 18 accidents.

- **Netflix:** Consists of users' movie ratings, provided by Netflix for the KDD Cup 2007 [Zhu 2018]. The dataset contains 480,189 users, each having up to 17,770 movie ratings. The number of ratings per user ranges from a minimum of 1 to a maximum of 17,653.
- **Kosarak:** Provided by Ferenc Bondon, contains click-stream data from a Hungarian news portal [Zhu 2018]. It includes 990,002 users, each with access to a domain of 41,270 unique links. The number of links a user accessed in this dataset ranges from 1 to 2498 links.
- **SharkFin:** A dataset made to simulate scenarios where transaction sizes are concentrated away from the unit size. It contains 515,596 users and a domain of 1,306 items [Oliveira 2025]. The number of the user set size ranges from 1 to 165 items.

**Metric.** To evaluate the identification of relevant items, we measure both the number of correctly identified items and their relative ranking. For this, we use Normalized Discounted Cumulative Gain (NDCG), a widely used metric that captures relevance and ranking accuracy. NDCG compares the estimated ranking to an ideal one where all relevant items are present and ordered by their true importance. Scores are normalized between $0$ and $1$, with higher values indicating better performance. Given a ranked list of $k$ items with scores $\{rel_1, rel_2, \ldots, rel_k\}$, NDCG is computed in Equation 5 as the ratio between the Discounted Cumulative Gain (DCG) of the estimated ranking and the Ideal DCG (IDCG), where items are ideally sorted by descending relevance $rel_i$.

$$\mathbf{NDCG}_k = \frac{\mathrm{DCG}_k}{\mathrm{IDCG}_k}, \qquad \mathbf{DCG}_k = \sum_{i=1}^{k} \frac{\mathrm{rel}_i}{\log_2(i+1)} \tag{5}$$

## 7. Results

We now present our findings from our experimental analysis. To evaluate the performance of each solution, we assess them in two main benchmarks: their effectiveness in identifying and ranking items and itemsets across varying levels of privacy strictness, and across varying target quantities of itemsets under a fixed privacy budget. The first evaluation will provide an insight into the impact of introduced noise in each solution, while the second examines how the target $k$ of the number of itemsets impacts the solutions' performance.

**Table 1. Item Mining Evaluation for k=64 Varying the Privacy Budget $\epsilon$.**

| | ACCIDENTS | | SHARKFIN | | NETFLIX | | KOSARAK | |
|---|---|---|---|---|---|---|---|---|
| $\epsilon$ | SVIM | DDIM | SVIM | DDIM | SVIM | DDIM | SVIM | DDIM |
| 0.2 | 0.34 | 0.57 | 0.17 | 0.41 | 0.02 | 0.00 | 0.13 | 0.06 |
| 0.6 | 0.67 | 0.87 | 0.44 | 0.54 | 0.01 | 0.04 | 0.17 | 0.19 |
| 1.0 | 0.77 | 0.94 | 0.54 | 0.70 | 0.02 | 0.16 | 0.23 | 0.28 |
| 2.0 | 0.83 | 0.96 | 0.71 | 0.85 | 0.15 | 0.29 | 0.29 | 0.34 |
| 3.0 | 0.88 | 0.98 | 0.80 | 0.88 | 0.37 | 0.50 | 0.33 | 0.41 |

Generally, as the privacy budget $\epsilon$ increases, utility tends to improve for both solutions. As foreseen at Definition 1, higher $\epsilon$ values introduce less noise to the data as reports become less private. Particularly for item mining, our solution shows the best relative performance against SVIM in high privacy settings. In Table 1, at $\epsilon = 1$, SVIM averaged an NDCG of $0.39$ across all tested datasets, while DDIM reached $0.52$. Notably, for the Accidents dataset, our solution achieved the highest performance, attaining an NDCG of $0.87$ at a strong privacy guarantee of $\epsilon = 0.6$, compared to $0.67$ SVIM under the same conditions. Similar trends are observed across the other datasets, with our framework consistently outperforming SVIM for most privacy budgets, achieving $0.50$ in the Netflix dataset and $0.41$ in the Kosarak dataset at looser privacy guarantees. These results are significant, as in Apriori-based frameworks, the quality of item mining estimates directly impacts subsequent itemset mining performance; this holds for both frameworks.

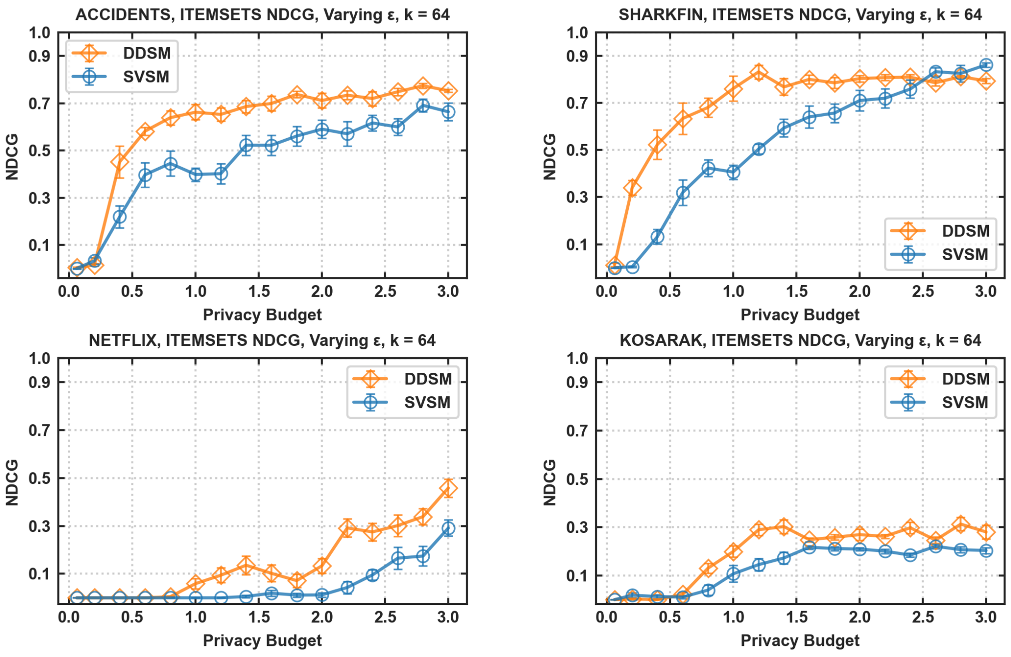

**Figure 4. Itemset Mining Evaluation for k=64 Varying the Privacy Budget.**

At the itemset mining task, shown in Figure 4, our framework performs notably well under strict privacy constraints, *i.e.*, at lower values of $\epsilon$. Specifically, in Accidents, DDSM reaches a score of $0.70$ at $\epsilon = 1.60$, whereas SVSM only attains a comparable result at $\epsilon = 2.6$. Similarly, in SharkFin, DDSM reaches $0.83$ at $\epsilon = 1.2$, whereas SVSM surpasses this value only at a substantially looser privacy setting of $\epsilon = 3.0$. These results express the strengths of our solution. For the Accidents dataset, the data-dependent approach used in our framework allows for greater leveraging of the dataset's large minimum transaction length of $18$ items. By querying transaction lengths and computing an optimal parameter for PSFO with Equation 2, our framework enables more effective data estimation tailored to the underlying dataset profile. Moreover, the SharkFin dataset results demonstrate that our framework performs particularly well in the hypothetical scenario where the minimum transaction size is one, but the heavy hitters in the size distribution are concentrated at larger sizes. In this context, DDSM, despite not using the unit size as $\ell$ for fetching the candidates with PSFO like SVSM, our solution consistently outperforms it across most tested privacy budgets. This demonstrates that our optimization function, combined with the data-dependent approach, remains resilient to skewed length distributions by dynamically computing optimal parameters through a distribution-aware optimization process, effectively avoiding the pitfalls of static parameter choices.

The consistently low scores observed for Kosarak and Netflix highlight a key limitation of our framework, which may reflect a broader issue inherent to Apriori-based itemset mining. In Figure 4, both solutions struggle to achieve high NDCG scores; our framework consistently outperforms SVSM. This difficulty stems from the high sparsity of these datasets, where most transaction sizes fall within a narrow range while the domain's cardinality extends far beyond the intervals where heavy hitters concentrate. Additionally, large domain cardinality limits the FO's ability to reliably estimate patterns. As the domain grows, identifying frequent patterns becomes increasingly complex, reducing utility. This is captured by the variance expressions of the FOs in Section 2.2, which increase with domain size, as shown in Subsections 2.2.1 and 2.2.2. Since PSFO relies on these FOs to generate reports, its accuracy is directly affected by this variance growth. Netflix, while also featuring high cardinality, is considerably less sparse, allowing more reliable estimations. Despite these challenges, our framework achieves competitive NDCG scores at the highest tested privacy budget.

**Table 2. Itemset Mining Evaluation for $\epsilon = 2.2$ Varying $k$.**

| | ACCIDENTS | | SHARKFIN | | NETFLIX | | KOSARAK | |
|---|---|---|---|---|---|---|---|---|
| $k$ | SVSM | DDSM | SVSM | DDSM | SVSM | DDSM | SVSM | DDSM |
| 20 | 0.21 | 0.62 | 0.56 | 0.76 | 0.05 | 0.28 | 0.18 | 0.40 |
| 30 | 0.29 | 0.66 | 0.67 | 0.76 | 0.01 | 0.36 | 0.19 | 0.33 |
| 40 | 0.48 | 0.61 | 0.71 | 0.71 | 0.03 | 0.21 | 0.23 | 0.30 |
| 50 | 0.61 | 0.69 | 0.74 | 0.69 | 0.03 | 0.21 | 0.20 | 0.35 |
| 60 | 0.59 | 0.75 | 0.74 | 0.70 | 0.07 | 0.27 | 0.19 | 0.29 |
| 70 | 0.67 | 0.72 | 0.77 | 0.82 | 0.04 | 0.20 | 0.18 | 0.29 |

Table 2 summarizes the utility of each solution across commonly used $k$ values in the literature. Generally, as $k$ increases, distinguishing between itemsets becomes harder, leading to ranking inconsistencies and reduced utility. In the Accidents dataset, our solution achieves nearly a threefold improvement over SVSM at $k = 20$. For the SharkFin dataset, while DDSM slightly falls behind at $k = 50$ and $k = 60$, it outperforms SVSM at all other tested values of $k$. In Netflix and Kosarak, our framework consistently maintains an average score advantage of approximately $0.17$ over to SVSM.

## 8. Conclusion

This work presented a data-dependent approach for frequent itemset mining under LDP. We introduced metadata queries, formalized an optimization problem involving the reporting mechanism's parameters, and proposed an objective function that captures dataset distribution characteristics to compute an optimal transaction length for PSFO. These contributions were incorporated into a framework comprising DDIM for item mining and DDSM for itemset mining. Extensive experimentation across diverse datasets demonstrated the superior and versatile performance of our framework, with analyses detailing how dataset-specific properties affected utility outcomes. Our solution consistently outperformed our competitor, particularly excelling in high-privacy scenarios and varied data profiles. However, it showed reduced effectiveness on sparse datasets with large domain cardinality. While this study advances a data-dependent strategy, user group partitioning remains hard-coded. Future work should explore dynamic, data-driven data partitioning strategies and further examine the sampling error effects it introduces, progressing toward a fully data-dependent framework with enhanced reliability.

## Acknowledgements

This work was partially supported by Lenovo, as part of its R&D investment under Brazil's Informatics Law. It was also partially funded by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ) (grant # 316729/2021-3).

## References

[Abadi et al. 2016] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, New York, NY, USA. Association for Computing Machinery.

[Chan et al. 2012] Chan, T. H. H., Li, M., Shi, E., and Xu, W. (2012). Differentially private continual monitoring of heavy hitters from distributed streams. In *Privacy Enhancing Technologies: 12th International Symposium, PETS 2012, Vigo, Spain, July 11-13, 2012. Proceedings 12*, pages 140–159. Springer.

[Chen and Wang 2022] Chen, Z. and Wang, J. (2022). Ldp-fpminer: Fp-tree based frequent itemset mining with local differential privacy. *arXiv preprint arXiv:2209.01333*.

[Dwork 2011] Dwork, C. (2011). A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95.

[Dwork et al. 2006] Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In Halevi, S. and Rabin, T., editors, *Theory of Cryptography*, pages 265–284. Springer Berlin Heidelberg.

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

[Erlingsson et al. 2014] Erlingsson, U., Pihur, V., and Korolova, A. (2014). Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA. Association for Computing Machinery.

[Filho and Machado 2023] Filho, J. S. C. and Machado, J. C. (2023). Felip: A local differentially private approach to frequency estimation on multidimensional datasets. In *International Conference on Extending Database Technology*.

[Frenkel and Isaac 2018] Frenkel, S. and Isaac, M. (2018). Facebook security breach exposes accounts of 50 million users. `https://www.nytimes.com/2018/09/28/technology/facebook-hack-data-breach.html`. Accessed: 2024-01-12.

[Geurts et al. 2003] Geurts, K., Wets, G., Brijs, T., and Vanhoof, K. (2003). Profiling high frequency accident locations using association rules. In *Proceedings of the 82nd Annual Transportation Research Board, Washington DC. (USA), January 12-16*, page 18pp.

[K et al. 2021] K, M. V., Immanuel Johnraja, J., Jeba Leelipushpam, G., Jebaveerasingh Jebadurai, J., and Santhosam, I. B. (2021). Security and privacy issues in the internet of things – a survey. In *2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*.

[Kairouz et al. 2016] Kairouz, P., Bonawitz, K., and Ramage, D. (2016). Discrete distribution estimation under local privacy. In *International Conference on Machine Learning*, pages 2436–2444. PMLR.

[Li et al. 2025] Li, Y., Huang, C., Cheng, M., Lv, T., Zhao, Y., Sun, Y., and Yuan, Y. (2025). Privminer: a similar-first approach to frequent itemset mining under local differential privacy. *World Wide Web*, 28(2):25.

[Lin et al. 2020] Lin, F., Muzumdar, K., Laptev, N. P., Curelea, M.-V., Lee, S., and Sankar, S. (2020). Fast dimensional analysis for root cause investigation in a large-scale service environment. *Proc. ACM Meas. Anal. Comput. Syst.*, 4(2).

[Neto et al. 2024] Neto, A. A. M., Neto, E. D., Filho, J. C., and Machado, J. (2024). Locally differentially private and consistent frequency estimation of longitudinal data. In *Anais do XXXIX Simpósio Brasileiro de Bancos de Dados*, pages 367–380, Porto Alegre, RS, Brasil. SBC.

[Oliveira 2025] Oliveira, R. (2025). Sharkfin. Accessed: 2025-04-06.

[Qin et al. 2016] Qin, Z., Yang, Y., Yu, T., Khalil, I., Xiao, X., and Ren, K. (2016). Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 192–203, New York, NY, USA. Association for Computing Machinery.

[Satariano et al. 2018] Satariano, A., Perlroth, N., and Tsang, A. (2018). Marriott hacking exposes data of up to 500 million guests. Accessed: 2024-01-12.

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

[Wang et al. 2018] Wang, T., Li, N., and Jha, S. (2018). Locally differentially private frequent itemset mining. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 127–143.

[Warner 1965] Warner, S. L. (1965). Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69.

[Zhao et al. 2023] Zhao, D., Zhao, S.-Y., Chen, H., Liu, R.-X., Li, C.-P., and Zhang, X.-Y. (2023). Hadamard encoding based frequent itemset mining under local differential privacy. *Journal of Computer Science and Technology*, 38(6):1403–1422.

[Zhu 2018] Zhu, E. (2018). Set similarity search benchmarks. Accessed: 2025-04-06.

[Zhu et al. 2017] Zhu, T., Li, G., Zhou, W., and Yu, P. S. (2017). Differentially private data publishing and analysis: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 29(8):1619–1638.