Data Poisoning Attacks to Locally Differentially Private Frequent Itemset Mining Protocols

Wei Tong
State Key Laboratory of Novel Software Technology, Nanjing University
Nanjing, China
weitong@outlook.com

Haoyu Chen
State Key Laboratory of Novel Software Technology, Nanjing University
Nanjing, China
corheyc@gmail.com

Jiacheng Niu
State Key Laboratory of Novel Software Technology, Nanjing University
Nanjing, China
lingdangsmoke@gmail.com

Sheng Zhong*
State Key Laboratory of Novel Software Technology, Nanjing University
Nanjing, China
zhongsheng@nju.edu.cn

Abstract

Local differential privacy (LDP) provides a way for an untrusted data collector to aggregate users' data without violating their privacy. Various privacy-preserving data analysis tasks have been studied under the protection of LDP, such as frequency estimation, frequent itemset mining, and machine learning. Despite its privacy-preserving properties, recent research has demonstrated the vulnerability of certain LDP protocols to data poisoning attacks. However, existing data poisoning attacks are focused on basic statistics under LDP, such as frequency estimation and mean/variance estimation. As an important data analysis task, the security of LDP frequent itemset mining has yet to be thoroughly examined. In this paper, we aim to address this issue by presenting novel and practical data poisoning attacks against LDP frequent itemset mining protocols. By introducing a unified attack framework with composable attack operations, our data poisoning attack can successfully manipulate the state-of-the-art LDP frequent itemset mining protocols and has the potential to be adapted to other protocols with similar structures. We conduct extensive experiments on three datasets to compare the proposed attack with four baseline attacks. The results demonstrate the severity of the threat and the effectiveness of the proposed attack.

CCS Concepts

• **Security and privacy → Privacy-preserving protocols**.

Keywords

Local Differential Privacy; Frequent Itemset Mining; Poisoning Attack

*Corresponding author

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3670298

ACM Reference Format:
Wei Tong, Haoyu Chen, Jiacheng Niu, and Sheng Zhong. 2024. Data Poisoning Attacks to Locally Differentially Private Frequent Itemset Mining Protocols. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3670298

1 Introduction

The increasing popularity of data analysis using user data has raised concerns over potential violations of privacy. As a promising solution for privacy-preserving data collection and analysis, Differential Privacy (DP) [14, 16] and its local version, Local Differential Privacy (LDP) [17, 22] have been widely studied. By adding perturbations to the data prior to reporting it, LDP enables data collection and analysis without compromising user privacy. Many LDP protocols have been proposed by both the academic and industrial communities. Various data analysis tasks, *e.g.*, frequency estimation [17, 37], heavy hitters [5, 39], and frequent itemset mining [30, 38], have been studied under the notion of local differential privacy. Major technology companies, including Google, Apple, and Microsoft, have used LDP to collect user data in widely used systems and applications, such as Chrome [17], iOS [3], and Windows [13].

Despite the promising applications of local differential privacy, recent work [11, 12, 41] has demonstrated the vulnerability of LDP protocols to data poisoning attacks. This is because the data collection in LDP is performed in a distributed manner, and some users may be malicious in real-world scenarios, allowing them to submit carefully crafted values to manipulate the aggregation results. In addition, some LDP protocols employ encoding schemes to spread the influence of a reported value and reduce communication complexity, making the data poisoning attacks more powerful. The local perturbation nature of LDP protocols also makes the data poisoning attacks more concealed, making it difficult for the data collector to distinguish between the poisoned data and the normal data. Previous studies have shown that poisoning attacks can significantly impact the performance of frequency estimation [11, 12], heavy hitters [11], and key-value data aggregation [41] in LDP protocols. However, frequent itemset mining protocols, which aim to discover frequently occurring events, patterns, or associations, have not

been investigated in terms of their vulnerability to data poisoning attacks. Despite being an indispensable and essential component of many data analysis tasks, as well as being widely used in various applications, the security of LDP frequent itemset mining protocols remains unexplored.

In this work, we investigate the security of LDP frequent itemset mining. Our focus is to examine the susceptibility of LDP frequent itemset mining protocols to data poisoning attacks. We propose a novel framework to demonstrate how malicious users can manipulate the results of frequent itemset mining by submitting carefully crafted data to the collector. These attacks can result in the identification of incorrect frequent itemsets and can have a significant impact on pattern discovery and association rule mining tasks. For instance, an attacker could undermine an advertising campaign strategy or promote a harmful product combination by exploiting the vulnerabilities in LDP frequent itemset mining protocols.

Although previous studies [11, 12] have shown the feasibility of attacking the LDP frequency estimation, which is a basic component for the LDP frequent itemset mining protocols [30, 38], or attacking the LDP heavy hitter identification, which shares a similar goal of identifying the most frequent items, it is important to note that successfully attacking these protocols does not necessarily imply the ability to successfully attack LDP frequent itemset mining protocols. Compared with the existing attacks on other LDP protocols, there are three unique challenges to highlight for attacking LDP frequent itemset mining protocols. (1) *The user data is in a more varied format.* In frequent itemset mining, each user is associated with a set of items, while in frequency estimation/heavy hitter identification and key-value aggregation, each user is associated with a single item or a pair of key-value data, respectively. This makes it challenging to craft user data as the domain of set-valued data grows exponentially with the number of items. (2) *The goal of the attacker is more difficult to achieve.* Unlike in frequency estimation and key-value aggregation, where the goal is to estimate the frequencies, the goal in frequent itemset mining is to rank the itemsets and output the top-k itemsets as results. Simply changing the frequencies of some itemsets may not guarantee a change in the ranking. (3) *The protocols are generally more sophisticated.* An LDP frequent itemset mining protocol usually consists of multiple phases of interactions with users, which creates the opportunity for multiple rounds of poisoning. While the multiple phases of interactions indeed present a larger attack surface for the attacker, they also create a more complicated task for the attacker to exploit and manage. Specifically, it is challenging for an attacker to coordinate the different phases in a manner that maximizes the attack performance with a limited number of malicious users.

To address the challenges, we leverage a general framework for LDP frequent itemset mining protocols and identify the possible attack surfaces by analyzing the phases that are most susceptible to be manipulated. We observe that the majority of current LDP frequent itemset mining protocols have two crucial phases: (a) pruning the domain by constructing a candidate set of itemsets, and (b) estimating the frequencies of candidate itemsets and selecting the top-k as the final results. As both phases rely on users' data for aggregation, attackers can leverage these phases to manipulate the results of the mining process. Additionally, some protocols employ an adaptive approach to determine parameters during execution, offering another avenue for attackers to exploit. In particular, we propose our novel *Adaptive Orchestration Attack* (AOA), which is built upon three types of composable attack operations. Given an LDP frequent itemset mining protocol, these attack operations enable the attacker to estimate the attack resource available for each phase, select a refined set of targets, and allocate resources effectively to generate poisoned data to achieve a satisfactory attack outcome.

We apply the proposed attack to two state-of-the-art set-valued frequent item mining protocols, LDPMiner [30] and SVIM [38], and a leading frequent itemset mining protocol, SVSM [38]. Set-valued frequent item mining is a special case of frequent itemset mining, where each user still possesses a set of items, *i.e.*, an itemset, but its goal is to identify the frequent itemsets with only one item. We first consider the scenario where the attack is well-informed. Then, we extend our attack to the cases with partial-knowledge attackers and and a man-in-the-middle attack scenario. In addition, we compare the performance of our attack with four baseline attacks, including two random attacks and two improved variants of a state-of-the-art attack against LDP mechanisms to demonstrate the superior effectiveness of our proposed attack.

The contributions of this paper are summarized as follows:

- To the best of our knowledge, we are the first to study data poisoning attacks to LDP frequent itemset mining protocols.
- We propose an effective attack, which is feasible under various practical threat models and can successfully compromise major state-of-the-art LDP frequent itemset mining protocols under a unified framework, with the help of composable attack operations.
- We demonstrate the effectiveness of our attacks against two set-valued frequent item mining protocols and a frequent itemset mining protocol on a synthetic dataset and two real-world datasets, compared with four baselines. Moreover, some potential defenses for countering the proposed attacks have been discussed and evaluated.

2 Preliminaries

2.1 Local Differential Privacy

In the local setting of differential privacy, each *user* perturbs its input x by using a perturbation mechanism $\mathcal{M}$ and sends $\mathcal{M}(x)$ to the *aggregator*. Below we review the formal definition of local differential privacy (LDP).

Definition 2.1 (Local Differential Privacy). A randomized mechanism $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ satisfies ϵ-local differential privacy, if and only if for any two inputs x and x' and for any output y of $\mathcal{M}$,

$$\Pr[\mathcal{M}(x) = y] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}(x') = y].$$

Frequency oracles. One of the major parts that are used in the LDP frequent itemset mining is frequency oracle (FO) [38], which is a basic component that can estimate the frequencies of items in a domain. Four commonly used frequency estimation protocols are Generalized Randomized Response (GRR) [17], Succinct Histogram (SH) [6], Optimized Local Hashing (OLH) [37], and RAPPOR [17]. For a frequency oracle FO, let $\Psi_{\mathrm{FO}(\epsilon)}(x_j)$ be the function of perturbing the value x_j of user j, $\langle x_j \rangle$ be the encoded value of x_j under the FO, and $\Phi_{\mathrm{FO}(\epsilon)}(R, t)$ be the function of aggregating the reports

R and getting the frequency of item t. The framework in [37] shows that the aggregated relative frequency of item t can be computed as

$$\tilde{f}_t = \frac{\tilde{c}_t}{n} = \frac{\sum_{j=1}^{n} \mathbb{I}_{\text{supp}(y_j)}(t) - q}{n(p-q)}, \tag{1}$$

where y_j is the perturbed value reported by user j; n is the number of users; p and q are the perturbation parameters; and $\mathbb{I}_{\text{supp}(y_j)}(t)$ is an indicator function that returns 1 if $t \in \text{supp}(y_j)$, otherwise 0. The support function $\text{supp}(y_j)$ returns all the input values that y_j supports (*i.e.*, the input values that can produce y_j in the LDP protocol).

Attacks to LDP FOs. For the sake of completeness, we review the study of attacks to the LDP frequency estimation. In [11], Cao et al. investigate how to poison LDP frequency estimation. In their paper, the concept of frequency gain is introduced to formulate the changes that can be achieved by falsely reported values. Formally, the frequency gain on an item t is

$$\Delta\tilde{f}_t = \frac{\sum_{\hat{y}\in\hat{Y}}(\mathbb{I}_{\text{supp}(\hat{y})}(t) - \sum_{y\in Y}\mathbb{I}_{\text{supp}(y)}(t)/n)}{(n+m)(p-q)} \tag{2}$$

Based on the frequency gain, they have proposed the maximal gain attack (MGA) [11] against LDP frequency estimation, which can promote the frequencies of chosen items. Specifically, given a set of items T, MGA tries to maximize the total frequency gain on T. The total frequency gain that can be achieved by the attacker can be considered as the attack resource for targeting the protocols. Although a theoretical bound of the maximum value of the attack resource has been given by Cao et al. [11], the attacker is not always able to obtain this maximum value in practice. For example, in OLH and SH, finding the maximum frequency gain contributed by a single reported value will cost an exponential search complexity.

2.2 Set-Valued Frequent Itemset Mining

We consider two tasks, *i.e.*, the set-valued frequent item mining [30, 38] and frequent itemset mining [38], under the definition of local differential privacy. We assume that there are n users, and each user $j \in [n]$ holds a subset of a set of d items $\mathcal{I} = \{1, 2, \ldots, d\}$. Denote the item subset, *i.e.*, *transaction*, of user j by $S_j \subseteq \mathcal{I}$. The frequency of an item t is defined as $f_t = \sum_{S_j} \mathbb{I}(t \in S_j)/n$; and the frequency of the itemset I is defined as $f_I = \sum_{S_j} \mathbb{I}(I \subseteq S_j)/n$. The frequent item/itemset mining task is to find the top-k most frequent items/itemsets in the corresponding domains.

Then we provide an overview of the family of locally differentially private frequent itemset mining protocols. To illustrate the basic idea of the differentially private frequent itemset mining, we consider two widely used mechanisms in this family: SVIM/SVSM [38] and LDPMiner [30]. Based on the analysis from [38], a frequent itemset mining protocol consists of three major parts: (1) FO choosing and parameter setting (ConfigPro); (2) Domain pruning (PruneDom); and (3) Estimation and top-k selection (SelectTop). For the sake of completeness, we review LDPMiner [30], SVIM [38], SVSM [38], FIML [26], and PrivSet [35] with respect to the above frequent itemset mining framework.

LDPMiner. The three parts of LDPMiner are

ConfigPro. LDPMiner sets the parameters: privacy budget ϵ_1 and ϵ_2 for PruneDom and SelectTop, respectively; padding size $l_1 = L$ for PruneDom and $l_2 = 2k$ for SelectTop. The SH mechanism is chosen as FO in PruneDom and the RAPPOR mechanism [17] is chosen as FO in SelectTop.

PruneDom. The protocol narrows down the item domain by finding the top $2k$ items as a candidate set C by using SH with privacy budget ϵ_1. This step lets user j pad its set of items S_j to length l_1 if $|S_j| < l_1$ before she/he selects an item t_j randomly from S_j and applies $\Psi_{\text{SH}(\epsilon_1)}(t_j)$.

SelectTop. LDPMiner lets each user report $\Psi_{\text{RAPPOR}(\epsilon_2)}(t_j)$, where t_j is randomly selected from $S_j \cap C$ (if $|S_j \cap S| < l_2$, first pads the intersection to size of l_2). Then LDPMiner estimates the frequencies of items in C by applying $\Phi_{\text{RAPPOR}(\epsilon_2)}(R, t)$. Then LDPMiner selects k items in C with the highest estimated frequencies as the top-k frequent items.

SVIM. The three parts of the SVIM protocol are

ConfigPro. The protocol first partitions the users into three groups and sets the initial system parameters: privacy budget ϵ; padding size $l_1 = 1$ for PruneDom and $l_2 = L$ for SelectTop, where L is the 90-th percentile of all the cardinalities of the intersection between the value of each user and the candidate set. GRR will be chosen as the FO if $d < l(4l-1)e^{\epsilon} + 1$ (for $l = l_1, l_2$), where $d = |I|$ for PruneDom and $d = 2k$ for SelectTop; otherwise, OLH will be chosen as the FO.

PruneDom. The protocol uses the first group of users to narrow down the item domain by finding the top $2k$ items and sets them as the candidate set C. In this part, each user first pads the set of items to length l_1 with dummy items and reports only one item t_j from her/his set of items by using $\Psi_{\text{FO}(\epsilon)}(t_j)$.

SelectTop. SVIM estimates the frequencies of $2k$ candidate items by using the FO under settings decided by the former steps as $\Psi_{\text{FO}(\epsilon)}(t_j)$, where t_j is sampled from a padded set with length l_2 of $S_j \cap C$. Then, SVIM estimates the frequencies of items in C by applying $\Phi_{\text{FO}(\epsilon)}(R, t)$ for $t \in C$, and updates the estimation with a factor to fix the bias induced by padding. Then SVIM selects k items in C with the highest estimated frequencies as the top-k frequent items.

SVSM. The three parts of the SVSM protocol are similar to those of SVIM. SVSM can be regarded as a modified version of SVIM, where the *items* are replaced by the itemsets. However, the domain of all itemsets is exponentially larger than the domain of items, making it infeasible to estimate the frequencies of all itemsets. SVSM uses the estimated item frequencies from SVIM to help itself prune the size of the domain of itemsets in the PruneDom phase.

ConfigPro. Users are partitioned into three groups, similar to SVIM, and the protocol decides the system parameters: privacy budget ϵ; padding size l for SelectTop. GRR is chosen as the FO if $d < l(4l-1)e^{\epsilon} + 1$, where $d = 2k$ for SelectTop; otherwise, OLH is chosen as the FO.

PruneDom. After calling SVIM with the first group of users, SVSM gets the top-k items $\mathcal{K}$ along with their estimated frequencies: $\{\tilde{f}_x | x \in \mathcal{K}\}$. The candidate itemsets are selected by guessing the frequencies of itemsets as $\tilde{f}_I = \prod_{t\in I} \frac{0.9\tilde{f}_t}{\max_{x\in\mathcal{K}} \tilde{f}_x}$ from the power set of $\mathcal{K}$. The $2k$ itemsets with the highest guessed frequencies are selected as the candidate itemsets $\mathcal{S}$.

SelectTop. SVSM estimates the frequencies of $2k$ candidate itemsets by having user j report $\Psi_{\mathrm{FO}(\epsilon)}(I_j)$, where I_j is sampled from the set of padded itemsets of length l. Then, SVSM estimates the frequencies of the itemsets in $\mathcal{S}$ by applying $\Phi_{\mathrm{FO}(\epsilon)}(R, I)$ for $I \in \mathcal{S}$, and updates the estimate with a factor to fix the bias induced by the padding. Then SVSM selects k itemsets in $\mathcal{S}$ with the highest estimated frequencies as the top-k frequent itemsets.

FIML. The FIML protocol is able to find both the frequent items and frequent itemsets. We denote these two functions as FIML-I and FIML-IS in this paper, and most parts of them share similar steps. The three parts of the FIML protocol are

ConfigPro. The protocol first divides the users into three groups and sets the privacy budget ϵ. The OLH mechanism is chosen as FO in PruneDom, and the GRR mechanism with a binary domain is chosen as FO in SelectTop.

PruneDom. For frequent items, the protocol prunes the domain by finding the top $1.5k$ items using OLH and sets them as the candidate set C. In this part, each user samples and reports only one item t_j from their set of items using $\Psi_{\mathrm{OLH}(\epsilon)}(t_j)$. For frequent itemsets, the protocol constructs the candidate itemsets from the mined top-k items and prunes the set of candidate itemsets to a size of $1.5k$.

SelectTop. In this part, FIML allows the aggregator to choose an item/itemset randomly for each user from the candidate set and send it to the user. Each user responds by using function $\Psi_{\mathrm{GRR}(\epsilon)}(\cdot)$ to indicate whether the received item/itemset is one of her/his top-k items/itemsets. Then, the aggregator uses $\Phi_{\mathrm{GRR}(\epsilon)}(\cdot)$ to estimate the frequency of each candidate item/itemset and identify the top-k items/itemsets.

PrivSet. The PrivSet protocol [35] is a set-valued data aggregation mechanism, and we can wrap it into a naive top-k item mining protocol by simply selecting k items with the highest estimated frequency as the final result.

ConfigPro. The PrivSet protocol sets the parameters: privacy budget ϵ, padding size l, and domain size d. Then, the report set size κ is determined based on these three parameters to minimize the square error of the estimation.

PruneDom. The PrivSet protocol does not include such a phase.

SelectTop. In PrivSet, reports from all users are collected, and the protocol identifies the top-k items from the aggregated result.

2.3 Threat Model

Attacker's capability. We follow the assumption made in previous work that the attacker can inject/corrupt users and make them report arbitrary values to the aggregator that conform to the formats of the LDP protocols [11, 12]. The corrupted users can create the original values, the encrypted values, and the corrupted values. In addition, we assume that the attacker can provide poisoned data in a man-in-the-middle (MITM) fashion by eavesdropping, crafting, and replaying perturbed values from benign users to the aggregator. We assume that the attacker can inject/corrupt or intercept reports from at most $m \leq n$ users. If we assume that the attacker will inject/corrupt or intercept as many users as possible, then we have that the number of honest users is $n - m$. The set of honest users is denoted by U^b and the set of corrupted users by U^c.

Attacker's background knowledge. We assume that the attacker knows the predefined parameters of the LDP protocols. Since the aggregator has no idea which user is malicious or benign, it will provide each user with the implementation/setup for running the LDP protocols. However, the attacker has no prior knowledge of the parameters that are adaptively determined by the protocols. We also consider two major types of attackers who have different levels of knowledge about the information regarding the frequent itemset mining process: *full-knowledge* and *partial-knowledge* attackers.

Full-knowledge: The attacker knows the true frequencies of all items/itemsets. Although the assumption is somewhat strong, there are some scenarios where it is applicable. For example, there are two aggregators who are competitors. One of them has completed the data aggregation and wants to prevent the other from mining the (genuine) frequent items. In addition, we can use this type of attacker to estimate the maximal severity of such threats.

Partial-knowledge: The attacker only knows the items/itemsets on the injected/corrupted users. The attacker can estimate the true frequencies of all items/itemsets based on the held data by the injected/corrupted users. A variant of the partial-knowledge scenario is the *man-in-the-middle (MITM) attack*, where the attacker can only estimate the frequencies of items/itemsets based on intercepted perturbed reports instead of the true values of the items/itemsets.

Attacker's goal. Suppose an attacker tries to manipulate top-k frequent itemset mining such that the aggregator will get false results. To poison the results of frequent itemset mining, the attacker carefully selects the responses to the aggregator in the interactive process of the LDP protocol. Denoted by the set of responses of all users $\mathcal{R}$, the set of responses of all benign users $\mathcal{R}^b$, and the set of responses of all corrupted users $\mathcal{R}^c$. For each user j, the response $R_j \in \mathcal{R}$ is a sequence of perturbed values $y_{j,1}, y_{j,2}, \ldots$ generated by the LDP protocol. If the users have only a single interaction with the aggregator, then $R_j = y_j$. Specifically, the goal of the attacker is to lower the accuracy of the aggregation results. Formally, the attacker's goal is

$$\min \quad \sum_{t \in \mathcal{I}_k} \mathbb{I}(\tilde{f}_t^* > \tilde{f}_{k+1}^*) \tag{3}$$

where $\mathcal{I}_k$ is the set of true top-k items before the attack and $\mathbb{I}(cond)$ is an indicator function that returns 1 if $cond$ is true; otherwise 0.

3 Attack Framework and Operations

The major building blocks of LDP frequent itemset mining protocols are FOs. The PruneDom and SelectTop of a LDP frequent itemset mining protocol use FOs as tools to select the candidates, set configurations, and choose top-k results based on users' reports. Therefore, the basic idea for the attacks is to leverage the manipulation of the FOs with respect to the upper-level LDP frequent itemset mining protocols. Specifically, we introduce an attack toolkit including three operations for attacking against LDP frequent itemset mining protocols. The first operation is *attack resource estimation*, which can estimate the amount of resources that can be used for the attack given a certain number of corrupted users. The second operation is *target set refinement*, which allows the attacker to select a candidate set of target itemsets for the attack. The third operation is *poisoned data generation*. Given the amount of attack resources and the set of targets, the attacker tries to allocate the resources

Table 1: Notation

Notation	Description
x_j, $\langle x_j \rangle$	value and encoded value of user j
$\mathcal{R}$, $\mathcal{R}_j$	the responses of all users and the response of user j
$\mathcal{R}^b$, $\mathcal{R}^c$	the set of responses of benign and corrupted users
$\Psi_{\mathrm{FO}(\epsilon)}(\cdot)$	function for perturbing values
$\Phi_{\mathrm{FO}(\epsilon)}(\cdot,\cdot)$	function for estimating frequencies
y_j, $\mathrm{supp}(y_j)$	perturbed value of user j and its support
$S(y)$	set of items/itemsets supported by report y
$\mathcal{I}$, d	domain of items and its size
S_j	set of items of user j
f_t, $\tilde{f}_t$	frequency of item t and its estimation
f_I, $\tilde{f}_I$	frequency of itemset I and its estimation
n, m	size of benign and corrupted users
U^b, U^c	sets benign and corrupted users
$\mathcal{C}$, $\mathcal{S}$	candidate item/itemset set
$\mathcal{K}$	estimated top-k items
T, L	attacker's target set and its size

and generate the poisoned responses strategically to maximize the effect of the attack.

3.1 Overview

Attack surfaces. We define a poisoning attack based on a given LDP frequent itemset mining protocol with ConfigPro, PruneDom, and SelectTop. By choosing a set of responses $\mathcal{R}^c$ of the corrupted users, the attacker can potentially manipulate the set of candidate items/itemsets generated by PruneDom, the estimated frequencies of the items/itemsets in the candidate set via SelectTop, and the parameters chosen in ConfigPro.

ConfigPro: An LDP frequent itemset mining protocol tries to choose a type of encoding format, which will be used in the following SelectTop step. The goal of the attacker is to trigger the protocol to choose a configuration of encoding format that can be leveraged to amplify the attack ability in the following step. Specifically, the padding length in SVIM and SVSM is decided by all users, where the attacker can greedily increase support for larger length values or smaller ones to influence the padding length choice.

PruneDom: The protocol tries to select a set of candidates $\mathcal{C}$ containing the top-k results ($|\mathcal{C}| \geq k$) from the item domain $\mathcal{I}$ or itemset domain $2^{\mathcal{I}}$, respectively. Therefore, the goal of the attacker is to manipulate the protocol in this phase so that as many non-top-$|\mathcal{C}|$ items/itemsets as possible can be pushed into $\mathcal{C}$. The attacker can estimate the itemset frequencies of the benign users and therefore identify its targets as those itemsets with high frequency but not in the genuine top-k results. Then the attacker refines its target set until the attack resources meet the requirement. Finally, the attacker will craft the poisoned reports and submit them to the aggregator.

SelectTop: The attacker tries to make as many non-top-k items or itemsets in $\mathcal{C}$ as top-k results by crafting the reports of corrupted users, which is similar to the goal of manipulating PruneDom. However, there are three differences when trying to manipulate SelectTop: (1) SelectTop uses different parameters (*e.g.*, FO, padding size, privacy budget) compared to PruneDom; (2) the item/itemsets domain is $\mathcal{C}$ in SelectTop instead of $\mathcal{I}$ or $2^{\mathcal{I}}$ in PruneDom for frequent item mining and frequent itemset mining, respectively; (3) true top-k items/itemsets remaining in the candidate set are usually with high ranks, meaning that making more progress in this phase will cost more attack resources on each target.

Attack operations. To make the above attack framework possible, we introduce three attack operations: *Attack Resource Estimation*, *Target Set Refinement*, and *Poisoned Data Generation*. For each part of the frequent itemset mining protocol, the attacker uses these three operations to perform an effective attack. Specifically, the attack resource estimation operation is employed to estimate the available attack resources for data poisoning, given a specified number of injected/corrupted users or maliciously modified reports. Then the target set refinement operation refines the target set based on the frequencies of items or itemsets. This operation aims to select targets from the domain of items or itemsets with the highest likelihood of being promoted to the top-k results through the impact of poisoned data. Subsequently, the poisoned data generation operation allocates the estimated attack resources to the selected target itemsets. This operation determines the poisoned reports needed to be crafted in the LDP frequent itemset mining protocol.

These three attack operations can be applied to any of the aforementioned attack surfaces within an LDP frequent item mining protocol. We design the attack resource estimation and poisoned data generation operations in terms of different LDP FOs because different frequent itemset mining protocols use different LDP FOs as the primitives in their design: SVSM and SVIM use GRR and OLH; LDPMiner uses SH and RAPPOR, and propose a unified target set refinement operation. We use AREst, TSRef, and PDGen to denote these three attack operations, respectively. In addition, we use Operation-FO to denote the specified versions of an attack operation Operation with respect to a given LDP frequency oracle FO, *e.g.*, AREst-GRR, PDGen-OLH. The details of these three operations are shown in Section 3.2, Section 3.3, and Section 3.4, respectively.

3.2 Attack Resource Estimation

We define the attack resources as the supports provided by the reports of users. Based on the frequency gain formulation (Eq. (2)) in [11], given a set of corrupted users, the amount of attack resources can be maximized by maximizing the frequency gain of each corrupted user with crafted reports. We consider four FO primitives: GRR, OLH, SH, and RAPPOR.

GRR. Given the perturbation function of GRR:

$$y = \Psi_{\mathrm{GRR}(\epsilon)}(t) = \langle v \rangle,$$

we have that item t is supported by the reported value if and only if the reported item v is the same as item t based on the analysis in [11]. Therefore, each reported value y can support only one item. Then we have the amount of resources produced by corrupted users U^c as $\Omega_{\mathrm{GRR}}(U^c) = |U^c| = m$.

OLH. According to the analysis in [11], the attacker needs to find an appropriate seed and hash value to try to support more items in the target set. Specifically, the attacker's goal is to find a seed r and a hash value v such that as many items in the target set as possible are hashed to the reported value v. MGA [11] adopts a method of finding the best seed by enumerating h (say 1,000)

random hash functions, without estimating the amount of attack resources that can be used. Below we characterize the amount of attack resources. Let α denote the probability that any given item is supported by a randomly reported value. We assume that there are h times of random sampling from the hash function space, and the corresponding reported values are $\hat{y}^{(1)}, \hat{y}^{(2)}, \ldots, \hat{y}^{(h)}$. For any given set of target items T, let $\text{supp}(\hat{y}^{(i)}; T)$ be the amount of items supported in T by the reported value $\hat{y}^{(i)}$. With h times of random sampling, we have the maximum number of supported items among them is

$$s^* = \max_{i=1,2,\ldots,h} \{\text{supp}(\hat{y}^{(i)}; T)\}, \tag{4}$$

and the cumulative probability distribution of s^* is

$$F_{s^*}(l) = \prod_{i=1}^{h} \left[\sum_{j=0}^{l} \binom{b}{j} \alpha^j \beta^{b-j} \right], \text{ for } l = 0, 1, \ldots, b. \tag{5}$$

where $\beta = 1 - \alpha$ and $b = |T|$. Thus the expectation of s^* is

$$E_{s^*}(h, b, \alpha) \triangleq \mathbb{E}[s^*] = \sum_{i=0}^{b} (1 - F_{s^*}(i)), \tag{6}$$

which is a function with parameters: the times of random sampling h, the size b of the target set, and the probability α.

For OLH, we have that a randomly chosen reported value will support any item with a probability of $1/d$ (*i.e.*, $\alpha = 1/d$), where d is the size of item domain considered in the aggregation, and then we have that the expectation of the amount of resources produced by corrupted users U^c is

$$\mathbb{E}\left[\Omega_{\text{OLH}}(U^c)\right] = \mathbb{E}\left[\text{supp}(\mathcal{R}^c; T)\right] = \sum_{j=1}^{m} E_{s^*}(h_j, |T|, 1/d),$$

where h_j is the times of random sampling for the j-th corrupted user to produce the reported value, and we can set all the $h_j = h$ if we assume all the corrupted users choose the same times of random sampling.

SH. SH is a special case of OLH with the size of the encoded domain as 2. Therefore, we can use the same attack resource estimation operation as OLH for SH.

RAPPOR. The RAPPOR protocol allows a report to support multiple itemsets. This means that, given a set of corrupted users U^c with a size of m, the amount of resources required for manipulating RAPPOR is $m \times L$, where L represents the size of the target set. Alternatively, if the attacker chooses all the itemsets as targets, the amount of resources allowed could be $m \times d$.

3.3 Target Set Refinement

Since the item domain is large and the amount of attack resources is limited, it is important to select a set of target items to be the focus of the attack. Basically, there are two ways to manipulate the results of frequent item mining: reduce the frequencies of items in I_k (i.e., the true top-k items), or increase the frequencies of non-top-k items and make them the top-k result.

However, due to the property of LDP mechanisms, if other users' reports remain unchanged, a user can only increase the frequency of a particular item, not decrease it. Thus, our goal is to push as many non-top-k items as possible into the final top-k result, and we denote the top-k result after attack by $\mathcal{I}_k^*$. Intuitively, the attacker should always pick those items with high frequency but not in $\mathcal{I}_k$. If the attacker chooses a set of items $\{\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_L\}$ and wants to push all of them into the top-k results, the frequency of any chosen item should be larger than the $(k+1-L)$-th item before the attack, and we call this item a *pivot item*. Denote by $x_1, x_2, \ldots, x_d$ the sorted items based on their frequencies that are contributed by the benign users from high to low and the corresponding frequencies contributed by the benign users are denoted by $\tilde{f}_1^b, \tilde{f}_2^b, \ldots, \tilde{f}_d^b$. The gap between any item t and the pivot item is defined as $\text{Gap}_t = \max\{\tilde{f}_{x_{k+1-L}}^b - \tilde{f}_t^b, 0\}$.

Denote by the frequencies contributed by the corrupted users as $\tilde{f}_1^c, \tilde{f}_2^c, \ldots, \tilde{f}_d^c$. Clearly, item t can be pushed into the top-k results if the attacker can make a frequency gain on t such that

$$f^*(t) = \tilde{f}_t^c - \tilde{f}_{x_{k+1-L}}^c \geq \text{Gap}_t.$$

After the attack, the estimation of item t's total frequency is $\tilde{f}_t^b + \tilde{f}_t^c$. Below we specify how to determine the number of items to attack and refine the target set. The attacker sorts all the items based on their frequencies that are contributed by the benign users from high to low and picks items from ranked $k+1$ as the candidates in the target set. The frequency increment required to push L items into the top-k result is $\sum_{i=1}^{L} (\tilde{f}_{x_{k+1-L}}^b - \tilde{f}_{x_{k+i}}^b)$.

We can choose the largest L as long as the attack resources are sufficient. Formally, the size of the target set is determined as

$$L^* = \arg\max_{L} \sum_{i=1}^{L} (\tilde{f}_{x_{k+1-L}}^b - \tilde{f}_{x_{k+i}}^b), \tag{7}$$

$$\text{subject to } \sum_{i=1}^{L} (\tilde{f}_{x_{k+1-L}}^b - \tilde{f}_{x_{k+i}}^b) \leq \mathbb{E}[\Omega_{\text{FO}}(U^c)]. \tag{8}$$

and the target set is $T^* = \{x_{k+1}, \ldots, x_{k+L^*}\}$.

3.4 Poisoned Data Generation

For simplicity, we let the refined target set as $T^* = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{L^*}\}$. Our goal is to construct a set of poisoned responses $\mathcal{R}^c$ for the corrupted users U^c such that

$$f^*(\hat{x}) \geq \text{Gap}_{\hat{x}}, \ \forall \hat{x} \in T^*. \tag{9}$$

Below we present how to allocate the attack resources with respect to the resource constraint and T^* to generate poisoned reports for the different FOs used in the LDP frequent itemset mining protocols.

GRR. Each reported value $\hat{y}$ can only support at most one item, *i.e.*, $|S(\hat{y})| \leq 1$, and increase the frequency of supported item by

$$\text{INC}_{\text{GRR}} = \frac{1}{n(p-q)} = \frac{e^\epsilon + 1}{(n+m)(e^\epsilon - 1)}, \tag{10}$$

where p is the probability of reporting the true value of a user in GRR, and q is the probability of reporting a perturbed value. Therefore, for each $\hat{x} \in T^*$, it requires $\lceil \text{Gap}_{\hat{x}} / \text{INC}_{\text{GRR}} \rceil$ corrupted users to report $\langle \hat{x} \rangle$ such that item $\hat{x}$ can get enough increase of the frequency, and the total number of corrupted users required for the set of target items T^* is $\left\lceil \frac{\sum_{\hat{x} \in T^*} \text{Gap}_{\hat{x}}}{\text{INC}_{\text{GRR}}} \right\rceil$.

OLH. Hash functions are used in OLH to map the items to an encoded domain, which makes a reported value can support multiple

Algorithm 1: Attack resource allocation for OLH

Input: T^*, $\text{Gap}_{\hat{x}}$ (for $\hat{x} \in T^*$), h
Output: $\mathcal{R}^c$
Function Sample($\hat{x}_{\text{axis}}$)
  $sup_{\max} \leftarrow 0$;
  for $i = 0 \rightarrow h$ **do**
    $r \leftarrow \text{RandomInt}()$;
    $sup \leftarrow 0$;
    for $\hat{x} \in T^*$ **do**
      if $H_r(\hat{x}) = H_r(\hat{x}_{\text{axis}})$ **then**
        $sup \leftarrow sup + 1$;
    if $sup > sup_{\max}$ **then**
      $y \leftarrow \langle r, H_r(\hat{x}_{\text{axis}})\rangle$;
  return y;
$index \leftarrow \arg\max_{\hat{x}} \text{Gap}_{\hat{x}}$;
$\mathcal{R}^c \leftarrow \emptyset$;
while $\text{Gap}_{index} > 0$ **do**
  $\hat{y} \leftarrow \text{Sample}(index)$;
  $\mathcal{R}^c \leftarrow \mathcal{R}^c \cup \hat{y}$;
  for $\hat{x} \in T^*$ **do**
    $\text{Gap}_{\hat{x}} \leftarrow \text{Gap}_{\hat{x}} - \tilde{f}^*_{\hat{x}}(\hat{y})$;
  $index \leftarrow \arg\max_{\hat{x}} \text{Gap}_{\hat{x}}$;
return $\mathcal{R}^c$;

different items. For an item $\hat{x} \in T^*$, let $\hat{y} = \langle r, H_r(\hat{x})\rangle$ to support it, where H_r is a hash function with a random seed r. The reported value $\hat{y}$ can support a chosen item $\hat{x}_{\text{axis}}$ and other items in T^* as many as possible and the total number items $\hat{y}$ supports is

$$1 + E_{s^*}(h, |T^*| - 1, 1/d),$$

where $E_{s^*}(h, b, \alpha)$ is the function defined in Eq. (6).

For the sampling sequence $\hat{y}^{(1)}, \hat{y}^{(2)}, \ldots, \hat{y}^{(h)}$, the maximum support s^*, which is defined in Eq. (4), and the corresponding reported value $\hat{y}$, the amount of resources, denoted by $S(\hat{y}; t)$, allocated to any item $t \in \mathcal{I}$ satisfies

$$\mathbb{E}[S(\hat{y}; t)] = \begin{cases} 1, & t = \hat{x}_{\text{axis}}, \\ \dfrac{E_{s^*}}{|T^*| - 1}, & t \in T^* \text{ and } t \neq \hat{x}_{\text{axis}}, \\ 1/d, & t \notin T^*. \end{cases} \tag{11}$$

Then the actual increase of the frequency on item t by reporting poisoned response $\hat{y}$ is

$$\Delta\tilde{f}^*_t(\hat{y}) = \text{INC}_{\text{OLH}} \cdot \left(S(\hat{y}; t) - \frac{1}{d}\right), \tag{12}$$

where $\text{INC}_{\text{OLH}} = \frac{1}{(n+m)(p-1/d)}$.

Based on the above formulation, we can find that given a fixed total amount of resources, the support for each target item in T^* is divided equally on expectation. For the items that are not in T^*, a certain amount of resources will still be allocated to them due to the randomness of hash functions. In addition, the total amount of resources may be not able to adequately support all the items in T^*. We have proposed an algorithm based on the water-filling technique for allocating the attack resources. Details of the algorithm are shown in Algorithm 1.

SH. Similar to OLH, SH is also a kind of local hashing protocol. It is a special case of OLH with the size of the encoded domain as 2. Therefore, we can use the same poisoned data generation operation as OLH for SH.

RAPPOR. The attack resource estimation of RAPPOR in Section 3.2 shows that the attack resources $\Omega_{\text{RAPPOR}}(U^c) = m \times L$, given the set of corrupted users U^c.

However, we note that if the attacker uses all mL attack resources, every m attack resources allocated for attacking RAPPOR are coupled and can only be allocated to a single target. Each target receives its m supports and these supports cannot be allocated to other targets. This implies that any target cannot receive more than m supports when the attacker tries to promote its frequency. Therefore, a succinct way to generate poisoned reports with respect to RAPPOR FO is to set the bits representing the itemsets in the target set T^* to 1 and the other bits to 0.

4 Attacking LDP Frequent Itemset Mining Protocols

We specify the attacks against two well-adopted set-valued frequent item mining protocols: LDPMiner [30] and SVIM [38]. In addition, we describe how to apply the attacks against a state-of-the-art frequent itemset mining protocol: SVSM [38] and discuss how to adopt the proposed attack for an attacker with limited knowledge.

4.1 Attacking LDPMiner

Manipulating PruneDom. Given a fixed padding length l, each benign user pads its value (*i.e.*, a set of items) to the padding length and submits a randomly picked item from the set by using SH. The attacker can estimate the genuine frequency and expected frequency gain to acquire the attack resource requirements and amounts for every combination of target items by the attack operation AREst-SH. The attacker thus picks out an available target set where the attack resource amount can meet the requirement by TSRef.

By submitting values supporting these target items, the attacker can promote the target items into a candidate set by allocating the attack resources with PDGen-SH. Note that the attacker needs to push at least $k + 1$ non-top-k items into the $2k$ candidate set C to squeeze out genuine top-k results.

Manipulating SelectTop. To manipulate SelectTop of LDPMiner, the attacker first considers the item domain that shrinks from $\mathcal{I}$ to C and the aggregator picks top-k rather than top-$2k$ frequent items as the result compared with manipulating PruneDom. Usually, the attacker has squeezed out some top-k items in the PruneDom phase. The left genuine top-k items have a higher frequency than those squeezed out, which means the frequency bound becomes higher, and squeezing out more true top-k items becomes harder. Specifically, the attacker employs AREst-RAPPOR and PDGen-RAPPOR to estimate the attack resources and generate poisoned data.

4.2 Attacking SVIM

Manipulating PruneDom. In the PruneDom phase of SVIM, the padding length l is set to 1, which is equivalent to no padding at all. Since users have different sizes of values (sets of items), the estimation can be biased. Another difference from LDPMiner is that

SVIM chooses an adapted FO instead of SH, but the choice is out of the attacker's control since the FO is chosen based on the item domain and padding length, which are set in advance instead of being determined by the collected reports from users. Similar to LDPMiner, the attacker can let the corrupted users submit poisoned reports to increase the frequency estimation of target items by using AREst-FO, TSRef, and PDGen-FO, where FO is GRR or OLH based on the setting of the SVIM protocol. In addition, we will also show how to manipulate SVIM to choose a certain FO when the protocol uses an adaptive and data-driven way to determine the padding length.

Manipulating ConfigPro and SelectTop. A unique challenge for SVIM is that the protocol may employ an adaptive method to choose the FO from GRR or OLH, and the attacker does not know in advance which FO will be chosen. In SVIM, OLH is chosen if

$$d \geq l(4l-1)e^{\epsilon}+1, \tag{13}$$

otherwise GRR is chosen as the FO [38]. In the above equation, d is the domain size, which is fixed for each phase, and l is the padding size. The aggregator uses OLH to estimate the distribution of the length of users' values and sets l as the 90th percentile of the value length among all users for the SelectTop phase. This allows the attacker to manipulate the value of l by having the corrupted users falsely report the value length.

Therefore, the first step to manipulate SelectTop of SVIM is to craft corrupted users' reports to trick the aggregator into choosing a certain FO. In Eq. (13) we can notice that OLH is chosen if l is small enough, otherwise GRR is chosen. Hence, the attacker has two options: 1) let the corrupted users submit reports to support as many target items as possible while having as small a value length as possible to decrease the padding length l so that the aggregator will choose OLH as the FO; 2) increase the padding length l to make GRR be chosen as the FO by letting the corrupted users report larger value lengths.

Intuitively, both options seem viable. However, we find that the former strategy hardly works because the item domain size d already shrinks to $2k$ in SelectTop. Since d is quite small and the right-hand side of Eq. (13) grows asymptotically proportional to the square of l, the padding size must be very small for OLH to be chosen. In our experiment, we find that even if the attacker controls up to 10% of the users to submit crafted reports, the padding length is still too large to get the OLH to be chosen as the FO. Moreover, it may also limit the attack capability to lower the padding length. Therefore, we choose the later option of manipulating the aggregator to set a longer padding length and choose GRR as the FO. Let's explain why this is the better strategy with an example. For the first option, suppose all the reports of the benign users provide 100 supports for different length values, the attacker must provide 900 supports for lower values to make the $90th$ percentile in the attacker's control. In contrast, if the attacker tries to increase the estimate of the $90th$ percentile by providing reports that support higher length values, only 11 supports are needed to make the $90th$ percentile under the control of the attacker. In addition, since the attacker has manipulated the aggregator to set a longer padding length, the benign users will pad their value with more dummy items, giving the attacker a greater opportunity to push more target items into the final top-k results.

Given that the SVIM protocol employs OLH to estimate the padding size l, the attacker utilizes AREst-OLH, TSRef, and PDGen-OLH to manipulate the aggregator to choose GRR as the FO. The attacker manipulates SVIM by using AREst-GRR to estimate the supports that can be allocated for the attack, TSRef to select targets, and PDGen-GRR to generate poisoned reports to affect the estimation of top-k results.

4.3 Attacking SVSM

Manipulating PruneDom. SVSM performs the SVIM protocol as the first step of its PruneDom. After the top-k frequent items are mined by the SVIM, SVSM uses the result to pruned the domain of itemsets. We note that the aggregator has no further interaction with users after the SVIM protocol is executed in the PruneDom of SVIM. Therefore, the attacker can only manipulate the first step of SVSM's PruneDom phase by the attack in Section 4.2 (including manipulating PruneDom and SelectTop phases of SVIM), and cannot influence the following parts of this phase. Since the determination of the set of candidate itemsets is based on the results of SVIM (*i.e.*, the top-k frequent items), if we can successfully attack SVIM, the performance of SVSM will also be significantly degraded.

Manipulating ConfigPro and SelectTop. We use a similar way as in SVIM to manipulate these two parts of SVSM by treating the itemsets in SVSM as items in SVIM. First, we manipulate the server to choose a longer padding length by committing reports that support long padding length values. Then we estimate the true itemset frequencies from our knowledge and choose target itemsets to be supported in the final estimation part.

4.4 Attacking FIML

Manipulating PruneDom. FIML utilizes OLH to estimate the frequencies of items and construct candidate sets. Consequently, an attacker can estimate the genuine frequency and obtain an attack resource estimation using the AREst-OLH attack operation. The attacker selects targets using TSRef and generates poisoned reports using PDGen-OLH. To manipulate the results and undermine the accuracy of the top-k results, the attacker must include at least $k+1$ non-top-k items in the candidate set C, which has a size of $1.5k$.

Manipulating SelectTop. In the protocol's SelectTop phase, a membership query is performed by selecting an random item/itemset from the candidate set for each user and asking the user whether the chosen item/itemset is one of her/his top-k items/itemsets. In FIML, the GRR mechanism with a binary domain is used as FO for this step. When responding to a query, the attacker follows a specific strategy: if the received item/itemset is in the attacker's target set, the attacker reports 1; otherwise, the attacker reports 0.

4.5 Adapting to Scenarios with Less Information

The major assumption of the full-knowledge attack is that the attacker knows the true frequencies of all items/itemsets, which could be a restrictive assumption in some cases. We try to remove this assumption and adapt our *Adaptive Orchestration Attack* (AOA) to more practical scenarios. Specifically, we introduce two attacks, which have limited information about the prior knowledge, against LDP frequent itemset mining protocols:

Partial-knowledge attack. Because the attacker has compromised a group of users, she/he can use the values of those users to estimate the frequencies of items. Specifically, in the full-knowledge attack, we assume that the attacker knows the true frequencies of items or itemsets, $\{f_t\}_{t\in\mathcal{I}}$ or $\{f_I\}_{I\subseteq\mathcal{I}}$, respectively. The partial-knowledge attack only requires that the attacker knows the transactions possessed by the corrupted users, and the attacker uses these transactions to estimate the frequencies of items. For the partial-knowledge attack, due to the limit of the observation, it may be difficult for the attacker to accurately estimate the frequencies of items. Previous work [19] shows that some statistical techniques, *e.g.*, Bootstrapping, can be used to refine the estimated frequencies in this case.

Man-in-the-middle (MITM) attack. In the case where the attacker can eavesdrop on the messages between the users and the collector, the attacker can capture the perturbed reports from the users. Thus, the attacker can also use this knowledge to estimate the frequencies from benign users. The attacker can exploit the aggregation part of the LDP protocol and obtain the estimated frequencies of items/itemsets. Unlike the partial-knowledge attack, the MITM attacker can only obtain the perturbed data of the LDP protocol. When the privacy budget is low, the perturbed reports of the users may be inaccurate, which leads to a poor target set determination for the attacker. However, the low privacy budget does not affect the attacker's simulation of the data collector's frequency estimation, because the attacker received exactly the same perturbed data from users as the collector.

We note that the attack resource estimation and the poisoned data generation operations do not rely on the knowledge of the frequencies of items/itemsets and such scenarios with less information only affect the target set refinement operation.

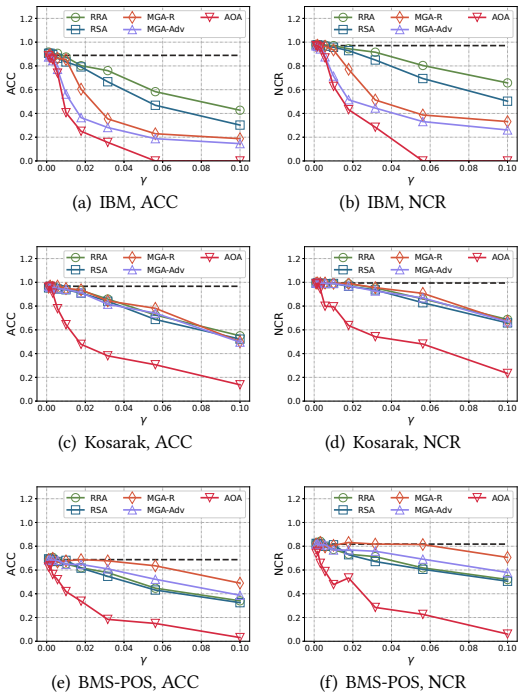
5 Evaluation

In this section, we introduce the setup for experiments. Next, we present the experimental results for attacking LDP frequent itemset mining protocol, SVSM, LDP frequent item mining protocols, LDPMiner and SVIM. We also evaluate the impact of different parameters of the LDP protocols and the attacks with limited knowledge. The codes are available in https://github.com/CorneyHeY/Poison-Attack-LDP-Frequent-Itemset-Mining-CCS2024. Due to the limit of space, more results will be included in the long version of this paper [34].

5.1 Setup

We compare the proposed attack with four other attacks:

- *Random Report Attack* (RRA): For each corrupted user j, RRA randomly selects a sequence of perturbed values (a report) for it to answer the aggregator's requests in the LDP frequent itemset mining protocols.
- *Random Set-Value Attack* (RSA): RSA randomly selects a subset of items for each corrupted user. The size of the chosen subset is equal to the cardinality of the original set of items held by the corresponding user.
- Variants of *Maximal Gain Attack* (MGA) [11]: MGA is a data poisoning attack against LDP frequency estimation, which cannot

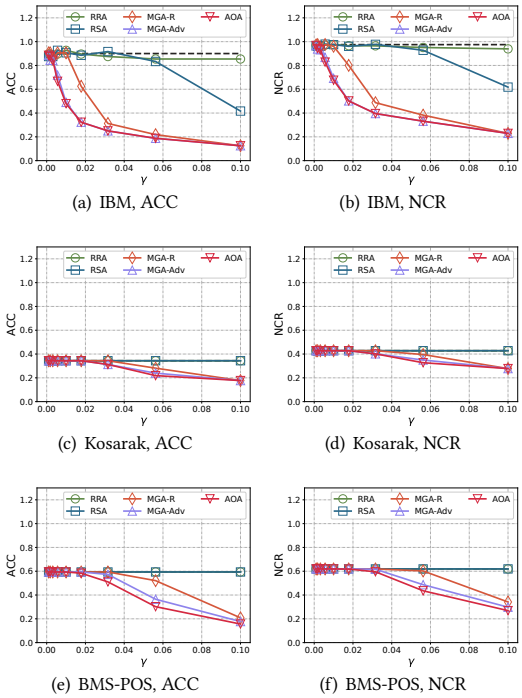


(a) IBM, ACC (b) IBM, NCR

(c) Kosarak, ACC (d) Kosarak, NCR

(e) BMS-POS, ACC (f) BMS-POS, NCR

Figure 1: Attacking SVSM, with $k = 32$, $\epsilon = 4.0$ (black dashed line - no attack).

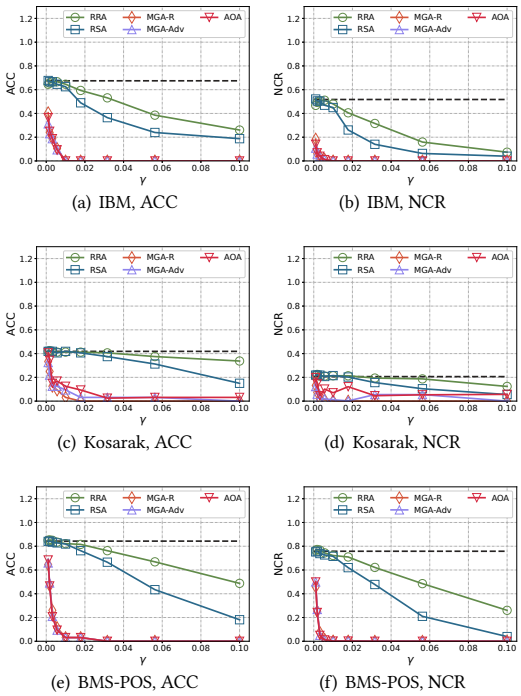


(a) IBM, ACC (b) IBM, NCR

(c) Kosarak, ACC (d) Kosarak, NCR

(e) BMS-POS, ACC (f) BMS-POS, NCR

Figure 2: Attacking FIML-IS, with $k = 32$, $\epsilon = 4.0$ (black dashed line - no attack).

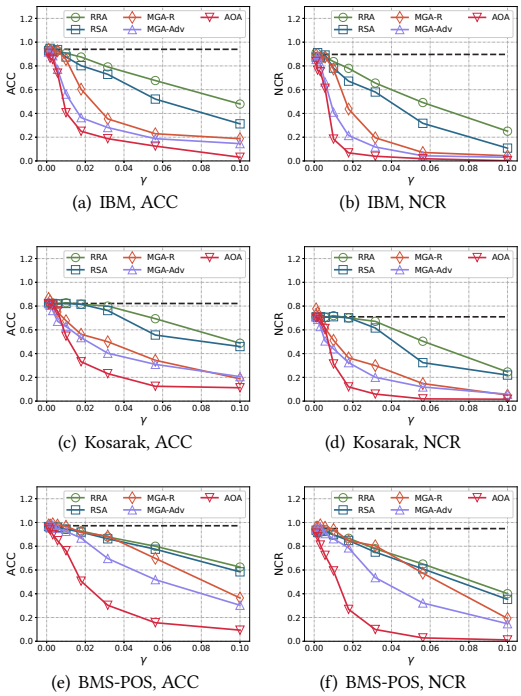


(a) IBM, ACC (b) IBM, NCR

(c) Kosarak, ACC (d) Kosarak, NCR

(e) BMS-POS, ACC (f) BMS-POS, NCR

Figure 3: Attacking LDPMiner, with $k = 32$, $\epsilon = 4.0$ (black dashed line - no attack).



(a) IBM, ACC (b) IBM, NCR

(c) Kosarak, ACC (d) Kosarak, NCR

(e) BMS-POS, ACC (f) BMS-POS, NCR

Figure 4: Attacking SVIM, with $k = 32$, $\epsilon = 4.0$ (black dashed line - no attack).

be directly applied to attack against LDP frequent itemset mining because it can only promote the frequencies of items when the targets have been determined. Therefore, we consider two baseline attacks that incorporate the proposed attack operations into MGA for attacking LDP frequent itemset mining protocols. 1) MGA-R: this attack first estimates the frequencies of items, uniformly randomly selects k items from the non-top-k items as the targets, and then maximizes the frequency gains of these items by MGA; 2) MGA-Adv: this attack uses AREst to estimate the attack resources, uses TSRef to get the set of target items, and then promotes the frequencies of items in the target set by MGA.

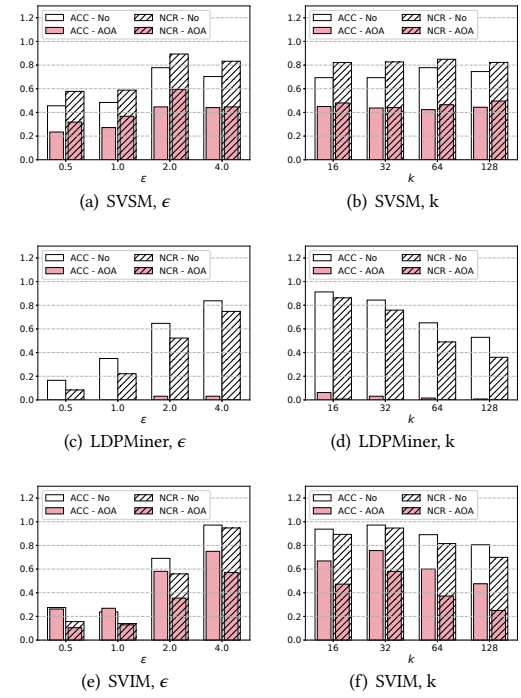
Datasets. We evaluate the proposed attacks on three datasets.

- IBM Synthesize [1]: A dataset generated by IBM Synthetic Data Generator [1], containing 1.8 million transactions with 1,000 items.
- BMS-POS [24]: A dataset of points of sale data, containing more than half a million users and 1,657 items [2].
- Kosarak [7]: A dataset of clickstreams on a Hungarian news portal, containing more than one million users and the domain of items is 42,000.

Metrics. We adopt two metrics to measure the performance of the proposed attacks.

- Accuracy (ACC). Let $\mathcal{K}^*$ be the true top-k items/itemsets, and $\mathcal{K}'$ be the top-k result mined by the LDP frequent itemset mining protocol under attack: ACC = $|\mathcal{K}^* \cap \mathcal{K}'|/k$.

[1] https://github.com/zakimjz/IBMGenerator, provided by Mohammed J. Zaki.
[2] https://github.com/cpearce/HARM, prepossessed by Chris Pearce.



(a) SVSM, ϵ (b) SVSM, k

(c) LDPMiner, ϵ (d) LDPMiner, k

(e) SVIM, ϵ (f) SVIM, k

Figure 5: AOA on BMS-POS with changing ϵ and k; Default: $\gamma = 0.01$, $k = 32$, and $\epsilon = 4.0$. The y-axis is ACC/NCR.

- Normalized Cumulative Rank (NCR). Let $w(x)$ be the weight of an item/itemset x, for the item or itemset with the highest frequency its weight is k, and for others, the weight decreases by one according to their ranks. For the item/itemset with rank k, its weight is 1, and for $x \notin \mathcal{K}^*$, $w(x) = 0$: NCR = $\frac{\sum_{x \in \mathcal{K}^*} w(x)}{\sum_{x' \in \mathcal{K}'} w(x')}$.

We note that an attacker tries to manipulate an LDP frequent itemset mining protocol to degrade its performance. Therefore, for the metrics in our experiments, **lower values of the evaluated metrics mean better attack performance.**

5.2 Results for LDP Frequent Itemset Mining

We first present the evaluation of the attack performance against the frequent itemset mining protocols, SVSM [38] and FIML-IS [26]. We set the privacy budget of both SVSM and FIML-IS as $\epsilon = 4.0$, and the protocol tries to find top-k itemsets with $k = 32$. We change the fraction of malicious users γ from 0.001 to 0.1.

Figure 1 and Figure 2 illustrate the effect of γ on the attack performance on SVSM and FIML-IS, respectively. In general, we can observe that both the baseline attacks and our attack can degrade the performance of the LDP frequent itemset mining protocol to a lower level with more malicious users, and our attack, AOA, can cause the most serious damage to SVSM. Our analysis of the IBM Synthesize dataset reveals that MGA-R and MGA-Adv also exhibit favorable attack performance when γ is sufficiently high compared with other two random baseline attacks, while AOA still exhibiting superior attack performance in various cases.

For SVSM, Figure 1c and Figure 1d show the results on the Kosarak dataset, and Figure 1e and Figure 1f show the results on the BMS-POS dataset. We can observe that our attack AOA can significantly undermine the protocol, while the other four attacks fail to produce effective attacks. Additionally, it is worth noting that MGA-Adv demonstrates comparatively strong performance on the IBM Synthesize dataset, but performs inadequately on both the BMS-POS and Kosarak datasets. This disparity can be attributed to the fact that real-world datasets (*e.g.*, BMS-POS and Kosarak) often contain itemsets with vastly differing frequencies, making it difficult for attacks on LDP frequency estimation to be successful against them. For FIML-IS, it is observed from Figure 2 that the advantage of AOA becomes less significant compared to other attacks. The reason for this is that the SelectTop phase of FIML-IS restricts the reports of users to membership tests only. While this approach reduces the attack surface, it also results in a limitation on the performance of the protocol when the domain of itemsets is large, e.g., for Kosarak and BMS-POS.

5.3 Results for LDP Frequent Item Mining

We evaluate the attack performance against the set-valued frequent item mining protocols, LDPMiner [30], SVIM [38], and FIML-I [26]. We set $\epsilon = 4.0$ and $k = 32$, and change the fraction of malicious users γ from 0.001 to 0.1. Figure 3 and Figure 4 show the impact of γ on the attack performance.

For LDPMiner, we can find that both the ACC and NCR drop rapidly with γ increasing when it is attacked by AOA, and metrics decrease to less than 0.1 when $\gamma \geq 0.03$ for all three datasets. We can observe that both MGA-R and MGA-Adv also achieve a good attack performance against LDPMiner because LDPMiner uses basic

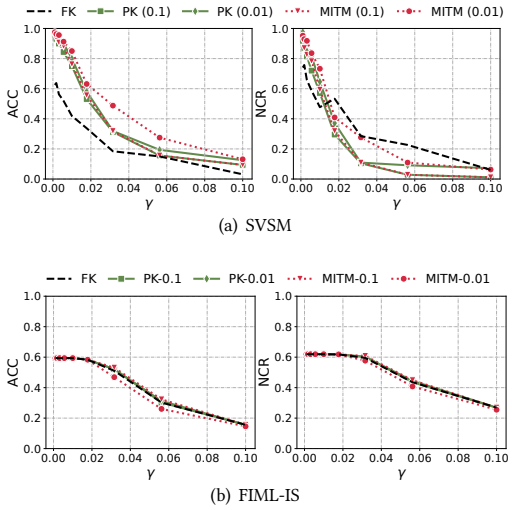


Figure 6: Attacking the frequent itemset mining protocols with Limited Knowledge on BMS-POS.

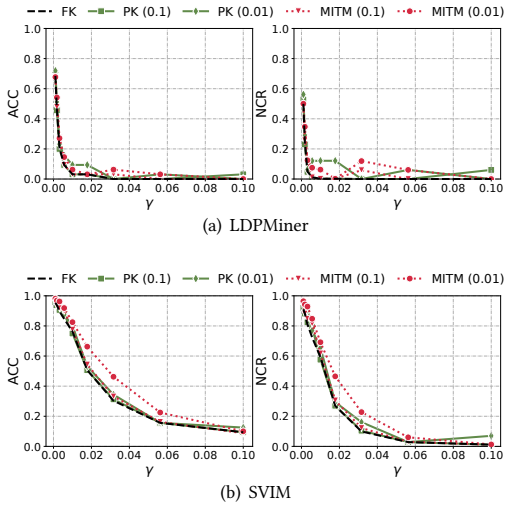
RAPPOR [17] as the FO in the SelectTop phase. Since basic RAPPOR submits a d-bit array with each bit representing support for each item/itemset, thus MGA-Adv and AOA can both provide support for all items/itemsets in the target set, even without the attack resource estimation and allocation operations in our proposed attack.

For SVIM, AOA significantly outperforms other attacks when the fraction of malicious users is in a reasonable range. When the fraction of malicious users is too small ($\gamma < 0.01$), none of the attacks can significantly degrade the performance of SVIM. With γ increasing, AOA can significantly degrade the performance of SVIM. For example, when γ increases to about 0.03, AOA can reduce ACC from 0.82 to 0.23 and NCR from 0.71 to 0.06 on the Kosarak dataset, and ACC from 0.97 to 0.30 and NCR from 0.95 to 0.10 on the BMS-POS dataset. For the same reason as that for SVSM, MGA-Adv can also achieve a satisfactory performance on the IBM dataset. Both AOA and MGA-Adv can degrade the metrics of SVIM on the IBM Synthesize dataset to less than 0.05 when $\gamma \geq 0.05$. Nevertheless, AOA still outperforms all the baseline attacks for various cases. In addition, when the number of malicious users is large ($\gamma = 0.10$), AOA is able to degrade the NCR to almost 0 for all the three datasets as we can observe in Figure 4b, Figure 4d, and Figure 4f. Another interesting observation from our experiments is that SVIM has a better performance compared to LDPMiner, and SVIM is also more robust than LDPMiner against the data poisoning attacks.

Similar results can be observed for FIML-I, as in the case of FIML-IS, due to the restricted attack surface resulting from the SelectTop phase of the protocol. However, it is worth noting that the proposed attacks still demonstrate the best attack performance in various cases.

5.4 Impact of Different Parameters

We investigate the impact of the privacy budget ϵ and the output size k on the attack performance. Specifically, we aim to examine the robustness of the LDP frequent itemset mining protocols with



(a) LDPMiner

(b) SVIM

Figure 7: Attacking the frequent item mining protocols with Limited Knowledge on BMS-POS.

different parameters against the proposed poisoning attack. We set $\gamma = 0.01$ and $k = 32$, and change ϵ from 0.5 to 4.0 to evaluate the impact of different privacy budgets. We set $\gamma = 0.01$ and $\epsilon = 4.0$, and change k from 16 to 128 to evaluate the impact of different output sizes. The results of changing ϵ and k are shown in Figure 5. We define the drop ratio as $\frac{\text{Met}^* - \text{Met}'}{\text{Met}^*}$, where Met^* represents the value of the metric in the absence of an attack and Met' is the value of the metric under attack.

For SVIM and LDPMiner, in general, we observe that the drop ratios increase with the privacy budget increases. These findings suggest that protocols with stronger privacy protection (smaller privacy budget) may be more robust to data poisoning attacks. However, it is important to note that a smaller privacy budget does not guarantee satisfactory performance even without the attack. For SVSM, the drop ratios in NCR are found to be 0.45, 0.38, 0.34, 0.46, respectively, and the drop ratios are 0.49, 0.44, 0.43, 0.37 in ACC, respectively. It is worth noting that in these experiments, we set the fraction of corrupted users to be 0.01, which means only 1% of the reports have been poisoned. This setting allows us to evaluate the impact of different parameters. While a stronger attack intensity may overwhelm the impact of parameters, we emphasize that a realistic setting could set γ up to 0.1, bringing a significant improvement in the drop ratios.

For SVSM, both the NCR and the ACC are not significantly affected by the change in k. For LDPMiner, the drop ratios in NCR and ACC are found to be larger than 0.95 in most cases. As for SVIM, the drop ratios in NCR are 0.47, 0.39, 0.54, and 0.64 for $k = 16$, 32, 64, and 128, respectively. The drop ratios in ACC exhibit a similar trend. Essentially, the results indicate that as k increases, the attack becomes more powerful. A larger top-k set of items or itemsets creates a wider attack surface, allowing the attacker more room to refine the target set and manipulate the results.

5.5 Evaluating Attacks with Limited Knowledge

In this set of experiments, we evaluate attacks with limited knowledge. We set $\epsilon = 4.0$ and $k = 32$, comparing the performance of the partial-knowledge attack (PK), the man-in-the-middle attack (MITM), and the full-knowledge attack (FK) in two scenarios. In these scenarios, the PK attacker or the MITM attacker obtains 1% or 10% of the knowledge about the frequencies of itemsets, denoted by PK-0.01 and PK-0.1 in the figures, or intercepts 1% or 10% of the communication between the users and the data aggregator, denoted by MITM-0.01 and MITM-0.1 in the figures.

The results (Figure 6 and Figure 7) show that both the PK attack and the MITM attack can achieve results comparable to those of the FK attack. This further demonstrates the feasibility of our data poisoning attacks against LDP frequent itemset mining protocols. Specifically, we observe that when 10% of the knowledge or communication can be obtained or intercepted by the PK attacker or the MITM attacker, respectively, the attack performance of the attacks with limited knowledge is almost as good as that of the full-knowledge attack across various cases. However, when only 1% of the knowledge or communication can be obtained or intercepted by the PK attacker or the MITM attacker, respectively, the attack performance degrades due to the inaccurate estimation of the frequencies under this setting.

6 Related Work

6.1 LDP Frequent Itemset Mining

Differential privacy [15, 16] has become the de-facto approach for privacy-preserving data analysis. When the data is collected from distributed users and there is no trusted data collector, local differential privacy (LDP) [17, 22] has been widely adopted in various tasks, such as frequency estimation [6, 17, 36, 37, 40, 42], heavy hitter identification [5, 20, 39, 43], frequent itemset mining [30, 38], and distributed learning [28, 29, 32].

In this paper, we focus on the LDP frequent itemset mining protocols. Qin et al. [30] propose LDPMiner, an LDP frequent item mining protocol, which is a special case of frequent itemset mining, and the mining result is the top-k frequent items rather than itemsets. Frequent item mining is also a type of heavy hitter identification, but unlike the previous work, LDPMiner is performed on set-valued data (i.e., each user has a set of items), while the typical LDP heavy hitter identification protocols [5, 39] are for the cases where each user has only a single value. Wang et al. [38] propose a more general protocol for LDP frequent item mining, named SVIM, which leverages a padding FO technique to improve the performance of the frequent item estimation. They also extend their work to propose SVSM, the first LDP frequent itemset mining protocol. Li et al. [26] also propose a frequent itemset mining protocl, named FIML, which incurs low communication and computation costs by using Random Response in the user response mechanism. Wang et al. [35] propose a delicate Frequency Oracle called PrivSet, working on set-valued data to estimate user item distribution, using an exponential mechanism as the randomization mechanism. We wrap it into a single-phase frequent item mining protocol and apply our attack to further illustrate its versatility.

6.2 Data Poisoning Attacks

In recent years, data poisoning attacks have gained tremendous popularity in machine learning, *e.g.*, [8, 18, 21, 25, 33], where the attacker contributes manipulated training data and attempts to influence the learned model. Beyond the interest in poisoning attacks against machine learning, recent work has explored data poisoning attacks against LDP protocols, *e.g.*, [4, 11, 12, 27, 41].

Cheu et al. [12] propose the attacks against LDP frequency estimation and heavy hitter identification protocols with the goal to degrade the aggregation performance, and the authors also explore the relationship between the attack bias and the injected data distribution. In [11], Cao et al. propose an attack called maximal gain attack, with the aim of promoting the frequencies of target items by letting the fake users report crafted values against LDP frequency estimation and heavy hitter identification protocols. Although the data poisoning attacks on heavy hitter identification, which is similar to frequent item mining, have been studied in these two works, their attacks can only be applied to single-valued heavy hitter identification, e.g.,PEM [39], and cannot apply well to set-valued heavy hitter identification (*i.e.*, frequent item mining), *e.g.*, LDPMiner [30], SVIM [38], which we have considered in this paper. Besides the frequency estimation task over single-valued data, Wu et al. [41] investigate the data poisoning attack against frequency estimation and mean over key-value data, and in [27], Li et al. investigate how to manipulate the estimation of the mean and the variance against LDP protocols.

In the face of the risks of data poisoning attacks in LDP protocols, prior research, *e.g.*, [2, 11, 23, 31], also proposes some approaches to mitigate the attacks. Cao et al. [11] propose two types of defenses against attacks at the level of FOs, namely normalization and fake user detection, which can mitigate the attacks to some extent. Cryptographic techniques, such as Multi-Party Computation and Zero-Knowledge Proof, are also employed to ensure the honest behavior of users, *e.g.*, [23, 31]. Kato et al. [23] propose Cryptographic Randomized Response Technique (CRRT) and use it to build a cryptographic version of FOs, which is equivalent to employing a Trusted Third Party (TTP) to execute the FO.

In this work, we demonstrate that the LDP frequent itemset mining protocols are also vulnerable to data poisoning attacks. Compared with previous work, our work focus different tasks (*i.e.*, frequent itemset mining) and different types of data (*i.e.*, set-valued data), which make the previous work cannot apply well to the problem in this paper.

7 Potential Defenses

In this section, we delve into potential defenses against data poisoning attacks for locally differentially private frequent itemset mining. Additionally, we explore whether these potential defenses, along with existing ones against such attacks on LDP frequency estimation, can effectively mitigate the proposed attacks through extensive experimental evaluation.

7.1 Securing the FOs

To counteract the attack, one possible approach is to secure the frequency oracles used in frequent itemset mining protocols. While some defenses for LDP frequency estimation have been explored in [11], such as normalizing estimated frequencies or detecting malicious users, the effectiveness of these defenses against GRR or SH remains unclear. Furthermore, the proposed defenses have been shown to lack satisfactory performance against OLH. An important observation is that malicious users often try to craft the LDP response to support more preimages. To mitigate the attack effect on FOs, one potential approach is to limit the number of items a response can support by clipping it, which would restrict the ability of malicious users.

Specifically, we examine a defense strategy called FilterFO, which filters user reports supporting items larger than the threshold. For OLH, we discard all reports with supports larger than θ, where $\theta \in [1, d]$ is a parameter to trade off between utility and security. The false positive rate (FPR) of the filter, i.e., the tail distribution for the number of supports from a benign report y is close to the tail of binomial ditribution $B(d, 1/g)$:

$$\Pr\left[S(y) \geq \theta\right] = \frac{I_{(1/g)}\left(\lceil\theta\rceil - 1, d - \lceil\theta\rceil + 1\right) + I_{(1/g)}\left(\lceil\theta\rceil, d - \lceil\theta\rceil\right)}{2}$$

where d represents the size of the domain of items/itemsets, g represents the size of the outputs, I is the regularized incomplete beta function. For RAPPOR, the expectation of the number of support from a benign report y for threshold θ is $\mathbb{E}[S(y)] = 1 + (d-2)q$, where $q = \frac{1}{\exp(\epsilon/2)+1}$ represents the probability to flip a bit of the reported d-bit array. The distribution of the number of support from a benign report y is

$$\Pr[S(y) \geq \theta] = q \cdot I_q(\lceil\theta\rceil, d - \lceil\theta\rceil) + (1-q) \cdot I_q(\lceil\theta\rceil - 1, d - \lceil\theta\rceil + 1)$$

This defense strategy shares similar idea with the detect fake user method proposed in [11], but is more neat and can be applied to various FOs.

Another type of defense aims to secure the FOs by using cryptographic techniques to verify the reports, e.g., [23, 31], in order to prevent the manipulation of the output of $\Psi_{\mathrm{FO}(\epsilon)}(\cdot)$ for a given input item/itemset. However, this type of defense faces a major drawback: there is a significant computational overhead. The use of cryptographic techniques also weakens the motivation for deploying LDP because good performance can be achieved by incorporating cryptography with central differential privacy if more computational overheads are allowed, as shown in [9, 10]. Additionally, this type of defense cannot mitigate the manipulation of the inputs, i.e., changing the value of a user before using $\Psi_{\mathrm{FO}(\epsilon)}(\cdot)$ to perturb it. We denote this type of defense as CryptoFO in the following parts of this paper.

7.2 Configuring the Protocols with Uncertainty

Although securing the underlying FOs is a valid approach for defending against attacks on LDP frequent itemset mining, the deployment of such a defense may face challenges. LDP frequent itemset mining protocols often use FOs as black-box building blocks, making it impractical to modify the details of FOs in some cases. We also explore an alternative strategy to defending against attacks by introducing randomness into the LDP frequent itemset mining. For example, the knowledge of the padding size and the number of candidates forms a key ability of the attacker to make the data poisoning attack successful. If the aggregator chooses the padding

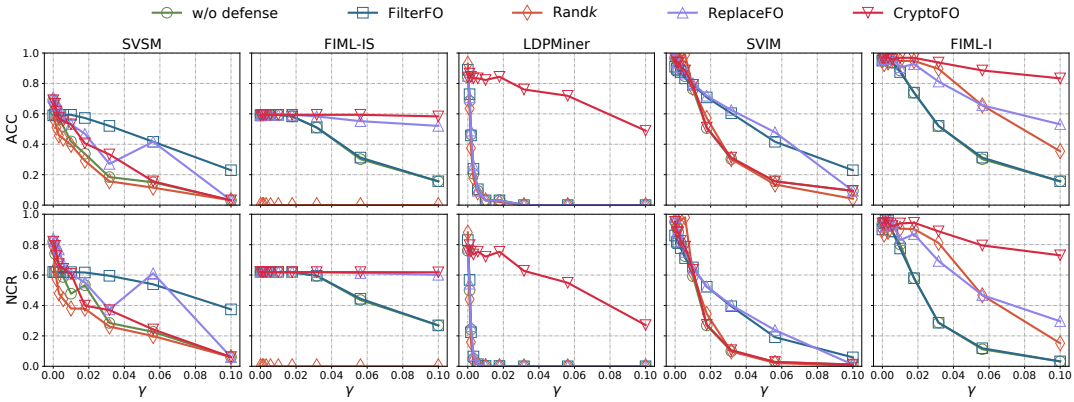


Figure 8: Defenses against AOA on BMS-POS dataset, with $k = 32$, $\epsilon = 4.0$.

size and determines the number of candidates randomly, the attacks may be mitigated to a great extent. For example, in SVSM, LDPMiner, and SVIM, the size of the candidate set is set to $2k$, while it is set to $1.5k$ in FIML. The padding size will affect the choice of FOs in SVSM and SVIM.

Specifically, we consider two countermeasures in this part: (1) Rand$k$: using a randomly chosen $k' > k$ to replace the original mining objective, aiming to disrupt the attacker's target set estimation step; (2) ReplaceFO: it is difficult to dynamically change the FO, but the aggregator can replace the FO used in the protocols. As aforementioned, OLH can achieve good performance when there is no attack, but it also leaves a larger attack surface for the attacker. Therefore, ReplaceFO replaces OLH with GRR in the protocols.

7.3 Experimental Results

We empirically evaluate the effectiveness of the four aforementioned defenses against the proposed attack. It is important to note that CryptoFO employs cryptographic techniques to prevent the poisoning of reports sent to the aggregator. In our evaluation, we let corrupted users to manipulate the mining results by crafting their original values (inputs to the perturbation function $\Psi_{\mathrm{FO}(\epsilon)}(\cdot)$).

We have conducted experiments with $k = 32$, $\epsilon = 4.0$, and $\gamma = 0.001$, 0.01, or 0.1 to evaluate the effectiveness of various defense strategies on BMS-POS dataset. Figure 8 shows the experimental results for scenarios with no defense, FilterFO, Rand$k$, ReplaceFO, and CryptoFO. In our experiments, the filter threshold of FilterFO is set as $\frac{2d}{g}$ for OLH and $\frac{2d}{\exp(\epsilon/2)+1}$ for RAPPOR. Our observations indicate that the defense strategies generally show some effectiveness, but they cannot fully mitigate the impact of the attack. For FIML-IS, LDPMiner, and FIML-I, CryptoFO achieves better defense performance by preventing the manipulation of LDP mechanism outputs, thereby significantly limiting the attacker's capabilities. However, it is worth noting that this defense may not be practical as it weakens the motivation and restricts the deployment scenarios for LDP protocols.

Among the other defenses, ReplaceFO demonstrates better performance across a wider range of settings compared to the other two defenses for FIML-IS. By replacing OLH with GRR, ReplaceFO effectively restricts the attacker's ability to carry out the attack. FilterFO can achieve better performance for SVSM and SVIM because the OLH is used in these two protocols. It is worth noting that, in general, for attacks involving a large number of corrupted users (*i.e.*, $\gamma = 0.1$), none of the defenses can achieve satisfactory performance. Additionally, it may cause some performance degradation to LDP protocols when there is no attack. The results highlight the limited effectiveness of the current defenses, indicating the need for more systematic defenses against our attacks.

8 Conclusion

In this paper, we conduct the first systematic study on data poisoning attacks to LDP frequent itemset mining protocols. We propose a unified attack framework that can significantly degrade the performance of the essential LDP protocols for frequent itemset mining by carefully crafting the reports from malicious users. We evaluate the proposed attack with extensive experiments on a range of scenarios and the results demonstrate the effectiveness of our attack against LDP frequent itemset mining protocols, even with a limited proportion of malicious participants. We have also discussed and evaluated potential defenses against our attacks. In conclusion, our work sheds light on the security of LDP frequent itemset mining protocols and highlights the need for further research in this area to ensure the robustness and reliability of these protocols in real-world scenarios.

Acknowledgments

We thank the anonymous reviewers for their comments. This work was supported by the National Natural Science Foundation of China (62372226, 62272215, 62002159) and the Leading-Edge Technology Program (BK20202001) of the Natural Science Foundation of Jiangsu Province, China.

References

[1] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. 1996. Fast Discovery of Association Rules. In *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, USA, 307–328.

[2] Andris Ambainis, Markus Jakobsson, and Helger Lipmaa. 2004. Cryptographic Randomized Response Techniques. In *Public Key Cryptography - PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004*, Vol. 2947. Springer, Berlin, Heidelberg, 425–438.

[3] Apple. 2017. Apple Differential Privacy Technical Overview. https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf

[4] Héber Hwang Arcolezi, Sébastien Gambs, Jean-François Couchot, and Catuscia Palamidessi. 2023. On the Risks of Collecting Multidimensional Data Under Local Differential Privacy. *Proc. VLDB Endow.* 16, 5 (2023), 1126–1139.

[5] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. 2017. Practical Locally Private Heavy Hitters. In *Advances in Neural Information Processing Systems 30: NeurIPS 2017, Long Beach, CA, USA, December 4-9, 2017*. Neural Information Processing Systems Foundation, USA, 2288–2296.

[6] Raef Bassily and Adam D. Smith. 2015. Local, Private, Efficient Protocols for Succinct Histograms. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. ACM, USA, 127–135.

[7] Austin R. Benson, Ravi Kumar, and Andrew Tomkins. 2018. A Discrete Choice Model for Subset Selection. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*. ACM, USA, 37–45.

[8] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning Attacks against Support Vector Machines. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, USA, 1467–1474.

[9] Jonas Böhler and Florian Kerschbaum. 2020. Secure Multi-party Computation of Differentially Private Median. In *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*. USENIX Association, USA, 2147–2164.

[10] Jonas Böhler and Florian Kerschbaum. 2021. Secure Multi-party Computation of Differentially Private Heavy Hitters. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*. ACM, USA, 2361–2377.

[11] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2021. Data Poisoning Attacks to Local Differential Privacy Protocols. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. USENIX Association, USA, 947–964.

[12] Albert Cheu, Adam D. Smith, and Jonathan R. Ullman. 2021. Manipulation Attacks in Local Differential Privacy. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, USA, 883–900.

[13] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *Advances in Neural Information Processing Systems 30: NeurIPS 2017, Long Beach, CA, USA, December 4-9, 2017*. Neural Information Processing Systems Foundation, USA, 3571–3580.

[14] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, Vol. 3876. Springer, Germany, 265–284.

[15] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2016. Calibrating Noise to Sensitivity in Private Data Analysis. *J. Priv. Confidentiality* 7, 3 (2016), 17–51.

[16] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3-4 (2014), 211–407.

[17] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. What Can We Learn Privately: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*. ACM, USA, 1054–1067.

[18] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2020. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*. USENIX Association, USA, 1605–1622.

[19] Minghong Fang, Minghao Sun, Qi Li, Neil Zhenqiang Gong, Jin Tian, and Jia Liu. 2021. Data Poisoning Attacks and Defenses to Crowdsourcing Systems. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. ACM / IW3C2, USA, 969–980.

[20] Justin Hsu, Sanjeev Khanna, and Aaron Roth. 2012. Distributed Private Heavy Hitters. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, Vol. 7391. Springer, Germany, 461–472.

[21] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, San Francisco, CA, USA, 21-23 May 2018*. IEEE Computer Society, USA, 19–35.

[22] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2008. What Can We Learn Privately?. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, Philadelphia, PA, USA, October 25-28, 2008*. IEEE Computer Society, USA, 531–540.

[23] Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. 2021. Preventing Manipulation Attack in Local Differential Privacy Using Verifiable Randomization Mechanism. In *Data and Applications Security and Privacy XXXV - 35th Annual IFIP WG 11.3 Conference, DBSec 2021, Calgary, Canada, July 19-20, 2021*, Vol. 12840. Springer, Berlin, Heidelberg, 43–60.

[24] KDD Cup 2000. 2000. BMS-POS: Online retailer website clickstream analysis. https://www.kdd.org/kdd-cup/view/kdd-cup-2000

[25] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data Poisoning Attacks on Factorization-Based Collaborative Filtering. In *Annual Conference on Neural Information Processing Systems 2016, Barcelona, Spain, December 5-10, 2016*. Neural Information Processing Systems Foundation, USA, 1885–1893.

[26] Junhui Li, Wensheng Gan, Yijie Gui, Yongdong Wu, and Philip S. Yu. 2022. Frequent Itemset Mining with Local Differential Privacy. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*. ACM, USA, 1146–1155.

[27] Xiaoguang Li, Ninghui Li, Wenhai Sun, Neil Zhenqiang Gong, and Hui Li. 2023. Fine-grained poisoning attack to local differential privacy protocols for mean and variance estimation. In *Proceedings of the 32nd USENIX Conference on Security Symposium, Anaheim, CA, USA*. USENIX Association, USA, Article 98, 18 pages.

[28] Ruixuan Liu, Yang Cao, Masatoshi Yoshikawa, and Hong Chen. 2020. FedSel: Federated SGD Under Local Differential Privacy with Top-k Dimension Selection. In *Database Systems for Advanced Applications - 25th International Conference, DASFAA 2020, Jeju, South Korea, September 24-27, 2020, Proceedings, Part I*, Vol. 12112. Springer, Germany, 485–501.

[29] Vasyl Pihur, Aleksandra Korolova, Frederick Liu, Subhash Sankuratripati, Moti Yung, Dachuan Huang, and Ruogu Zeng. 2022. Differentially-Private "Draw and Discard" Machine Learning: Training Distributed Model from Enormous Crowds. In *Cyber Security, Cryptology, and Machine Learning - 6th International Symposium, CSCML 2022, Be'er Sheva, Israel, June 30 - July 1, 2022, Proceedings (Lecture Notes in Computer Science, Vol. 13301)*. Springer, Cham, 468–486.

[30] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. 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, Vienna, Austria, October 24-28, 2016*. ACM, USA, 192–203.

[31] Shaorui Song, Lei Xu, and Liehuang Zhu. 2023. Efficient Defenses Against Output Poisoning Attacks on Local Differential Privacy. *IEEE Trans. Inf. Forensics Secur.* 18 (2023), 5506–5521.

[32] Lichao Sun, Jianwei Qian, and Xun Chen. 2021. LDP-FL: Practical Private Aggregation in Federated Learning with Local Differential Privacy. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, Virtual Event / Montreal, Canada, IJCAI 2021, 19-27 August 2021*. ijcai.org, USA, 1571–1578.

[33] Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. 2023. A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning. *ACM Comput. Surv.* 55, 8 (2023), 166:1–166:35.

[34] Wei Tong, Haoyu Chen, Jiacheng Niu, and Sheng Zhong. 2024. Data Poisoning Attacks to Locally Differentially Private Frequent Itemset Mining Protocols. https://drive.google.com/file/d/1gzp00t1fMjmx9ShuxlCxhPgCDNOa2Jmh/view?usp=drive_link. [Online; accessed 28-June-2024].

[35] Shaowei Wang, Liusheng Huang, Yiwen Nie, Pengzhan Wang, Hongli Xu, and Wei Yang. 2018. PrivSet: Set-Valued Data Analyses with Locale Differential Privacy. In *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*. IEEE, USA, 1088–1096.

[36] Shaowei Wang, Yuqiu Qian, Jiachun Du, Wei Yang, Liusheng Huang, and Hongli Xu. 2020. Set-valued Data Publication with Local Privacy: Tight Error Bounds and Efficient Mechanisms. *Proc. VLDB Endow.* 13, 8 (2020), 1234–1247.

[37] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally Differentially Private Protocols for Frequency Estimation. In *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*. USENIX Association, USA, 729–745.

[38] Tianhao Wang, Ninghui Li, and Somesh Jha. 2018. Locally Differentially Private Frequent Itemset Mining. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, San Francisco, CA,USA, 21-23 May 2018*. IEEE Computer Society, USA, 127–143.

[39] Tianhao Wang, Ninghui Li, and Somesh Jha. 2021. Locally Differentially Private Heavy Hitter Identification. *IEEE Trans. Dependable Secur. Comput.* 18, 2 (2021), 982–993.

[40] Tianhao Wang, Milan Lopuhaä-Zwakenberg, Zitao Li, Boris Skoric, and Ninghui Li. 2020. Locally Differentially Private Frequency Estimation with Consistency. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, CA, USA, February 23-26, 2020*. The Internet Society, USA, 16 pages.

[41] Yongji Wu, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2022. Poisoning Attacks to Local Differential Privacy Protocols for Key-Value Data. In *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*. USENIX Association, USA, 519–536.

[42] Qingqing Ye, Haibo Hu, Xiaofeng Meng, and Huadi Zheng. 2019. PrivKV: Key-Value Data Collection with Local Differential Privacy. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, USA, 317–331.

[43] Youwen Zhu, Yiran Cao, Qiao Xue, Qihui Wu, and Yushu Zhang. 2024. Heavy Hitter Identification Over Large-Domain Set-Valued Data With Local Differential Privacy. *IEEE Trans. Inf. Forensics Secur.* 19 (2024), 414–426.