Poisoning Attacks to Local Differential Privacy Protocols for Key-Value Data

Yongji Wu, Xiaoyu Cao, Jinyuan Jia, Neil Zhenqiang Gong

Duke University

{yongji.wu769, xiaoyu.cao, jinyuan.jia, neil.gong}@duke.edu

Abstract

Local Differential Privacy (LDP) protocols enable an untrusted server to perform privacy-preserving, federated data analytics. Various LDP protocols have been developed for different types of data such as categorical data, numerical data, and key-value data. Due to their distributed settings, LDP protocols are fundamentally vulnerable to *poisoning attacks*, in which fake users manipulate the server's analytics results via sending carefully crafted data to the server. However, existing poisoning attacks focused on LDP protocols for simple data types such as categorical and numerical data, leaving the security of LDP protocols for more advanced data types such as key-value data unexplored.

In this work, we aim to bridge the gap by introducing novel poisoning attacks to LDP protocols for key-value data. In such a LDP protocol, a server aims to simultaneously estimate the frequency and mean value of each key among some users, each of whom possesses a set of key-value pairs. Our poisoning attacks aim to simultaneously maximize the frequencies and mean values of some attacker-chosen target keys via sending carefully crafted data from some fake users to the sever. Specifically, since our attacks have two objectives, we formulate them as a two-objective optimization problem. Moreover, we propose a method to approximately solve the two-objective optimization problem, from which we obtain the optimal crafted data the fake users should send to the server. We demonstrate the effectiveness of our attacks to three LDP protocols for key-value data both theoretically and empirically. We also explore two defenses against our attacks, which are effective in some scenarios but have limited effectiveness in other scenarios. Our results highlight the needs for new defenses against our poisoning attacks.

1 Introduction

Nowadays, many Internet services rely on users' data. However, it poses significant challenges to users' privacy for a server to collect raw data from users. *Local Differential Privacy (LDP)* [16] aims to address the challenges. Specifically,

LDP is a variant of differential privacy [15] under a local setting, where each user locally perturbs his/her data before sending it to an untrusted server. The server aggregates the perturbed data and obtains the statistics of interest. LDP ensures that even if the server is compromised, users' privacy is still well-protected. Due to its promising resilience against untrusted server, LDP has been widely deployed by Internet giants such as Google [16], Apple [40], and Microsoft [13].

Moreover, LDP protocols have been proposed for different types of data, such as categorical data [16, 30, 39, 45–47], numerical data [13, 14], multidimensional data [44, 52], and key-value data [24, 49]. For instance, in recommender systems, each user rates a set of items (e.g., products), where an item and a rating can be viewed as a key and a value, respectively. Thus, each user possesses a set of key-value pairs. In current recommender systems, users send their raw key-value pairs to the server. However, given access to users' raw key-value pairs, an untrusted server can infer users' sensitive attributes (e.g., gender, age, sexual orientation) via attribute inference attacks [22, 32]. LDP protocols enable a server to collect frequency (i.e., popularity) and mean value (i.e., mean rating) of each key from users without accessing their raw key-value pairs and thus protect users' rating-behavior privacy. The collected frequencies and mean values can be used to rank keys and make recommendations to users.

However, due to the distributed settings, LDP protocols are vulnerable to *poisoning attacks* [7, 11], in which an attacker injects fake users into the system and manipulates the server's analytics results via sending carefully crafted data from the fake users to the server. Specifically, Cheu et al. [11] showed that poisoning attacks can degrade the overall performance for indiscriminate items, while Cao et al. [7] showed that poisoning attacks can promote attacker-chosen target items in LDP protocols for frequency estimation and heavy hitter identification. However, these studies focused on simple data types such as categorical data and numerical data, in which each user possesses a single categorical item or numerical value. The security of LDP protocols for more advanced data types such as key-value data is largely unexplored.

In this work, we aim to bridge this gap. Specifically, we perform a systematic study on poisoning attacks to LDP protocols for key-value data. In our poisoning attacks, an attacker aims to simultaneously promote the estimated frequencies and mean values for some attacker-chosen *target keys*. An attacker can inject some fake users into the system and send carefully crafted data to the server to achieve the attack goals. Our attacks pose severe security threats to LDP protocols for key-value data. For example, when such a LDP protocol is deployed to collect popularity and mean ratings of mobile apps in a mobile-app recommender system, an attacker can use our attacks to promote a malicious app's popularity and mean rating such that it may be recommended to more people.

However, different from the poisoning attacks to LDP protocols for simple data types [7, 11], poisoning attacks to the LDP protocols for key-value data face new challenges. Specifically, key-value data are inherently heterogeneous, i.e., keys are categorical and values are numerical. Moreover, there are correlations between the keys and the values. In particular, the estimated mean value of a key depends on the estimated frequency of the key. Furthermore, each user may possess more than one key-value pair, while each user only has a single item or numerical value in LDP protocols for categorical and numerical data. Therefore, existing poisoning attacks are insufficient for LDP protocols for key-value data.

To address the challenges, we formulate our poisoning attacks as a *two-objective optimization problem*, which explicitly captures the attacker's two objectives on promoting both the estimated frequencies and mean values of the target keys. Specifically, we define the *frequency gain* (or *mean gain*) as the difference between the total estimated frequency (or mean value) of the target keys before and after attack. The expected frequency gain and expected mean gain are the two objective functions in our two-objective optimization problem, where the expectation is taken over the randomness in a LDP protocol. Moreover, we propose a method, called *maximal gain attack (M2GA)*, to approximately solve the two-objective optimization problem. The solution corresponds to the crafted data fake users should send to the server. Specifically, M2GA can exactly maximize the expected frequency gain and approximately maximize the expected mean gain.

To demonstrate the effectiveness of M2GA, we also propose two baseline poisoning attacks, called *random message attack (RMA)* and *random key-value pair attack (RKVA)*. In RMA, each fake user sends a random message in the domain allowed by the LDP protocol to the server, while in RKVA, each fake user picks a random target key, associates the largest allowable value with it, and perturbs the key-value pair following the LDP protocol before sending it to the server.

We apply our attacks to three state-of-the-art LDP protocols for key-value data, e.g., PrivKVM [49], PCKV-UE [24], and PCKV-GRR [24]. Moreover, we evaluate our attacks both theoretically and empirically. Theoretically, we derive the expected frequency gains of our attacks exactly. However, it

is challenging to derive the expected mean gains exactly because they involve divisions of random variables. To address the challenge, we derive the expected mean gains approximately via relaxing the divisions of random variables. We note that prior work [7, 11] found security-privacy trade-offs in LDP protocols for categorical and numerical data, i.e., such a LDP protocol is more vulnerable to poisoning attacks when it is more privacy-preserving. One interesting finding from our theoretical analysis is that, such security-privacy trade-off does not necessarily hold in LDP protocols for key-value data. For instance, in M2GA to PrivKVM [49], the expected frequency gain increases (i.e., more vulnerable to M2GA) as the privacy budget decreases (i.e., more privacy-preserving) when an attacker selects one target key; the expected frequency gain does not depend on the privacy budget when an attacker selects two target keys; and the expected frequency gain decreases as the privacy budget decreases when an attacker selects more than two target keys. Empirically, we evaluate our attacks on multiple datasets. Our results show that M2GA can successfully promote the estimated frequencies and mean values of the target keys, and that M2GA substantially outperforms the two baseline attacks.

We also explore two defenses against our poisoning attacks. Specifically, in one defense, the server uses one-class classifier to detect fake users via treating users' data sent to the server as their features. PrivKVM requires multiple communication rounds between the users and the server. Therefore, in our second defense, the server detects fake users in PrivKVM via checking the consistency of their data sent to the server in multiple rounds. Our intuition is that a fake user sends highly correlated data to the server in multiple rounds, while a genuine user does not. Our empirical results show that our defenses are effective in some scenarios. For instance, when the fraction of fake users and the number of target keys are small, M2GA achieves negligible frequency gains and mean gains when the second defense is deployed. However, the defenses are ineffective in other scenarios, e.g., when the fraction of fake users or the number of target keys is large for the second defense, which highlights the needs for new defense mechanisms against our attacks.

Our contributions can be summarized as follows:

- To the best of our knowledge, we are the first to study poisoning attacks to LDP protocols for key-value data.

- We formulate our attacks as a two-objective optimization problem, which aims to maximize both the expected frequency gain and expected mean gain of the target keys.

- We evaluate our attacks on three state-of-the-art LDP protocols for key-value data both theoretically and empirically.

- We investigate two defenses against our attacks. Our results show that the defenses can defend against our attacks in some scenarios but not in others, which highlights that new defenses are needed to mitigate our attacks.

2 Related Work

Poisoning Attacks to LDP   Two concurrent studies [7, 11] proposed poisoning attacks to LDP protocols for categorical and numerical data. In these LDP protocols, each user holds a single item or numerical value, and a server aims to estimate the frequencies of items or identify heavy hitters that have the largest item frequencies. Cheu et al. [11] showed that an attacker can downgrade the accuracy of the estimated item frequencies or the identified heavy hitters for indiscriminate items via injecting fake users into the system. Cao et al. [7] showed that an attacker can increase the estimated frequencies for attacker-chosen target items or promote them to be identified as heavy hitters. In particular, Cao et al. formulated their poisoning attacks as a single-objective optimization problem, where the objective function is to maximize the frequency gains for the target items. As we discussed in Introduction, these poisoning attacks are insufficient for LDP protocols for key-value data.

In particular, our work differs from [7] in the following aspects. First, we formulate a two-objective optimization problem for key-value data instead of the single-objective one. Second, our solutions to the optimization problems are different. Third, we propose different defenses against the poisoning attacks. Fourth, we observe different privacy-security trade-off. Specifically, Cao et al. [7] found that when the privacy guarantee is stronger, a protocol becomes less secure to poisoning attacks. We do not necessarily observe such privacy-security trade-off both theoretically (in some cases) and empirically for LDP protocols for key-value data.

Poisoning Attacks to ML   Poisoning attacks to machine learning systems have been studied extensively [6, 8, 10, 17–19, 23, 26, 28, 29, 34, 36–38, 48]. In these attacks, an attacker manipulates the training phase of a machine learning system via poisoning some carefully selected training examples or tampering the training process. For instance, training-data poisoning attacks have been studied for support vector machines [6], neural networks [10, 23, 36], and recommender systems [18, 19, 27, 34, 48]. Training-process poisoning attacks have been studied for federated learning [4, 5, 17]. Our poisoning attacks differ from these ones because the computational process of LDP protocols is significantly different from that of machine learning training phases.

3 Preliminaries

Before we dive into details, we summarize the important notations we use in Table 1.

3.1 LDP Protocols for Key-Value Data

Suppose we have n users, we have a dictionary $\mathcal{K}$ of d keys (i.e., $\mathcal{K} = \{1, 2, \cdots, d\}$), and each user possesses a set of KV

symbol	representation
n	# genuine users
m	# fake users
β	fraction of fake users
$\mathcal{K}$	dictionary of keys
d	# keys
$\langle k, v \rangle$	key-value pair
f_k	frequency of k
m_k	mean value of k
ε	privacy budget
ℓ	padding length
r	# target keys
G_f	frequency gain
G_m	mean gain

Table 1: Notations used in this work.

pairs $\langle k, v \rangle$, where $k \in \mathcal{K}$ and $v \in [-1, 1]$. Note that, without loss of generality, we assume the values are transformed into the range $[-1, 1]$. A server aims to estimate the frequency and mean value of each key among the n users. The frequency of a key is the fraction of users who possess the key, while the mean value of a key is the average of the values in the KV pairs that contain the key. Formally, the true frequency f_k and mean value m_k for each key k are defined as follows:

$$f_k = \frac{\sum_{u=1}^{n} \mathbb{I}_{S_u}(\langle k, \cdot \rangle)}{n}, \quad m_k = \frac{\sum_{u \in \{1, \cdots, n\}, \langle k, v \rangle \in S_u} v}{n \cdot f_k},$$

where S_u is the set of KV pairs possessed by user u and $\mathbb{I}_{S_u}(\langle k, \cdot \rangle)$ is an indicator function that equals 1 if one KV pair in S_u contains the key k and equals 0 otherwise.

Framework of LDP Protocols for Key-Value Data   In LDP protocols, each user randomly perturbs its KV pairs and sends the perturbed data (called *message*) to the server. Roughly speaking, in LDP, any two sets of KV pairs are perturbed to the same message with close probabilities. State-of-the-art LDP protocols [24, 49] for key-value data consist of the following three key steps.

- **Sample:** A user randomly samples a key from the dictionary and constructs a KV pair based on the sampled key.

- **Perturb:** The user perturbs the constructed KV pair to obtain the message that should be sent to the server.

- **Aggregate:** The server estimates the frequency and mean value of each key via aggregating the messages from all users. We denote by $\hat{f}_k$ and $\hat{m}_k$ the estimated frequency and mean value of a key k.

Next, we briefly review three state-of-the-art LDP protocols for key-value data, i.e., PrivKVM [49], PCKV-UE [24], and PCKV-GRR [24].

3.2 PrivKVM

PrivKVM utilizes an iterative procedure, where the aforementioned three steps are performed for N_{iter} rounds. Specifically, after each round, the server has an estimated mean value $\hat{m}_k$ for each key k, which is used to construct messages for the users who do not possess the key k in the next round. Next, we describe the three steps in each round.

Sample For each user, PrivKVM samples a key k from the dictionary uniformly at random. If the user possesses k, then the Sample step returns the user's KV pair $\langle k, v \rangle$, otherwise the Sample step constructs a KV pair $\langle k, v = \hat{m}_k \rangle$ ($\hat{m}_k$ is the estimated mean value in the previous round and is set to 0 in the first round). The value v in the KV pair is then discretized to $v^* = 1$ with a probability of $\frac{1+v}{2}$ and $v^* = -1$ with a probability $\frac{1-v}{2}$. Finally, the Sample step returns a KV pair $\langle k, v^* \rangle$ and a flag indicating whether k is possessed by the user or not.

Perturb First, the user perturbs the discretized value v^* to be v' based on the following rule:

$$v' = \begin{cases} v^* & \text{w.p. } \frac{e^{\varepsilon_2}}{1+e^{\varepsilon_2}} \\ -v^* & \text{w.p. } \frac{1}{1+e^{\varepsilon_2}} \end{cases}, \qquad (1)$$

where w.p. is short for with probability. Then, the user further perturbs the $\langle k, v' \rangle$ pair to be $\langle k_p, v'_p \rangle$. Specifically, if the user possesses the key k, then $\langle k_p, v'_p \rangle$ is obtained based on the following perturbation rule:

$$\langle k_p, v'_p \rangle = \begin{cases} \langle 1, v' \rangle & \text{w.p. } \frac{e^{\varepsilon_1}}{1+e^{\varepsilon_1}} \\ \langle 0, 0 \rangle & \text{w.p. } \frac{1}{1+e^{\varepsilon_1}} \end{cases}. \qquad (2)$$

If the user does not have k, then $\langle k_p, v'_p \rangle$ is obtained as follows:

$$\langle k_p, v'_p \rangle = \begin{cases} \langle 0, 0 \rangle & \text{w.p. } \frac{e^{\varepsilon_1}}{1+e^{\varepsilon_1}} \\ \langle 1, v' \rangle & \text{w.p. } \frac{1}{1+e^{\varepsilon_1}} \end{cases}. \qquad (3)$$

Finally, the user sends the pair $\langle k_p, v'_p \rangle$ and the index of key k to the server.

Aggregate We denote by n_k the number of users reporting the index of key k and the tuple $\langle 1, \cdot \rangle$. Then, the server computes the estimated frequency of k as follows:

$$\hat{f}_k = \frac{p - 1 + n_k/n}{2p - 1}, \qquad (4)$$

where $p = \frac{e^{\varepsilon_1}}{e^{\varepsilon_1}+1}$. Then, the server counts the number of users n_1^k (or n_{-1}^k) that report the index of key k and the tuple $\langle 1, 1 \rangle$ (or $\langle 1, -1 \rangle$). The server computes the estimated mean value of k as follows:

$$\hat{m}_k = \frac{\hat{n}_1^k - \hat{n}_{-1}^k}{n_k}, \qquad (5)$$

where $\hat{n}_1^k$ and $\hat{n}_{-1}^k$ are defined as follows:

$$\hat{n}_1^k = \frac{p-1}{2p-1} \cdot n_k + \frac{n_1^k}{2p-1}, \qquad (6)$$

$$\hat{n}_{-1}^k = \frac{p-1}{2p-1} \cdot n_k + \frac{n_{-1}^k}{2p-1}, \qquad (7)$$

where $p = \frac{e^{\varepsilon_2}}{e^{\varepsilon_2}+1}$. We note that frequency estimation is only conducted in the first round, while mean estimation uses the results after N_{iter} rounds. The privacy budget ε_1 is only allocated to the first round, while the privacy budget ε_2 is equally allocated for each round. Specifically, we have $\varepsilon_1 = \frac{\varepsilon}{2}$ and $\varepsilon_2 = \frac{\varepsilon}{2N_{\text{iter}}}$, where ε is the overall privacy budget.

3.3 PCKV-UE and PCKV-GRR

PCKV-UE and PCKV-GRR are two protocols from the PCKV family [24]. PCKV improves PrivKVM by utilizing a *padding-and-sampling* strategy in the Sample step to reduce the variance of frequency and mean value estimation. Moreover, unlike PrivKVM that performs aforementioned three steps for multiple rounds, PCKV only requires a single round. The two protocols PCKV-UE and PCKV-GRR mainly differ in the Perturb step and the Aggregate step, while sharing a common Sample step. Specifically, we have the following workflow:

Sample Suppose a user u has a set of KV pairs S_u. If $|S_u| < \ell$, where ℓ is called *padding length* and is a parameter of the protocols, then the user u pads the set S_u with dummy KV pairs $\{\langle d+1, 0 \rangle, \langle d+2, 0 \rangle, \ldots, \langle d+l-|S_u|, 0 \rangle\}$.

Note that the maximum number of dummy KV pairs is ℓ when S_u is an empty set. After the padding, a random KV pair $\langle k, v \rangle$ is drawn from the padded set. The value v is then discretized in the same way as PrivKVM, i.e., the value v is discretized to $v^* = 1$ with a probability of $\frac{1+v}{2}$ and $v^* = -1$ with a probability $\frac{1-v}{2}$.

Perturb We denote $d' = d + l$ and $\mathcal{K}' = \{1, 2, \cdots, d+l\}$ (the dictionary with dummy keys). The Perturb steps for PCKV-UE and PKCV-GRR are as follows:

- **PCKV-UE:** PCKV-UE leverages Unary Encoding (UE) to perturb KV pairs. In particular, a perturbed vector $\mathbf{y} \in \{1, -1, 0\}^{d'}$ is sent to the server, where $y[i]$ contains value information of key i and is obtained as follows:

$$\mathbf{y}[k] = \begin{cases} v^*, & \text{w.p.} & a \cdot p \\ -v^*, & \text{w.p.} & a \cdot (1-p) \\ 0, & \text{w.p.} & 1 - a \end{cases}, \qquad (8)$$

$$\mathbf{y}[i] = \begin{cases} 1, & \text{w.p.} & b/2 \\ -1, & \text{w.p.} & b/2 \\ 0, & \text{w.p.} & 1 - b \end{cases}, i \in \mathcal{K}' \setminus \{k\}, \qquad (9)$$

where a, b, and p are as follows:

$$a = \frac{1}{2}, b = \frac{2}{e^{\varepsilon} + 3}, p = e^{\varepsilon}/(e^{\varepsilon} + 1). \qquad (10)$$

- **PCKV-GRR:** PCKV-GRR leverages Generalized Random Response (GRR) to perturb KV pairs. Specifically, the KV pair $\langle k, v^* \rangle$ is randomly perturbed into $\langle k', v' \rangle$ as follows:

$$\langle k', v' \rangle = \begin{cases} \langle k, v^* \rangle, & \text{w.p.} \quad a \cdot p \\ \langle k, -v^* \rangle, & \text{w.p.} \quad a \cdot (1-p) \\ \langle i, 1 \rangle, & \text{w.p.} \quad b \cdot 0.5 \\ \langle i, -1 \rangle, & \text{w.p.} \quad b \cdot 0.5 \end{cases}, \qquad (11)$$

where $i \in \mathcal{K}' \setminus \{k\}$ and a, b, and p are as follows:

$$a = \frac{\ell(e^{\varepsilon} - 1) + 2}{\ell(e^{\varepsilon} - 1) + 2d'}, b = \frac{1-a}{d'-1}, p = \frac{\ell(e^{\varepsilon} - 1) + 1}{\ell(e^{\varepsilon} - 1) + 2}. \qquad (12)$$

The perturbed KV pair $\langle k', v' \rangle$ is sent to the server.

Aggregate Due to the difference in Perturb step, the Aggregate steps for PCKV-UE and PCKV-GRR are also different. Given a key k, we respectively use n_1^k and n_{-1}^k to denote the number of users that support the KV pairs $\langle k, 1 \rangle$ and $\langle k, -1 \rangle$. In particular, they can be computed as follows:

- **PCKV-UE**: Recall that, in PCKV-UE, $\mathbf{y}[k]$ contains the value information of the key k. We say $\mathbf{y}[k]$ supports $\langle k, 1 \rangle$ (or $\langle k, -1 \rangle$) if $\mathbf{y}[k] = 1$ (or $\mathbf{y}[k] = -1$). Then, we can compute n_1^k (or n_{-1}^k) as the number of users whose perturbed vectors satisfy $\mathbf{y}[k] = 1$ (or $\mathbf{y}[k] = -1$).

- **PCKV-GRR**: In PCKV-GRR, each user sends a single perturbed KV pair $\langle k', v' \rangle$ to the server. Similar to PCKV-UE, we say $\langle k', v' \rangle$ supports $\langle k, 1 \rangle$ (or $\langle k, -1 \rangle$) if $k' = k$ and $v' = 1$ (or $v' = -1$). Then, we can compute n_1^k (or n_{-1}^k) as the number of users whose perturbed KV pairs satisfy $k' = k$ and $v' = 1$ (or $v' = -1$).

Given n_1^k and n_{-1}^k, the server can estimate the frequency of key k as follows:

$$\hat{f}_k = \frac{\left(n_1^k + n_{-1}^k\right)/n - b}{a - b} \cdot \ell, \qquad (13)$$

The estimated mean value of the key k is computed as follows:

$$\hat{m}_k = \ell \left(\hat{n}_1^k - \hat{n}_{-1}^k \right) / \left(n\hat{f}_k \right), \qquad (14)$$

where

$$\begin{bmatrix} \hat{n}_1^k \\ \hat{n}_{-1}^k \end{bmatrix} = A^{-1} \begin{bmatrix} n_1^k - nb/2 \\ n_{-1}^k - nb/2 \end{bmatrix}, \qquad (15)$$

$$A = \begin{bmatrix} ap - \frac{b}{2} & a(1-p) - \frac{b}{2} \\ a(1-p) - \frac{b}{2} & ap - \frac{b}{2} \end{bmatrix}. \qquad (16)$$

We note that in all the three LDP protocols, the server can clip the estimated frequency $\hat{f}_k$ to be $\frac{1}{n}$ if it is smaller than $\frac{1}{n}$ and to be 1 if it is larger than 1. Moreover, the server can clip the support counts $\hat{n}_1^k$ and $\hat{n}_{-1}^k$ into the range of $[0, \frac{n\hat{f}_k}{\ell}]$ in PCKV-UE and PCKV-GRR, as well as the range of $[0, n_k]$ in PrivKVM, before using them to estimate the mean value.

4 Threat Model

Attacker's capability and background knowledge We assume that the attacker is able to inject some fake users into the system. Previous measurement study [41] has shown that an attacker can easily obtain a large number of fake/compromised users in online web services such as Twitter and Facebook. Specifically, we assume that the attacker has access to m fake users. Together with the n genuine users, the server estimates frequencies and mean values of keys among the $n + m$ users. For each fake user, the attacker can arbitrarily craft its message sent to the server. An attacker has access to the parameters of the LDP protocol since the LDP protocol is executed on a user side. Specifically, an attacker has access to the dictionary of keys, as well as the implementation details of the Sample and Perturb steps of the LDP protocol.

Attacker's goal An attacker aims to promote some target keys. We assume r target keys and denote them as a set $\mathbb{T} = \{k_1, k_2, \cdots, k_r\}$. The attacker aims to increase the estimated frequencies and mean values of the target keys via sending carefully crafted messages from the fake users to the server. Without loss of generality, we assume the m fake users have IDs $n+1, n+2, \cdots, n+m$. We denote the set of messages the fake users send to the server as $\mathbb{Y} = \{y_i\}_{i=n+1}^{n+m}$, where y_i is the message fake user i sends to the server. We denote by $\hat{f}_k$ and $\tilde{f}_k$ the estimated frequency of key k among the n genuine users and all the $n + m$ users, respectively. Moreover, we denote by $G_f(\mathbb{Y}) = \sum_{k \in \mathbb{T}} \mathbb{E}[\Delta \hat{f}_k]$ the *frequency gain* of the target keys, where $\Delta \hat{f}_k = \tilde{f}_k - \hat{f}_k$ and the expectation is taken over the randomness in a LDP protocol.

Similarly, we denote by $\hat{m}_k$ and $\tilde{m}_k$ the estimated mean value of key k among the n genuine users and all the $n + m$ users, respectively. Furthermore, we denote by $G_m(\mathbb{Y}) = \sum_{k \in \mathbb{T}} \mathbb{E}[\Delta \hat{m}_k]$ the *mean gain* of the target keys, where $\Delta \hat{m}_k = \tilde{m}_k - \hat{m}_k$ and the expectation is taken over the randomness in a LDP protocol. An attacker aims to simultaneously maximize the frequency gain and mean gain via carefully crafting the messages $\mathbb{Y}$. We propose to formulate such an attack goal as the following two-objective optimization problem:

$$\max_{\mathbb{Y}} \begin{bmatrix} G_f(\mathbb{Y}) \\ G_m(\mathbb{Y}) \end{bmatrix}. \qquad (17)$$

Note that we consider the target keys are weighted equally for simplicity. However, our formulation can be extended

to the scenario where the attacker assigns different weights to different target keys in the frequency and mean gains. A method to solve the two-objective optimization problem is a poisoning attack to a LDP protocol for key-value data.

5 Our Attacks

We first introduce our three attacks and then apply them to PrivKVM, PCKV-UE, and PCKV-GRR.

5.1 Three Attacks

We propose *Maximal Gain Attack (M2GA)*, which solves the two-objective optimization problem to construct the optimal messages the fake users should send to the server. To show the effectiveness of M2GA, we also propose two baseline attacks: *Random Message Attack (RMA)* and *Random Key-Value Pair Attack (RKVA)*. Next, we describe them one by one.

5.1.1 M2GA

Our idea is to unify the frequency gain and mean gain for different LDP protocols under the same framework, based on which we transform the two-objective optimization problem to be one that is easier to solve.

Unifying the frequency gain We first observe that the estimated frequency $\hat{f}_k$ can be unified as Eq. 13. In particular, as discussed in Section 3.3, PCKV-UE and PCKV-GRR use Eq. 13 to calculate $\hat{f}_k$. We can also use Eq. 13 to calculate $\hat{f}_k$ in PrivKVM, where the parameters a, b, and l are set as follows:

$$a = \frac{e^{\varepsilon_1}}{e^{\varepsilon_1}+1}, b = \frac{1}{e^{\varepsilon_1}+1}, \ell = 1. \tag{18}$$

Therefore, we can represent the frequency gain $G_f(\mathbb{Y})$ as:

$$G_f(\mathbb{Y}) = \sum_{k \in \mathbb{T}} \mathbb{E}\left[\tilde{f}_k - \hat{f}_k\right]$$

$$= \sum_{k \in \mathbb{T}} \ell \left\{ \mathbb{E}\left[\frac{(n_1^k + n_{-1}^k + \tilde{n}_1^k + \tilde{n}_{-1}^k)/(n+m) - b}{a - b}\right] - \mathbb{E}\left[\frac{(n_1^k + n_{-1}^k)/n - b}{a - b}\right] \right\},$$

where n_1^k and n_{-1}^k respectively are the support counts of $\langle k, 1 \rangle$ and $\langle k, -1 \rangle$ among the n genuine users, while $\tilde{n}_1^k$ and $\tilde{n}_{-1}^k$ are the ones among the m fake users. We note that the messages $\mathbb{Y}$ only affect the term $\sum_{k \in \mathbb{T}} \mathbb{E}\left[\frac{(\tilde{n}_1^k + \tilde{n}_{-1}^k)}{(n+m)(a-b)}\right]$ and the denominator $(n+m)(a-b)$ is irrelevant in the optimization for a given setting of LDP protocol. Therefore, optimizing the frequency gain is equivalent to optimizing the following:

$$\max_{\mathbb{Y}} \sum_{k \in \mathbb{T}} (\mathbb{E}[\tilde{n}_1^k] + \mathbb{E}[\tilde{n}_{-1}^k]). \tag{19}$$

Moreover, the frequency gain can be simplified as follows:

$$G_f(\mathbb{Y}) = \frac{\ell}{(n+m)(a-b)} \sum_{k \in \mathbb{T}} (\mathbb{E}[\tilde{n}_1^k] + \mathbb{E}[\tilde{n}_{-1}^k]) - c, \tag{20}$$

where $c = \sum_{k \in \mathbb{T}} \frac{m\ell(n_1^k + n_{-1}^k)}{n(n+m)(a-b)} = \frac{m\ell}{n+m}(f_{\mathbb{T}} + \frac{rb}{a-b})$. $f_{\mathbb{T}} = \sum_{k \in \mathbb{T}} f_k$ is the sum of the true frequencies of all target keys, which is a constant.

Unifying the mean gain Similar to frequency estimation, the estimated mean value can also be unified in the following equation:

$$\hat{m}_k = \frac{\left(n_1^k - n_{-1}^k\right)(a-b)}{a(2p-1)\left(n_1^k + n_{-1}^k - nb\right)}, \tag{21}$$

where the parameters a, b, p, and l are described in Section 3.3 for PCKV-UE and PCKV-GRR, and they are set as follows for PrivKVM:

$$a = 1, b = 0, p = \frac{e^{\varepsilon_2}}{e^{\varepsilon_2}+1}, l = 1. \tag{22}$$

Then, we can represent the mean gain $G_m(\mathbb{Y})$ as follows:

$$G_m(\mathbb{Y}) = \sum_{k \in \mathbb{T}} \mathbb{E}[\tilde{m}_k - \hat{m}_k]$$

$$= \sum_{k \in \mathbb{T}} \left\{ \mathbb{E}\left[\frac{\left(n_1^k - n_{-1}^k + \tilde{n}_1^k - \tilde{n}_{-1}^k\right)(a-b)}{a(2p-1)\left(n_1^k + n_{-1}^k + \tilde{n}_1^k + \tilde{n}_{-1}^k - (n+m)b\right)}\right] - \mathbb{E}\left[\frac{\left(n_1^k - n_{-1}^k\right)(a-b)}{a(2p-1)\left(n_1^k + n_{-1}^k - nb\right)}\right] \right\}. \tag{23}$$

However, unlike the frequency gain, it is non-trivial to compute the two expectations above because they involve divisions between random variables. Specifically, since n_1^k and n_2^k are random variables, both the numerator and the denominator are random variables. To address the challenge, we propose to use the first-order Taylor expansion of functions of random variables [9] to approximately compute G_m. Specifically, given two random variables X and Y, the first-order Taylor expansion means the following:

$$\mathbb{E}\left[\frac{X}{Y}\right] \approx \frac{\mathbb{E}[X]}{\mathbb{E}[Y]}. \tag{24}$$

Note that we have the following:

$$\mathbb{E}[n_1^k - n_{-1}^k] = n\frac{f_k}{\ell}a(2p-1)m_k,$$

$$\mathbb{E}[n_1^k + n_{-1}^k - nb] = n\frac{f_k}{\ell}(a-b),$$

where f_k and m_k are the true frequency and mean value of k. Thus, based on the first-order Taylor expansion, we have:

$$G_m \approx \sum_{k \in \mathbb{T}} \left(\frac{a-b}{a(2p-1)} \frac{nf_k a(2p-1)m_k/\ell + \mathbb{E}[\tilde{n}_1^k] - \mathbb{E}[\tilde{n}_{-1}^k]}{nf_k(a-b)/\ell + \mathbb{E}[\tilde{n}_1^k] + \mathbb{E}[\tilde{n}_{-1}^k] - mb} - m_k \right). \tag{25}$$

For simplicity, we denote $c_1^k = nf_k a(2p-1)m_k/\ell$ and $c_2^k = nf_k(a-b)/\ell - mb$. Then, we approximate optimizing the mean gain as follows:

$$\max_{\mathbb{Y}} \sum_{k\in\mathbb{T}} \frac{\mathbb{E}[\tilde{n}_1^k] - \mathbb{E}[\tilde{n}_{-1}^k] + c_1^k}{\mathbb{E}[\tilde{n}_1^k] + \mathbb{E}[\tilde{n}_{-1}^k] + c_2^k}. \tag{26}$$

Reformulated two-objective optimization problem By combining Eq. 19 and 26, we re-formulate our two-objective optimization problem as follows:

$$\max_{\mathbb{Y}} \begin{bmatrix} \sum_{k\in\mathbb{T}}\left(\mathbb{E}[\tilde{n}_1^k] + \mathbb{E}[\tilde{n}_{-1}^k]\right) \\ \sum_{k\in\mathbb{T}} \frac{\mathbb{E}[\tilde{n}_1^k] - \mathbb{E}[\tilde{n}_{-1}^k] + c_1^k}{\mathbb{E}[\tilde{n}_1^k] + \mathbb{E}[\tilde{n}_{-1}^k] + c_2^k} \end{bmatrix}. \tag{27}$$

5.1.2 RMA

In this baseline attack, each fake user picks a message uniformly at random from the message domain allowed by a LDP protocol and sends it to the server.

5.1.3 RKVA

RMA does not consider any information about the target keys. Different from RMA, RKVA considers the target keys. Specifically, each fake user picks a random target key k, pairs it with an extreme value 1, and the constructed KV pair is viewed as the fake user's KV pair. Then, the constructed KV pair is processed by the LDP protocol and the resulting message is sent to the server.

5.2 Attacking PrivKVM

5.2.1 M2GA

Recall that PrivKVM is an iterative procedure, in which the frequency estimation is performed in the first round while the mean estimation is performed in each round and the estimated mean values in the last round are used. Solving Eq. 27 exactly is non-trivial. Therefore, we propose a two-step approximate solution, which first optimizes the frequency gain and then approximately optimizes the mean gain.

Since each user in PrivKVM sends a key index and a tuple to the server, we craft such message for each fake user such that Eq. 19 is maximized. For PrivKVM, a fake user can only inject a single key to be counted by the server. That is, a fake user can only increase either $\tilde{n}_1^k$ or $\tilde{n}_{-1}^k$ for a single key k. Therefore, for each fake user, we randomly select a target key k, and send the index k and the tuple $\langle 1, \cdot \rangle$ to the server, where the reported value does not influence the frequency gain and we will discuss it for optimizing the mean gain. Thus, we have $\sum_{k\in\mathbb{T}}(\mathbb{E}[\tilde{n}_1^k] + \mathbb{E}[\tilde{n}_{-1}^k]) = m$, and we have the frequency gain as $G_f = \frac{m}{(n+m)(a-b)} - c$, where the parameters a, b are defined in Eq. 18. In practice, for each target key, $\frac{m}{r}$ fake users send messages including the target key to the server.

For mean estimation, we attack each round of PrivKVM. Specifically, in PrivKVM, the value sent to the server is either 1 or -1. Therefore, to increase the estimated mean, a fake user always sends value 1 for a target key. In other words, we have $\mathbb{E}[\tilde{n}_1^k] = \frac{m}{r}$ and $\mathbb{E}[\tilde{n}_{-1}^k] = 0$ for each target key k. The mean gain is as follows: $G_m \approx \sum_{k\in\mathbb{T}} \frac{a-b}{a(2p-1)} \frac{nf_k a(2p-1)m_k/\ell + m/r}{nf_k(a-b)/\ell + m/r - mb} - m_k$, where a, b, p are given in Eq. 22.

To summarize, each fake user sends a random target key and value 1 to the server in each round of PrivKVM.

5.2.2 RMA

In RMA, each fake user randomly chooses a key k from the entire dictionary. Then, the fake user randomly chooses a tuple to report to the server. Specifically, $\langle 0, 0 \rangle$ is chosen with a probability of $\frac{1}{2}$, while $\langle 1, -1 \rangle$ and $\langle 1, 1 \rangle$ are each chosen with probability $\frac{1}{4}$.

Therefore, we have a probability of $\frac{1}{2d}$ that the message of a fake user supports key k, and the KV pairs $\langle k, 1 \rangle$ and $\langle k, -1 \rangle$ would be supported with equal probabilities. Thus, we have $\mathbb{E}[\tilde{n}_1^k] = \mathbb{E}[\tilde{n}_{-1}^k] = \frac{m}{4d}$. By plugging the values into Eq. 20 and Eq. 25, we have the frequency gain of $G_f = \frac{mr}{2(n+m)(a-b)d} - c$, and the mean gain of $G_m \approx \sum_{k\in\mathbb{T}} \frac{a-b}{a(2p-1)} \frac{nf_k a(2p-1)m_k}{nf_k(a-b)+m/(2d)-mb} - m_k$. Again, we note that the parameters a, b, p are different in G_f and G_m.

5.2.3 RKVA

In RKVA, each fake user picks a target key k uniformly at random and the fake user's tuple is $\langle 1, 1 \rangle$. This tuple is perturbed according to the Perturb step of the PrivKVM protocol. The perturbed tuple still supports k with a probability of $\frac{e^{\varepsilon_1}}{e^{\varepsilon_1}+1}$, and the value is inverted with a probability of $\frac{1}{1+e^{\varepsilon_2}}$.

Therefore, we have $\mathbb{E}[\tilde{n}_1^k] = \mathbb{E}[\tilde{n}_{-1}^k] = \frac{me^{\varepsilon_1}}{2r(e^{\varepsilon_1}+1)}$. $\mathbb{E}[\tilde{n}_1^k] = \frac{me^{\varepsilon_1}e^{\varepsilon_2}}{r(e^{\varepsilon_1}+1)(1+e^{\varepsilon_2})}$ and $\mathbb{E}[\tilde{n}_{-1}^k] = \frac{me^{\varepsilon_1}}{r(e^{\varepsilon_1}+1)(1+e^{\varepsilon_2})}$. The frequency gain is $G_f = \frac{me^{\varepsilon_1}}{(n+m)(a-b)(e^{\varepsilon_1}+1)} - c$, and the mean gain is $G_m \approx \sum_{k\in\mathbb{T}} \frac{a-b}{a(2p-1)} \frac{nf_k a(2p-1)m_k + me^{\varepsilon_1}(e^{\varepsilon_2}-1)/r(e^{\varepsilon_1}+1)(1+e^{\varepsilon_2})}{nf_k(a-b)+me^{\varepsilon_1}/(r(e^{\varepsilon_1}+1))-mb} - m_k$.

5.3 Attacking PCKV-UE

5.3.1 M2GA

In PCKV-UE, each user sends a vector of length $d + \ell$ to the server, and each dimension is checked independently on whether it supports the corresponding key. Therefore, a single user could support multiple keys.

For each fake user, we put a 1 or -1 in all the dimensions corresponding to the target keys. Therefore, a single fake user can increase $\tilde{n}_1^k$ or $\tilde{n}_{-1}^k$ for all $k \in \mathbb{T}$. For the remaining dimensions, if we simply leave them as 0, the server may easily detect that these messages are from fake users. To address this issue, we sample some dimensions and set them

to 1 or -1, such that the vectors we craft for the fake users would have the same number of 1 bits and -1 bits as the expectation of the genuine users'. Specifically, if a genuine user samples a KV pair $\langle \cdot, 1 \rangle$ (or $\langle \cdot, -1 \rangle$) to report, the perturbed vector would have $\lfloor ap + (d'-1)(b/2) \rfloor$ 1 (or -1) bits and $a(1-p) + (d'-1)(b/2)$ -1 (or 1) bits on expectation. We note that this form of disguise does not affect the frequency gain and mean gain for the target keys.

Therefore, we have $\tilde{n}_1^k + \tilde{n}_{-1}^k = m$ for each target key k. The frequency gain is $G_f = \frac{mr\ell}{(n+m)(a-b)} - c$. To further maximize the mean gain, we solve the optimization problem of Eq. 26 in the similar way as for PrivKVM. Specifically, we only need to maximize $\tilde{n}_1^k - \tilde{n}_{-1}^k$ for each target key k under the constraints of $\tilde{n}_1^k \geq 0, \tilde{n}_{-1}^k \geq 0$ and $\tilde{n}_1^k + \tilde{n}_{-1}^k = m$. Therefore, we have the following optimal solution $\tilde{n}_1^k = m$ and $\tilde{n}_{-1}^k = 0$. Thus, we obtain the mean gain as: $G_m \approx \sum_{k \in \mathbb{T}} \frac{a-b}{a(2p-1)} \frac{nf_k a(2p-1)m_k/\ell+m}{nf_k(a-b)/\ell+m(1-b)} - m_k$.

To summarize, each fake user sets the dimensions corresponding to the target keys to 1 in its vector. Moreover, to evade possible detection, each fake user randomly samples some other dimensions of its vector and set them to be 1 or -1 such that the vector has the same number of 1 bits and -1 bits as the expected 1 bits and -1 bits in a genuine user's vector.

5.3.2 RMA

In PCKV-UE, a message is a vector. For each fake user, we randomly sample the value of each dimension of the vector. Specifically, each dimension is randomly set to 1, -1, or 0 with an equal probability of $\frac{1}{3}$. Therefore, we have $\mathbb{E}[\tilde{n}_1^k] = \mathbb{E}[\tilde{n}_{-1}^k] = \frac{m}{3}$. The frequency gain is $G_f = \frac{2mr\ell}{3(n+m)(a-b)} - c$. The mean gain is $G_m \approx \sum_{k \in \mathbb{T}} \frac{a-b}{a(2p-1)} \frac{nf_k a(2p-1)m_k/\ell}{nf_k(a-b)/\ell+2m/3-mb} - m_k$.

5.3.3 RKVA

For each fake user, a random target key k is sampled from $\mathbb{T}$ and paired with a value of 1. The constructed KV pair is then perturbed by the PCKV-UE protocol. As the perturbation is independent for each dimension, we only need to focus on the dimensions corresponding to the target keys in $\mathbb{T}$. Similar to PrivKVM, if k is selected, its value remains the same with probability ap and gets inversed to -1 with a probability of $a(1-p)$. However, different from PrivKVM, if a key other than k is selected, the perturbed vector supports k with probability b and the value is 1 or -1 with equal probability $\frac{b}{2}$. Therefore, we have $\mathbb{E}[\tilde{n}_1^k] = \frac{map+m(r-1)b/2}{r}$ and $\mathbb{E}[\tilde{n}_{-1}^k] = \frac{ma(1-p)+m(r-1)b/2}{r}$. The frequency gain and the mean gain are respectively as follows: $G_f = \frac{ma\ell+m(r-1)b\ell}{(n+m)(a-b)} - c$ and $G_m \approx \sum_{k \in \mathbb{T}} \frac{nf_k m_k/\ell+m/r}{nf_k/\ell+m/r} - m_k$.

5.4 Attacking PCKV-GRR

5.4.1 M2GA

In PCKV-GRR, each user sends a KV pair as the message to the server. A user supports a KV pair if and only if the message he/she sent to the server is exactly the KV pair. Therefore, similar to PrivKVM, a fake user can only increase the support count $\tilde{n}_1^k$ or $\tilde{n}_{-1}^k$ of a single target key. Similar to PrivKVM, we have $\sum_{k \in \mathbb{T}} \mathbb{E}[\tilde{n}_1^k + \tilde{n}_{-1}^k] = m$. The frequency gain is $G_f = \frac{m\ell}{(n+m)(a-b)} - c$.

The mean gain for each target key is then maximized in the same way as each round of PrivKVM, where we set $\tilde{n}_1^k = \frac{m}{r}$ and $\tilde{n}_{-1}^k = 0$, i.e., we set all the values in the KV pairs sent to the server to 1. Thus, the mean gain is $G_m \approx \sum_{k \in \mathbb{T}} \frac{a-b}{a(2p-1)} \frac{nf_k a(2p-1)m_k/\ell+m/r}{nf_k(a-b)/\ell+m/r-mb} - m_k$.

To summarize, each fake user sends a random target key and value 1 to the server.

5.4.2 RMA

For each fake user, we randomly select a target key and set its corresponding value as -1 or 1 uniformly at random, which is the KV pair sent to the server. Therefore, we have $\mathbb{E}[\tilde{n}_1^k] = \mathbb{E}[\tilde{n}_{-1}^k] = \frac{m}{2d'}$. We have the frequency gain as $G_f = \frac{mr\ell}{(n+m)(a-b)d'} - c$, while the mean gain as $G_m^k \approx \sum_{k \in \mathbb{T}} \frac{a-b}{a(2p-1)} \frac{nf_k a(2p-1)m_k/\ell}{nf_k(a-b)/\ell+m/d'-mb} - m_k$.

5.4.3 RKVA

For each fake user, we randomly select a target key k from $\mathbb{T}$ and choose its value 1 to construct a KV pair. The KV pair is then perturbed according to the Perturb step of the PCKV-GRR protocol. The KV pair after perturbation still keeps k as its key with a probability of a, and a key other than k gets perturbed to k with a probability of b. Therefore, we get $\mathbb{E}[\tilde{n}_1^k] = \frac{map+m(r-1)b/2}{r}$ and $\mathbb{E}[\tilde{n}_{-1}^k] = \frac{ma(1-p)+m(r-1)b/2}{r}$. We have $G_f = \frac{ma\ell+m(r-1)b\ell}{(n+m)(a-b)} - c$ and $G_m \approx \sum_{k \in \mathbb{T}} \frac{nf_k m_k/\ell+m/r}{nf_k/\ell+m/r} - m_k$.

5.5 Theoretical Analysis

Table 2 summarizes the analytical forms of the frequency gains of our three attacks for the three LDP protocols, while Table 3 summarizes the analytical forms of the approximate mean gains of our three attacks for the three LDP protocols. We have replaced the protocol-dependent parameters a, b, and p in the analytical forms. We note that we do not consider clipping the estimated frequency $\hat{f}_k$ and support counts n_1^k, n_{-1}^k when deriving the analytical forms.

First, we observe that M2GA outperforms RMA and RKVA. This is because M2GA crafts the fake users' messages via solving a two-objective optimization problem. In particular, our two objectives are non-convex. However, M2GA

Table 2: Frequency gains of the three attacks for PrivKVM, PCKV-UE, and PCKV-GRR. $\beta = \frac{m}{n}$ is the fraction of fake users, $f_{\mathbb{T}} = \sum_{k\in\mathbb{T}} f_k$ is the sum of true frequencies of the target keys, ε is the privacy budget, d is the total number of keys, ℓ is the padding length, $d' = d + \ell$ is the padded dictionary size, and r is the number of target keys.

	PrivKVM	PCKV-UE	PCKV-GRR
M2GA	$\frac{\beta}{1+\beta}\left[1 - f_{\mathbb{T}} + \frac{2-r}{e^{\varepsilon/2}-1}\right]$	$\frac{\beta\ell}{1+\beta}\left[2r - f_{\mathbb{T}} + \frac{4r}{e^{\varepsilon}-1}\right]$	$\frac{\beta}{1+\beta}\left[(1-f_{\mathbb{T}})\ell + \frac{2(d'-r)}{e^{\varepsilon}-1}\right]$
RMA	$\frac{\beta}{1+\beta}\left[\frac{(e^{\varepsilon/2}-2d+1)r}{2(e^{\varepsilon/2}-1)d} - f_{\mathbb{T}}\right]$	$\frac{\beta\ell}{1+\beta}\left[\frac{4e^{\varepsilon}r}{3(e^{\varepsilon}-1)} - f_{\mathbb{T}}\right]$	$\frac{\beta(r-f_{\mathbb{T}}d')\ell}{(1+\beta)d'}$
RKVA	$\frac{\beta}{1+\beta}\left[1 - f_{\mathbb{T}} + \frac{1-r}{e^{\varepsilon/2}-1}\right]$	$\frac{\beta\ell}{1+\beta}\left(1 - f_{\mathbb{T}}\right)$	$\frac{\beta\ell}{1+\beta}\left(1 - f_{\mathbb{T}}\right)$

Table 3: Approximate mean gains of the three attacks for PrivKVM, PCKV-UE, and PCKV-GRR. f_k is the true frequency of key k, m_k is the true mean value of k, $\varepsilon_2 = \frac{\varepsilon}{2N_{\text{iter}}}$ is the privacy budget in each round of PrivKVM, and N_{iter} is the number of rounds.

	PrivKVM	PCKV-UE	PCKV-GRR
M2GA	$\sum_{k\in\mathbb{T}}\frac{f_k m_k r + \frac{e^{\varepsilon_2}+1}{e^{\varepsilon_2}-1}\beta}{f_k r + \beta} - m_k$	$\sum_{k\in\mathbb{T}}\frac{2\beta\ell(e^{\varepsilon}+1)+(e^{\varepsilon}-1)f_k m_k}{2\beta\ell(e^{\varepsilon}+1)+(e^{\varepsilon}-1)f_k} - m_k$	$\sum_{k\in\mathbb{T}}\frac{(e^{\varepsilon}-1)(\beta\ell+f_k m_k r)+2\beta d'}{\beta[(e^{\varepsilon}-1)\ell+2(d'-r)]+(e^{\varepsilon}-1)f_k r} - m_k$
RMA	$\sum_{k\in\mathbb{T}}\frac{2f_k m_k d}{2f_k d + \beta} - m_k$	$\sum_{k\in\mathbb{T}}\frac{3(e^{\varepsilon}-1)f_k m_k}{3(e^{\varepsilon}-1)f_k + 4e^{\varepsilon}\beta\ell} - m_k$	$\sum_{k\in\mathbb{T}}\frac{f_k m_k d'}{f_k d' + \beta\ell} - m_k$
RKVA	$\sum_{k\in\mathbb{T}}\frac{f_k m_k r(e^{\varepsilon/2}+1)+e^{\varepsilon/2}\beta}{f_k(e^{\varepsilon/2}+1)+e^{\varepsilon/2}\beta} - m_k$	$\sum_{k\in\mathbb{T}}\frac{f_k m_k r + \beta\ell}{f_k r + \beta\ell} - m_k$	$\sum_{k\in\mathbb{T}}\frac{f_k m_k r + \beta\ell}{f_k r + \beta\ell} - m_k$

achieves the optimal frequency gain for the three LDP protocols in all cases, as we discussed when describing M2GA for each protocol. Moreover, in a given execution of a LDP protocol, M2GA also achieves the optimal mean gain if there is only one target key k (i.e., $r = 1$) and $n_1^k \geq n_{-1}^k > \frac{(n+m)b}{2}$ (Appendix A.1 shows the proof). Second, we observe that the frequency gain of an attack increases as the fraction of fake users increases. However, we do not have this observation for mean gains. We suspect the reason is that the mean gain depends on the estimated frequency and that we approximate the mean gains via Taylor expansion. Third, the frequency gain is larger when the total true frequencies $f_{\mathbb{T}}$ of the target keys is smaller. This is because the frequency gain is the difference between the estimated frequencies before and after attack. Moreover, the approximate mean gain becomes larger when the true mean value of each target key becomes smaller.

Prior work [7, 11] observed trade-off between security against poisoning attacks and privacy in LDP protocols for categorical and numerical data, i.e., such a LDP protocol is more vulnerable to poisoning attacks if it uses a smaller privacy budget. Our fourth observation is that such security-privacy trade-off does not necessarily hold in LDP protocols for key-value data. In particular, while we observe such security-privacy trade-off for the frequency gains of M2GA to PCKV-UE and PCKV-GRR, how the privacy budget ε influences the frequency gain of M2GA to PrivKVM depends on r, the number of target keys. Specifically, the frequency gain of M2GA to PrivKVM increases, does not change, and decreases as the privacy budget ε decreases when $r = 1$, $r = 2$, and $r > 2$, respectively. The *approximate* mean gain of M2GA

Table 4: Dataset statistics. #records indicates the total number of KV pairs in a dataset, while the 90th-percentile refers to that of the number of KV pairs possessed per user.

Dataset	#users	#keys	#records	90th-percentile
Synthetic	100,000	100	100,000	1.0
Clothing	105,508	5,850	192,198	3.0
TalkingData	60,822	320	1,327,468	34.0
MovieLens-1M	943	1,682	100,000	244.4

in Table 3 has such security-privacy trade-off as we can verify that the derivative of the approximate mean gain of M2GA with respect to ε is negative. However, we do not necessarily observe such security-privacy trade-off for the *true* mean gains of M2GA in our experiments. This is because the approximate mean gains are obtained using Taylor expansion of the true ones and the protocols clip frequencies and support counts in practice.

6 Evaluation

6.1 Experimental Setup

6.1.1 Datasets

We evaluate our three attacks, i.e., M2GA, RMA, and RKVA, on a synthetic dataset and three real-world datasets. The statistics of the four datasets are shown in Table 4.

- **Synthetic**: Following [24,49], we create a synthetic dataset to evaluate our attacks. In particular, we generate 10^5 users
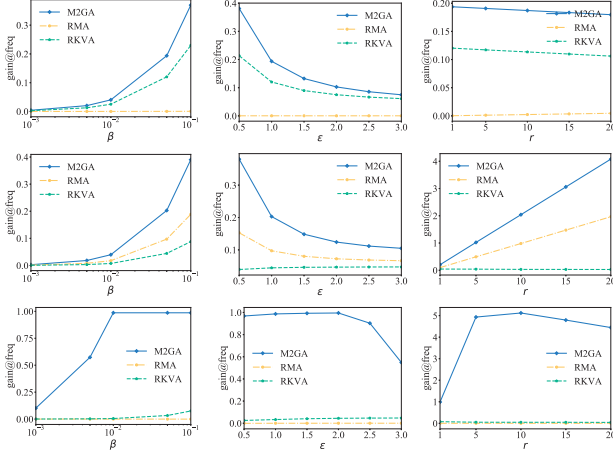
Figure 1: Impact of different parameters (β, ε, r) on the frequency gains on Synthetic. The three rows are for PrivKVM, PCKV-UE, and PCKV-GRR, respectively.

Figure 2: Impact of different parameters (β, ε, r) on the mean gains on Synthetic. The three rows are for PrivKVM, PCKV-UE, and PCKV-GRR, respectively.

and 100 keys. Each user has a single KV pair. The keys and the values follow a zero-mean Gaussian distribution, where the standard deviation is 15 for keys and 1 for values.

- **Clothing** [1]: This is a clothing fit dataset for product size recommendation. It contains users' rating scores for different products. We treat each product as a key and view each rating score as a value. Note that each user may have multiple pairs of ⟨product, rating score⟩.

- **TalkingData** [3]: This dataset contains mobile apps downloaded by users on their mobile devices. In particular, we treat each category of mobile apps as a key and view the number of apps downloaded by a user in a category as a value. A user may have multiple KV pairs.

- **MovieLens-1M** [25]: This dataset contains users' rating scores for different movies. Each movie is a key and each rating score is a value. A user may rate multiple movies.

We scale the values in each dataset such that they fall into the range of $[-1, 1]$.

6.1.2 Evaluation Metrics

gain@freq and gain@mean We use *frequency gain (gain@freq)* and *mean gain (gain@mean)* of a set of target keys as the evaluation metrics. In particular, given a set of target keys $\mathbb{T}$, gain@freq is computed as $\sum_{k \in \mathbb{T}} \mathbb{E}[\Delta \hat{f}_k]$ and gain@mean is computed as $\sum_{k \in \mathbb{T}} \mathbb{E}[\Delta \hat{m}_k]$, where $\Delta \hat{f}_k$ and $\Delta \hat{m}_k$ respectively measure the frequency gain and mean gain for the target key k. Note that frequency gain and mean gain involve expectations. In our experiments, we average the results over 100 trials to compute the expectations. Since in our experiments, we clip the estimated frequencies and support
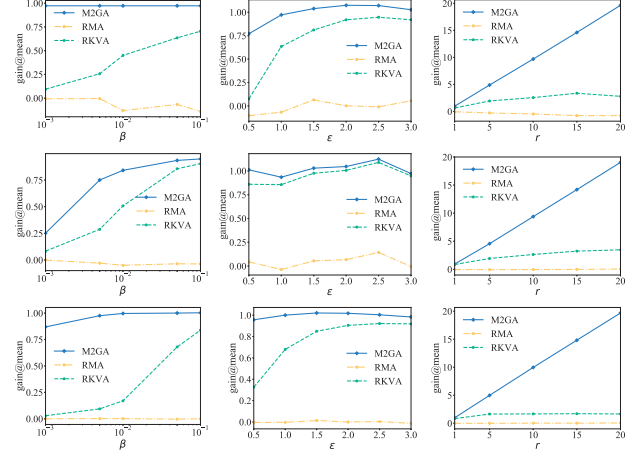
counts in the LDP protocols, the frequency gains may not be the same as those in Table 2.

ASR for recommender systems We also consider recommender system as a downstream application. Specifically, the server first collects the frequency and mean value (i.e., average rating score) of each item/key from users using LDP protocols and then recommends top-t items to all users based on the statistics. In this downstream application, the attacker's goal is to promote the target items/keys to be among the top-t items recommended by the system. Therefore, we use *attack success rate (ASR)* as our metric, which we define as the fraction of target items that are in the t recommended items after attack. Note that the target items are not among the t recommended ones before attack.

We consider three different cases of recommender systems, i.e., frequency-based recommender system (Case 1), score-based recommender system (Case 2), and frequency-score-based recommender system (Case 3). In Case 1, the recommender system recommends the most popular t items, i.e., the t items with the largest estimated frequencies. Ties are broken by selecting the item with higher estimated average rating score. In Case 2, the recommender system recommends t items with the highest estimated average rating scores. Ties are broken by selecting the item with larger estimated frequency. In Case 3, the recommender system considers both the popularity and the average rating score of an item. Specifically, the recommender system calculates the product of the estimated frequency and (uncalibrated) average rating score of each item, and recommends the t items with the largest products. Ties are broken randomly. Roughly speaking, the product of the estimated frequency and average rating score of an item is the item's estimated total rating scores.
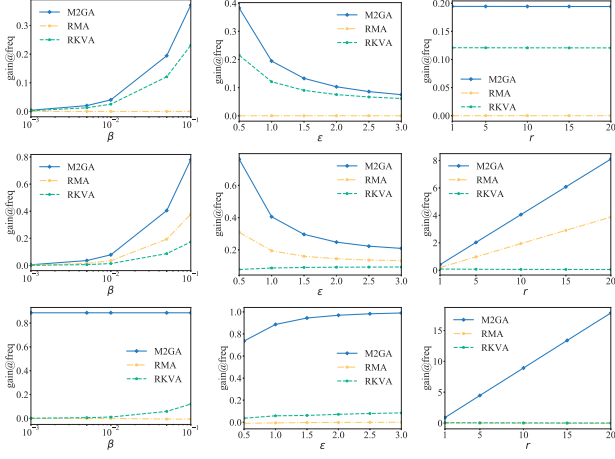
Figure 3: Impact of different parameters (β, ε, r) on the frequency gains on Clothing. The three rows are for PrivKVM, PCKV-UE, and PCKV-GRR, respectively.
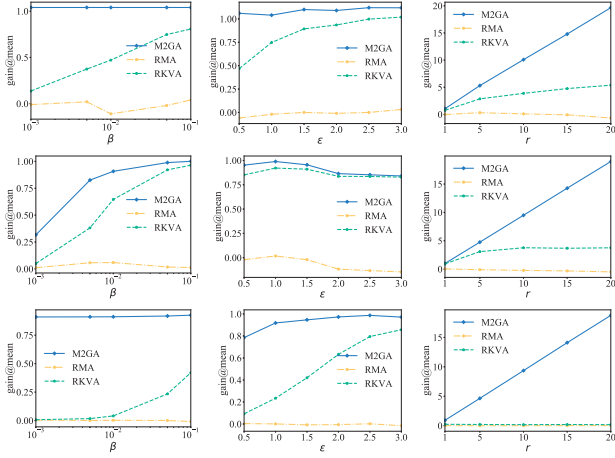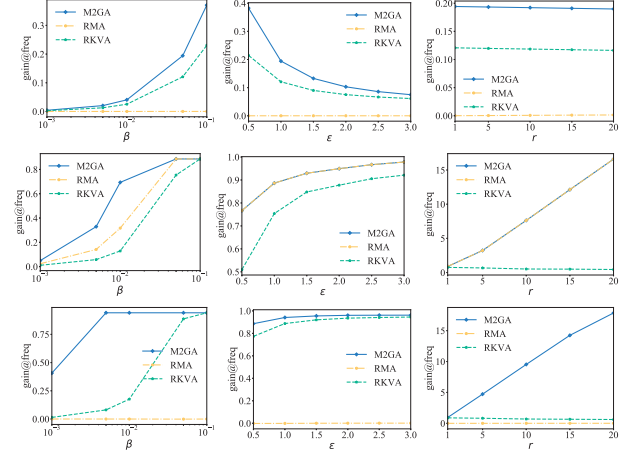

Figure 5: Impact of different parameters (β, ε, r) on the frequency gains on TalkingData. The three rows are for PrivKVM, PCKV-UE, and PCKV-GRR, respectively.

Figure 4: Impact of different parameters (β, ε, r) on the mean gains on Clothing. The three rows are for PrivKVM, PCKV-UE, and PCKV-GRR, respectively.
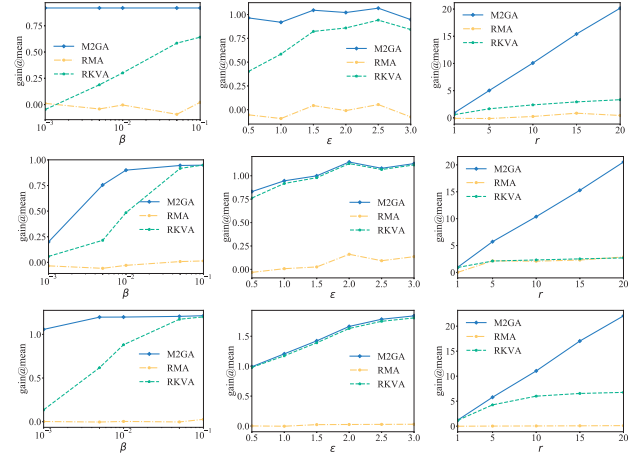

Figure 6: Impact of different parameters (β, ε, r) on the mean gains on TalkingData. The three rows are for PrivKVM, PCKV-UE, and PCKV-GRR, respectively.

6.1.3 Parameter Settings

The parameters involved are β (the fraction of fake users), ε (the privacy budget), and r (the number of target keys). PrivKVM further involves N_{iter} (the number of rounds), while PCKV-UE and PCKV-GRR further involve ℓ (the padding length). Unless otherwise mentioned, we set the default values of these parameters as follows: $\beta = 0.05$, $\varepsilon = 1.0$, $r = 1$, $N_{\text{iter}} = 10$, $\ell = 1$ for Synthetic, $\ell = 2$ for Clothing, $\ell = 20$ for TalkingData, and $\ell = 100$ for MovieLens-1M. We set ℓ differently for different datasets to consider their different characteristics, which is suggested by [24]. We set $r = 10$ and $t = 20$ by default when evaluating our attacks to the recommender system downstream application. We randomly sample r keys from the entire dictionary as the target keys for

each dataset. We vary one parameter while keeping the others fixed to their default values, to investigate its impact on the frequency and mean gains. We note that we clip the estimated frequencies and support counts in the LDP protocols as we described in Section 3.3.

6.2 Experimental Results

Figure 1–Figure 8 show the frequency gains and mean gains of our attacks on the four datasets. Figure 9 shows the ASRs of M2GA to the recommender systems in different cases on Clothing dataset. Moreover, we also explore the impact of N_{iter} on our attacks for PrivKVM, and the results are shown in Figure 10. Note that we don't show the results of frequency estimation since the frequencies of keys are estimated only in
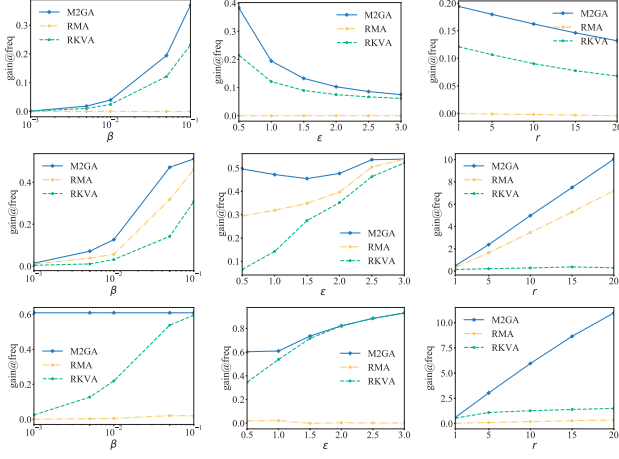
Figure 7: Impact of different parameters (β, ε, r) on the frequency gains on MovieLens-1M. The three rows are for PrivKVM, PCKV-UE, and PCKV-GRR, respectively.

Figure 8: Impact of different parameters (β, ε, r) on the mean gains on MovieLens-1M. The three rows are for PrivKVM, PCKV-UE, and PCKV-GRR, respectively.

the first round and thus are not affected by N_{iter}. We have the following observations:

- In all scenarios, M2GA achieves larger frequency and mean gains than the two baseline attacks (RMA and RKVA). This is because M2GA is an optimization based attack.

- RKVA achieves larger frequency gains than RMA except PCKV-UE, as RKVA considers target keys. RMA achieves a larger frequency gain for PCKV-UE because the target key RKVA samples gets perturbed and the perturbed message in PCKV-UE continues to support this target key with a probability of $1/2$, while a target key is supported with a probability of $2/3$ in RMA.

- M2GA and RKVA achieve larger frequency and mean gains as the number of fake users (i.e., β) increases. However, the frequency/mean gains of RMA may increase, not change, or fluctuate as β increases in different datasets and for different LDP protocols.

- The security-privacy trade-off does not necessarily hold. In particular, we observe security-privacy trade-off with respect to the frequency gains of M2GA (for PrivKVM, this is because we set $r = 1$), i.e., the frequency gains of M2GA decrease as ε increases. However, the mean gains of M2GA may increase, fluctuate, or decrease as ε increases in different datasets and for different LDP protocols.

- The mean gains of M2GA increase as the number of target keys (i.e., r) increases for all the three LDP protocols. The frequency gains of M2GA decrease as r increases for PrivKVM. This is because a fake user can only increase the estimated frequency for a single target key. The frequency gains of M2GA increase as r increases for PCKV-UE. This is because a fake user in M2GA can simultaneously support
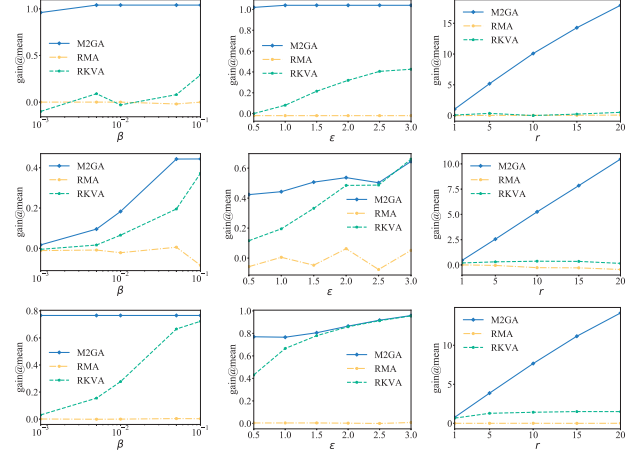
all the target keys. However, the frequency gains of M2GA for PCKV-GRR have different trends on different datasets. We find that this is mainly caused by clipping the estimated frequencies and support counts in PCKV-GRR.

- M2GA achieves high ASRs towards the recommender systems in different cases. Specifically, in Case 1 and Case 3, when $\beta \geq 0.01$ and $\varepsilon \geq 1$, M2GA achieves close-to-1 ASRs. Our results mean that the recommender system recommends almost all target items under M2GA. In Case 2, M2GA still achieves ASRs that are close to 1 for PrivKVM, while the ASRs for PCKV-UE and PCKV-GRR are close to 0.5, which means that half of the target keys/items are among the recommended t items. The ASRs for PCKV-UE and PCKV-GRR are smaller in Case 2 because the estimated average rating scores of many non-target keys are 1 in PCKV-UE and PCKV-GRR.

- Our strongest attack, i.e., M2GA, is effective for different N_{iter}. Specifically, the estimated mean after attack is consistently 1.0 when N_{iter} ranges from 1 to 10.

7 Defenses

Cao et al. [7] proposed three defenses against poisoning attacks to LDP protocols for categorical data. However, these defenses cannot be directly applied to defend against our attacks. This is because these defenses rely on the assumption that each user only holds one single item. In contrast, we consider key-value data, where each user usually has multiple KV pairs. We explore two methods to detect fake users as defenses against our poisoning attacks. For both methods, we assume the server knows the KV pairs sent from each user. For one-class classifier based detection, we further assume the server knows λ fraction of genuine users as ground truth.
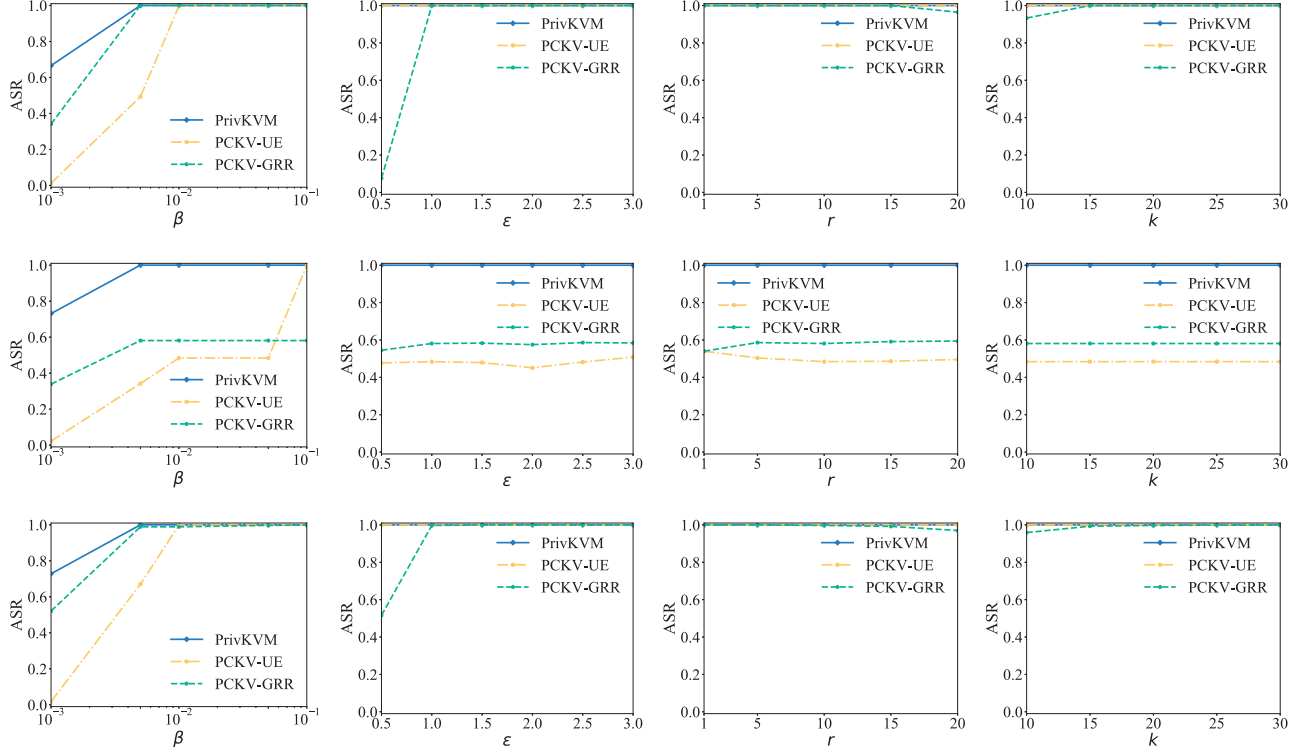
Figure 9: Impact of β, r, ε, and k on ASR of M2GA towards recommender systems (first row: Case 1, second row: Case 2, and third row: Case 3). Three LDP protocols and Clothing dataset are used.
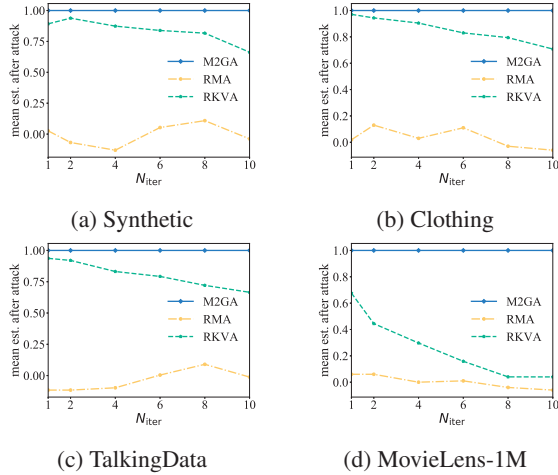

(a) Synthetic

(b) Clothing

(c) TalkingData

(d) MovieLens-1M

Figure 10: Impact of N_{iter} on the estimated mean value after attack for PrivKVM on the four datasets.

7.1 One-class Classifier (OC) based Detection

Detecting fake users is essentially an anomaly detection problem, where we aim to distinguish fake users as outliers from the genuine ones. Therefore, we can leverage the one-class machine learning classifiers that are commonly used for anomaly (outlier) detection to detect fake users. Specifically, we treat each user's messages sent to the server as its features. For PrivKVM, we concatenate each user's messages in multiple rounds as a single feature vector. We can then use these features as training data to fit an outlier detection classifier. In our experiments, we use isolation forest [35]. An isolation forest trains an ensemble of randomly partitioned trees to detect outliers. After training, the isolation forest can categorize the users into two groups. We assume the server already knows λ fraction of the genuine users as ground truth. Moreover, the server treats the group which includes more ground-truth genuine users as the "genuine" group and the other one as the "fake" group. The users in the "fake" group are considered as fake users and are excluded from aggregation. The server only uses the messages sent by users in the "genuine" group to estimate the frequencies and mean values. In our experiments, we use the implementation of isolation forest in Scikit-learn [2].

7.2 Anomaly Score (AS) based Detection

We note that, multiple rounds of communications are conducted in PrivKVM, allowing us to check the consistency of the messages sent by a user in different rounds. Based on this observation, we propose a method to detect fake users for PrivKVM. Recall that, in PrivKVM, each user sends a
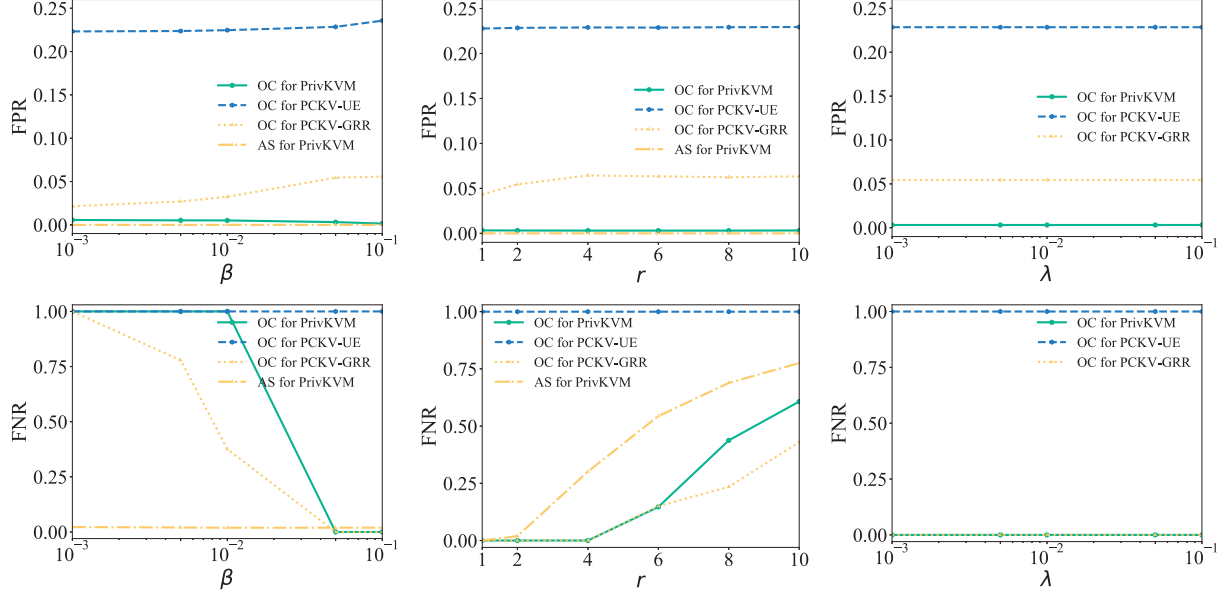
Figure 11: Impact of β, r, and λ on FPR (first row) and FNR (second row) of detecting fake users against M2GA on TalkingData.

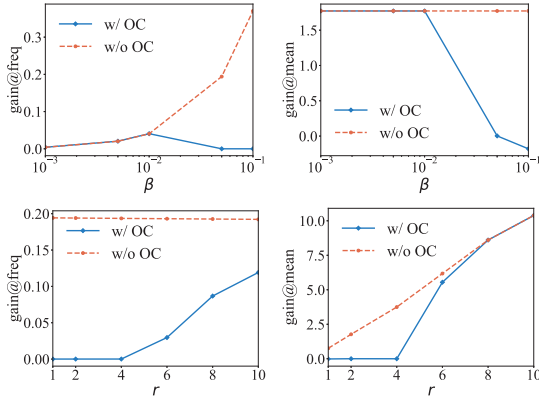
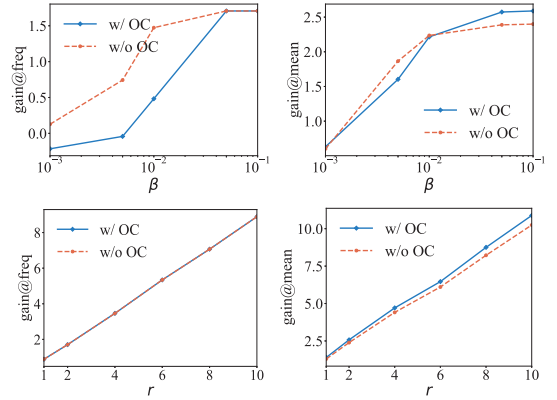Figure 12: Impact of β and r on the defense effectiveness of OC against M2GA for PrivKVM on TalkingData.

Figure 13: Impact of β and r on the defense effectiveness of OC against M2GA for PCKV-UE on TalkingData.

perturbed KV pair and the index of a key to the server in each round. Since the key is randomly sampled from the large dictionary, it is unlikely that the same key is repeatedly selected in multiple rounds for genuine users. However, since a fake user promotes a target key in each round, it may send the same key to the server in multiple rounds, especially when the number of target keys is small.

Based on this intuition, we assign an *anomaly score* to each user, which we define as the maximum number of rounds in which the user sends the same index of key to the server. Specifically, in round t, the server computes the number of rounds $N_{k,u}^t$ in which the user u has sent key k to the server. The anomaly score of user u in round t is the maximum $N_{k,u}^t$ over possible k's. If the anomaly score for a user is no smaller

than η (called *anomaly threshold*), then we mark the user as a fake one. We calculate the anomaly score of each user and detect fake users in each round. When a user is detected as fake in a certain round, we exclude the user in the subsequent rounds for mean estimation. Moreover, we re-estimate the frequencies of keys based on the messages sent by users in the first round by removing the ones belonging to the detected fake users.

7.3 Experiments

7.3.1 Experimental Setup

Unless otherwise mentioned, we adopt the following default parameters: β = 0.05, $r = 2$, ε = 1.0, $N_{iter} = 10$, and η =

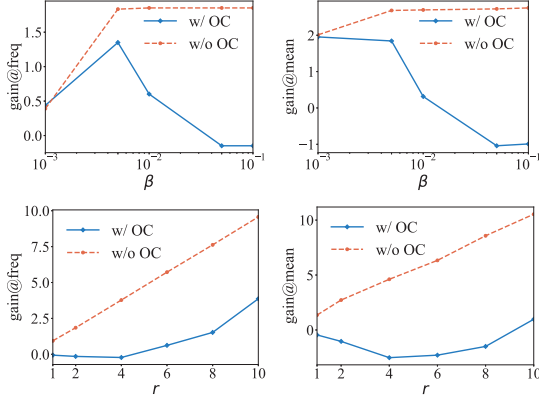Figure 14: Impact of β and *r* on the defense effectiveness of OC against M2GA for PCKV-GRR on TalkingData.

Figure 15: Impact of β and *r* on the defense effectiveness of AS against M2GA for PrivKVM on TalkingData.

2. We adopt *frequency gain (gain@freq)* and *mean gain (gain@mean)* of a set of target keys as the evaluation metrics (please refer to Section 6.1.2 for details). Moreover, we also consider *False Positive Rate (FPR)* (or *False Negative Rate (FNR)*), which is the fraction of genuine (or fake) users that are detected as fake (or genuine). We vary one parameter while keeping the others fixed to their default values to study the impact of it on the effectiveness of our defenses. Moreover, we evaluate M2GA since it is the strongest attack.

7.3.2 Experimental Results

Figure 11 shows the impact of β, *r*, and λ on the FPRs and FNRs of OC and AS against M2GA on TalkingData dataset. Figure 12, 13, and 14 show the impact of β and *r* on the defense effectiveness of OC against M2GA on TalkingData dataset for PrivKVM, PCKV-UE, and PCKV-GRR, respectively. Figure 15 shows the impact of β and *r* on the defense effectiveness of AS against M2GA for PrivKVM on Talking-Data dataset. Note that in Figure 15, we set β = 0.001 when exploring the impact of *r* to better illustrate the impact of *r* (AS is not effective in the default setting β = 0.05 regardless of *r*).

Our key observation is that the defenses are effective in some scenarios but have limited effectiveness in other scenarios. For instance, when β is small or *r* is large, OC fails to detect the fake users with high FNR. Moreover, OC has high FPR (e.g., 22% for PCKV-UE), which results in utility loss as a large fraction of genuine users are excluded from aggregation. For instance, when β = 0.05 and *r* = 2, OC for PCKV-GRR has a FPR of 5.5% and the mean gain is -1.04, which means that the estimated mean decreases by 1.04 compared to the estimated mean without attack and defense. Similarly, when β or *r* is small, AS can detect a large fraction or all of fake users, and thus the frequency and mean gains of M2GA under AS become close to 0. However, when β or *r* is large, the frequency gain and/or mean gain increase
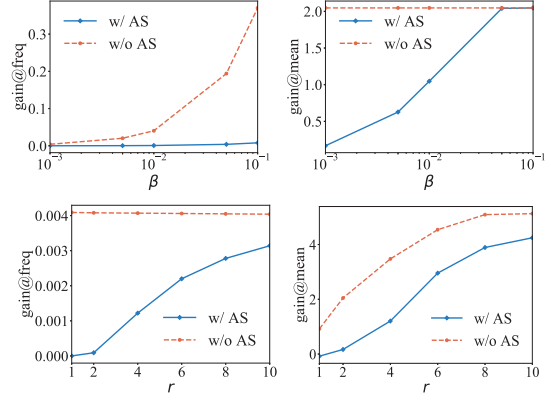
substantially. Our results show that new defenses are needed to defend against our attacks.

7.4 Other Defenses

Another defense is to use verifiable computing in the LDP protocols. For instance, the server may leverage homomorphic encryption [33] when collecting the key-value pairs. However, such methods incur large computational overhead on the user side, downgrading the user experience. Other potential defenses include detecting fake users based on additional information about the users, e.g., their social connections [12, 20, 21, 31, 42, 43, 50] or registration information [51]. Nevertheless, these detection methods are not applicable when the needed information is not available.

8 Conclusion and Future Work

In this paper, we conduct the first systematic study on poisoning attacks to LDP protocols for key-value data. We show such poisoning attacks can be formulated as a two-objective optimization problem. Our results show that an attacker can promote the estimated frequencies and mean values of attacker-chosen target keys. We also explore two defenses, which are effective in some scenarios but are ineffective in others. An interesting future work is to study defenses against our attacks.

Acknowledgments

References

[1] Clothing fit dataset for size recommendation. https://www.kaggle.com/rmisra/clothing-fit-dataset-for-size-recommendation.

[2] Scikit-learn isolation forest. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html.

[3] Talkingdata mobile user demographics. https://www.kaggle.com/c/talkingdata-mobile-user-demographics.

[4] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *AISTATS*, 2020.

[5] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *ICML*, 2019.

[6] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *ICML*, 2012.

[7] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Data poisoning attacks to local differential privacy protocols. In *USENIX Security Symposium*, 2021.

[8] Nicholas Carlini. Poisoning the unlabeled dataset of semi-supervised learning. *arXiv preprint arXiv:2105.01622*, 2021.

[9] G. Casella and R. Berger. *Statistical Inference*. Duxbury, 1990.

[10] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[11] Albert Cheu, Adam Smith, and Jonathan Ullman. Manipulation attacks in local differential privacy. In *IEEE S & P*, 2021.

[12] George Danezis and Prateek Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *NDSS*, 2009.

[13] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. *arXiv preprint arXiv:1712.01524*, 2017.

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

[15] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[16] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, 2014.

[17] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to byzantine-robust federated learning. In *USENIX Security Symposium*, 2020.

[18] Minghong Fang, Neil Zhenqiang Gong, and Jia Liu. Influence function based data poisoning attacks to top-n recommender systems. In *WWW*, 2020.

[19] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. Poisoning attacks to graph-based recommender systems. In *ACSAC*, 2018.

[20] Hao Fu, Xing Xie, Yong Rui, Neil Zhenqiang Gong, Guangzhong Sun, and Enhong Chen. Robust spammer detection in microblogs: Leveraging user carefulness. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(6), 2017.

[21] Neil Zhenqiang Gong, Mario Frank, and Prateek Mittal. Sybilbelief: A semi-supervised learning approach for structure-based sybil detection. *IEEE Transactions on Information Forensics and Security*, 9(6), 2014.

[22] Neil Zhenqiang Gong and Bin Liu. You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors. In *USENIX Security Symposium*, 2016.

[23] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[24] Xiaolan Gu, Ming Li, Yueqiang Cheng, Li Xiong, and Yang Cao. {PCKV}: Locally differentially private correlated key-value data collection with optimized utility. In *USENIX Security Symposium*, 2020.

[25] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[26] Seira Hidano, Takao Murakami, Shuichi Katsumata, Shinsaku Kiyomoto, and Goichiro Hanaoka. Exposing private user behaviors of collaborative filtering via model inversion techniques. *Proc. Priv. Enhancing Technol.*, 2020.

[27] Hai Huang, Jiaming Mu, Neil Zhenqiang Gong, Qi Li, Bin Liu, and Mingwei Xu. Data poisoning attacks to deep learning based recommender systems. In *NDSS*, 2021.

[28] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. Model-reuse attacks on deep learning systems. In *CCS*, 2018.

[29] Yujie Ji, Xinyang Zhang, and Ting Wang. Backdoor attacks against learning systems. In *CNS*, 2017.

[30] Jinyuan Jia and Neil Zhenqiang Gong. Calibrate: Frequency estimation and heavy hitter identification with local differential privacy via incorporating prior knowledge. In *INFOCOM*, 2019.

[31] Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Random walk based fake account detection in online social networks. In *DSN*, 2017.

[32] Jinyuan Jia, Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. Attriinfer: Inferring user attributes in online social networks using markov random fields. In *WWW*, 2017.

[33] Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. Preventing manipulation attack in local differential privacy using verifiable randomization mechanism. *arXiv*, 2021.

[34] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *NeurIPS*, 2016.

[35] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *ICDM*, 2008.

[36] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *NDSS*, 2018.

[37] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *AISec*, 2017.

[38] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D Joseph, Benjamin IP Rubinstein, Udam Saini, Charles A Sutton, J Doug Tygar, and Kai Xia. Exploiting machine learning to subvert your spam filter. *LEET*, 2008.

[39] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *CCS*, 2016.

[40] Apple Differential Privacy Team. Learning with privacy at scale. *Machine Learning Journal*, 2017.

[41] Kurt Thomas, Damon McCoy, Chris Grier, Alek Kolcz, and Vern Paxson. Trafficking fraudulent accounts: The role of the underground market in twitter spam and abuse. In *USENIX Security Symposium*, 2013.

[42] Binghui Wang, Neil Zhenqiang Gong, and Hao Fu. Gang: Detecting fraudulent users in online social networks via guilt-by-association on directed graphs. In *ICDM*, 2017.

[43] Binghui Wang, Jinyuan Jia, and Neil Zhenqiang Gong. Graph-based security and privacy analytics via collective classification with joint weight learning and propagation. In *NDSS*, 2019.

[44] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. Collecting and analyzing multidimensional data with local differential privacy. In *ICDE*, 2019.

[45] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally differentially private protocols for frequency estimation. In *USENIX Security Symposium*, 2017.

[46] Tianhao Wang, Ninghui Li, and Somesh Jha. Locally differentially private frequent itemset mining. In *IEEE S & P*, 2018.

[47] Tianhao Wang, Ninghui Li, and Somesh Jha. Locally differentially private heavy hitter identification. *IEEE Transactions on Dependable and Secure Computing*, 2019.

[48] Guolei Yang, Neil Zhenqiang Gong, and Ying Cai. Fake co-visitation injection attacks to recommender systems. In *NDSS*, 2017.

[49] Qingqing Ye, Haibo Hu, Xiaofeng Meng, and Huadi Zheng. Privkv: Key-value data collection with local differential privacy. In *IEEE S & P*, 2019.

[50] Haifeng Yu, Michael Kaminsky, Phillip B Gibbons, and Abraham Flaxman. Sybilguard: defending against sybil attacks via social networks. In *SIGCOMM*, 2006.

[51] Dong Yuan, Yuanli Miao, Neil Zhenqiang Gong, Zheng Yang, Qi Li, Dawn Song, Qian Wang, and Xiao Liang. Detecting fake accounts in online social networks at the time of registrations. In *CCS*, 2019.

[52] Zhikun Zhang, Tianhao Wang, Ninghui Li, Shibo He, and Jiming Chen. Calm: Consistent adaptive local marginal for marginal release under local differential privacy. In *CCS*, 2018.

A   Appendix

A.1   Optimality of M2GA

Theorem 1. *In a given execution of any of the three LDP protocols, M2GA achieves the optimal mean gain if there is only one target key k and $n_1^k \geq n_{-1}^k > \frac{(n+m)b}{2}$.*

Proof. When there is only one target key k, the mean gain in a given execution of a LDP protocol can be written as $G_m(\mathbb{Y}) = \tilde{m}_k - \hat{m}_k$, where the second term is irrelevant to the attack. Therefore, $G_m(\mathbb{Y})$ is maximized when $\tilde{m}_k$ is maximized. According to Equation (21), we have the following equation:

$$\tilde{m}_k = \frac{\left(n_1^k - n_{-1}^k + \tilde{n}_1^k - \tilde{n}_{-1}^k\right)(a-b)}{a(2p-1)\left(n_1^k + n_{-1}^k + \tilde{n}_1^k + \tilde{n}_{-1}^k - (n+m)b\right)}, \quad (28)$$

where n_1^k and n_{-1}^k are constants in a given execution. For simplicity, we let $x = n_1^k + n_{-1}^k - (n+m)b$, $y = n_1^k - n_{-1}^k$, and $z = \frac{a-b}{a(2p-1)}$. Then we can rewrite $\tilde{m}_k$ as follows:

$$\tilde{m}_k = z \cdot \frac{y + \tilde{n}_1^k - \tilde{n}_{-1}^k}{x + \tilde{n}_1^k + \tilde{n}_{-1}^k}, \quad (29)$$

where $z > 0$. Taking the partial derivative with respect to $\tilde{n}_1^k$ and $\tilde{n}_{-1}^k$, we have the following equations:

$$\frac{\partial \tilde{m}_k}{\partial \tilde{n}_1^k} = \frac{z}{(x + \tilde{n}_1^k + \tilde{n}_{-1}^k)^2} \cdot (2\tilde{n}_{-1}^k + x - y) \quad (30)$$

$$\frac{\partial \tilde{m}_k}{\partial \tilde{n}_{-1}^k} = \frac{z}{(x + \tilde{n}_1^k + \tilde{n}_{-1}^k)^2} \cdot (-2\tilde{n}_1^k - x - y) \quad (31)$$

If $n_1^k \geq n_{-1}^k > \frac{(n+m)b}{2}$, we have $x - y > 0$ and $x + y > 0$. Since $\tilde{n}_{-1}^k$ and $\tilde{n}_1^k$ are both in the range $[0, m]$, we have $\frac{\partial \tilde{m}_k}{\partial \tilde{n}_1^k} > 0$ and $\frac{\partial \tilde{m}_k}{\partial \tilde{n}_{-1}^k} < 0$. Therefore, $\tilde{m}_k$ reaches the maximum value when $\tilde{n}_1^k = m$ and $\tilde{n}_{-1}^k = 0$, which is what M2GA does. In other words, M2GA maximizes the mean gain $G_m(\mathbb{Y})$ for the given execution.

$\square$