Mitigating Data Poisoning Attacks to Local Differential Privacy

Xiaolin Li
Purdue University
West Lafayette, IN, USA
li4955@purdue.edu

Ninghui Li
Purdue University
West Lafayette, IN, USA
ninghui@purdue.edu

Boyang Wang
The University of Cincinnati
Cincinnati, OH, USA
boyang.wang@uc.edu

Wenhai Sun
Purdue University
West Lafayette, IN, USA
whsun@purdue.edu

Abstract

The distributed nature of local differential privacy (LDP) invites data poisoning attacks and poses unforeseen threats to the underlying LDP-supported applications. In this paper, we propose a comprehensive mitigation framework for popular frequency estimation, which contains a suite of novel defenses, including malicious user detection, attack pattern recognition, and damaged utility recovery. In addition to existing attacks, we explore new adaptive adversarial activities for our mitigation design. For detection, we present a new method to precisely identify bogus reports and thus LDP aggregation can be performed over the "clean" data. When the attack behavior becomes stealthy and direct filtering out malicious users is difficult, we further propose a detection that can effectively recognize hidden adversarial patterns, thus facilitating the decision-making of service providers. These detection methods require no additional data and attack information and incur minimal computational cost. Our experiment demonstrates their excellent performance and substantial improvement over previous work in various settings. In addition, we conduct an empirical analysis of LDP post-processing for corrupted data recovery and propose a new post-processing method, through which we reveal new insights into protocol recommendations in practice and key design principles for future research.

1 Introduction

Local differential privacy is a promising privacy-enhancing tool [8, 9, 16, 22, 23, 28] and widely used in many applications [11, 18, 20, 24]. Recent studies show that LDP is vulnerable to data poisoning attacks, i.e., its results can be manipulated by a small portion of malicious local users [5, 6, 14, 15, 21, 27]. This emerging threat urges people to rethink the security implications of LDP and highlights the pressing need for effective defense for vulnerable LDP protocols.

The existing research centered on attack exploration and understanding of adversarial behavior, while little attention has been paid to a systematic study of countermeasures. In this work, we focus on mitigating data poisoning attacks on state-of-the-art categorical frequency oracles (CFOs) for frequency estimation [6, 13, 23]. In the literature, the detection methods were briefly discussed supplementary to the main effort of attack discovery [5] and suffer from many limitations. For instance, in the state-of-the-art maximal gain attack (MGA) [5], the attackers set the target item indexes to 1 in their reports to boost the corresponding frequencies. Meanwhile, they also set a fixed number of non-target indexes to masquerade as benign users. Despite being intuitive, such a fixed number may leak attack patterns. Therefore, a practical detection should consider stealthier attacks that can adaptively set non-target indexes (see Section 3.2). In addition, the existing detection methods often require extra knowledge and incur non-negligible false positives and time costs. On the other hand, post-processing as a utility boosting method for LDP was also adapted to tackle post-attack data recovery [5, 12, 19]. However, there is a lack of investigations on their comparative performance and understanding of the relationship between the for-attack methods and those for no-attack settings, which is crucial for attack-aware post-processing design. In this paper, we want to answer the following research questions to provide guidance in developing a reliable and informed mitigation framework that is adaptive to various challenging attack scenarios.

RQ-1. *Can the attacker go beyond the state-of-the-art MGA [5] and launch a more adaptive attack as mentioned above for elevated stealthiness?* MGA is an intuitive adversarial strategy. A discussion on more advanced attack behaviors is paramount for us to understand the attacker's capabilities and build a meaningful foundation for better defense.

RQ-2. *Can we accurately and efficiently identify malicious users?* The current fake user detection targets MGA only with high false positives [5]. It also incurs significant computational overhead, which is not friendly to time-sensitive applications.

RQ-3. *Can we still effectively detect attack behaviors without additional data and attack knowledge when identifying bogus reports is challenging?* Fake user detection is not always possible against adaptive, stealthy attacks. It is important to capture abnormal behavior with minimal cost for rapid and informed decision-making.

RQ-4. *Given positive results from the detections mentioned in **RQ-2** and **RQ-3**, can we recover the corrupted data utility? How good are the current post-processing methods for attack suppression and utility boosting?* Prior work studied the attack [5, 19] and non-attack [26] settings independently. Discussing them with comparative analyses helps us understand their respective enabling components and interpret their performance for practical recommendations and new designs in the future.

In this paper, we study defenses against advanced data poisoning attacks on state-of-the-art CFOs protocols, i.e., GRR [13], OUE [23], OLH [23] and HST [6]. We investigate extending existing attacks to

This paper was accepted by ACM CCS 2025 (https://doi.org/10.1145/3719027.3744839).

new LDP settings and study the potentials to achieve more hidden attack behaviors. We propose a new metric to effectively measure the attack efficacy across various LDP protocols and data settings.

Based on a realistic set of attack variants, we propose a mitigation framework consisting of novel attack detections and attack-resilient post-processing methods for data recovery. In particular, we present a new *differential statistical anomaly detection* to find fake LDP participants controlled by the attacker. The method leverages the abnormal patterns of bogus reports to identify malicious users. Our experiment shows a substantial performance improvement compared to the prior work [5] (e.g., about 0.8 versus 0.3 for $F1$ scores in most cases for MGA) while enabling fake user identification in an adaptive MGA setting for the first time. To respond to the challenges of detecting bogus reports with our new attack strategy, we propose an *abnormal statistics detection* by using the inherent LDP characteristics to statistically differentiate the attack and non-attack scenarios. The experiment reports a detection accuracy of 100% in almost all tested settings. We consider our detection methods *zero-shot* since they only depend on known data knowledge and are agnostic to attack details. In addition, we significantly reduce the time costs for running detections and make them suitable for applications that are sensitive to latency in practice.

In this work, we further study utility recovery under the data poisoning attack. We propose a novel LDP post-processing method, *robust segment normalization*, and empirically study the performance of state-of-the-art for attack and non-attack purposes at the same time. We find that the widely adopted post-processing method – *consistency* (i.e., non-negative frequency estimates and sum to 1) [26], plays a more important role in effectively recovering corrupted data than other existing strategies. Based on the experimental results, we make recommendations to help service providers select appropriate LDP post-processing methods under varying attack influences. The relevant code is provided at **https://github.com/Marvin-huoshan/MDPA_LDP/**. We summarize our contributions below.

- We comprehensively investigate the mitigation against the data poisoning attack on LDP frequency estimation, including new attack exploration, novel detection methods, and attack-resilient utility recovery, which advances the much-needed defensive technology development and generates new knowledge for LDP security.
- Our research reveals a new stealthy attack strategy on existing CFO protocols, deepening our understanding of attacker capabilities. We also present a metric to help measure the attack efficacy over diverse protocols and data settings.
- We propose novel zero-shot detection methods for underlying attacks to identify malicious users and hidden attack patterns. The experimental results show significant improvement in detection accuracy and time cost compared to the state-of-the-art.
- We design a new LDP post-processing method and empirically study the impact of post-processing in the presence of attackers with varying capabilities for the first time. The study produces new insight into the recoverability of corrupted LDP data. The experimental results also help with recommendations for practical deployment.

2 Background and Related Work

2.1 Local Differential Privacy

LDP enables n users to share their data v with an untrusted server through a local perturbation function $\Psi(\cdot)$ such that only obfuscated items $\Psi(v)$ is obtained by the server. Formally,

Definition 1. *(ϵ-Local Differential Privacy [8]). An algorithm $\Psi(\cdot) : \mathcal{D} \to \hat{\mathcal{D}}$ satisfies ϵ-LDP if for any $v_1, v_2 \in \mathcal{D}$ and for $y \in \hat{\mathcal{D}}$,* $\Pr[\Psi(v_1) = y] \leq e^{\epsilon} \Pr[\Psi(v_2) = y]$.

2.1.1 Categorical Frequency Oracles. We briefly introduce some state-of-the-art CFOs used in this paper for frequency estimation in LDP. We assume that there are d items in $\mathcal{D}$. $v \in [d]$ is the index of the item in the encoding space, where $[d]$ is $\{1, ..., d\}$ for simplicity.

Generalized Randomized Response (GRR) [13]. In GRR, a user directly encodes their item v. The perturbation function keeps v with probability $p = \frac{e^{\epsilon}}{e^{\epsilon}+d-1}$ and changes it to another item with probability $q = \frac{1}{e^{\epsilon}+d-1}$. By collecting perturbed user report $y^{(j)}$ from all n users, the frequency of each item can be estimated by the aggregation function $\Phi_{\text{GRR}}(v) = \frac{C_v - nq}{n(p-q)}$, where C_v is the count of v instances in report y.

Optimal Unary Encoding (OUE) [23]. OUE achieves the theoretical lower bound of L_2 errors. In OUE, each item v is encoded into a one-hot vector $\mathbf{v} = [0, \ldots, 1, \ldots, 0]$ with 1 in the v-th position only. $\mathbf{v}$ is further randomized to the report $\mathbf{y}$ by flipping 1 to 0 with probability $p = \frac{1}{2}$ and 0 to 1 with probability $q = \frac{1}{e^{\epsilon}+1}$. We obtain the frequency of item v by $\Phi_{\text{OUE}}(v) = \frac{\sum_{j=1}^{n} \mathbf{y}^{(j)}[v] - \frac{n}{e^{\epsilon}+1}}{n(\frac{1}{2} - \frac{1}{e^{\epsilon}+1})}$.

Optimal Local Hashing (OLH) [23]. OLH can also achieve the same minimum L_2 error as OUE. It is preferred for large domain sizes by encoding input items to a smaller domain of size $g = \lfloor e^{\epsilon} + 1 \rfloor \ll d$. In the *user setting* of OLH, each user randomly selects a hash function h from a universal hash family $\mathbf{H}$ to encode $v \in [d]$ into $v_h \in [g]$. The perturbation function keeps $\hat{v}_h = v_h$ with probability $p = \frac{e^{\epsilon}}{e^{\epsilon}+g-1}$ and changes it to another hash value with probability $q = \frac{1}{e^{\epsilon}+g-1}$. Given the reports $y^{(j)} = \langle h^{(j)}, \hat{v}_h^{(j)} \rangle$ from all n users, the frequency of each item can be estimated by $\Phi_{\text{OLH}}(v) = \frac{C_v - \frac{n}{g}}{n(\frac{e^{\epsilon}}{e^{\epsilon}+g-1} - \frac{1}{g})}$, where $C_v = |\{j \mid h^{(j)}(v) = \hat{v}_h^{(j)}\}|$ counts the number of reports that support the item v.

ExplicitHist (HST) [6]. In HST, a uniform public vector $\mathbf{s}$ of length d is generated. HST becomes OLH with $g = 2$ [23]. The user j randomizes the v-th element $\mathbf{s}^{(j)}[v]$ to $\frac{e^{\epsilon}+1}{e^{\epsilon}-1} \times \mathbf{s}^{(j)}[v]$ with probability $\frac{e^{\epsilon}}{e^{\epsilon}+1}$ and to $-\frac{e^{\epsilon}+1}{e^{\epsilon}-1} \times \mathbf{s}^{(j)}[v]$ with probability $\frac{1}{e^{\epsilon}+1}$. The frequency of each item thus can be estimated by $\Phi_{\text{HST}}(v) = \frac{1}{n} \sum_{j=1}^{n} y^{(j)} \times \mathbf{s}^{(j)}[v]$.

2.1.2 LDP Post Processing. Consistency condition (i.e., non-negative frequency estimates and sum to 1) is widely used along with CFOs as a post-processing strategy to improve the utility [26]. We briefly describe Norm-Sub and Base-Cut as they are based on consistency and do not assume prior data knowledge (versus Power and PowerNS [26]). They were also recommended in prior work for non-attack situations [26]. We denote the estimated frequency as $\tilde{f}$ and the one after post-processing as $\tilde{f}'$.

Norm-Sub. Norm-Sub adjusts a frequency estimate to $\tilde{f}'_v = \max(\tilde{f}_v + \Delta, 0)$ by converting negative estimates to zero and adding Δ to the remaining estimates. As a result, the sum of all frequency estimates equals 1, i.e., $\sum_{v \in \mathcal{D}} \max(\tilde{f}_v + \Delta, 0) = 1$.

Base-Cut. Base-Cut does not consider *sum to 1*. Instead, it simply keeps the estimates that are above a sensitivity threshold and sets the rest to zero.

Post-processing methods are also adopted for utility recovery under data poisoning attacks.

Normalization. The server re-calibrates the frequency of each item v by $\tilde{f}'_v = \frac{\tilde{f}_v - \tilde{f}_{min}}{\sum_v (\tilde{f}_v - \tilde{f}_{min})}$, where $\tilde{f}_{min}$ is the smallest estimate [5].

LDPRecover. In LDPRecover [19], the observed frequency $\tilde{f}_Z(v)$ of an item is assumed to be a combination of the genuine component $\tilde{f}_X(v)$ and bogus component $\tilde{f}_Y(v)$, weighted by the proportions of genuine and fake users. LDPRecover divides items into two sets based on their observed frequencies and focuses on restoring genuine frequencies $\tilde{f}'_X$ by solving a constraint inference problem. LDPRecover also ensures the consistency condition.

In addition, LDPGuard [12] used two rounds of reports to estimate attack details, including the percentage of fake users and attack strategies, to inform the recovery, which may not be practical. We do not consider it in this paper.

We propose a new post-processing method in this work and empirically study it with the above state-of-the-art approaches in both attack and non-attack environments. We expect this will generate new knowledge about the recoverability of corrupted utility and shed light on attack-resilient designs in the future.

2.2 Attack Detection

Frequent itemset anomaly detection (FIAD) was proposed in [5] to detect fake users for MGA. The intuition is that the reports of fake users always support a set of target items regardless of LDP perturbation. The method adopted frequent itemset mining to find malicious users by checking their supported items. However, FIAD fails when r (i.e., the number of target items) is small and becomes ineffective against the adaptive MGA attack. FIAD also suffers from a high false positive rate (refer to [5] for more details). A conditional probability-based attack detection was also presented in [5] to detect polluted item frequencies instead of bogus users. It depends on ground-truth data knowledge, such as fake user percentage and attacked items, which may not be available in practice.

In this paper, we propose a novel method for fake user detection, which requires no prior data and attack knowledge. It significantly outperforms the state-of-the-art and produces fewer false positives. In addition, we present a new detection approach for scenarios where identifying bogus reports is challenging.

3 Mitigation Overview

In this section, we overview the proposed mitigation. We introduce the threat model, target attacks, and metrics for evaluation.

3.1 Threat Model

Attacker's Capability and Goal. Consistent with prior work [5, 14], we assume that the attacker can control $m = \beta \cdot n$ fake users, where n is the total number of users and $\beta \in [0, 1]$ is the percentage of fake users. Since the LDP perturbation function is on the user end, the attacker can circumvent it and directly craft bogus values in its output domain $\hat{\mathcal{D}}$. As a result, the fake reports will be injected and aggregated with benign ones on the server side. The attacker also knows the related information, such as privacy budget ϵ, the item domain $\mathcal{D}$ and its size d, and the support set $S(y)$, which is a set of items that report y supports [23].

The attacker's goal is to increase the estimated frequencies of a set of r target items $T = \{t_1, t_2, ..., t_r\}$. To this end, the attacker carefully crafts the perturbed values $\hat{Y}$ to maximize the overall gain of target items:

$$\max_{\hat{Y}} \sum_{t \in T} \mathbb{E}\left[\Delta \tilde{f}_t\right] \tag{1}$$

where $\Delta \tilde{f}_t = \tilde{f}_{t,after} - \tilde{f}_{t,before}$ is the frequency gain of target item t after the attack. We also consider a baseline attack with the same goal, but the attacker can only provide false values in the input domain of the perturbation. Thus its behavior is indistinguishable from honest users.

Knowledge of Defender. We assume that the defender knows nothing about attack details (e.g., β and T) and underlying data, other than the LDP parameters (e.g., ϵ and n) and received reports.

3.2 Attacks

Maximal Gain Attack. MGA is the state-of-the-art data poisoning attack on CFOs [5]. The main idea is to craft the perturbed values for the fake users via solving the optimization problem in Eq. (1). The original MGA only supports OUE, OLH-User (i.e., the user setting) and GRR. We extend it to OLH-Server (i.e., the server setting) and HST. We briefly describe it here and refer readers to [5] for details of the original MGA.

- **OUE.** For each fake user, the attacker initializes a zero vector $y^{(j)}$ of length d and sets $y_t^{(j)} = 1$ for all $t \in T$. To further hide traces, the attacker randomly sets $l = \lfloor p + (d-1)q - r \rfloor$ non-target bits to ensure the number of 1's matches the expected number in the report of a genuine user.
- **OLH-User.** In OLH-User, users randomly select the hash function. Therefore, each fake user will choose a hash function $h^{(j)}$ that maps all items in T to $v_h^{(j)}$, i.e., $\sum_{t \in T} \mathbb{1}_{S(y^{(j)})}(t) = r$ and submit the report $y^{(j)} = \langle h^{(j)}, v_h^{(j)} \rangle$ to the server. $\mathbb{1}_{S(y^{(j)})}(v)$ is a characteristic function and outputs 1 if $y^{(j)}$ supports item t. The number of items supported by the hash value is made close to $\sum_{v \in \mathcal{D}} \mathbb{1}_{S(y^{(j)})}(v) = \frac{d}{g}$ to further hide the attack.
- **OLH-Server.** In OLH-Server, the server chooses the hash function $h^{(j)}$ such that the attacker aims to find a hash value $v_h = \arg\max \left(\sum_{t \in T} \mathbb{1}_{S(y^{(j)})}(t) \right)$ that maps the most items $t \in T$ with $h^{(j)}$.
- **HST-User.** In this setting, a user samples her public vector $\mathbf{s}^{(j)}$. For each fake user, the attacker initializes a public vector $\mathbf{s}^{(j)} = [-1, -1, \ldots, -1]$ of length d, and sets $\mathbf{s}^{(j)}[t] = 1$ for all $t \in T$. Since $\mathbf{s}^{(j)}$ is expected to follow uniform distribution (i.e., equal numbers of -1 and 1), the attacker randomly sets $l = \lfloor d/2 -$

$r\rfloor$ non-target positions to 1 in $\mathbf{s}^{(j)}$. $y^{(j)}$ is then set to $\frac{e^\epsilon+1}{e^\epsilon-1}$ to maximize the frequencies of the target set.

- **HST-Server.** In HST-Server, the server sets a public binary vector for each user uniformly. Fake users can only promote the expected frequencies of the aggregated target set, i.e., $\sum_{t\in T}\left[\frac{1}{n}\sum_{j=1}^{n} y^{(j)}\times \mathbf{s}^{(j)}[t]\right]$, by manipulating $y^{(j)}$.
- **GRR.** A user report is a perturbed item in GRR. We have $\sum_{t\in T}\mathbb{1}_{S(y^{(j)})}(t)\le 1$ and $\sum_{t\in T}\mathbb{1}_{S(y^{(j)})}(t)=1$ when $y^{(j)}$ is a target item in T. Therefore, MGA selects any $t\in T$ for each fake user.

Adaptive Maximal Gain Attack (MGA-A). MGA-A is an enhanced MGA that can evade the FIAD detection [5] by crafting a perturbed value that only supports a subset of the target set T. Specifically, the attacker randomly selects and supports an itemset of size r' from $\binom{r}{r'}$ possible subsets for each fake user, where $r'<r$. MGA-A was shown to significantly reduce the effectiveness of FIAD, especially when $r'\le 2$, while maintaining high attack efficacy (refer to [5] for more details). MGA-A was originally designed for OUE. In this paper, we extend it to cover OLH and HST. MGA-A cannot be applied to GRR which only reports a single value.

Adaptive Pattern Attack. Though MGA-A may evade FIAD, our detection method can still effectively identify fake users (see Section 5.2). In this paper, we discover a new attack strategy, *adaptive pattern attack* (APA), where the attacker can further hide attack patterns by strategically setting bits in the crafted report. In MGA-A, the support of the values in a crafted report is matched with the expected support of a genuine user, for instance, setting $\sum_{v\in\mathcal{D}}\mathbb{1}_{S(y^{(j)})}(v)=\lfloor p+(d-1)q\rfloor$ in OUE; the proposed APA further tweaks the support at non-target positions to maximize the estimated frequencies of target items while reducing detectability. Specifically, APA can be applied to the following CFOs.

- **OUE.** For the number $k\in[0,d]$ of 1's in the report produced by the LDP, APA generates $\omega[k]$, which represents the number of fake users whose perturbed reports support k items, such that $\sum_k \omega[k]=m$. m is the total number of fake users. For different k, $\omega[k]$ fake users are selected and $l=\max(\lfloor k-r\rfloor,0)$ non-target items are randomly set to 1 in their reports.
- **OLH-User.** For any user report $y^{(j)}$, let $k\in[0,d]$ be the number of supported items $v\in\mathcal{D}$. For all k, $\omega[k]$ is generated such that $\sum_k\omega[k]=m$. For a specific fake user, assuming its assigned target support count is k, APA ensures that the items supported by the hash value $v_h^{(j)}$ are close to k, i.e., $\sum_{v\in\mathcal{D}}\mathbb{1}_{S(y^{(j)})}(v)=k$.
- **HST-User.** For $k\in[0,d]$ bits set to 1 in a report, $\omega[k]$ is generated and $\sum_k\omega[k]=m$. For different k, $\omega[k]$ fake users are selected and $l=\max(\lfloor k-r\rfloor,0)$ non-target positions are randomly set to 1 in the report. The report y_j is then set to $\frac{e^\epsilon+1}{e^\epsilon-1}$ to maximize the frequencies of the target set.

Note that in the server setting of OLH and HST, the attacker is significantly constrained in selecting hash functions or setting bits in the public vector since these are assigned by the server; in GRR, the report is a single value. Therefore, we do not consider APA in these protocols.

Baseline Attack. The baseline attack is a universal strategy that follows an LDP protocol but supplies bogus data in the input domain of LDP [5, 14]. Therefore, the behavior of a baseline attacker is indistinguishable from that of an honest user without prior knowledge of the ground truth data. We use the baseline attack as a benchmark and propose a new metric based on it for a more meaningful interpretation and comparison between our recovery methods and existing ones (see Section 3.3.1).

3.3 Workflow

We overview our mitigation for data poisoning attacks on LDP in Figure 1. The defense occurs on the server side and contains two modules: *attack detection* and *post-processing*. We propose two novel detection methods. The first is to identify fake users. The other is to detect abnormal statistics in the LDP estimates. The detection results will inform mitigation strategies in the following steps. The post-processing will further reshape the LDP estimates to suppress the attack gain and boost data utility.

Path 1: ⓐ ⇒ ⓑ ⇒ ⓒ. Given the collected LDP reports, our *fake user detection* attempts to identify malicious users (**step ⓐ**). The corresponding user reports will be subsequently removed from the report collection before being sent to the LDP aggregation (**step ⓑ**). At this point, we consider the collected dataset to be clean without attack influence. The post-processing methods will be used to further boost the data utility (**step ⓒ**). The experiment shows that our detection outperforms the state-of-the-art in identifying bogus users in MGA and MGA-A attacks.

Path 2: ⓐ ⇒ ⓓ ⇒ ⓔ ⇒ ⓕ. If detecting fake users is challenging (e.g., under APA or weak attacks close to the baseline attack), the collected data will first be sent to the LDP aggregation function (**step ⓓ**). The estimated result will be further examined by looking for abnormal statistics (**step ⓔ**). Should an attack be recognized, depending on the data practices and available resources, the server either terminates the protocol and recollects data, or post-processes the result to reduce the attack gain and recover utility (**step ⓕ**).

Path 3: ⓐ ⇒ ⓓ ⇒ ⓔ ⇒ ⓖ. This path is identical to **Path 2** except for the last step, where no attack is detected. In this case, the LDP estimate will be post-processed to increase utility (**step ⓖ**).

3.3.1 Metrics. For fake user detection, we measure the $F1$ score while we evaluate the abnormal statistics detection using *accuracy*.

We consider both *utility boosting* and *attack suppression* for LDP post-processing in the presence of data poisoning attacks. We use the widely adopted metric, *mean square error* (MSE) [23, 26] for utility. For attack suppression, frequency gain reduction is an intuitive metric to measure the change of the attack gain after post-processing. However, this metric is inherently limited in interpreting a meaningful comparison of the attack recoverability across various post-processing methods. For instance, a positive gain reduction, without additional context information, is a weak indicator of how much of the attack has been contained. In other words, it is unclear if the attack can still significantly impact the LDP result. In addition, this metric varies given different target itemset size r and malicious user percentage β. A more precise measure of attack influence at the per-fake-user and per-item level is desired. In this paper, we present a new metric, *item gain ratio* (IGR), inspired by prior work [15] to address the above issues. IGR measures a normalized frequency gain change per target item caused by a fake user after post-processing versus the attack gain by the baseline attack.

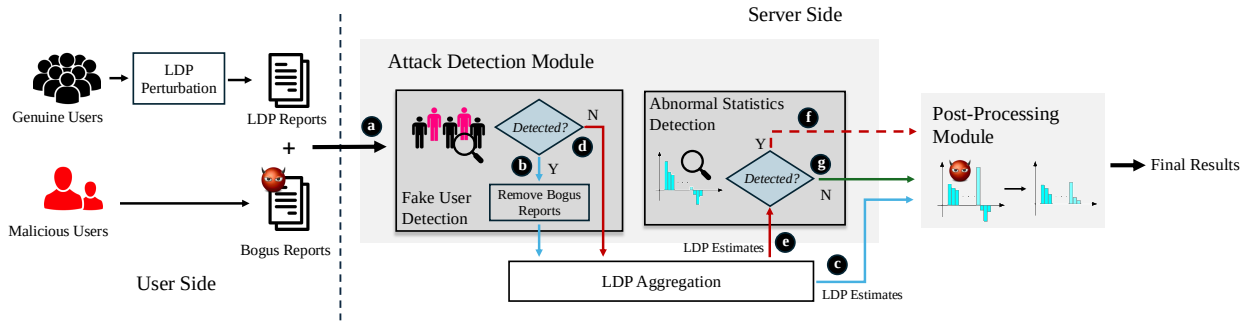

Figure 1: Mitigation workflow for data poisoning attacks on LDP

This is because the baseline represents the minimum damage the attacker can always cause in any circumstances. Formally,

$$\text{IGR} = \frac{\sum_{t\in T}\mathbb{E}(\tilde{f}'_{t,recovery} - \tilde{f}_{t,before})}{\sum_{t\in T}\mathbb{E}(\tilde{f}_{t,base} - \tilde{f}_{t,before})\cdot r} \tag{2}$$

where $\tilde{f}_{t,before}$ is the frequency before attack, $\tilde{f}'_{t,recovery}$ is the frequency after post-processing, and $\tilde{f}_{t,base}$ is the frequency under a baseline attack.

Intuitively, when IGR $\gg 1/r$, the attack is still considered to be strong compared to the baseline at the per-item level. When it decreases and approaches $1/r$, the efficacy of the attack gets close to that of the baseline attack, which is generally considered no longer a significant threat. When IGR $\in [0, \frac{1}{r}]$, the attack becomes even weaker than the baseline. A negative IGR indicates excessively suppressed frequencies of target items, which may lead to deteriorated data utility.

4 Attack Detection

In this section, we introduce the proposed attack detection methods.

4.1 Fake User Detection

In prior work [5], FIAD uses frequent itemset mining to find malicious itemsets by checking if the number of supporting users is greater than a predefined threshold. As a result, all the supporting users for these itemsets will be considered malicious. However, the detection accuracy is largely contingent on how precise we can derive the threshold, which is often challenging in practice. Moreover, the mining process is time-consuming, leading to delayed detection. In this paper, we propose a novel *differential statistical anomaly detection* (Diffstats) to overcome the above limitations. Diffstats (in Algorithm 1) identifies malicious users by adopting a different strategy that looks at the statistical differences, i.e., $E_{sq}(k)$ and E_{freq}, between fake and genuine reports. Our detection includes two steps. First, Diffstats computes the discrepancy $E_{sq}(k)$ between the observed frequency O^k (line 1) of bits set to 1 in the reports of n users and the expected frequency Y^k (line 2). It then leverages $E_{sq}(k)$ to determine a candidate fake user set $\mathcal{U}_s$ from all users $\mathcal{U}$ (line 6-10). Second, to reduce false positives, Diffstats calculates E_{freq} for all $\{\mathcal{U}/\mathcal{U}_{sc}\}$ (line 15), where $\mathcal{U}_{sc}$ are subsets of $\mathcal{U}_s$ (line 14). Consequently, the smaller E_{freq} is, the more likely

Algorithm 1 Diffstats **for Fake User Detection**

Input: Users' reports $\mathcal{Y}$.
Output: Fake users $\mathcal{U}_f$.

1: Set $O^k = |\{y^{(j)} \in \mathcal{Y} \mid \sum_{v\in\mathcal{D}} \mathbb{1}_{S(y^{(j)})}(v) = k\}|$
2: Set $Y^k = n \cdot P(X = k)$.
3: Initialize the minimum E_{freq} as $E_{min} = +\infty, \mathcal{U}_f = \emptyset$
4: Set $\mathcal{K} = \{0, 1, 2, ..., d\}$
5: **while** $\mathcal{K} \neq \emptyset$ **do**
6: Let $\delta = \arg\min_{k\in\mathcal{K}} E_{sq}(k)$
7: Update $\mathcal{K} = \{\mathcal{K}\backslash\delta\}$
8: Initialize candidate fake users set $\mathcal{U}_s = \emptyset$
9: **for** $k \in \mathcal{K}$ **do**
10: Add users to $\mathcal{U}_s$ whose reports contain k-bit "1".
11: **end for**
12: For the top-L supported items $\mathcal{S}_L$ in the common support set S, compute all combinations $\mathcal{P}(\mathcal{S}_L)$ in $\mathcal{U}_s$.
13: **for** s in $\mathcal{P}(\mathcal{S}_L)$ **do**
14: Obtain subset $\mathcal{U}_{sc} \subseteq \mathcal{U}_s$ that supports s.
15: Compute E_{freq} for all users in $\{\mathcal{U}\backslash\mathcal{U}_{sc}\}$.
16: **if** $E_{freq} < E_{min}$ **then**
17: Update $E_{min} = E_{freq}$ and $\mathcal{U}_f = \mathcal{U}_{sc}$
18: **end if**
19: **end for**
20: **end while**
21: **return** $\mathcal{U}_f$

that the corresponding $\{\mathcal{U}/\mathcal{U}_{sc}\}$ contains more genuine users and $\mathcal{U}_{sc}$ is malicious (line 16-17). The two steps are performed iteratively for all k until producing the final result of fake users $\mathcal{U}_f$. The detailed description of identifying $\mathcal{U}_{sc}$ for reduced false positives is provided in Section 4.1.3. In addition, our method is much faster in fake user identification compared to FIAD (see Section 5.2).

4.1.1 Frequency Approximation. In this section, we first describe deriving the frequency X of the number of "1" bits for one user and then extend it to Y^k for n users. We take OUE as an example and later discuss OLH, HST, and GRR.

For the OUE perturbation, we consider the event of keeping the bit of 1 (i.e., the user's item) a Bernoulli trial, i.e., $X_a \sim \text{Bernoulli}(p)$

and flip the remaining $d-1$ bits independently to 1 with probability q. As a result, we may use a binomial distribution for this process, i.e., $X_b \sim B(d-1, q)$. The distribution X of the number of "1" bits in a user's report can be estimated as the sum of two random variables, X_a and X_b, following a Poisson binomial distribution $X \sim PB(p_1, p_2, \ldots, p_d)$, where $X = \sum_{i=1}^{d} X_i$ and each X_i is an independent Bernoulli random variable with expected value $E[X_i] = p_i$. $p_i = p$ if $i = v$, and $p_i = q$ otherwise. We subsequently approximate $X \sim PB(p_1, p_2, \ldots, p_d)$ using a binomial distribution $X \sim B(d, \tilde{p})$, where $\tilde{p} = \frac{p+(d-1)q}{d}$ is the mean of p_i. The approximated error bound can be derived as $dist(PB, B) \leq (1-\tilde{p}^{d+1}-\tilde{q}^{d+1})\frac{\sum_{i=1}^{d}(p_i-\tilde{p})^2}{(d+1)\cdot\tilde{p}\cdot\tilde{q}}$ according to [10], where $\tilde{q} = 1-\tilde{p}$. $dist(PB, B)$ gets close to 0 if and only if the ratio R_{var} of the variance of PB to that of B approaches 1. To see this, for $B(d, \tilde{p})$ and its variance $Var(B) = d\tilde{p}(1-\tilde{p})$, we have $R_{var} = \frac{Var(PB)}{Var(B)} = 1 - \frac{(d-1)(p-q)^2}{d^2\tilde{p}(1-\tilde{p})}$. Since d is typically large in underlying applications (e.g., $d > 100$ items), R_{var} approaches 1, resulting in $dist(PB, B)$ close to 0. Therefore, we can readily approximate $X \sim B(d, \frac{p+(d-1)q}{d})$ for a single user. Since the perturbations for n users are i.i.d. processes, the expected frequency Y^k of bits set to "1" in the reports of n genuine users is

$$Y^k \approx n \cdot P(X = k),$$

where $P(X = k) = \binom{d}{k}\tilde{p}^k(1-\tilde{p})^{d-k}$ is the probability mass function of the binomial distribution, given that exactly k bits in a single user's vector are set to 1.

Other Target CFOs Protocols. The expected frequency Y^k serves as a baseline for calculating errors E_{freq} and E_{sq}. Under different CFOs, the distribution X varies given distinct encoding and perturbation functions, which thus results in different Y^k. In OLH, a malicous user crafts a fake report $y^{(j)} = \langle h^{(j)}, v_h^{(j)} \rangle$ with $\sum_{t\in T} \mathbb{1}_{S(y^{(j)})}(t) = r$. OLH requires that the choice of the hash function from $\mathbf{H}$ is uniform. This ensures the expected size of the support set of the perturbed value $y^{(j)}$ in the input domain $\mathcal{D}$ is $\frac{d}{g}$. Therefore, $X \sim B(d, \frac{1}{g})$. Likewise, in HST, we consider the number of 1's in the public vector. Assuming an ideal and uniform public vector, the distribution is modeled as $X \sim B(d, \frac{1}{2})$. GRR is more vulnerable to MGA attacks compared to OUE and OLH when $d > (2r-1)(e^{\epsilon}-1) + 3r$ [5]. Since each user only reports a single value, our fake user detection and prior work do not apply. However, the attack can still be caught by our abnormal statistics detection (see Section 4.2).

4.1.2 Quantifying Frequency Discrepancies. In this subsection, we elaborate on quantifying the errors E_{sq} and E_{freq} to measure the gap between observed frequency O^k and expected frequency Y^k, which helps us identify malicious users. E_{sq} indicates the error between O^k and Y^k for a particular k for all n users. Therefore, we have

$$E_{sq}(k) = (O^k - Y^k)^2$$

where $O^k = |\{y^{(j)} \in Y \mid \sum_{v\in\mathcal{D}} \mathbb{1}_{S(y^{(j)})}(v) = k\}|$. We define the error E_{freq} as the chi-square statistic in the chi-square goodness-of-fit test [17]. Formally,

$$E_{freq}(O^k, Y^k) = \sum_k \frac{(O^k - Y^k)^2}{Y^k}, \tag{3}$$

where the degree of freedom is $k-1$. Thus, E_{freq} is the sum of $E_{sq}(k)$ for all k. Note that our analyses below are generic and apply to OUE, OLH, and HST.

Error Analysis for MGA and MGA-A. MGA and MGA-A adopt the same attack strategy by setting additional bits up to the expected number of "1" in a report of a genuine user. Therefore, the following results apply to both.

Theorem 4.1. *For m fake users and $k \in [0, d]$, the expected error $\mathbb{E}\left[E_{sq}(k)\right]$ of the MGA and MGA-A is*

$$\mathbb{E}\left[E_{sq}^{MGA}(k)\right] = \begin{cases} m^2 \cdot (P(X=k)-1)^2 + Var(O_{MGA}^k), & \text{if } k = l_g, \\ m^2 \cdot (P(X=k))^2 + Var(O_{MGA}^k), & \text{otherwise,} \end{cases}$$

where $l_g = \lfloor p + (d-1)q \rfloor$ is the expected number of "1" in a genuine user's report and $Var(O_{MGA}^k) = (n-m)P(X=k)(1-P(X=k))$ denotes the variance of the observed frequency O_{MGA}^k under the MGA attack.

Proof. See Appendix A □

Given Theorem 4.1, we further derive E_{freq} in Theorem 4.2.

Theorem 4.2. *The error $E_{freq}(O_{MGA}^k, Y^k)$ between the observed frequency O_{MGA}^k under MGA or MGA-A attack and the expected frequency Y^k is*

$$E_{freq}(O_{MGA}^k, Y^k) = \frac{m^2}{n}\left(\frac{1}{P(X=l_g)} - 1\right) + \frac{(n-m)\cdot d}{n} \tag{4}$$

Proof. See Appendix B □

Theorem 4.1 and Theorem 4.2 show that both errors have a quadratic relationship with m, i.e., the number of fake users. They are sensitive to the change of m, which is desirable. This aligns with our intuition that more fake users generally lead to a stronger attack with more skewed statistics, thus benefiting the detection. In addition, the fake users in MGA and MGA-A set l_g "1" in their reports while in practice the probability $P(X = l_g)$ for genuine users is low. Therefore, E_{freq} becomes evident in the presence of attacks. The detection performance also relies on the domain size d. For simplicity, we approximate $X \sim N(\mu = d\tilde{p}, \sigma = \sqrt{d\tilde{p}(1-\tilde{p})})$. Since $\mathbb{E}\left[P(X = l_g)\right] = \mathbb{E}\left[f(x = \mu)\right] = \frac{1}{\sigma\sqrt{2\pi}} = \frac{1}{\sqrt{2\pi d\tilde{p}(1-\tilde{p})}}$, a small d gives rise to a large $P(X = l_g)$ and a small E_{freq}, thus rendering the detection less effective.

Error Analysis for APA. We analyze the expected error $\mathbb{E}\left[E_{sq}^{APA}(k)\right]$ for APA by Theorem 4.3 below.

Theorem 4.3. *For m fake users and $k \in [0, d]$, the expected error $\mathbb{E}\left[E_{sq}^{APA}(k)\right]$ under APA attack is*

$$\mathbb{E}\left[E_{sq}^{APA}(k)\right] = (m \cdot P(X=k) - \omega[k])^2 + Var(O_{APA}^k).$$

$\omega[k] \in [0, m]$ is the number of attacker vectors in which exact k bits are set to 1 and $Var(O^k_{APA}) = (n-m)P(X=k)(1-P(X=k))$ denotes the variance of the observed frequency O^k_{APA} under the APA attack.

Proof. See Appendix C. □

Theorem 4.4 gives the error E_{freq} for the APA attack.

Theorem 4.4. *The error $E_{freq}(O^k_{APA}, Y^k)$ between the observed frequency O^k_{APA} under APA attack and the expected frequency Y^k is*

$$E_{freq}(O^k_{APA}, Y^k) = \frac{1}{n}\sum_{k=0}^{d}\frac{(m \cdot P(X=k) - \omega[k])^2}{P(X=k)} + \frac{(n-m)\cdot d}{n}$$

Proof. See Appendix D □

Unlike in MGA and MGA-A, Theorem 4.3 shows that $E^{APA}_{sq}(k)$ in APA is affected by not only m but also $\omega[k]$. A similar relationship is observed in E_{freq} in Theorem 4.4. In other words, the attacker can hide traces by adjusting $\omega[k]$ for different k to offset the impact of m, which makes the detection challenging in practice. We further theoretically analyze the detection performance under different attacks in Section 4.1.4.

4.1.3 Put All Together. Diffstats employs E_{sq} and E_{freq} to identify potential bogus reports, as in Algorithm 1. Specifically, E_{sq} is adopted to find fake user candidates, while E_{freq} is subsequently used to minimize false positive rates and refine the final result.

To generate a candidate set $\mathcal{U}_s$ of fake users, we first determine $\delta = \arg\min_{k\in\mathcal{K}} E_{sq}(k)$ for $\mathcal{K} = [0, 1, ..., k]$. Subsequently, we add users into $\mathcal{U}_s$ whose reports contain k-bit 1's for $k \in \mathcal{K} \setminus \delta$, as their $E_{sq}(k)$ is relatively large compared to $E_{sq}(\delta)$. However, the derived $\mathcal{U}_s$ may contain genuine users.

To reduce false positives, we measure the error E_{freq} of a group of "genuine" users $\{\mathcal{U}\backslash\mathcal{U}_s\}$. The smaller the derived E_{freq}, the higher the likelihood that $\mathcal{U}_s$ is malicious. Note that this counters the intuition of directly measuring the distance between malicious and benign groups. This is because O^k and Y^k are anticipated to characterize statistically similar populations by definition, and in practice, we only know the expected pattern Y^k of genuine users.

To filter out "clean" users, we examine the behavioral differences between attackers and benign users. We observe that the fake users consistently set corresponding bits of the target items to 1 in their reports, which is unlikely among genuine users. After determining the common support set $\mathcal{S} = \{S_i\}_{i\in[1,...,d]}$ by summing up the bits at i-th position of the user reports, we derive top-L supported items $\mathcal{S}_L$ from $\mathcal{S}$. For all combinations $\mathcal{P}(\mathcal{S}_L)$ of $\mathcal{S}_L$, Diffstats identifies a subset $\mathcal{U}_{sc}$ of $\mathcal{U}_s$ and computes E_{freq} for "clean" users in $\{\mathcal{U}\backslash\mathcal{U}_{sc}\}$. If the error is smaller than the current minimum, $\mathcal{U}_{sc}$ is likely to be malicious since the attacker aims to maximize the attack gain. The algorithm will eventually determine a group of users $\mathcal{U}_f$ as fake users by iterating through all $k \in \mathcal{K}$.

4.1.4 Performance Analysis of Diffstats. We theoretically analyze the performance of the detection for different attacks. In MGA-A, the attacker can adjust r to evade FIAD detection (e.g., $r \le 2$ in [5]). Theoretically, a large r helps Diffstats distinguish honest and malicious users through common support. On the other hand, the candidate set $\mathcal{U}_s$ also facilitates this process. The experiments show that our detection can effectively identify fake users even with a small r. We analyze the error relationship R between APA and MGA (MGA-A) in Theorem 4.5.

Theorem 4.5. *For $k = \lfloor p + (d-1)q \rfloor$ the relationship between $\mathbb{E}\left[E^{MGA}_{sq}(k)\right]$ and $\mathbb{E}\left[E^{APA}_{sq}(k)\right]$ satisfies*

$$\begin{cases} \mathbb{E}\left[E^{MGA}_{sq}(k)\right] = \mathbb{E}\left[E^{APA}_{sq}(k)\right], & \text{if } \omega[k] = m, \\ \mathbb{E}\left[E^{MGA}_{sq}(k)\right] > \mathbb{E}\left[E^{APA}_{sq}(k)\right], & \text{if } \omega[k] < m \end{cases}$$

Proof. See Appendix E □

In addition to m, $\mathbb{E}[E^{APA}_{sq}(k)]$ is also affected by $\omega[k]$, which is the number of attacker vectors where k bits are set to 1. Adjusting $\omega[k]$ will effectively change the stealthiness of APA. Theorem 4.5 shows that when $\omega[k = l_g] = m$, APA is in the worst-case scenario and becomes equivalent to MGA; when $\omega[k] < m$, APA is stealthier than MGA. According to Theorem 4.3, when $\omega[k] = m \cdot P(X=k)$ for all k, $\mathbb{E}[E^{APA}_{sq}(k)]$ reaches its lower bound $Var(O^k_{APA})$, which represents the optimal attack strategy for APA. In this case, it is difficult to identify suspicious users even if m is large. To address this issue, we present a new detection method below.

4.2 Abnormal Statistics Detection

We propose *abnormal statistics detection* (ASD) to detect the APA attack and the MGA and MGA-A attacks in GRR. The design is inspired by the observation that the sum of the true item counts $\sum_{i=1}^{d} C_i$ should not exceed n; thus the sum of all perturbed counts $\sum_{i=1}^{d} \tilde{C}_i > n$ may indicate the presence of the attack that promotes the target items by amplifying the corresponding frequencies. However, $\sum \tilde{C}_i$ may exceed n due to random LDP noise, making the naïve use of this condition unreliable. To address this challenge, we derive a threshold ξ that divides $\{\tilde{C}_i\}_{i\in[d]}$ into two subsets, $\mathcal{A} = \{\tilde{C}_i | \tilde{C}_i > \xi\}$ and $\mathcal{B} = \{\tilde{C}_i | \tilde{C}_i \le \xi\}$, such that for the items in $\mathcal{A}$, the sum of their perturbed counts should be close to n but still not surpass it. Since the attack gain is unlikely to exist in the lower count domain $\mathcal{B}$, $\sum_{\tilde{C}_i\in\mathcal{A}} \tilde{C}_i > n$ will be a tighter and more sensitive detection condition. We elaborate on how to find such ξ below.

4.2.1 Determining ξ. Directly determining ξ as the lower bound of $\mathcal{A}$ is challenging. Instead, we attempt to find ξ as the upper bound of $\mathcal{B}$, which is anticipated to contain much fewer items compared to $\mathcal{A}$. As a result, $\mathcal{A}$ can be derived by excluding $\mathcal{B}$ in the full domain.

Intuitively, $\mathcal{B}$ should at least contain items whose true frequencies $f_i = 0$, because the expected sum of their perturbed counts would not contribute to n. Thus, we are looking for a heuristic $\mathcal{B} = \{\tilde{C}_i | f_i = 0\}$ for $i \in [d]$. It is difficult to derive the precise upper bound ξ of $\mathcal{B}$ without underlying data knowledge, such as true item frequencies. Instead, we aim to find an approximation $\xi(\gamma)$ with confidence level γ. To this end, we first estimate the distribution of $\tilde{C}_i$. Prior work [23] shows that $\tilde{C}_i \sim N(n \cdot f_i,\ \sigma_i^2)$. Theorem 4.6 gives the upper bound $\xi(\gamma)$ of the set $\mathcal{B}$ below.

Theorem 4.6. *For a given confidence level γ, the upper bound of $\mathcal{B} = \{\tilde{C}_i | f_i = 0\}$ for $i \in [d]$ is*

$$\xi(\gamma) = Z(\gamma) \cdot \sqrt{\frac{nq(1-q)}{(p-q)^2}}$$

Proof. See Appendix F. □

$\xi(\gamma)$ indicates that we have confidence γ that all the items in set $\mathcal{B}$ have their true frequencies $f_i = 0$. A larger γ also results in a larger $\xi(\gamma)$, which increases the likelihood that all the items with $f_i = 0$ are included in $\mathcal{B}$. On the other hand, $\mathcal{A}$ may also contain zero-frequency items with probability $1 - \gamma$. Thus the introduced error Err may cause expected sum $\mathbb{E}[\sum_{\tilde{C}_i \in \mathcal{A}} \tilde{C}_i] > n$ and affect detection accuracy. It is challenging to precisely calculate Err since we need to know the true frequencies f_i (see Appendix G). Alternatively, we approximate the error as

$$Err = |\mathcal{B}| \cdot \xi(\gamma) \cdot (1 - \gamma)$$

where $|\mathcal{B}|$ is the size of the set $\mathcal{B}$ and $\xi(\gamma) \cdot (1 - \gamma)$ indicates the occurrence of an error event that $\mathcal{A}$ contains an item of $f_i = 0$ with probability $1 - \gamma$. This error is an approximation of the true error (Eq. (5) in Appendix G) and represents the upper-bound of the error corresponding to the items with $f_i = 0$.

Therefore, we formulate the detection as an optimization problem to find the minimum γ

$$\begin{aligned} \min \quad & \gamma \\ \text{s.t.} \quad & Err < \lambda \cdot n \\ & 0 < \lambda < 1 \end{aligned}$$

The condition confines the error below a predefined threshold $\lambda \cdot n$ while ensuring a minimum γ to keep $\mathbb{E}[\sum_{\tilde{C}_i \in \mathcal{A}} \tilde{C}_i]$ close to n.

4.2.2 Performance Analysis of ASD. Since we estimate $\tilde{C}_i \sim N(n \cdot f_i,\ \sigma_i^2)$, a larger n makes the distribution closer to the normal distribution according to the central limit theorem and thus may lead to better detection results. We may also derive $\sum_{i=1}^{d} \tilde{C}_i \sim N(n, \sum_{i=1}^{d} \sigma_i^2)$ due to independent $\tilde{C}_i$. We can see that with more items, the variance will increase and thus it is difficult to satisfy $\mathbb{E}[\sum_{\tilde{C}_i \in \mathcal{A}} \tilde{C}_i] \leq n$. The detection accuracy is also expected to decline. On the other hand, a weak attack with a large ϵ and a small β will cause the attacked $\tilde{C}_i$ to be close to their expected true count $n \cdot f_i$. Therefore, the detection will be less effective by satisfying the condition $\sum_{\tilde{C}_i \in \mathcal{A}} \tilde{C}_i > n$.

5 Detection Evaluation

5.1 Experimental Setup

Datasets. We use one synthetic dataset and two real-world datasets to evaluate the proposed detection methods.

- **zipf.** We synthesize a dataset containing $d = 1,024$ items and $n = 1,000,000$ users satisfying a zipf distribution, where $s = 1.5$.
- **emoji.** We use $d = 1,496$ emojis from [1] and treat their average "sent" statistics as the number of users, i.e., $n = 218,477$.
- **fire.** The dataset contains information on calls for the San Francisco Fire Department [2]. We only use the "Alarms" records and take each unique unit ID as an item. Therefore, there are $d = 296$ items from $n = 723,090$ users.

The datasets and their pre-processing details, along with the source code, are provided at **https://github.com/Marvin-huoshan/MDPA_LDP/**.

Settings. The experiments were conducted on a server with Ubuntu 22.04.5 LTS, 2× AMD EPYC 9554 CPU, and 768GB RAM. We use a default $\epsilon = 1$ for LDP and $\beta = 0.05$, $r = 10$ for all attacks unless specified in the paper. We also set $r' = 4$ by default for APA for balanced stealthiness and efficacy of the attack. We follow the optimal APA attack strategy by setting $\omega[k] = \lfloor m \cdot P(X = k) \rfloor$ for all k as discussed in Section 4.1.4. In OLH-User, we allow the attacker to find an optimal hash function $h^{(j)}$ that maps all target items in T to $v_h^{(j)}$.

For Diffstats detecting fake users, we select $L = 6$ by default and set the hyperparameter $\lambda = 0.02$ in ASD detection.

Metrics. We measure $F1$ score for our method Diffstats to correctly identify fake users while minimizing false positives and false negatives. The results are an average of 10 trials. We measure the detection *accuracy* for ASD over a total of 40 instances including 20 attack and 20 no-attack cases.

In the experiment, we compare the proposed Diffstats with the state-of-the-art fake user detection, FIAD [5] and only shows the performance of ASD since no similar method exists for detecting MGA and MGA-A attacks on GRR and APA attack.

5.2 Results for Diffstats

The detection results against MGA are shown in Figure 2 and those against MGA-A in Figure 3. A higher $F1$ score indicates better detection performance, and shaded regions represent 95% confidence intervals (CI). Since the majority of the measured CIs are narrow, i.e., typically $[0.02, 0.05]$, which makes them difficult to see in the figures.

Detection for MGA. In general, our experiment shows that Diffstats outperforms FIAD by a large margin across all datasets with varying parameters. Both detection methods become less effective under the server setting of OLH and HST. This is because those protocols are naturally more robust to the attacks, which is consistent with prior results [15]. In addition, we have the following key observations.

- Diffstats performs almost constantly with $F1$ greater than 0.8 versus $F1$ of FIAD below 0.4 for a common setting of $\epsilon \leq 1$ across all datasets.
- We observe a performance drop of our method in the fire dataset of a small domain size d when $\epsilon = 1$ in OLH-User. This aligns with our analysis of the error E_{freq} in Eq. (4) that a small d leads to a loss in detection performance. At the same time, the attack becomes weak when it is difficult to find a hash function that covers all target items as ϵ grows. However, performance bounces back quickly with a larger ϵ since fewer genuine users are included in the common

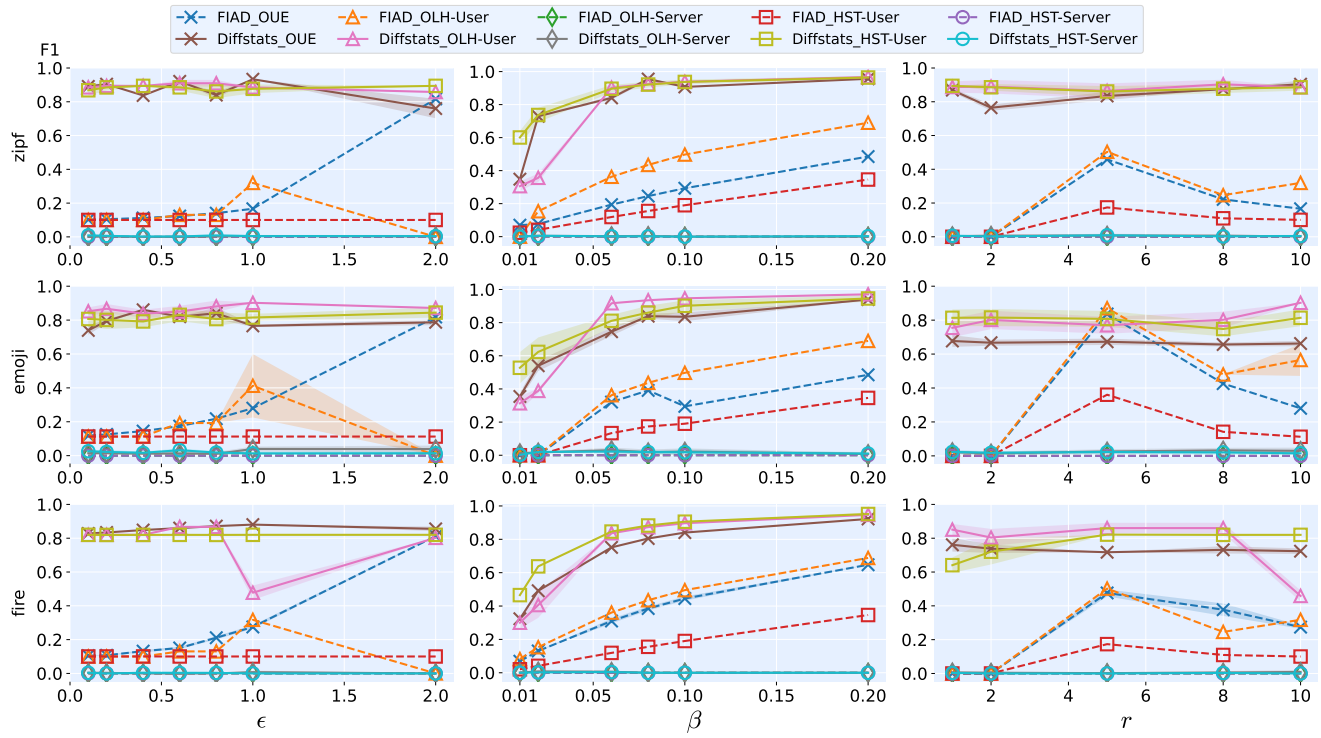

Figure 2: Performance comparison between the proposed Diffstats **and FIAD [5] against the MGA attack.**

support in Diffstats, thus reducing false positives. In contrast, the performance of FIAD for OLH-User is significantly affected given a large ϵ.

- We observe that the $F1$ of FIAD first increases and then reduces with growing ϵ for OLH-User with all three datasets. Given a smaller ϵ, though a stronger attack presents for it is easier for more fake users to find hash functions to cover all target items, more LDP noise also results in higher false positives, which is more prominent in detection performance (i.e., low $F1$ score). As the negative impact of LDP noise diminishes much faster than the difficulty of obtaining ideal hash functions with increasing ϵ, the combined effect makes the $F1$ score of FIAD first peak at about $\epsilon = 1$ and then decrease rapidly. This phenomenon is consistent across all datasets, with a wider CI in the emoji dataset. This is because the number of fake users in the emoji is much smaller than that in the other two, leading to a more "volatile" measurement.
- While our method performs stably in OUE across all ϵ with a narrow 95% CI typically ranging from 0.02 to 0.05, FIAD only becomes more effective as ϵ increases. With less LDP noise, the identified frequent itemsets in FIAD are likely to include the target items.
- The performance (i.e., $F1$ and CI) of both detection methods improves for stronger attacks with more malicious users (i.e., larger β) except for the server settings. Compared to FIAD, our method stays sensitive to small β and grows much faster to an $F1$ score of above 0.9 as β increases.
- Diffstats is insensitive to changes in the target set size r, consistently showing superior performance. The $F1$ score drops when r becomes larger for OLH-User as the attacker fails to find a qualified hash function. On the other hand, FIAD cannot identify fake users when $r \leq 2$. It performs best at $r = 5$ and worsens as r further grows. This is because a relatively large target set helps FIAD filter out malicious items in general. However, excessively large r leads to a higher false positive rate in FIAD.

Detection for MGA-A. For MGA-A attack, Figure 3 shows that:

- The performance gap between our method and FIAD becomes even clearer against the MGA-A attack. Diffstats shows a constantly high and stable $F1$ score while FIAD fails to detect fake users in most cases.
- Consistent with the MGA case, our detection is more effective when a strong attack with a large β is present.
- Compared to other protocols, Diffstats in HST-User is more subject to the changes of r'. Especially when r' gets larger, higher false positives may appear as more items exist in the common support.
- Diffstats performs better in zipf and emoji since the fire dataset has a small domain size d, making the detection more challenging.

We further evaluate our approach and FIAD for APA with the optimal strategy described in Section 4.1.4. The experiment confirms our theoretical analysis that APA is much stealthier than

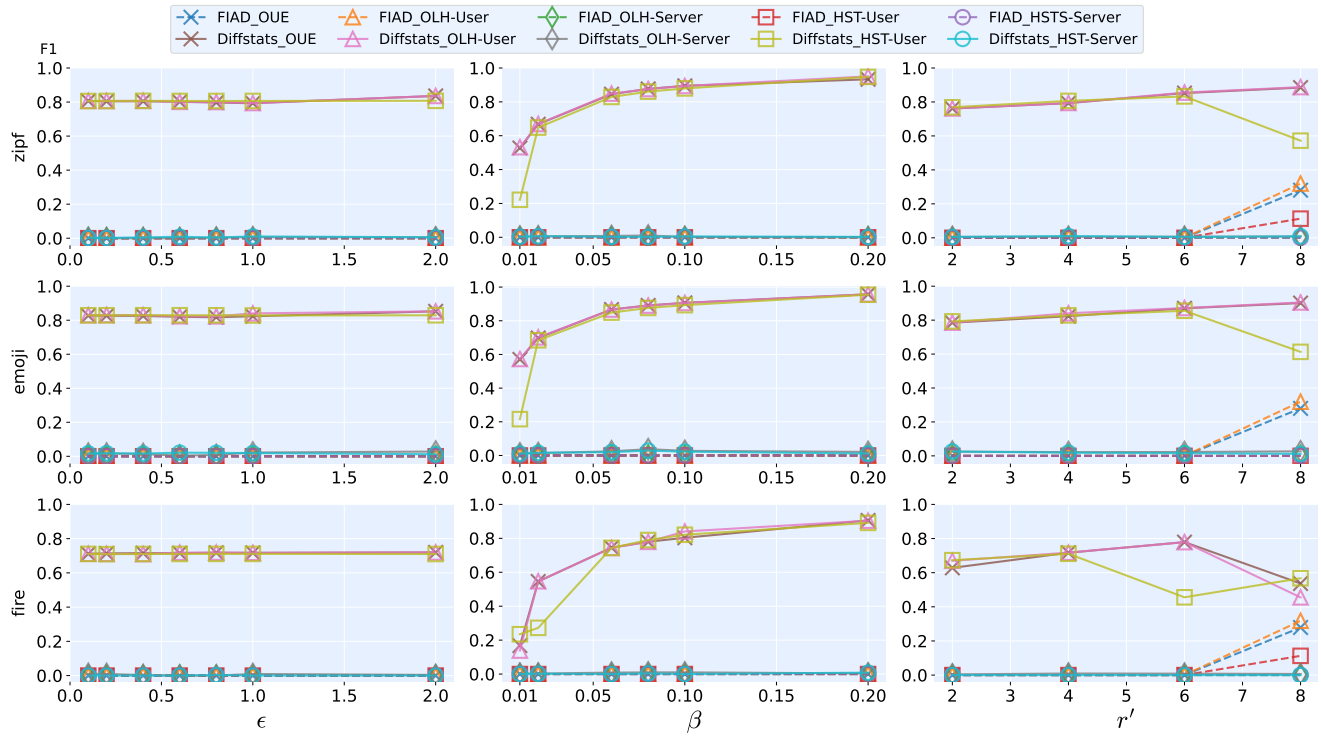

Figure 3: Performance comparison between the proposed Diffstats **and FIAD [5] against the MGA-A attack.**

existing data poisoning attacks of similar IGR and evades detection in all three datasets, i.e., $F1 = 0$ with $r' = 4$, $\epsilon \in [0.1, 1]$ and $\beta \in [0.01, 0.1]$. In addition, neither FIAD nor ours supports GRR.

Time Cost. We measure the time cost for Diffstats and FIAD in Table 1. Our experiments demonstrate up to about 120× detection speedup versus FIAD, with a 1-hour timeout[1]. Thus, Diffstats is more friendly to time-sensitive applications.

5.3 Results for ASD

We assess the detection accuracy of the proposed ASD method under challenging circumstances where malicious users cannot be identified under APA attack (Table 2), and MGA and MGA-A attacks on GRR (Table 3).

Detection for APA. Table 2 shows that our ASD can effectively detect the APA attack with various ϵ irrespective of the underlying LDP protocols. In general, a small β results in a decrease in accuracy since the attack also becomes weaker. We do not observe a performance drop until $\beta = 0.01$, which indicates the effectiveness of ASD even when the attacker controls a small portion of users in the system. The detection becomes stable with a narrower CI as β increases. Our analysis in Section 4.2.2 also shows that the performance improves with increasing n and decreases with a larger d. This is confirmed by our experiment, i.e., the best result is observed in zipf with $n = 1,000,000$ and $d = 1,024$, while the worst occurs in emoji with $n = 218,477$ and $d = 1,496$. With a small β, the detection performs better in OUE and OLH-User than in HST-User. This is

[1]37.04% of FIAD instances were not recorded due to the timeout.

Table 1: Time cost in seconds of FIAD and Diffstats**. Default $\epsilon = 1$, $r = 10$, and $\beta = 0.05$. "–" indicates passing the 1-hour timeout.**

Dataset		OUE FIAD / Ours	OLH-User FIAD / Ours	OLH-Server FIAD / Ours	HST-User FIAD / Ours	HST-Server FIAD / Ours
zipf	0.1	– / 117	– / 116	– / 103	– / 117	– / 104
	ϵ 0.5	– / 122	– / 124	– / 107	– / 115	– / 102
	1	– / 129	2,441 / 130	2,389 / 109	– / 114	– / 101
	1%	2,677 / 118	2,336 / 120	2,270 / 105	– / 106	– / 100
	β 5%	– / 128	2,368 / 131	2,357 / 107	– / 117	– / 100
	10%	– / 140	2,531 / 146	2,432 / 115	– / 129	– / 103
	1	2,586 / 123	2,190 / 126	2,326 / 111	– / 113	– / 101
	r 5	2,826 / 127	3,294 / 130	2,390 / 115	– / 119	– / 101
	10	– / 126	2,355 / 129	2,430 / 111	– / 117	– / 99
emoji	0.1	3,491 / 29	– / 28	– / 25	– / 28	– / 24
	ϵ 0.5	2,230 / 29	1,761 / 27	1,876 / 25	– / 27	– / 24
	1	1,171 / 30	1,045 / 28	1,077 / 25	– / 27	– / 24
	1%	1,206 / 27	1,062 / 27	1,075 / 25	– / 25	– / 25
	β 5%	1,169 / 28	1,030 / 30	1,056 / 24	– / 27	– / 25
	10%	3,073 / 31	1,007 / 32	1,072 / 26	– / 30	– / 23
	1	1,214 / 28	976 / 31	1,052 / 25	– / 27	– / 24
	r 5	1,187 / 29	1,037 / 30	1,070 / 25	– / 27	– / 24
	10	1,190 / 29	1,054 / 30	1,109 / 25	– / 27	– / 24
fire	0.1	1,043 / 109	– / 106	589 / 99	2,414 / 106	587 / 101
	ϵ 0.5	544 / 110	– / 112	289 / 108	1,786 / 108	592 / 99
	1	341 / 118	228 / 122	182 / 113	1,986 / 109	577 / 99
	1%	225 / 118	209 / 113	176 / 112	1,831 / 101	589 / 96
	β 5%	337 / 122	230 / 123	176 / 111	1,869 / 106	587 / 98
	10%	458 / 127	277 / 129	191 / 114	1,792 / 115	590 / 99
	1	195 / 120	170 / 122	170 / 111	557 / 106	586 / 97
	r 5	201 / 117	307 / 122	184 / 115	728 / 107	609 / 97
	10	396 / 125	238 / 124	180 / 112	1,435 / 107	616 / 98

Table 2: Detection accuracy with 95% CI of ASD against APA. We set $\beta = 0.1$ and $\epsilon = 0.5$ by default.

Dataset			OUE	OLH-User	HST-User
zipf	ϵ	0.1	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		0.5	1.00 [0.912, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		0.8	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		1	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
	β	1%	0.98 [0.88, 0.99]	0.88 [0.73, 0.95]	0.53 [0.36, 0.68]
		5%	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		7.5%	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		10%	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
emoji	ϵ	0.1	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		0.5	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		0.8	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		1	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
	β	1%	0.50 [0.33, 0.66]	0.50 [0.33, 0.66]	0.50 [0.33, 0.66]
		5%	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		7.5%	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		10%	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
fire	ϵ	0.1	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		0.5	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		0.8	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		1	1.00 [0.91, 1]	0.98 [0.88, 0.99]	1.00 [0.91, 1]
	β	1%	0.50 [0.33, 0.66]	0.65 [0.48, 0.79]	0.53 [0.36, 0.68]
		5%	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		7.5%	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]
		10%	1.00 [0.91, 1]	1.00 [0.91, 1]	1.00 [0.91, 1]

because the attack under the same setting on HST-User is weaker than the other two.

Detection for MGA and MGA-A in GRR. Similar to the detection against APA attack, Table 3 shows that ASD consistently achieves high accuracy ($\geq$ 88%) for all settings. Unlike in APA, the detection is still effective even with small β but is more subject to change of ϵ. In other words, a small ϵ results in low detection accuracy. The reason is that the variance of GRR is greater than that of other protocols, which introduces more LDP noise and negatively affects the detection performance.

Time Cost. Our experiment shows that ASD is very fast, consistently under 1*s* across all tested settings (see Table 4 and Table 5).

6 Attack Recovery of LDP Post-processing

Post-processing was previously intended to remove excessive LDP noise to boost utility in a compliant environment without attack concerns. Depending on underlying data tasks and available resources, service providers may not be able to recollect data when the ASD detection result is positive. Therefore, it is crucial to select a post-processing method that can reconstruct as many characteristics of the original data from polluted LDP estimates as possible. In this section, we empirically study the data recoverability of state-of-the-art LDP post-processing methods introduced in Section 2.1.2,

Table 3: Detection accuracy with 95% CI of ASD for GRR. We set $\beta = 0.1$, $\epsilon = 0.5$ and $r = 10$ by default.

Dataset	MGA			MGA-A		
zipf	ϵ	0.1	0.93 [0.79, 0.98]	ϵ	0.1	0.88 [0.73, 0.95]
		0.5	1.00 [0.91, 1]		0.5	1.00 [0.91, 1]
		0.8	1.00 [0.91, 1]		0.8	1.00 [0.91, 1]
		1	1.00 [0.91, 1]		1	1.00 [0.91, 1]
	β	1%	1.00 [0.91, 1]	β	1%	1.00 [0.91, 1]
		5%	1.00 [0.91, 1]		5%	1.00 [0.91, 1]
		7.5%	1.00 [0.91, 1]		7.5%	1.00 [0.91, 1]
		10%	1.00 [0.91, 1]		10%	1.00 [0.91, 1]
	r	1	1.00 [0.91, 1]	r'	2	1.00 [0.91, 1]
		2	1.00 [0.91, 1]		4	1.00 [0.91, 1]
		5	1.00 [0.91, 1]		6	1.00 [0.91, 1]
		10	1.00 [0.91, 1]		8	1.00 [0.91, 1]
emoji	ϵ	0.1	0.93 [0.79, 0.98]	ϵ	0.1	0.95 [0.83, 0.99]
		0.5	1.00 [0.91, 1]		0.5	1.00 [0.91, 1]
		0.8	1.00 [0.91, 1]		0.8	1.00 [0.91, 1]
		1	1.00 [0.91, 1]		1	1.00 [0.91, 1]
	β	1%	1.00 [0.91, 1]	β	1%	1.00 [0.91, 1]
		5%	1.00 [0.91, 1]		5%	1.00 [0.91, 1]
		7.5%	1.00 [0.91, 1]		7.5%	1.00 [0.91, 1]
		10%	1.00 [0.91, 1]		10%	1.00 [0.91, 1]
	r	1	1.00 [0.91, 1]	r'	2	1.00 [0.91, 1]
		2	1.00 [0.91, 1]		4	1.00 [0.91, 1]
		5	1.00 [0.91, 1]		6	1.00 [0.91, 1]
		10	1.00 [0.91, 1]		8	1.00 [0.91, 1]
fire	ϵ	0.1	1.00 [0.91, 1]	ϵ	0.1	1.00 [0.91, 1]
		0.5	1.00 [0.91, 1]		0.5	1.00 [0.91, 1]
		0.8	1.00 [0.91, 1]		0.8	1.00 [0.91, 1]
		1	1.00 [0.91, 1]		1	1.00 [0.91, 1]
	β	1%	1.00 [0.91, 1]	β	1%	1.00 [0.91, 1]
		5%	1.00 [0.91, 1]		5%	1.00 [0.91, 1]
		7.5%	1.00 [0.91, 1]		7.5%	1.00 [0.91, 1]
		10%	1.00 [0.91, 1]		10%	1.00 [0.91, 1]
	r	1	1.00 [0.91, 1]	r'	2	1.00 [0.91, 1]
		2	1.00 [0.91, 1]		4	1.00 [0.91, 1]
		5	1.00 [0.91, 1]		6	1.00 [0.91, 1]
		10	1.00 [0.91, 1]		8	1.00 [0.91, 1]

including Norm-Sub and Base-Cut [26] for regular no-attack scenarios and the one specifically designed for attack recovery, i.e., Normalization [5] and LDPRecover [19]. In addition, we propose a new post-processing method, *robust segment normalization* (RSN), which strikes a balance between robustness and accuracy. We first describe the design of RSN below.

6.1 Robust Segment Normalization

In Norm-Sub, Δ introduces positive bias for low frequencies and negative bias for high frequencies. This approach may reduce the frequencies of all non-target items to 0 under a substantial attack; normalization reduces the attack influences by scaling down all

Table 4: Time cost (ms) of ASD against APA. Default $\epsilon = 0.5$, $r = 10$, and $\beta = 0.1$.

Dataset			OUE	OLH-User	HST-User
zipf	ϵ	0.1	539	546	554
		0.5	375	370	356
		1	360	369	348
	β	1%	542	539	525
		5%	363	366	347
		10%	349	351	348
emoji	ϵ	0.1	592	592	615
		0.5	451	428	486
		1	416	421	411
	β	1%	597	591	735
		5%	424	411	411
		10%	411	410	411
fire	ϵ	0.1	514	557	631
		0.5	283	294	290
		1	244	249	241
	β	1%	449	454	481
		5%	274	272	287
		10%	348	318	286

frequency estimates multiplicatively and introduces significant negative errors to high frequencies.

Our RSN is inspired by existing methods and aims to control the incurred bias more precisely. We observe that the perturbation errors within high-frequency estimates $\mathcal{H}$ are relatively small compared to their true frequencies. Thus, a minimal error adjustment is needed there. On the other hand, we want to find a region $\mathcal{L}$ that contains $\tilde{C}_i$ whose corresponding frequency after LDP may become negative. We only add Δ to this region instead of the full domain in Norm-Sub to minimize the impact of bias on high-frequency estimates. Since $\tilde{C}_i \sim N(n \cdot f_i, \sigma_i^2)$ (see Section 4.2.1), we may find a f_i whose minimum perturbed count $\tilde{C}_i = 0$ via $n \cdot f_i - 2 \cdot \sigma_i = 0$. $\mathcal{L}$ can be approximated by finding the items with perturbed counts less than $n \cdot f_i + 2 \cdot \sigma_i$. RSN centers on recalibrating estimates within $\mathcal{L}$ since only those frequencies can contribute to negative values after LDP aggregation. This, therefore, reduces the impact of bias on high-frequency estimates. Δ can be derived by $\sum_{\tilde{C}_i \in \mathcal{L}} \max(\tilde{C}_i + \Delta, 0) + \sum_{\tilde{C}_j \in \mathcal{H}} \max(\tilde{C}_j, 0) = \sum \tilde{C}_i$. In other words, we only adjust low frequencies to become non-negative while keeping high-frequency estimates unchanged. To further maintain a *consistency* condition, the final result will be $\tilde{f}_i' = \frac{\tilde{C}_i}{\sum \tilde{C}_i}$. We use multiplicative rather than additive correction to avoid the case where a non-target item in $\mathcal{H}$ is subtracted to 0. In this way, RSN maintains the relative ratios among the high-frequency values, which are desired for many applications.

6.2 Evaluation

We empirically study the attack resilience and utility boost of state-of-the-art post-processing methods. The experimental setting remains the same as the detection evaluation in Section 5. The results are an average of 10 trials.

Table 5: Time cost (ms) of ASD for GRR. Default $\epsilon = 0.5$, $r = 10$, and $\beta = 0.1$.

Dataset	MGA			MGA-A		
zipf	ϵ	0.1	530	ϵ	0.1	619
		0.5	380		0.5	381
		1	372		1	378
	β	1%	530	β	1%	543
		5%	357		5%	359
		10%	355		10%	356
	r	1	621	r'	2	533
		5	363		6	362
		10	363		8	363
emoji	ϵ	0.1	568	ϵ	0.1	564
		0.5	421		0.5	418
		1	425		1	418
	β	1%	576	β	1%	572
		5%	403		5%	403
		10%	402		10%	405
	r	1	571	r'	2	581
		5	400		6	403
		10	397		8	400
fire	ϵ	0.1	469	ϵ	0.1	461
		0.5	302		0.5	303
		1	295		1	295
	β	1%	471	β	1%	542
		5%	289		5%	284
		10%	289		10%	282
	r	1	455	r'	2	473
		5	285		6	290
		10	282		8	291

6.2.1 Utility Boost after Fake User Removal. We first consider the situation where MGA and MGA-A attacks are applied to OUE, OLH and HST. The attack influence can be effectively minimized by our detection and thus the underlying data is considered to be clean. We also consider GRR without attack for a comprehensive evaluation of major CFOs. We show the results for different datasets in Figure 4.

Overall, our RSN performs best by exhibiting the lowest MSE in most settings. LDPRecover and Normalization underperform the regular post-processing approaches, which is expected. The performance of the attack-centered methods is less affected by ϵ compared to other methods whose MSE decreases as ϵ grows. It is interesting to see that for small ϵ LDPRecover performs better than Base-Cut recommended in prior work [26].

In GRR, Normalization shows lower MSE than other methods when ϵ is small, which is more obvious with emoji and fire. This is because the high variance of GRR and large noise significantly affect the accuracy of the LDP result, on which the approaches, such as Norm-Sub, rely for utility optimization, while Normalization largely ignores. This is evidenced by consistent MSE values (i.e., about 10^{-4}) of Normalization for all LDP protocols.

6.2.2 Attack Recovery. We consider the utility recoverability of the post-processing methods for the APA attack as well as MGA and MGA-A on the GRR protocol.

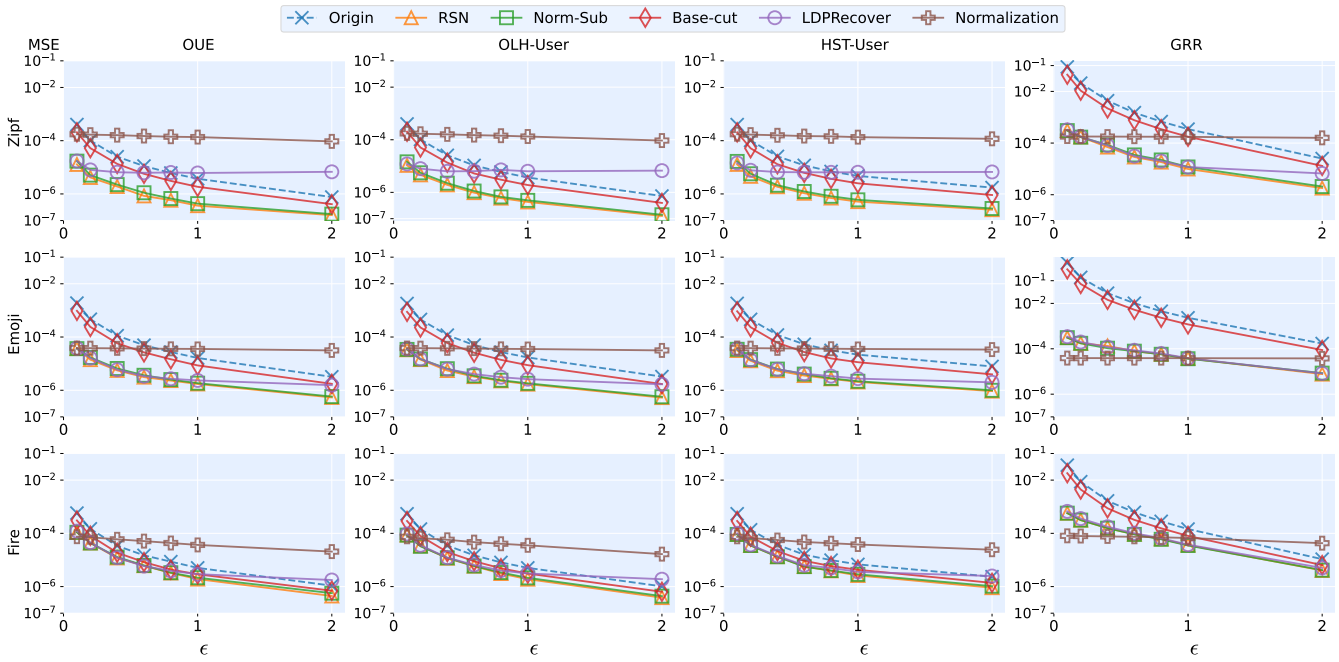

Figure 4: Utility boost of different post-processing methods without attack.

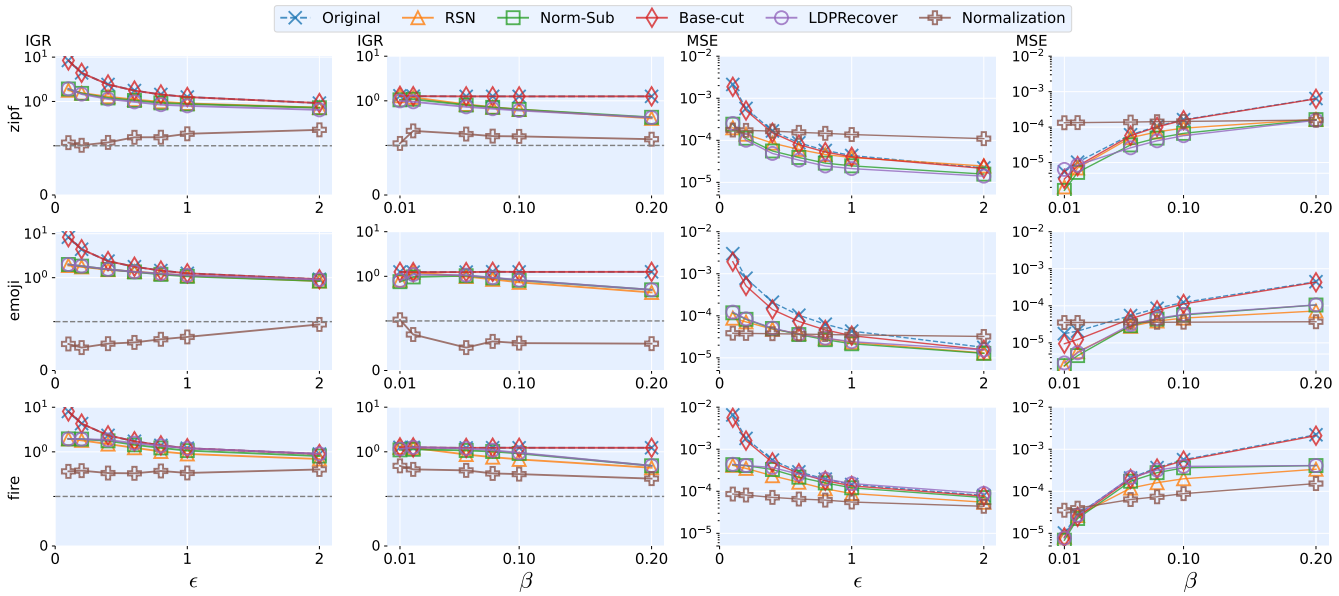

Figure 5: Utility recoverability of different post-processing methods under APA attack on OUE.

Results for APA Attack. As the results are consistent across different CFO protocols, we only discuss OUE in Figure 5 and leave others in Figure 8 and 9 in Appendix H.

For attack gain suppression, the experimental results are mixed for resilience-optimized methods. On the one hand, Normalization shows lower IGR closer to the baseline given various ϵ and β. On the other hand, LDPRecover does not show the expected advantages in reducing attack gain. Its performance is on par with that of Norm-Sub and RSN in zipf, while it underperforms RSN in emoji and fire.

For utility boosting, despite superior performance in attack gain reduction, Normalization does not produce lower MSE than other

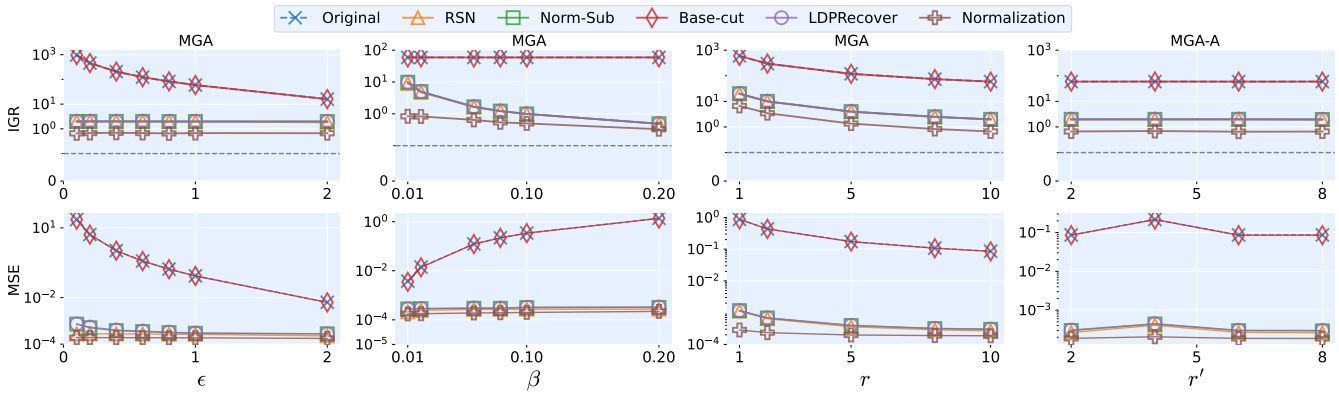

Figure 6: Utility recoverability of different post-processing methods under MGA and MGA-A on GRR in zipf.

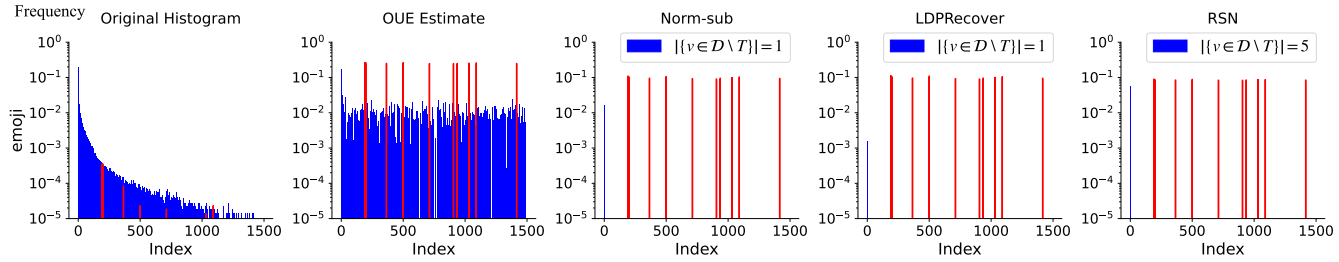

Figure 7: Comparison of histograms after Norm-Sub, LDPRecover, and RSN. The target item frequencies are highlighted in red. $\epsilon = 0.5, r = 10$ and $\beta = 0.05$.

methods due to its coarse-grained bias control. Rather, it again shows the context-agnostic nature of utility recovery with steady MSE values, rendering it an unreliable method. Since Norm-Sub, LDPRecover, and RSN are all based on the *consistency*, they exhibit similar performance in the experiment while RSN exhibits lower MSE in the fire dataset.

Results for MGA and MGA-A attack. Here we only show MGA and MGA-A with varying r' in zipf in Figure 6 due to similar results in other settings (see Figure 10 and 11 in Appendix H).

Normalization outperforms other methods in reducing attack gain in all settings for GRR. LDPRecover, Norm-Sub and RSN are similar in mitigating attack influence. Regarding MSE, all methods are close with slightly better performance of Normalization. Base-Cut shows the worst in both attack resilience and data accuracy. As r' increases in MGA-A, both IGR and MSE remain stable for all methods, because each fake user in GRR can support only one target item in their report with probability $1/r$, independent of r'.

6.2.3 Summary. It is interesting to see that the regular post-processing methods (e.g., Norm-Sub) perform similarly to state-of-the-art attack-focused methods (e.g., LDPRecover) in data recovery. It is speculated that the *consistency* condition plays a crucial role, not only benefiting utility but also attack suppression. When there is no attack or the attack influence has been minimized (e.g., fake users can be identified or in the server setting of OLH and HST), our method RSN is recommended for utility boosting. When a strong attack is present in OUE, OLH-User, and HST-User, RSN and Norm-Sub are both preferred methods for data recovery. In general, Normalization is not recommended since it is not adaptive to the underlying data and LDP settings except when large errors are expected (e.g., GRR with large d and small ϵ). Our additional experiment in Figure 7 shows that given similar MSE, RSN retains more original high-frequency items compared to Norm-Sub and LDPRecover. As a result, RSN would be a preferred post-processing method in general for balanced attack mitigation and data recovery.

7 Discussion

Extension to Heavy Hitter Identification. Prior study [5] shows that PEM protocol for heavy hitter identification [3, 4, 25] is also vulnerable to MGA attack. As a result, the target items could be falsely recognized as the top-k. Our detection methods can help identify attack behaviors. Specifically, since PEM interactively applies OLH for partial vector perturbation, we may adopt Diffstats to detect malicious users for each iteration. However, as the size of the encoded domain in PEM is smaller than that of the original item domain, it may affect the detection performance as discussed in Section 4.1.2. Likewise, ASD should also be applicable but could be less effective due to the focus on top-k items in PEM, which prevents full domain aggregation analysis.

Extension to Numerical Data. The recent work [15] recommended the SW protocol over binning-based CFOs for numerical

distribution estimation thanks to the more robust performance of SW against the data poisoning attack and better results of their proposed detection method on SW (i.e., higher AUC values for attack existence). Our research can be extended to enhance the practical robustness of the binning-based CFO protocols. Specifically, CFOs with binning apply, for example, OUE or OLH, to estimate a histogram and adopt *consistency* post-processing to enforce the distribution characteristics. Since the targeted attack in [15] is a variant of MGA, our proposed detection methods can be adapted to detect the attack and identify malicious users. Along with our post-processing method, we may significantly enhance the corrupted data recovery for vulnerable binning-based CFOs and thus provide reliable alternatives for numerical data in hostile environments.

8 Conclusion

Data poisoning attacks pose an imminent threat to current LDP implementations and urge the research community to rethink the security implications in privacy-enhancing technologies. Our newly discovered attack strategies unveil the dynamics of the threat landscape. To mitigate the emerging adversaries, we propose novel detection methods with high detection accuracy and low overhead. In addition, our new post-processing method and follow-up analysis of existing approaches reveal key design principles and highlight the importance of LDP post-processing in attack resilience, which we believe will benefit the research in the future.

9 Acknowledgments

We would like to thank the Shepherd and anonymous reviewers for their insightful comments and guidance. This paper was supported in part by NSF grants CNS-2238680, CNS-2207204, and CNS-2247794.

References

[1] 2024. Emoji Stats. https://www.emojistats.org/
[2] 2024. San Francisco Fire Department Calls for Service. http://bit.ly/336sddL
[3] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. 2017. Practical locally private heavy hitters. Advances in Neural Information Processing Systems 30 (2017).
[4] Raef Bassily and Adam Smith. 2015. Local, private, efficient protocols for succinct histograms. In Proceedings of the forty-seventh annual ACM symposium on Theory of computing. 127–135.
[5] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2021. Data poisoning attacks to local differential privacy protocols. In 30th USENIX Security Symposium. 947–964.
[6] Albert Cheu, Adam Smith, and Jonathan Ullman. 2021. Manipulation attacks in local differential privacy. In 2021 IEEE Symposium on Security and Privacy. IEEE, 883–900.
[7] Abraham De Moivre. 2020. The doctrine of chances: A method of calculating the probabilities of events in play. Routledge.
[8] John C Duchi, Michael I Jordan, and Martin J Wainwright. 2013. Local privacy and statistical minimax rates. In 2013 IEEE 54th annual Symposium on Foundations of Computer Science. IEEE, 429–438.
[9] John C Duchi, Michael I Jordan, and Martin J Wainwright. 2018. Minimax optimal procedures for locally private estimation. J. Amer. Statist. Assoc. 113, 521 (2018), 182–201.
[10] Werner Ehm. 1991. Binomial approximation to the Poisson binomial distribution. Statistics & Probability Letters 11, 1 (1991), 7–16.
[11] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. 1054–1067.
[12] Kai Huang, Gaoya Ouyang, Qingqing Ye, Haibo Hu, Bolong Zheng, Xi Zhao, Ruiyuan Zhang, and Xiaofang Zhou. 2024. LDPGuard: Defenses against data poisoning attacks to local differential privacy protocols. IEEE Transactions on Knowledge and Data Engineering (2024).
[13] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2014. Extremal mechanisms for local differential privacy. Advances in Neural Information Processing Systems 27 (2014).
[14] 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 32nd USENIX Security Symposium. 1739–1756.
[15] Xiaoguang Li, Zitao Li, Ninghui Li, and Wenhai Sun. 2025. On the Robustness of LDP Protocols for Numerical Attributes under Data Poisoning Attacks. In 32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025.
[16] Zitao Li, Tianhao Wang, Milan Lopuhaä-Zwakenberg, Ninghui Li, and Boris Škoric. 2020. Estimating numerical distributions under local differential privacy. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 621–635.
[17] Karl Pearson. 1896. VII. Mathematical contributions to the theory of evolution.—III. Regression, heredity, and panmixia. Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character 187 (1896), 253–318.
[18] 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. 192–203.
[19] Xinyue Sun, Qingqing Ye, Haibo Hu, Jiawei Duan, Tianyu Wo, Jie Xu, and Renyu Yang. 2024. LDPRecover: Recovering frequencies from poisoning attacks against local differential privacy. In 2024 IEEE 40th International Conference on Data Engineering. 1619–1631.
[20] ADP Team et al. 2017. Learning with privacy at scale. Apple Mach. Learn. J 1, 8 (2017), 1–25.
[21] Wei Tong, Haoyu Chen, Jiacheng Niu, and Sheng Zhong. 2024. Data Poisoning Attacks to Locally Differentially Private Frequent Itemset Mining Protocols. arXiv preprint arXiv:2406.19466 (2024).
[22] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. 2019. Collecting and analyzing multidimensional data with local differential privacy. In 2019 IEEE 35th International Conference on Data Engineering. IEEE, 638–649.
[23] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally differentially private protocols for frequency estimation. In 26th USENIX Security Symposium. 729–745.
[24] Tianhao Wang, Bolin Ding, Jingren Zhou, Cheng Hong, Zhicong Huang, Ninghui Li, and Somesh Jha. 2019. Answering multi-dimensional analytical queries under local differential privacy. In Proceedings of the 2019 International Conference on Management of Data. 159–176.
[25] Tianhao Wang, Ninghui Li, and Somesh Jha. 2019. Locally differentially private heavy hitter identification. IEEE Transactions on Dependable and Secure Computing 18, 2 (2019), 982–993.
[26] Tianhao Wang, Milan Lopuhaa-Zwakenberg, Zitao Li, Boris Skoric, and Ninghui Li. 2020. Locally Differentially Private Frequency Estimation with Consistency. In Proceedings of the NDSS Symposium.
[27] 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. 519–536.
[28] 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. IEEE, 317–331.

Appendix A Proof of Theorem 4.1

Proof.

$$\begin{aligned}
\mathbb{E}\left[E_{sq}^{MGA}(k)\right] &= \mathbb{E}\left[(O_{MGA}^k - Y^k)^2\right] \\
&= \mathbb{E}\left[(O_{MGA}^k)^2\right] - 2Y^k\mathbb{E}\left[O_{MGA}^k\right] + (Y^k)^2 \\
&= (\mathbb{E}\left[(O_{MGA}^k)^2\right] - \mathbb{E}\left[O_{MGA}^k\right]^2) \\
&+ (\mathbb{E}\left[O_{MGA}^k\right]^2 - 2Y^k\mathbb{E}\left[O_{MGA}^k\right] + (Y^k)^2) \\
&= Var(O_{MGA}^k) + (\mathbb{E}[O_{MGA}^k] - Y^k)^2
\end{aligned}$$

Let $W_{benign}^k \sim B(n-m,\ P(X=k))$ denote the frequency of bits set to 1 in the reports of $n-m$ benign users and W_{fake}^k be the frequency of bits set to 1 in the reports of m fake users. $W_{fake}^k = m$

if $k = l_g$ and $W^k_{fake} = 0$ if $k \neq l_g$. Thus, the observed frequency is $O^k_{MGA} = W^k_{benign} + W^k_{fake}$. Note that $Y^k = nP(X = k)$.

Given the independence between W^k_{benign} and W^k_{fake}, when $k = l_g$, we have $\mathbb{E}\left[O^k_{MGA}\right] = \mathbb{E}[W^k_{benign}]+m = (n-m)P(X = k)+m$ and $Var(O^k_{MGA}) = Var(W^k_{benign}) = (n-m)P(X = k)(1-P(X = k))$. Therefore,

$$\begin{aligned}
&\mathbb{E}\left[E^{MGA}_{sq}(k)\right] = Var(O^k_{MGA}) + (\mathbb{E}[O^k_{MGA}] - Y^k)^2 \\
&= (n-m)P(X=k)(1-P(X=k)) \\
&+ ((n-m)P(X=k) + m - nP(X=k))^2 \\
&= (n-m)P(X=k)(1-P(X=k)) + m^2 \cdot (P(X=k)-1)^2.
\end{aligned}$$

When $k \neq l_g$, since $W^k_{fake} = 0$, we have $\mathbb{E}\left[O^k_{MGA}\right] = \mathbb{E}[W^k_{benign}] = (n-m)P(X = k)$ and $Var(O^k_{MGA}) = Var(W^k_{benign}) = (n-m)P(X = k)(1-P(X = k))$. Therefore,

$$\begin{aligned}
&\mathbb{E}\left[E^{MGA}_{sq}(k)\right] = Var(O^k_{MGA}) + (\mathbb{E}[O^k_{MGA}] - Y^k)^2 \\
&= (n-m)P(X=k)(1-P(X=k)) \\
&+ ((n-m)P(X=k) - nP(X=k))^2 \\
&= (n-m)P(X=k)(1-P(X=k)) + m^2 \cdot (P(X=k))^2.
\end{aligned}$$

□

Appendix B Proof of Theorem 4.2

Proof.

$$\begin{aligned}
E_{freq}(O^k_{MGA}, Y^k) &= \sum_k \frac{(O^k_{MGA} - Y^k)^2}{Y^k} \\
&= \frac{(O^{l_g}_{MGA} - Y^{l_g})^2}{Y^{l_g}} + \sum_{\substack{k=0 \\ k\neq l_g}}^{d} \frac{(O^k_{MGA} - Y^k)^2}{Y^k}
\end{aligned}$$

According to Theorem 4.1, when $k = l_g$, we have

$$\begin{aligned}
&\frac{(O^{l_g}_{MGA} - Y^{l_g})^2}{Y^{l_g}} \\
&= \frac{m^2 \cdot (P(X=l_g)-1)^2 + (n-m)P(X=l_g)(1-P(X=l_g))}{n \cdot P(X=l_g)}.
\end{aligned}$$

When $k \neq l_g$,

$$\begin{aligned}
&\frac{(O^k_{MGA} - Y^k)^2}{Y^k} \\
&= \frac{m^2 \cdot (P(X=k))^2 + (n-m)P(X=k)(1-P(X=k))}{n \cdot P(X=k)} \\
&= \frac{m^2 \cdot P(X=k) + (n-m)(1-P(X=k))}{n}.
\end{aligned}$$

According to Eq. (3) we have

$$\begin{aligned}
&E_{freq}(O^k_{MGA}, Y^k) \\
&= \frac{m^2 \cdot (P(X=l_g)-1)^2 + (n-m)P(X=l_g)(1-P(X=l_g))}{n \cdot P(X=l_g)} \\
&+ \sum_{\substack{k=0 \\ k\neq l_g}}^{d} \frac{m^2 \cdot P(X=k) + (n-m)(1-p(X=k))}{n} \\
&= \frac{m^2}{n}\left(\frac{(P(X=l_g)-1)^2}{P(X=l_g)} + \sum_{\substack{k=0 \\ k\neq l_g}}^{d} P(X=k)\right) \\
&+ \frac{n-m}{n}\left((1-P(x=l_g) + \sum_{\substack{k=0 \\ k\neq l_g}}^{d}(1-p(X=k)))\right) \\
&= \frac{m^2}{n}\left(\frac{(P(X=l_g)-1)^2}{P(X=l_g)} + \sum_{\substack{k=0 \\ k\neq l_g}}^{d} P(X=k)\right) + \frac{(n-m)\cdot d}{n}.
\end{aligned}$$

Given $\sum_{k=0}^{d} P(X=k) = 1$, we have $\sum_{\substack{k=0 \\ k\neq l_g}}^{d} P(X=k) = 1 - P(X=l_g)$. Therefore,

$$\begin{aligned}
&E_{freq}(O^k_{MGA}, Y^k) \\
&= \frac{m^2}{n}\left(\frac{(P(X=l_g)-1)^2}{P(X=l_g)} + \sum_{\substack{k=0 \\ k\neq l_g}}^{d} P(X=k)\right) + \frac{(n-m)\cdot d}{n} \\
&= \frac{m^2}{n}\left(\frac{1}{P(X=l_g)} - 1\right) + \frac{(n-m)\cdot d}{n}.
\end{aligned}$$

□

Appendix C Proof of Theorem 4.3

Proof. For APA, the proof here is similar to that of Theorem 4.1. We have $\mathbb{E}\left[E^{APA}_{sq}(k)\right] = Var(O^k_{APA}) + (\mathbb{E}[O^k_{APA}] - Y^k)^2$.

Let $W^k_{benign} \sim B(n-m,\ P(X=k))$ denote the frequency of bits set to 1 in the reports of $n-m$ benign users and W^k_{fake} be the frequency of bits set to 1 in the reports of m fake users. For APA, $W^k_{fake} = \omega[k]$. Thus, the observed frequency is $O^k_{APA} = W^k_{benign} + W^k_{fake}$. Note that $Y^k = nP(X = k)$.

Since W^k_{benign} is independent of W^k_{fake}, for each $k \in [0, d]$, we have

$$\mathbb{E}\left[O^k_{APA}\right] = \mathbb{E}[W^k_{benign}] + \omega[k] = (n-m)P(X=k) + \omega[k]$$

and

$$Var(O^k_{APA}) = Var(W^k_{benign}) = (n-m)P(X=k)(1-P(X=k)).$$

Therefore,

$$\begin{aligned}\mathbb{E}\left[E_{sq}^{APA}(k)\right] &= \mathbb{E}\left[(O_{APA}^k - Y^k)^2\right] \\ &= Var(O_{APA}^k) + (\mathbb{E}[O_{APA}^k] - Y^k)^2 \\ &= (n-m)P(X=k)(1-P(X=k)) \\ &\quad + (m \cdot P(X=k) - \omega[k])^2.\end{aligned}$$

□

Appendix D Proof of Theorem 4.4

Proof.

$$\begin{aligned}E_{freq}(O_{APA}^k, Y^k) &= \sum_k \frac{(O_{APA}^k - Y^k)^2}{Y^k} \\ &= \sum_{k=0}^{d} \frac{(O_{APA}^k - Y^k)^2}{Y^k}\end{aligned}$$

According to Theorem 4.3,

$$\begin{aligned}&\frac{(O_{APA}^k - Y^k)^2}{Y^k} \\ &= \frac{(m \cdot P(X=k) - \omega[k])^2 + (n-m)P(X=k)(1-P(X=k))}{n \cdot P(X=k)}\end{aligned}$$

As per Eq. (3), we have

$$\begin{aligned}&E_{freq}(O_{APA}^k, Y^k) \\ &= \sum_{k=0}^{d} \frac{(m \cdot P(X=k) - \omega[k])^2 + (n-m)P(X=k)(1-P(X=k))}{n \cdot P(X=k)} \\ &= \frac{1}{n}\sum_{k=0}^{d} \frac{(m \cdot P(X=k) - \omega[k])^2}{P(X=k)} + \frac{n-m}{n}\sum_{k=0}^{d}(1-P(X=k)) \\ &= \frac{1}{n}\sum_{k=0}^{d} \frac{(m \cdot P(X=k) - \omega[k])^2}{P(X=k)} + \frac{(n-m)\cdot d}{n}.\end{aligned}$$

□

Appendix E Proof of Theorem 4.5

Proof. Given Theorems 4.1 and 4.3, when $k = l_g$ we have

$$\begin{aligned}&\mathbb{E}\left[E_{sq}^{MGA}(k)\right] - \mathbb{E}\left[E_{sq}^{APA}(k)\right] \\ &= m^2 \cdot (P(X=k) - 1)^2 - (m \cdot P(X=k) - \omega[k])^2 \\ &= m^2 \cdot P(X=k)^2 - 2m^2 \cdot P(X=k) + m^2 \\ &\quad - (m^2 P(X=k)^2 - 2m \cdot \omega[k] \cdot P(X=k) + \omega[k]^2) \\ &= -2m^2 \cdot P(X=k) + m^2 + 2m \cdot \omega[k] \cdot P(X=k) - \omega[k]^2 \\ &= (\omega[k] - m)(m \cdot (2P(X=k) - 1) - \omega[k])\end{aligned}$$

According to the De Moivre-Laplace theorem [7], we can approximate $X \sim \mathcal{N}(\mu, \sigma)$ using a normal approximation, where $\mu = dp$ and $\sigma = \sqrt{dp(1-p)}$ with a typical requirement of $dp > 5$. Note that it is common in CFOs to have a large d (e.g., $d > 100$). For $\tilde{p} = \frac{p+(d-1)q}{d}$ and $q = \frac{1}{e^\epsilon+1} \in (0, \frac{1}{2})$, $dp > 5$ is easily satisfied when $\tilde{p} \approx q$. Thus, $\mathbb{E}\left[P(X = k = l_g)\right] = \mathbb{E}\left[f(x=\mu)\right] = \frac{1}{\sigma\sqrt{2\pi}} \leq \frac{1}{\sqrt{2\pi}} \approx 0.4$. For $\omega[k = l_g] \in [0, m]$, $R = \mathbb{E}\left[E_{sq}^{MGA}(k)\right] - \mathbb{E}\left[E_{sq}^{APA}(k)\right]$ is 0 when $\omega[k = l_g] = m$; $R > 0$ when $\omega[k = l_g] < m$. □

Appendix F Proof of Theorem 4.6

Proof. Since $\tilde{C}_i \sim N(n \cdot f_i,\ \sigma_i^2)$, we can obtain , $\xi(\gamma) = \mu_i + Z_\gamma \cdot \sigma_i$ for the item i and a given confidence level γ, where $\mu_i = n \cdot f_i$ and $Z_\gamma = \Phi^{-1}(\frac{1+\gamma}{2})$ is the z-score corresponding to the confidence level γ under the standard normal distribution. This upper bound ensures that $\tilde{C}_i$ corresponding to its true frequency μ_i lies within the interval with the specified confidence level. For items with $f_i = 0$, $\tilde{C}_i$ follows $N(0, \sigma_i^2)$ and $\sigma_i^2 = \frac{n(q(1-q))}{(p-q)^2}$. As a result, the upper bound of $\mathcal{B}$ is $\xi(\gamma) = Z(\gamma) \cdot \sqrt{\frac{nq(1-q)}{(p-q)^2}}$ with confidence γ. □

Appendix G Error Analysis of ASD

We analyze the error Err in ASD below. Since $\tilde{C}_i \sim N(n \cdot f_i,\ \sigma_i^2)$, for a given $\tilde{C}_j$, its probability of coming from different frequencies f_i is

$$f(\tilde{C}_j | f_i) = \frac{1}{\sqrt{2\pi\frac{\sigma_i^2}{(p-q)^2}}} \exp\left(-\frac{(\tilde{C}_j - n \cdot f_i)^2}{2\frac{\sigma_i^2}{(p-q)^2}}\right)$$

Therefore, we can derive the error Err

$$\begin{aligned}Err &= \mathbb{E}\left[n - n \cdot \sum_{i=1}^{d} \tilde{C}_i \cdot \Pr(\tilde{C}_i > \xi(\gamma) | f_i)\right] \\ &= n - \sum_{i=1}^{d}\left[n \cdot f_i \cdot (1 - \Pr(\tilde{C}_i > \xi(\gamma)|f_i)) + \sigma_i \phi\left(\frac{\xi(\gamma) - n \cdot f_i}{\sigma_i}\right)\right] \\ &= n \cdot \sum_{i=1}^{d} f_i \cdot \Pr(\tilde{C}_i < \xi(\gamma)|f_i) - \sum_{i=1}^{d} \sigma_i \phi\left(\frac{\xi(\gamma) - n \cdot f_i}{\sigma_i}\right) \\ &= n \cdot \sum_{i=1}^{d} f_i \cdot \int_{-\infty}^{\xi(\gamma)} f(\tilde{C}_i|f_i)\, d\tilde{C}_j - \sum_{i=1}^{d} \sigma_i \phi\left(\frac{\xi(\gamma) - n \cdot f_i}{\sigma_i}\right) \\ &= n \cdot \sum_{i=1}^{d} f_i \cdot \int_{-\infty}^{\xi(\gamma)} \frac{1}{\sqrt{2\pi\frac{\sigma_i^2}{(p-q)^2}}} \exp\left(-\frac{(\tilde{C}_i - n \cdot f_i)^2}{2\frac{\sigma_i^2}{(p-q)^2}}\right) \\ &\quad - \sum_{i=1}^{d} \sigma_i \phi\left(\frac{\xi(\gamma) - n \cdot f_i}{\sigma_i}\right) \qquad (5)\end{aligned}$$

where $\phi(z)$ is the probability density function of the standard normal distribution. As the detector lacks knowledge of the genuine frequency f_i, it is challenging to compute the accurate Err.

Appendix H Additional Results of Recovery

Figures 8 and 9 present additional recovery results for OLH-User and HST-User. LDPRecover performs similarly to Norm-Sub and RSN with the zipf dataset but falls behind RSN with emoji and fire. RSN consistently demonstrates better MSE performance, particularly with fire.

Figures 10 and 11 present the complete recovery results for MGA and MGA-A attacks under the GRR protocol with all three datasets. The results are in line with the conclusion in Section 5.3.

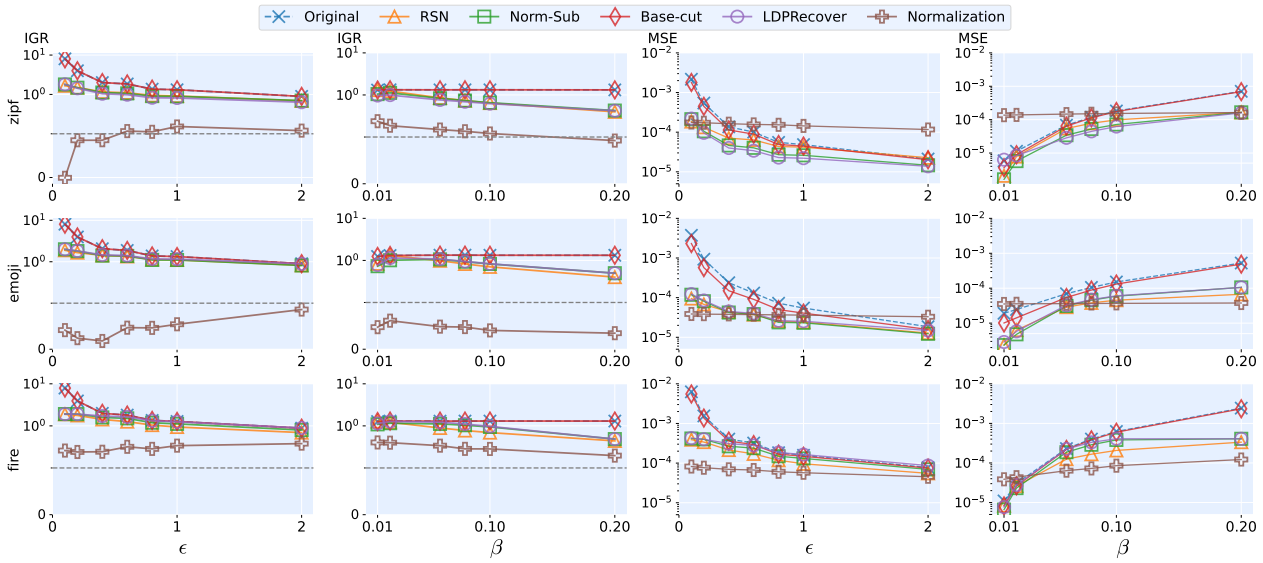

Figure 8: Utility recoverability of different post-processing methods under APA attack on OLH-User.

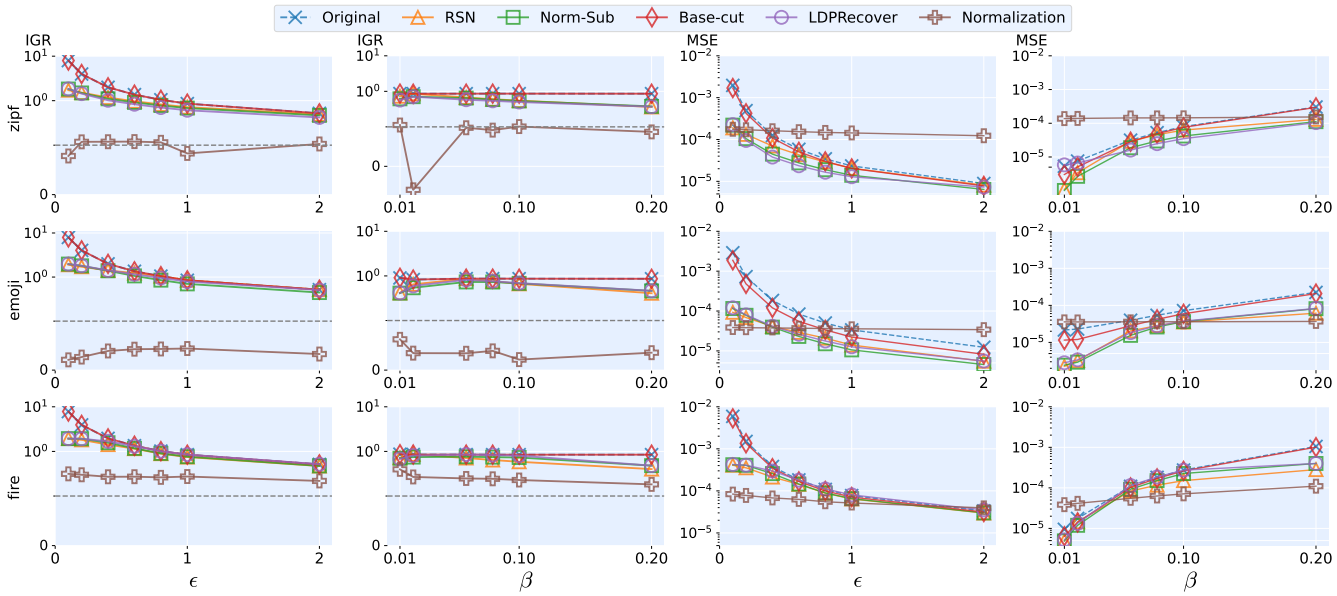

Figure 9: Utility recoverability of different post-processing methods under APA attack on HST-User.

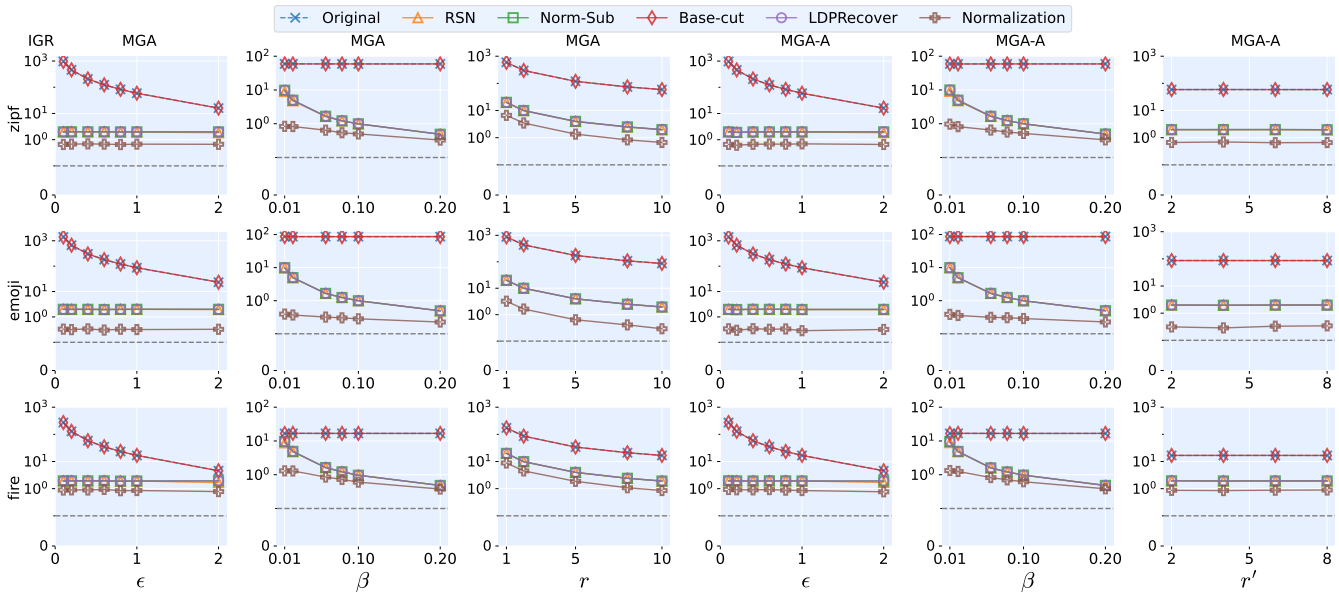

Figure 10: IGR for different post-processing methods under MGA and MGA-A attacks on GRR with emoji and fire.

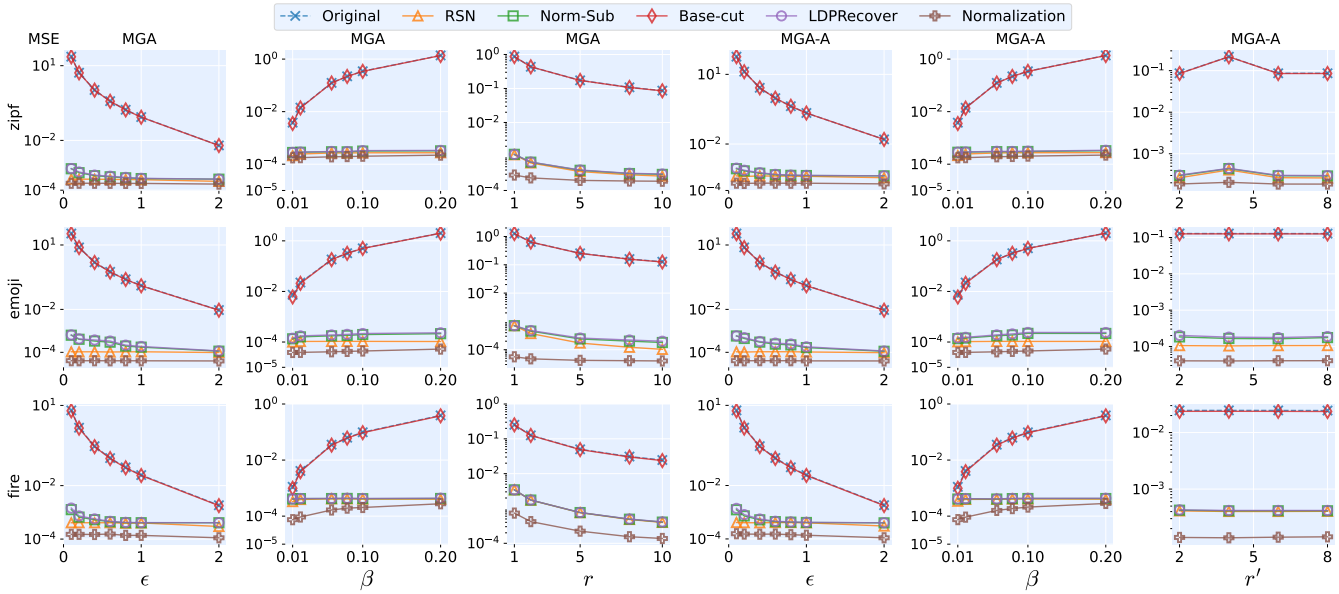

Figure 11: MSE for different post-processing methods under MGA and MGA-A attacks on GRR with emoji and fire.