

Fine-grained Manipulation Attack to Local Differential Privacy Protocols for Data Streams

Xinyu Li

Xi'an Jiaotong University
Xi'an, China
2213311049@stu.xjtu.edu.cn

Xuebin Ren

Xi'an Jiaotong University
Xi'an, China
xuebinren@mail.xjtu.edu.cn

Shusen Yang

Xi'an Jiaotong University
Xi'an, China
shusen yang@mail.xjtu.edu.cn

Liang Shi

Xi'an Jiaotong University
Xi'an, China
sl1624@stu.xjtu.edu.cn

Chia-Mu Yu

National Yang Ming Chiao Tung
University
Hsinchu, Taiwan
chiamuyu@gmail.com

Abstract

Local Differential Privacy (LDP) enables massive data collection and analysis while protecting end users' privacy against untrusted aggregators. It has been applied to various data types (e.g., categorical, numerical, and graph data) and application settings (e.g., static and streaming). Recent findings indicate that LDP protocols can be easily disrupted by poisoning or manipulation attacks, which leverage injected/corrupted fake users to send crafted data conforming to the LDP reports. However, current attacks primarily target static protocols, neglecting the security of LDP protocols in the streaming settings. Our research fills the gap by developing novel fine-grained manipulation attacks to LDP protocols for data streams. By reviewing the attack surfaces in existing algorithms, we introduce a unified attack framework with composable modules, which can manipulate the LDP estimated stream toward a target stream. Our attack framework can adapt to state-of-the-art streaming LDP algorithms with different analytic tasks (e.g., frequency and mean) and LDP models (event-level, user-level, w -event level). We validate our attacks theoretically and through extensive experiments on both synthetic and real-world datasets, and finally explore a possible defense mechanism for mitigating these attacks.

1 Introduction

Local Differential Privacy (LDP) [1–3] enables massive data collection and analytics while ensuring end-users' privacy without relying on a trusted aggregator. Due to the rigorous guarantee and easy implementation, it has been widely deployed at major companies like Google [4], Microsoft [5], and Apple [6]. Early studies on LDP focus on various static analytic tasks like frequency estimation [4, 7–12], mean/variance estimation [13], key-value data collection [14], frequent itemset mining [15–18] and graph data mining [19, 20]. Recent work proposes to adapt LDP protocols for more complicated streaming settings [21–25], which can realize continual data collection and analysis over streams. These work adopts different stream LDP models, including event-level LDP [21, 26], user-level LDP [22, 27], and w -event LDP [23]. In particular, w -event LDP can be easily extended to both event-level and (approximate) user-level LDP, thus being a popular paradigm [24].

Recently, data poisoning attacks, arising in general data community [28, 29], have also emerged as a threat to LDP protocols [30–32].

Research indicates that LDP aggregators are vulnerable to manipulated user data distributions. Cao *et al.* [30] initiated attacks on LDP frequency estimation and heavy-hitter detection by inflating target item frequencies. Wu *et al.* [31] targeted LDP for key-value data, aiming to boost both frequencies and mean values of selected keys using data from fictitious users. Li *et al.* [32] developed a fine-grained attack on LDP for mean and variance estimation, enabling precise manipulation of statistical estimates and demonstrating that a larger LDP privacy budget enhances attack effectiveness. However, these studies are all limited to *one-shot* attacks in static settings, and none of them consider the streaming data. Despite being an essential analytic setting and extensively used in various applications, the vulnerability of streaming data LDP protocols to data poisoning attacks remains unexplored.

In this work, we explore data poisoning attacks on streaming LDP protocols. Specifically, we focus on fine-grained manipulation attacks [32] of the stream of estimated statistics (e.g., frequency and mean), to persistently match a target stream of intended statistics at each timestamp, allowing for varied targets over time. We propose a novel framework to show how fake users can continuously manipulate the estimation over streams by submitting carefully crafted data to the aggregator at each timestamp, thus minimizing the overall gap between the released and intended statistics throughout the whole stream. These attacks can lead the online estimate sequence over massive users' streams close to a sequence with intended statistics. For example, a malicious company wants to manipulate seasonal consumer interest or preferences over time. It can leverage some fake users to send crafted data continuously, thus manipulating the preference trends to follow desired time-varying patterns. We identify three technical challenges of the attacks:

- **C1: Complicated optimization over streams.** Unlike poisoning attacks for static (non-streaming) LDP that only perform one-shot manipulation, attacks for streaming LDP require attackers to reconcile the correlations across timestamps and formulate attack strategies to optimize the attack performance over the whole stream. Directly applying the one-shot attacks into streaming settings cannot achieve the overall optimality of the attack.
- **C2: Fine-grained manipulation of statistics.** We focus on fine-grained manipulation of common statistics like frequency and mean estimation. On one hand, existing attacks for frequency

estimation like MGA [30] are not fine-grained with specific targets. On the other hand, existing fine-grained attacks [32] only work for mean and variance estimation in static settings, lacking consideration of frequency estimation and Challenge C1.

- **C3: Sophisticated mechanisms within LDP protocols.** Streaming LDP protocols typically consist of multiple coupled phases, choosing different submechanisms (i.e., publication or approximation) for statistical data release at different timestamps. It is challenging to design corresponding attack modules for different phases that can coordinate the choice of different strategies toward a specific target in a unified manner.

To address the above challenges, we propose a novel fine-grained manipulation attack framework against streaming LDP protocols, by formulating it as an optimization problem. We first summarize existing streaming LDP protocols into three phases with inherent attack surfaces. Based on the surfaces, we then propose two theory-driven attack modules for manipulating the publication and approximation strategies respectively, with consideration of both input and output poisoning. To coordinate the two attacks toward the overall optimization, we also propose a manipulation strategy determination module by mimicking the adaptive LDP protocol themselves. To demonstrate the effectiveness of our attack framework, we further propose both baseline and adaptive attacking algorithms against the state-of-the-art LDP protocols. Besides theoretical analysis, we conduct extensive experiments for performance evaluation. Finally, we discuss a possible defense with validations. Our contributions can be summarized as follows:

- To the best of our knowledge, we are the first to explore fine-grained data poisoning attacks against state-of-the-art LDP protocols for infinite data streams, which achieves not only arbitrary targets-driven LDP poisoning but also nearly-optimal attacking performance *on-the-fly* over the whole streams.
- We propose a general attack framework against LDP protocols for data streams, which considers both different knowledge assumptions (full and partial knowledge) and attacking modes (input and output poisoning). The unified framework with composable modules can effectively adapt to various analytical tasks (frequency and mean estimation etc) and streaming LDP models (w -event, event-level, and user-level LDP).
- We study the proposed attacks both theoretically and empirically. We present theoretical analysis of the attack performance, and discuss the sufficient conditions. We also implement all the proposed attacks and conduct experiments on both synthetic and real-world datasets. The results show that compared to the baseline attacks, our proposed attacks can achieve significant improvement in attack effectiveness. In addition, we explore a possible defense method against our attacks.

2 Background Knolwedge

We outline the background knowledge, defining Local Differential Privacy (LDP), and introduce Local Differential Privacy over Data Streams, our attacking targets.

2.1 Local Differential Privacy (LDP)

In LDP, \mathcal{M} is a randomized mechanism that perturbs each user's input v before being aggregated for data analytics.

Definition 2.1 (Local Differential Privacy). A mechanism \mathcal{M} with \mathcal{O} denotes the set of all possible outputs of \mathcal{M} satisfies ϵ -Local Differential Privacy (i.e., ϵ -LDP) if and only if, for $\forall v, v' \in \text{Dom}(\mathcal{M})$ and $\forall S \subseteq \mathcal{O}$ there is $\Pr[\mathcal{M}(v) \in S] \leq e^\epsilon \Pr[\mathcal{M}(v') \in S]$.

LDP also applies to the streaming setting with the similar definitions of event-level, user-level and w -event LDP, which are defined over different adjacency of stream prefixes. Event-level adjacency has at most one timeslot of difference while user-level adjacency can differ at all the user's contributed slots. w -neighboring means they share identical elements within a window of up to w timestamps. More details can refer to [33] and [23].

Definition 2.2. Let \mathcal{M} be a mechanism that takes as input a stream prefix $V_t = (v_1, v_2, \dots, v_t)$ consisting of an arbitrary number of consecutive input values v_t of a single user, and \mathcal{O} be the set of all possible outputs of \mathcal{M} . \mathcal{M} satisfies w -event (event-level, user-level resp.) ϵ -LDP if, for any w -neighboring (event-level, user-level adjacent resp.) stream prefixes V_t, V'_t with arbitrary t , and $\forall S \subseteq \mathcal{O}$ it holds $\Pr[\mathcal{M}(V_t) \in S] \leq e^\epsilon \Pr[\mathcal{M}(V'_t) \in S]$.

2.2 Frequency and Mean Estimation under LDP

Frequency and mean estimation are two common analytic tasks in streaming LDP. Frequency estimation is based on Frequency Oracles (FOs) [9], which estimate the frequency of any item ω_k in a domain $\Omega = \{\omega_1, \dots, \omega_d\}$ of size d . Three commonly used FO protocols are kRR [1], OUE [9] and Ada [34]. For every $j \in [n]$ user, a FO first perturbs its input item v_j to y_j and sends the output value y_j to the aggregator. Then the aggregator calculates the frequency of each distinct item k , denoted as $\hat{f}[k]$, as follows:

$$\hat{f}[k] = \frac{\frac{1}{n} \sum_{j=1}^n \mathbb{I}_{S(y_j)}^{(k)} - q}{p - q}, \quad (1)$$

where n is the total number of users, y_j is the output value from the j -th user, p and q are two perturbation parameters of FOs, and $S(y_j)$ is the support set of y_j (i.e., the input values that can produce y_j in FO). $\mathbb{I}_{S(y_j)}^{(k)}$ is a characteristic function, which is defined as 1 if $k \in S(y_j)$ and 0 otherwise. FOs provides unbiased estimation of the actual item frequencies $f[k]$, i.e., $\mathbb{E}(\hat{f}[k]) = f[k]$. [9] shows the variance of estimated frequency. For kRR, it is $\text{Var}(\hat{f}[k], \epsilon, n) = \frac{d-2+e^\epsilon}{n(e^\epsilon-1)^2} + \frac{f[k](d-2)}{n(e^\epsilon-1)}$. Since $\sum_{k=1}^d f[k] = 1$, we denote $\frac{1}{d} \sum_{k=1}^d \text{Var}(\hat{f}[k], \epsilon, n)$ as $\text{Var}(n, \epsilon) = \frac{d-2+e^\epsilon}{n(e^\epsilon-1)^2} + \frac{d-2}{nd(e^\epsilon-1)}$. For OUE, $\text{Var}(n, \epsilon) = \frac{1}{d} \sum_{k=1}^d \text{Var}(\hat{f}[k], \epsilon, n) = \frac{4e^\epsilon}{n(e^\epsilon-1)^2} + \frac{1}{nd}$. The variance of Ada is the smaller one of kRR and OUE.

Mean estimation usually adopts the Hybrid Mechanism [13], which merges Stochastic Rounding (SR) [35] and Piecewise Mechanism (PM) [13] for minimal error. For $\epsilon > 0.61$, it employs PM with probability $1 - e^{-\epsilon/2}$ and SR with $e^{-\epsilon/2}$. Below $\epsilon \leq 0.61$, it solely uses SR. Like FOs, the variance of HM can also be denoted as the function of population n and privacy budget ϵ , i.e., $\text{Var}(n, \epsilon)$.

2.3 LDP over Data Streams

2.3.1 System Models. We consider the stream data analytics, under the definition of local differential privacy. We assume there are n users and a central aggregator. At each timestamp t , each user $j \in [n]$ holds a value v_t^j from a domain Ω , either categorical $\Omega = \{\omega_1, \dots, \omega_d\}$ with the cardinality of $|\Omega| = d$ or numerical

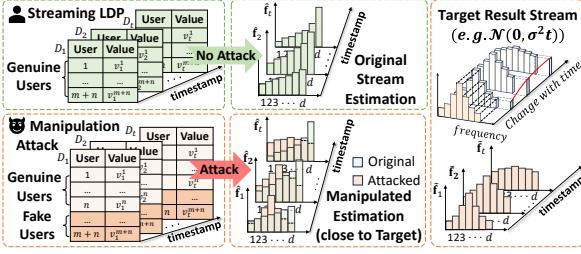


Figure 1: Manipulation to LDP Frequency Estimation over Streams.

$\Omega = [0, B]$ with a maximal differential bound B . Meanwhile, the aggregator aims to analyze the aggregate statistics (e.g., frequency or mean estimation) over all n users' data $v_t^1, v_t^2, \dots, v_t^n$ at each time t . With time evolves, each user actually has an infinite data stream $V^j = (v_1^j, v_2^j, \dots, v_t^j, \dots)$. Different LDP model (i.e., event-level, w -event LDP, user-level) can be adopted to provide specific DP guarantees to these users. Then LDP streaming analytic aims to derive a stream of aggregate statistics $\hat{f} = (\hat{f}_1, \hat{f}_2, \dots, \hat{f}_t, \dots)$ as close to the actual stream $f = (f_1, f_2, \dots, f_t, \dots)$ as possible *on-the-fly*, according to massive end users' LDP perturbed report stream. For any given stream with a length of T timestamps, it seeks to minimize the average estimate error as:

$$\min \frac{1}{d} \frac{1}{T} \sum_{t=1}^T \sum_{k=1}^d (\hat{f}_t[k] - f_t[k])^2. \quad (2)$$

In particular, two typical LDP analytic tasks can be considered: *Frequency estimation for categorical data* aims to estimate the frequency histogram $f_t = \langle f_t[1], f_t[2], \dots, f_t[d] \rangle$ over a domain Ω of size d . *Mean-value estimation for numerical data* estimates the mean value $f_t = \frac{1}{n} \sum_{j=1}^n v_t^j$ over all n users' data v_t^j in a transformed domain $[0, B]$ where B is the maximum value.

As w -event LDP can be easily extended to event-level and user-level ones, we primarily introduce the streaming protocols with w -event LDP and briefly discuss those with the other two models.

2.3.2 w -event Adaptive Budget-Division and Population-Division.

- **Private Dissimilarity Calculation.** At time t , LDP-IDS [23] invokes the FO to derive an initial estimation \hat{f}_t using budget $\epsilon_{t,1} = \frac{\epsilon}{2w}$ with whole population or population $|U_{t,1}| = \frac{n}{2w}$ with whole budget. Then, a private dissimilarity \overline{dis} can be calculated
$$\overline{dis} = \frac{1}{d} \sum_{k=1}^d (\hat{f}_t[k] - \hat{f}_{t-1}[k])^2 - \frac{1}{d} \sum_{k=1}^d \text{Var}(\hat{f}_t[k]). \quad (3)$$
- **Private Strategy Determination.** Some publication budget $\epsilon_{t,2}$ or population $|U_{t,2}|$ is assigned to derive a potential publication error err (equivalent to noise variance $\text{Var}(n, \epsilon_{t,2})$ for budget-division and $\text{Var}(|U_{t,2}|, \epsilon)$ for population-division) for possible publication strategy. Then \overline{dis} and err are compared to choose a strategy with less error. That is, if $\overline{dis} > err$, the publication is chosen; otherwise, approximation without spending budget.

- **Publication Budget Allocation.** If publication is chosen, LDP-IDS will consume $\epsilon_{t,2}$ budget with whole population or $|U_{t,2}|$ population with whole budget and release the estimates as \hat{f}_t ; otherwise, it will directly publish the last released statistics as an approximation and save budget or population for next publication. $\epsilon_{t,2}$ and $|U_{t,2}|$ is assigned based on different rules. In LDP budget distribution (LBD) and LDP population distribution (LPD), $\epsilon_{t,2}$ and $|U_{t,2}|$ is distributed in an exponentially decaying way to the timestamps when a publication is chosen. In LDP budget absorption (LBA) and LDP population absorption (LPA),

$\epsilon_{t,2}$ and $|U_{t,2}|$ is uniformly assigned as $\epsilon_{t,2} = \frac{\epsilon}{2w}$ and $|U_{t,2}| = \frac{n}{2w}$ first and then unused budget is absorbed at timestamps where approximation is chosen.

2.3.3 *Other w -event LDP Methods.* LDP-IDS also provides three baselines, LBU (LDP Budget Uniform), LPU (LDP Population Uniform), and LSP (LDP Sampling). Furthermore, RescueDP [36] and DSAT_w [37] can be extended to w -event LDP, as discussed in [23].

- **Private Dissimilarity Calculation.** DSAT_w calculates dis using budget $\epsilon_{t,1}$ with whole population or population $|U_{t,1}|$ with whole budget. RescueDP calculates feedback and PID errors without allocating budget or population. Other methods do not apply.
- **Private Strategy Determination.** They adopt a similar budget-division or population-division framework, except that DSAT_w modifies err as a dynamic threshold. LBU/LPU publishes always. LSP selects one timestamp in a window for publication and approximates other $w-1$ timestamps. RescueDP calculates the next sampling intervals according to PID error, publishes at the next sampling timestamps and approximates at other timestamps.
- **Publication Budget/Population Allocation.** DSAT_w dynamically allocates $\epsilon_{t,1}$ (or $|U_{t,1}|$) and $\epsilon_{t,2}$ (or $|U_{t,2}|$) by a PID controller. LBU/LPU allocates fixed budget ϵ/w or population n/w across timestamps. LSP allocates the entire ϵ and population at the sampling timestamp. RescueDP dynamically allocates budget or population at sampling timestamps and approximates with a Kalman filter.

2.3.4 *Event-level and User-Level Methods.* We briefly discuss event-level protocols, including ToPL [21] and (LDP extended) PeGaSus [38], and user-level protocols, including CGM [22] and (LDP extended) FAST [39].

- **Private Dissimilarity Calculation.** FAST calculates the feedback error and PID error. PeGaSus, ToPL and CGM do not apply.
- **Private Strategy Determination.** PeGaSus, ToPL and CGM publish at each timestamp. FAST calculates the next sampling interval according to PID error, publishes at next sampling timestamp and approximates the rest timestamps.
- **Publication Budget/Population Allocation.** PeGaSus allocates all privacy and population at each timestamps to satisfy event-level LDP and post-processes the estimates with Grouper and Smoother. ToPL consumes all privacy and population for mean estimation at each timestamp. FAST allocates budget for publication equally at each sampling timestamp. CGM distributes budget to one user's data stream for user-level LDP.

3 Threat Model

We detail attackers' abilities and goals in our proposed attacks.

3.1 Assumptions

We assume that the attacker can inject or compromise multiple users to manipulate the aggregator's estimation by sending crafted data. Specifically, the attacker may inject m fake users and blend them with n genuine users (or the attacker may compromise m out of $n+m$ as fake users), all participating in the streaming LDP protocols. Similar to [32], we assume a sufficient number of fake users is available, i.e., m is large enough. Existing study [40] indicates the low cost of acquiring fake accounts on platforms like Twitter, Google, and Hotmail (approximately 0.0004 – 0.03 US dollar per

Table 1: Notations.

Notation	Description
ϵ, w	Privacy budget, size of sliding windows
n, n^e	Number of genuine users, attacker-estimated n
f, f^e	Genuine distribution, attacker-estimated f
m	Number of fake users
d	Domain size of FOs
$\hat{f}_t, \hat{f}_t[k]$	Released distribution t , k -th item frequency in \hat{f}_t
$\tilde{f}_t, \tilde{f}_t[k]$	Target distribution at t , k -th item frequency in \tilde{f}_t
\bar{dis}, err	Private dissimilarity, potential publication error
$\hat{f}_t, \tilde{f}_t[k]$	Frequency for dis calculation, k -th item in \tilde{f}_t
$G_t, G_{O,t}$	Manipulation gap of IPMA/OPMA

account). Besides, we follow [32] to assume that the attacker can estimate the number of real users n as n^e from public sources. We also assume that attackers can estimate the input distribution f_t as f_t^e at every timestamp, where we further consider two major types of attackers who have different levels of knowledge about the information: *full-knowledge* and *partial-knowledge* as discussed in [41]. The former means attackers know the true statistics while the latter means attackers can compromise a subset of users and know their inputs for f_t^e estimation. A variant of the partial-knowledge is the *man-in-the-middle* (MITM) attack, where the attacker can only estimate the frequencies of items based on intercepted perturbed reports. Additionally, the attacker accesses LDP protocol parameters like ϵ and w from public documents [6] and detailed FO implementation information [30–32].

3.2 Attacker’s Goal

We consider the attacker aim to modify the estimated statistical result \hat{f}_t (e.g., frequency distribution or mean value) of streaming LDP protocols at each timestamp t , such that the estimated result stream $\hat{f} = \{\hat{f}_1, \hat{f}_2, \dots, \hat{f}_T\}$ to be as close to the target result stream $\tilde{f} = \{\tilde{f}_1, \tilde{f}_2, \dots, \tilde{f}_T\}$ as possible. Fig. 1 illustrates the attack scenario for frequency estimation. As shown, the attacker can set a target frequency distribution (e.g., $\tilde{f}_t = N(0, \sigma^2 t)$) at each timestamp, which forms a target stream. Without attack, the streaming LDP protocol continuously releases the estimated frequency histogram from the genuine users at each timestamp. Once attacked by some fake users, the released histogram is manipulated to be close to the target at each timestamp. To measure how successful the fine-grained attack is at each timestamp, we define the *manipulation gap* $G_t = \sum_{k=1}^d (\hat{f}_t[k] - \tilde{f}_t[k])^2 / d$ as the distance between the estimated result \hat{f}_t and the target result \tilde{f}_t at timestamp t . Then, average manipulation gap $G_{avg} = \frac{1}{T} \sum_{t=1}^T G_t$ can be used as the performance metric over the whole stream. A smaller gap G_{avg} implies a more successful fine-grained attack. For any given streams of T timestamps and d -dimensional statistics, the attacker seeks to minimize the average gap between them

$$\min \frac{1}{d} \frac{1}{T} \sum_{t=1}^T \sum_{k=1}^d (\hat{f}_t[k] - \tilde{f}_t[k])^2. \quad (4)$$

Note that, the target \tilde{f}_t changes by time t , manipulating the post-attack estimated result to follow specific time-varying patterns. Table 1 summarizes key notations.

4 Attack Framework and Modules

4.1 Overview

Attack Surfaces. As introduced in Sec. 2.3, existing streaming LDP protocols can be summarized into three phases, shown in Fig. 2(a), which also leaves surfaces for manipulation attacks.

(1) **Private Dissimilarity Calculation:** This step estimates a dissimilarity value \bar{dis} via FOs to measure the stream change at the current timestamp, which is compared with the potential publication error err in the following step of private strategy determination. Therefore, the attacker can manipulate \bar{dis} to control the private strategy determination, which can be leveraged to amplify the attack. Specifically, the dissimilarity \bar{dis} is calculated based on \tilde{f}_t which is derived by FOs, and can be manipulated to be increased or decreased.

(2) **Private Strategy Determination:** This step tries to select between publication and approximation at each timestamp to adaptively minimize the total noise introduced over the non-deterministic data stream, as illustrated in Sec. 4.4.2. The attacker also need a guideline to manipulate the strategy choice at each timestamp to minimize the overall gap throughout the stream. Interestingly, we found that the optimization problem of minimizing the average manipulation gap is very similar to that of minimizing the total noise in streaming LDP protocols. Therefore, the attacker can mimic the optimization strategy in streaming LDP protocols to adaptively manipulate the strategy choice in fine-grained attacks. Specifically, the manipulation gap of publication or approximation at each timestamp is compared for greedily choosing a better strategy.

(3) **Publication Budget/Population Allocation:** This step distributes the LDP budget/population and invokes FOs for the publication strategy. The attacker cannot directly affect the distribution process of publication budget/population since it is independent of user data. However, the invoked FOs can be manipulated to make the estimate \hat{f}_t at publication timestamps close to the target \tilde{f}_t . Note that, despite no direct manipulation on approximation timestamps, the manipulated result at a publication timestamp would be set as the approximate value on later approximation timestamps.

Attack Modules. Thus, we introduce three attack modules: Publication Manipulation Attack (PMA), Dissimilarity Manipulation Attack (DMA), and Manipulation Strategy Determination (MSD). At each timestamp, the attacker performs these modules to mount the attack. Specifically, based on current knowledge, the attacker first adopts MSD to determine whether publication or approximation is more beneficial in reducing the manipulation gap between \hat{f}_t and \tilde{f}_t . If MSD chooses the publication strategy, the attacker will invoke DMA to manipulate \bar{dis} and steer the LDP aggregator to choose publication. Otherwise, DMA will minimize \bar{dis} to make the aggregator choose the approximation strategy. If the publication strategy is chosen as expected, PMA will be further invoked to manipulate the released statistics \hat{f}_t to approach the target \tilde{f}_t . Fig. 2(b) summarizes the attack modules and interactions with the exposed attack surfaces of the adaptive streaming LDP protocols. Note that, these three attack modules can be applied to any streaming LDP algorithms with the aforementioned attack surfaces. For DMA and PMA, we also consider both input and output manipulation methods.

In what follows, we introduce PMA and DMA in Sec. 4.2 and Sec. 4.3, respectively. Then we design MSD in Sec. 4.4.

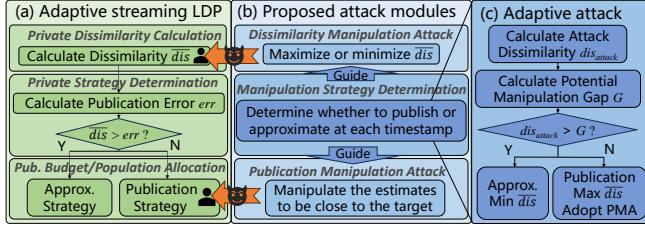


Figure 2: Overview of adaptive attack framework: (a) attack surfaces; (b) proposed attack modules; (c) proposed adaptive attack strategy.

4.2 Publication Manipulation Attacks (PMAs)

4.2.1 The Goal of PMAs. PMAs aim to manipulate the estimated statistic \hat{f}_t (e.g., frequency and mean estimate) via FOs to be as close to a target \tilde{f}_t as possible (at any timestamp t), i.e., minimizing the manipulation gap G_t . It can be formulated as follows.

$$\min G_t = \frac{1}{d} \sum_{k=1}^d (\hat{f}_t[k] - \tilde{f}_t[k])^2 \quad (5)$$

Eq. (5) aligns with our main objective (Eq. (4)), allowing the attacker to precisely target a specific distribution \tilde{f}_t with PMAs to manipulate LDP estimates at publication timestamps. Note that existing poisoning or manipulation attacks cannot directly apply here. Maximum gain attack (MGA) [30] works for frequency estimation but is not fine-grained to precisely manipulate each item, while existing fine-grained attacks [32] are only proposed for mean and variance estimation.

4.2.2 Input Publication Manipulation Attack (IPMA). We first introduce the Input Publication Manipulation Attack (IPMA), aimed at modifying \hat{f}_t by adjusting the inputs from fake users to approximate the target distribution.

IPMA can minimize the manipulation gap G_t by crafting the input values for the m controllable fake users. According to the FO protocols, once the input distribution f_t of $n+m$ total users' data can be manipulated to be close to the target distribution \tilde{f}_t , the estimates \hat{f}_t derived from FOs will also approach \tilde{f}_t . Therefore, the optimization in Eq. (5) can be transformed into the minimization of the gap between f_t and \tilde{f}_t . Besides, for $\forall k \in [1, \dots, d]$, suppose $m[k]$ represents the number of fake users whose input is the k -th item $\omega_k \in \Omega$, then the true frequency $f_t[k] \forall k \in [1, \dots, d]$ can be estimated as $f_t[k] \approx (m[k] + n^e \cdot f_t^e[k]) / (m + n^e)$. Specifically, we formulate IPMA as finding $m[k] \forall k \in [1, \dots, d]$ such that $f_t[k] = \tilde{f}_t[k]$ and the attacker can determine $m[k]$ for different k by solving the following convex optimization problem.

$$\begin{aligned} \min \frac{1}{d} \sum_{k=1}^d (\hat{f}_t[k] - \tilde{f}_t[k])^2 &= \frac{1}{d} \sum_{k=1}^d \left(\frac{m[k] + n^e \cdot f_t^e[k]}{n^e + m} - \tilde{f}_t[k] \right)^2 \\ \text{s.t. } \sum_{k=1}^d m[k] &= m, 0 \leq m[k] \leq m \end{aligned} \quad (6)$$

And the input values of fake users at timestamp t can be set as an arbitrary combination of $[v_1, \dots, v_m]$ ($\forall j : v_j \in \Omega = \{\omega_1, \dots, \omega_d\}$) that satisfies $\sum_{j=1}^m \mathbb{I}_{S(v_j)}^{(k)} = m[k], \forall k \in [1, \dots, d]$ where $\mathbb{I}_{S(v_j)}^{(k)}$ equals to 1 if $v_j = \omega_k$ and 0 otherwise.

Analysis of IPMA. Here we analyze the properties and sufficient conditions of IPMA.

THEOREM 4.1. *The expected manipulation gap for IPMA $G_{I,t}$ can be calculated as*

$$\begin{aligned} G_{I,t}(m, n, \epsilon) &= \mathbb{E} \left(\frac{1}{d} \sum_{k=1}^d (\hat{f}_t[k] - \tilde{f}_t[k])^2 \right) \\ &= \frac{1}{d} \sum_{k=1}^d \left(\frac{m^*[1] + n \cdot f_t[k]}{n + m} - \tilde{f}_t[k] \right)^2 + \text{Var}(n + m, \epsilon), \end{aligned} \quad (7)$$

where $(m^*[1], m^*[2], \dots, m^*[d])$ is the solution of the above optimization problem, $\text{Var}(n + m, \epsilon) = \frac{1}{d} \sum_{k=1}^d \text{Var}(\hat{f}_t[k])$ and ϵ is the privacy budget of the attacked FO.

PROOF. See Appendix A.1. \square

COROLLARY 4.2. *When ϵ is larger, $G_{I,t}(m, n, \epsilon)$ becomes smaller, implying a more successful attack.*

PROOF. See Appendix A.2. \square

Also, the relation between $G_{I,t}(m, n, \epsilon)$ and the number of users n and m could be illustrated as follows.

COROLLARY 4.3. *For $\forall \alpha > 1$, $G_{I,t}(m, n, \epsilon) > G_{I,t}(\alpha m, \alpha n, \epsilon)$.*

PROOF. See Appendix A.3. \square

Corollaries 4.2 and 4.3 show the relationship between the manipulation gap $G_{I,t}$ and the budget ϵ or population n .

COROLLARY 4.4. *For $\forall k \in [d]$, if the attacker has*

$$m \geq \max \left(\frac{n \cdot f_t[k]}{\tilde{f}_t[k]} - n, \frac{n \cdot \tilde{f}_t[k] - n \cdot f_t[k]}{1 - \tilde{f}_t[k]} \right), \quad (8)$$

fake users and the accurate knowledge of n and f_t , i.e., $n^e = n$ and $f_t^e = f_t$, then the expected manipulation gap is

$$G_{I,t}^*(m, n, \epsilon) = \text{Var}(n + m, \epsilon). \quad (9)$$

PROOF. See Appendix A.4. \square

4.2.3 Output Publication Manipulation Attack (OPMA). We then present the Output Publication Manipulation Attack (OPMA), which utilizes the LDP implementation details to manipulate the outputs sent to the aggregator directly.

We consider the attacker targets the FO at a publication timestamp t with privacy budget ϵ . The n real users' output are denoted as (y_1, y_2, \dots, y_n) , and the m fake users' outputs (z_1, z_2, \dots, z_m) . We use $m[k]$ to denote the number of fake users who send the k -th item to the aggregator (when using kRR) or who set the k -th bit of the encoding binary strings as 1 (when using OUE). That is, $m[k] = \mathbb{E} \left(\sum_{j=1}^m \mathbb{I}_{S(z_j)}^{(k)} \right)$. Using FOs' unbiased estimation, there is

$$\sum_{j=1}^n \mathbb{E} \left(\mathbb{I}_{S(y_j)}^{(k)} \right) = n (f_t[k] (p - q) + q) \approx n^e (f_t^e[k] (p - q) + q),$$

where p and q are FO parameters. So, the expected estimate $\mathbb{E}(\hat{f}_t[k])$ after the attack can be calculated as

$$\mathbb{E} \left(\frac{\sum_{j=1}^n \mathbb{I}_{S(y_j)}^{(k)} + \sum_{j=1}^m \mathbb{I}_{S(z_j)}^{(k)} - q (m + n^e)}{(m + n^e) (p - q)} \right) = \frac{A_k + m[k]}{(p - q) (m + n^e)},$$

where $A_k = n^e (p - q) f_t^e[k] - mq$.

Also, $\mathbb{E}(\frac{1}{d} \sum_{k=1}^d (\hat{f}_t[k] - \tilde{f}_t[k])^2)$ can be computed by

$$\frac{1}{d} \sum_{k=1}^d \left(\mathbb{E}(\hat{f}_t[k]) - \tilde{f}_t[k] \right)^2 + \frac{1}{d} \sum_{k=1}^d \text{Var}(\hat{f}_t[k]).$$

However, unlike IPMA perturbs all inputs (including those of fake users), only the input of genuine users is perturbed in FOs. Considering the estimate variance

$\text{Var}(\hat{\mathbf{f}}_t[k]) = \text{Var}\left(\frac{\sum_{j=1}^n \mathbb{I}_{S(y_j)}^{(k)} + \sum_{j=1}^m \mathbb{I}_{S(z_j)}^{(k)} - q(m+n^\epsilon)}{(m+n^\epsilon)(p-q)}\right)$ is usually small when there are a large number of users. In OPMA, we focus on minimizing the first term, i.e. $\min \frac{1}{d} \sum_{k=1}^d (\mathbb{E}(\hat{\mathbf{f}}_t[k]) - \tilde{\mathbf{f}}_t[k])^2$

There is $0 \leq m[k] \leq m$, and $\sum_{k=1}^d m[k] = m$ when kRR is used as the FO and only $0 \leq m[k] \leq m$ when using OUE. To simplify the computation, we try to minimize $\frac{1}{d} \sum_{k=1}^d |\mathbb{E}(\hat{\mathbf{f}}_t[k]) - \tilde{\mathbf{f}}_t[k]|$, instead of $\frac{1}{d} \sum_{k=1}^d (\mathbb{E}(\hat{\mathbf{f}}_t[k]) - \tilde{\mathbf{f}}_t[k])^2$. Specifically, there is

$$\sum_{k=1}^d |\mathbb{E}(\hat{\mathbf{f}}_t[k]) - \tilde{\mathbf{f}}_t[k]| = \frac{\sum_{k=1}^d |m[k] + C_k|}{(p-q)(m+n^\epsilon)},$$

where $C_k = A_k - \tilde{\mathbf{f}}_t[k](p-q)(m+n^\epsilon)$. Particularly, this simplifies the problem into a linear optimization problem, solvable in polynomial time [42].

Let $u_k = \frac{|m[k]+C_k|+(m[k]+C_k)}{2}$ and $v_k = \frac{|m[k]+C_k|-(m[k]+C_k)}{2}$, so the minimization of $\sum_{k=1}^d |m[k] + C_k|$ could be converted to

$$\min \sum_{k=1}^d u_k + v_k \quad \text{s.t. } C_k \leq u_k - v_k \leq m + C_k. \quad (10)$$

When attacking kRR, u_k, v_k also satisfy $\sum_{k=1}^d (u_k - v_k - C_k) = m$. We use HiGHS [42] to solve the optimization problem. After obtaining u_k and v_k , we calculate $m[k]$ as $u_k - v_k - C_k$.

Analysis of OPMA. Here we also analyze the properties and sufficient conditions of OPMA.

THEOREM 4.5. *The expected manipulation gap for OPMA $G_{O,t}$ can be calculated as*

$$\begin{aligned} G_{O,t}(m, n, \epsilon) &= \mathbb{E}\left(\frac{1}{d} \sum_{k=1}^d (\hat{\mathbf{f}}_t[k] - \tilde{\mathbf{f}}_t[k])^2\right) \\ &= \frac{1}{d} \sum_{k=1}^d \left(\frac{n f_t[k](p-q) + m^*[k] - mq}{(m+n)(p-q)} - \tilde{\mathbf{f}}_t[k] \right)^2 + \frac{n^2 \cdot \text{Var}(n, \epsilon)}{(m+n)^2}, \end{aligned} \quad (11)$$

where $(m^*[1], m^*[2], \dots, m^*[d])$ is the solution of the above optimization problem, p, q are parameters of FO and ϵ is the privacy budget of the attacked FO.

PROOF. See Appendix A.5. \square

COROLLARY 4.6. *For $\forall k \in [d]$, if the attacker has*

$$m \geq \max\left(\frac{n \tilde{\mathbf{f}}_t[k] - n f_t[k]}{\frac{1-q}{p-q} - \tilde{\mathbf{f}}_t[k]}, \frac{n f_t[k] - n \tilde{\mathbf{f}}_t[k]}{\frac{q}{p-q} + \tilde{\mathbf{f}}_t[k]}\right). \quad (12)$$

fake users and the accurate knowledge of n and f_t , i.e., $n^\epsilon = n$ and $\mathbf{f}_t^\epsilon = \mathbf{f}_t$, then the expected manipulation gap is

$$G_{O,t}^*(m, n, \epsilon) = \left(\frac{n}{m+n}\right)^2 \text{Var}(n, \epsilon) = \frac{n}{m+n} \text{Var}(n+m, \epsilon). \quad (13)$$

PROOF. See Appendix A.6. \square

For OPMA, we also explore its relation between manipulation gap and privacy budget ϵ .

COROLLARY 4.7. *When (12), $n^\epsilon = n$ and $\mathbf{f}_t^\epsilon = \mathbf{f}_t$ holds, $G_{O,t}$ becomes smaller as ϵ becomes larger, implying more successful attack.*

PROOF. See Appendix A.7. \square

Unlike IPMA, OPMA's optimization and $m^*[k]$ depend on ϵ , making it impossible to determine the relationship between $G_{O,t}(m, n, \epsilon)$ and ϵ without enough fake users. However, the impact of user numbers n and m on attack effectiveness is similar in OPMA and IPMA.

COROLLARY 4.8. *For $\forall \alpha > 1$, $G_{O,t}(m, n, \epsilon) > G_{O,t}(\alpha m, \alpha n, \epsilon)$.*

PROOF. See Appendix A.8. \square

Corollaries 4.7 and 4.8 also show the relation between the manipulation gap and the ϵ (or population) used when attacking adaptive streaming LDP protocols.

The manipulation gaps of PMAs, regardless of OPMA or IPMA, decrease with the increase of $m+n$ and/or ϵ . This implies the similarity between PMAs and FOs. For FOs, a larger ϵ or population n means a better estimation with a smaller variance. Similarly, for PMAs, a larger ϵ or population $m+n$ also means a better attack with a smaller manipulation gap. This inspires us that the strategy for minimizing the estimation error of streaming LDP protocols (i.e., Eq (2)), can be referenced for achieving our attack goal (i.e., Eq (4)). We will detail it in Sec. 4.4.

Based on the above introduction, one naive idea is manipulating the FOs at each publication timestamp by PMAs. However, it fails to achieve the optimal. Consider a simple situation where target $\tilde{\mathbf{f}}_t$ keeps as a constant across timestamps and there are enough fake users with accurate knowledge $n^\epsilon = n$, $\mathbf{f}_t^\epsilon = \mathbf{f}_t$. Then, at publication timestamps, PMAs can achieve the manipulation gap as Eq. (9) using IPMA or Eq. (13) using OPMA. At approximation timestamps, the manipulation gap equals to that of the last publication manipulation gap. In such a case, the optimal attack strategy is to find the minimal manipulation gap and try to keep it approximate the timestamps after it. Recall that a larger budget or population helps PMAs to achieve a smaller manipulation gap. Thus, PMAs achieve the optimal when one publication timestamp has the largest budget or population that the protocol can allocate and the rest timestamps directly approximate it. Unfortunately, PMAs can not control the strategy determination within LDP protocol. Specifically, the strategy is chosen according to the calculated dissimilarity dis , which depends on $\tilde{\mathbf{f}}_t$ and is uncontrolled by PMAs.

4.3 Dissimilarity Manipulation Attacks (DMAs)

4.3.1 The Goal of DMA. The adaptive streaming LDP protocol selects a per-timestamp strategy by comparing dissimilarity and potential publication error err . The aggregator calculates err using user population and privacy budget. For dissimilarity, it first aggregates users' LDP data to estimate $\tilde{\mathbf{f}}_t[k]$, then computes dissimilarity as $\overline{dis} = \frac{1}{d} \sum_{k=1}^d (\tilde{\mathbf{f}}_t[k] - \tilde{\mathbf{f}}_{t-1}[k])^2 - \frac{1}{d} \sum_{k=1}^d \text{Var}(\tilde{\mathbf{f}}_t[k])$. This data aggregation allows attackers to manipulate $\tilde{\mathbf{f}}_t[k]$.

The objective of DMAs is to manipulate \overline{dis} via $\tilde{\mathbf{f}}_t[k]$ to steer LDP protocol towards a desired strategy for enhanced attack outcomes. If attackers want the protocol to choose the publication strategy, they should increase the \overline{dis} (max \overline{dis}) so it surpasses the publication error. Conversely, to trigger the approximation strategy, they should reduce \overline{dis} (min \overline{dis}) to make it lower than the publication error. This manipulation allows attackers to control the protocol's Private Strategy Determination phase. And the goal of DMAs can be formulated as follows:

$$\max/\min \frac{1}{d} \sum_{k=1}^d (\tilde{\mathbf{f}}_t[k] - \tilde{\mathbf{f}}_{t-1}[k])^2 - \frac{1}{d} \sum_{k=1}^d \text{Var}(\tilde{\mathbf{f}}_t[k]). \quad (14)$$

4.3.2 Input Dissimilarity Manipulation Attack (IDMA). We also introduce the Input Dissimilarity Manipulation Attack (IDMA) first. Similar to the idea of manipulating the distribution of fake users' inputs of FOs in IPMA, IDMA can adjust the input distribution of FOs to either maximize or minimize the estimated dissimilarity between

timestamps. According to [23], dissimilarity is an unbiased estimate of the genuine dissimilarity $dis = \frac{1}{d} \sum_{k=1}^d (\mathbf{f}_t[k] - \hat{\mathbf{f}}_{t-1}[k])^2$. The attacker can manipulate $\mathbf{f}_t[k]$ to either align closely or deviate significantly from $\hat{\mathbf{f}}_{t-1}$, by controlling the number of fake users $m[k]$ per k -th element, to manipulate dis . This can be formulated as the following optimization problem.

$$\begin{aligned} & \max / \min \frac{1}{d} \sum_{k=1}^d (\mathbf{f}_t[k] - \hat{\mathbf{f}}_{t-1}[k])^2 \\ & \text{s.t. } \sum_{k=1}^d m[k] = m, 0 \leq m[k] \leq m \end{aligned} \quad (15)$$

where $\mathbf{f}_t[k]$ approximates to $\frac{m[k] + n^e \mathbf{f}_t^e[k]}{m + n^e}$.

In Eq. (15), the minimization of the estimated dissimilarity can be solved similarly as Eq. (6). In particular, the attacker only has to replace $\tilde{\mathbf{f}}_t[k]$ with $\hat{\mathbf{f}}_{t-1}[k]$. On the other hand, the maximization of the estimated dissimilarity, we use the following theorem.

THEOREM 4.9. Assume $k^* = \arg \max_{k \in [1, \dots, d]} (n^e \mathbf{f}_t^e[k] / (m + n^e) - \hat{\mathbf{f}}^{t-1}[k])$, the maximum of

$$\frac{1}{d} \sum_{k=1}^d (\mathbf{f}_t[k] - \tilde{\mathbf{f}}_t[k])^2 = \frac{1}{d} \sum_{k=1}^d \left(\frac{m[k] + n^e \cdot \mathbf{f}_t^e[k]}{m + n^e} - \tilde{\mathbf{f}}_t[k] \right)^2$$

is reached when $m[k^*] = m$ and $m[k] = 0$ ($k \neq k^*$).

PROOF. See Appendix A.9. \square

4.3.3 Output Dissimilarity Manipulation Attack (ODMA). IDMA can evolve into Output Dissimilarity Manipulation Attack (ODMA), where fake values are sent directly to the aggregator to control dissimilarity. According to [23], dissimilarity at time t is $\overline{dis} = \frac{1}{d} \sum_{k=1}^d (\mathbf{f}_t[k] - \hat{\mathbf{f}}_{t-1}[k])^2 - \frac{1}{d} \sum_{k=1}^d \text{Var}(\mathbf{f}_t[k])$. The aggregator computes the variance term as a constant $\text{Var}(n, \epsilon)$. To manipulate dissimilarity, only the first term is adjusted, i.e.,

$$\max / \min \mathbb{E} \left(\frac{1}{d} \sum_{k=1}^d (\tilde{\mathbf{f}}_t[k] - \hat{\mathbf{f}}_{t-1}[k])^2 \right). \quad (16)$$

OPMA can be adapted as ODMA to reduce dissimilarity by substituting $\tilde{\mathbf{f}}_t[k]$ with $\hat{\mathbf{f}}_{t-1}[k]$. For maximizing dissimilarity, we use the following theorem.

THEOREM 4.10. Assume $k^* = \arg \min_{k \in [1, \dots, d]} (\hat{\mathbf{f}}_{t-1}[k])$, the lower bound of

$$\begin{aligned} & \mathbb{E} \left(\frac{1}{d} \sum_{k=1}^d (\tilde{\mathbf{f}}_t[k] - \hat{\mathbf{f}}_{t-1}[k])^2 \right) \\ &= \frac{1}{d} \sum_{k=1}^d \left(\frac{n \mathbf{f}_t[k] (p - q) + m[k] - mq}{(m+n)(p-q)} - \hat{\mathbf{f}}_{t-1}[k] \right)^2 \\ &+ \frac{1}{d} \sum_{k=1}^d \text{Var} \left(\frac{\sum_{j=1}^n \mathbb{I}_{S(y_j)}^{(k)} + \sum_{j=1}^m \mathbb{I}_{S(z_j)}^{(k)} - q(m+n)}{(m+n)(p-q)} \right) \end{aligned}$$

can be maximized when $m[k^*] = m$ and $m[k] = 0$ ($k \neq k^*$), where (y_1, y_2, \dots, y_n) are real users' outputs, (z_1, z_2, \dots, z_m) are fake users' outputs and $m[k] = \mathbb{E}(\sum_{j=1}^m \mathbb{I}_{S(z_j)}^{(k)})$.

PROOF. See Appendix A.10. \square

According to Theorem 4.10, to maximize the dissimilarity, the attacker can first identify the least frequent item (denoted as k^* -th item) in the last release $\hat{\mathbf{f}}_{t-1}$, and then control all the fake users to send the data representing k^* -th item to the aggregator, i.e., send k^* -th item when using kRR or set the k^* -th bit of the encoding binary strings as 1 and other bits as 0 when using OUE.

4.4 Manipulation Strategy Determination

PMAs and DMAs provide composable modules for manipulating the per-timestamp estimate processes in streaming LDP protocols. With them, we first propose two baselines applicable to two special target streams.

4.4.1 Baseline Strategy for Special Targets

Baseline 1: Sampling Attack. Let us reconsider a simple situation where the target $\tilde{\mathbf{f}}_t$ remains as a constant with t . With DMAs, an attacker could steer LDP protocols into a desired strategy at each timestamp. Recall that for a constant target, the optimal attack can be achieved when the protocol chooses publication at one timestamp with allocating the largest budget or population while approximating at other timestamps. In a window of size w , the attacker needs to steer the protocol into approximation for $w-1$ timestamps and then into publication to make it allocate the largest budget or population. Then the attacker only needs to approximate until the end of the stream. With loss of attack effectiveness, we modify the above idea by publishing at the sampling timestamp in the window, and approximating at other timestamps, termed Sampling Attack, since only one publication with continuous approximation may raise aggregator's concern. This attack can achieve high attacking effectiveness against LSP by solely manipulating sampling timestamps.

Baseline 2: Uniform Attack. In contrast to the constant target, here we consider another special target stream $\tilde{\mathbf{f}}_t$ with huge fluctuations at each timestamp. Similar to the above analysis, at each publication timestamp, the attacker can achieve a manipulation gap as $G_{I,t}^*$ or $G_{O,t}^*$. Since the huge fluctuations of target, meaning large distance between two adjacent timestamps, the manipulation gap for approximation will be larger than that of the publication timestamp. Thus, always publication is more beneficial for the attack. We term this Uniform Attack, meaning attacking every publication timestamp uniformly. The uniform attack works well for LBU/LPU, which publishes at every timestamp.

The above baselines assume special targets. However, for arbitrary targets, adopting Uniform Attack to publish always will result in a publication budget or population not exceeding $\epsilon/2w$ or $(m+n)/2w$. When the budget or population is exactly or below $\epsilon/2w$ or $(m+n)/2w$, G_t meets or exceeds $G_{I(O),t}(m, n, \epsilon/2w)$ or $G_{I(O),t}(m/2w, n/2w, \epsilon)$. As analysed in Sec. 4.2, with a large w , G_t will become very large, which contradicts the objective of minimizing G_{avg} . When adopting the Sampling Attack for arbitrary targets, the protocol allocates at most a publication budget of $\epsilon/2$ or population $(m+n)/2$ at the sampling timestamp and zero at others. The average manipulation gap G_{avg} becomes $(G_{I(O),t}(m, n, \frac{\epsilon}{2}) + \frac{1}{d} \sum_{t=2}^w \sum_{k=1}^d (\tilde{\mathbf{f}}_t[k] - \hat{\mathbf{f}}_1[k])^2) / w$ for budget-division methods and $(G_{I(O),t}(\frac{m}{2}, \frac{n}{2}, \epsilon) + \frac{1}{d} \sum_{t=2}^w \sum_{k=1}^d (\tilde{\mathbf{f}}_t[k] - \hat{\mathbf{f}}_1[k])^2) / w$ for population-division methods, assuming the first timestamp is the sampling time. The attack effectiveness diminishes unless $\tilde{\mathbf{f}}_t$ remains constant.

4.4.2 Adaptive Strategy Chosen for Arbitrary Targets

Minimizing the time-average manipulation gap for an arbitrary target stream requires adaptive coordination of the two modules across timestamps. We propose a MSD module to achieve this by mimicking the idea of adaptively optimizing the estimation error in streaming LDP protocols as follows.

Optimization View of Streaming LDP. Adaptive streaming LDP protocols seek to minimize the average estimate error as an optimization problem in Eq. (2). Specifically, the estimation error $\text{error}_t = \sum_{k=1}^d (\hat{\mathbf{f}}_t[k] - \mathbf{f}_t[k])^2/d$ per timestamp can be denoted as a binary-state variable $\text{error}_t(i)$ concerning the *privacy strategy* of publication or approximation. It equals to either publication error err in the publication (denoted as $i = 1$) or approximation error (dissimilarity dis , where $\mathbb{E}(\overline{\text{dis}}) = \text{dis}$) in the approximation ($i = 0$). The former err equals the noise variance of FO (i.e., $\text{Var}(n_t, \epsilon_t)$) which is decided by the allocated user population n_t and budget ϵ_t . The latter dis can be denoted as $\sum_{k=1}^d (\hat{\mathbf{f}}_{t-1}[k] - \tilde{\mathbf{f}}_t[k])^2/d$, where $\hat{\mathbf{f}}_{t-1}$ is the last release at $t - 1$. So,

$$\text{error}_t(i) = \begin{cases} \text{dis} = \sum_{k=1}^d (\hat{\mathbf{f}}_{t-1}[k] - \tilde{\mathbf{f}}_t[k])^2/d, & i = 0 \text{ (approximation)} \\ \text{err} = \text{Var}(n_t, \epsilon_t), & i = 1 \text{ (publication)} \end{cases}$$

Then, the optimization problem in Eq. (2) is a strategy-chosen problem of finding a binary sequence Q for minimizing the time-average estimation error, i.e. $\arg \min_{Q \in \{0,1\}^T} \frac{1}{T} \sum_{t=1}^T \text{error}_t(Q[t])$. Adaptive streaming LDP protocols in Sec. 2.3 provide different online adaptive solutions to this problem by comparing err with dis and greedily choosing a *private strategy* with a smaller error.

Optimization View of Attack. The overall goal of our attack is formulated in Eq. (4). The *manipulation gap* $G_t = \sum_{k=1}^d (\hat{\mathbf{f}}_t[k] - \tilde{\mathbf{f}}_t[k])^2/d$ at timestamp t varies with the aggregator's choice between publication and approximation. If the aggregator chooses the approximation strategy, it will publish the previously estimated frequency $\hat{\mathbf{f}}_{t-1}$, so $G_t = \sum_{k=1}^d (\hat{\mathbf{f}}_{t-1}[k] - \tilde{\mathbf{f}}_t[k])^2/d$. Otherwise, it chooses the publication strategy, the aggregator will invoke the FO to publish a fresh estimate. The attacker will also initiate PMAs to manipulate the estimate to approach the target. Overall, when publication, G_t can be computed as $G_{I,t}(m_t^a, n_t^a, \epsilon_t)$ when using IPMA or $G_{O,t}(m_t^a, n_t^a, \epsilon_t)$ when using OPMA. m_t^a, n_t^a , and ϵ_t are the number of fake users and genuine users, and the privacy budget used for publication, respectively. Considering the aggregator's choice over two *privacy strategies* can be manipulated by DMAs, there are also two *attack strategies* from the attacker's view. They are approximation-based and publication-based strategies, i.e., manipulating the dissimilarity calculation to steer the aggregator to choose approximation and publication, respectively. And the *manipulation gap* G_t can be formulated as the following variable, where $i = 0$ and 1 represent the approximation and publication-based attack strategies, respectively.

$$G_t(i) = \begin{cases} \sum_{k=1}^d (\hat{\mathbf{f}}_{t-1}[k] - \tilde{\mathbf{f}}_t[k])^2/d, & i = 0 \text{ (approximation)} \\ G_{I(O),t}(m_t^a, n_t^a, \epsilon_t), & i = 1 \text{ (publication)} \end{cases}$$

Therefore, the optimization problem in Eq. (4) can also be seen as a *strategy-chosen problem*, i.e., seeking a binary vector $Q = \{0, 1\}^T$ of length T which can minimize the average manipulation gap $G_{avg} = \frac{1}{T} \sum_{t=1}^T G_t$ over the whole stream:

$$\arg \min_{Q \in \{0,1\}^T} \frac{1}{T} \sum_{t=1}^T G_t(Q[t]).$$

MSD: Adaptive Strategy Chosen. Comparing the strategy-chosen problems of the adaptive streaming LDP protocol to adaptive attack for arbitrary targets, it is easy to find that they share a similar problem structure. For publication, both $\text{error}_t(1)$ and $G_t(1)$ are determined similarly by privacy budget and population size; larger values result in smaller errors or manipulation gaps, enhancing estimation or attack effectiveness. For approximation, both

Algorithm 1: Adaptive Attack to LDP-IDS

```

Input: Target stream  $\tilde{\mathbf{f}} = \{\tilde{\mathbf{f}}_1, \tilde{\mathbf{f}}_2, \dots, \tilde{\mathbf{f}}_t, \dots\}$ 
Output: Manipulated released statistics  $\hat{\mathbf{f}} = \{\hat{\mathbf{f}}_1, \hat{\mathbf{f}}_2, \dots, \hat{\mathbf{f}}_t, \dots\}$ 
1 Initialize Queue of  $\epsilon_{t,2}$  or  $|U_{t,2}|$ ,  $Q_m = [0, \dots, 0]^{w-1}$ ;
2 Estimate the number of genuine users as  $n^e$ ;
3 for each timestamp  $t$  do
4   Get  $\mathbf{f}_t^e$  from fake users and estimate  $\epsilon_{t,2}$  or  $|U_{t,2}|$  using  $Q_m$ ;
5   Compute  $m^*$  for PMAs' optimization problem (e.g. Eq. (10));
6   Calculate attack dissimilarity  $\text{dis}_{attack} = \sum_{k=1}^d (\hat{\mathbf{f}}_{t-1}[k] - \tilde{\mathbf{f}}_t[k])^2/d$ ;
7   Calculate potential manipulation gap  $G(m, n^e, \epsilon_{t,2})$  for budget-division
    or  $G\left(\frac{m|U_{t,2}|}{(m+n^e)}, \frac{n^e|U_{t,2}|}{(m+n^e)}, \epsilon\right)$  for population-division;
// Manipulation Strategy Determination (MSD)
8   if  $\text{dis}_{attack} > G$  then
9     Adopt DMAs to maximize the dissimilarity  $\overline{\text{dis}}$ ;
10  else
11    Adopt DMAs to minimize the dissimilarity  $\overline{\text{dis}}$ ;
12  end
13  if LDP Aggregator chooses publication then
14    Launch PMAs (using  $m^*$  in line 5) to derive  $\hat{\mathbf{f}}_{attack}$ ;
15    return  $\hat{\mathbf{f}}_t = \hat{\mathbf{f}}_{attack}$ ;
16  else
17    return  $\hat{\mathbf{f}}_t = \hat{\mathbf{f}}_{t-1}$ ;
18  end
19 Pop the first item from  $Q_m$ , enqueue new  $\epsilon_{t,2}$  or  $|U_{t,2}|$ ;
20 end

```

$\text{error}_t(0)$ and $G_t(0)$ represents approximating the results of current timestamp (i.e., \mathbf{f}_t or $\tilde{\mathbf{f}}_t$) with the last one. The similar problem structure inspires us the design of MSD module, which solves the attack strategy-chosen problem in an online adaptive way as the private strategy-chosen problem of adaptive streaming LDP protocols. In particular, MSD guides the attacker to greedily choose between approximation and publication-based strategy with a better attack effect (i.e., smaller manipulation gap). Specifically, if $G_t(0)$ is less than $G_t(1)$, MSD chooses the approximation-based attack strategy; otherwise, MSD chooses the publication-based attack strategy.

5 Attacking Streaming LDP Protocols

In the following, we present the details of our Adaptive Attack against the adaptive streaming LDP protocols by integrating the above proposed MSD, PMAs, and DMAs.

5.1 Attacking Adaptive Streaming LDP

For LDP estimation, a larger privacy budget or population enhances data utility. Similarly, for attacks, a larger budget (Corollaries 4.2 and 4.7) or population (Corollaries 4.3 and 4.8) reduces the manipulation gap, thus increasing the attack effectiveness. This suggests adopting a streaming-LDP-like adaptive budget/population allocation could enhance attacks, as discussed in Sec. 4.4.2.

5.1.1 Algorithm details. Algorithm 1 shows the detailed framework of our proposed Adaptive Attack to LDP-IDS (adaptive framework of streaming LDP protocol [23]) with an arbitrary target stream $\tilde{\mathbf{f}} = \{\tilde{\mathbf{f}}_1, \tilde{\mathbf{f}}_2, \dots, \tilde{\mathbf{f}}_t, \dots\}$. At each timestamp t , the attacker first gets the inputs (or outputs in the MITM attack) of FOs from a portion of compromised users to estimate \mathbf{f}_t^e (line 4). Then, the attacker computes $(m^*[1], \dots, m^*[d])$ via solving the optimization problem of PMAs (line 5). After that, the attacker can calculate $\text{dis}_{attack} = \sum_{k=1}^d (\hat{\mathbf{f}}_{t-1}[k] - \tilde{\mathbf{f}}_t[k])^2/d$ as the manipulation gap for the approximation-based attack strategy (line 6), and the *potential manipulation gap* G (i.e., $G_{I,t}$ in IPMA or $G_{O,t}$ in OPMA) for the publication-based attack strategy (line 7). For budget-division protocols LBD/LBA, by estimating n and \mathbf{f}_t as n^e and \mathbf{f}_t^e , G is calculated as $G(m, n^e, \epsilon_{t,2})$ since $\epsilon_{t,2}$ budget with all users (n genuine and m

fake users) are used. For population-division protocols LPD/LPA, G equals to $G\left(\frac{m|U_{t,2}|}{(m+n^e)}, \frac{n^e|U_{t,2}|}{(m+n^e)}, \epsilon\right)$ as the whole budget with $|U_{t,2}|$ users ($\frac{n|U_{t,2}|}{m+n}$ genuine and $\frac{m|U_{t,2}|}{m+n}$ fake users) are used.

Then, MSD chooses a beneficial strategy by comparing dis_{attack} with G (line 8). If $dis_{attack} > G$, MSD will choose the publication-based attack strategy. Thus, DMAs will be launched to maximize dis in the streaming LDP protocol and enforce the LDP aggregator to choose publication (line 9). If the aggregator does so, PMAs will be further mounted to manipulate the FO in publication (line 14). If $dis_{attack} \leq G$, DMAs instead of PMAs will then be called to minimize dis (line 11). Fig. 2(c) illustrates the strategy selection-based adaptive attack, which interestingly mimics that of the strategy selection in the adaptive LDP estimation in Fig. 2(a).

Note that, the historical usage of budget $\epsilon_{t,2}$ or population $|U_{t,2}|$ for the previous $w - 1$ timestamps is needed for calculating G (line 4). So, the attacker can store them in a first-in-first-out queue Q_m (line 19). For example, when attacking LBD, $\epsilon_{t,2} = (\epsilon/2 - \sum_{i=1}^{w-1} Q_m[i])/2$. Current $\epsilon_{t,2}$ or $|U_{t,2}|$ at t can be obtained as follows. For the publication timestamp, the fake users will receive $\epsilon_{t,2}$ from the aggregator, $|U_{t,2}|$ can be estimated as $\frac{m_t(n^e+m)}{m}$ where m_t is the number of fake users sampled. For the approximation timestamp, $\epsilon_{t,2}$ or $|U_{t,2}|$ is 0.

5.1.2 Theoretical Analysis. Tables 3 and 2 summarize the effects of all proposed attacking methods. For LBU and LPU, only the Uniform Attack (including both Input and Output poisoning) is used. For LSP, only the Sampling Attack (also including Input and Output) is applied, and we abbreviated $G_{I(O),t}(m, n, \epsilon) + \sum_{t=(i-1)w+2}^{iw} \sum_{k=1}^d (\hat{f}_t[k] - \hat{f}_{(i-1)w+1}[k])^2/d$ as $G_{I(O),t}^S(m, n, \epsilon, i)$. In each attack, the same type of LDP poisoning (i.e., Input or Output) is assumed for both DMAs and PMAs. These methods employ Uniform, Sampling, and Adaptive Attacks, each available in input and output forms. Particularly, it is assumed that there are enough m fake users so that DMAs can always succeed in inducing the LDP aggregator into the desired strategy. Since the adaptive allocation of privacy budget and population results in varying sequences of $\epsilon_{t,2}$ and $U_{t,2}$, it is hard to calculate the exact average manipulation gap. Therefore, we estimate the upper and lower bounds of the manipulation gap based on the minimum and maximum privacy budget and population that can be allocated. For LBD, the privacy budget ranges from $\epsilon/2^{w+1}$ to $\epsilon/4$. For LBA, it ranges from $\epsilon/2w$ to $\epsilon/2$. For LPD, the user population is between $(m+n)/2^{w+1}$ and $(m+n)/4$. For LPA, the range is from $(m+n)/2w$ to $(m+n)/2$.

Tables 3 and 2 manifest that the Adaptive Attacks' performance always equals to or exceeds Uniform Attacks'. It is hard to compare Adaptive Attacks and Sampling Attacks directly while experiments show that Adaptive Attacks also outperform Sampling Attacks. Interestingly, we find that the baseline LDP algorithms are more robust to our attacks than the adaptive ones when adopting the same type of attacks. For example, attacking LBA has less manipulation gap than attacking LBU with Uniform Attacks. This raises concerns that, despite achieving higher utility, adaptive LDP methods may be less robust to fine-grained manipulation.

Table 2: Average manipulation gap of attacks against LDP-IDS baselines. t_a is the total number of attack timestamps.

	LBU	LPU	LSP
Input Uniform Attack(Sec. 4.4.1)	$G_{I,t}(m, n, \frac{\epsilon}{w})$	$G_{I,t}\left(\frac{m}{w}, \frac{n}{w}, \epsilon\right)$	Input Sampling Attack(Sec. 4.4.1) $\left(\sum_i G_{I,t}^S(m, n, \epsilon, i)\right)/t_a$
Output Uniform Attack(Sec. 4.4.1)	$G_{O,t}(m, n, \frac{\epsilon}{w})$	$G_{O,t}\left(\frac{m}{w}, \frac{n}{w}, \epsilon\right)$	Output Sampling Attack(Sec. 4.4.1) $\left(\sum_i G_{O,t}^S(m, n, \epsilon, i)\right)/t_a$

5.2 Attacking Other LDP Tasks and Protocols

- **Attacking Mean Estimation.** Besides frequency estimation, our methods can also apply to continual mean estimation over data streams [21, 23] with only replacing PMA with OPA [32] since both PMA and OPA benefits from a larger budget and population. For further information, refer to Appendix C.
- **Attacking Other Streaming LDP Protocols.** Our attack methods are applicable to other streaming LDPs like PeGaSus [38], FAST [39], DSAT_w [37], RescueDP [36], CGM [22] and ToPL [21]. To summarize, since DSAT_w adopts budget-division or population-division like LDP-IDS, all our proposed attacks are applicable. PeGaus, CGM and ToPL choose to publish at every timestamp like LBU/LPU, thus Uniform Attack are adopted. FAST and RecueDP samples timestamps for publication like LSP, so the Sampling Attack is considered. The post-processing of event-level LDP does not affect our attacks. Both LDP estimation and attacks benefit from a larger budget population (Corollaries 4.2, 4.3, 4.7, and 4.8), sharing similar properties. Thus, the post-processing for better LDP estimation can be directly exploited to enhance our attacks. More details are in Appendix D.

6 Performance Evaluation

6.1 Experimental Setup

6.1.1 Datasets. Three real-world datasets were used as follows. Results on synthetic datasets are shown in Appendix B.8.

- **Taxi**¹ comprises taxi trajectories in Beijing from Feb. 2nd to Feb. 8th, 2008. We extracted data from $N = 10,357$ taxis, each with $T = 885$ timestamps at 10-minute intervals, across 6 grid partitions ($d = 6$). We also utilize longitude data of Taxi dataset for numerical domain attacks, denoted as **Taxi-Latitude**.
- **Foursquare**² includes Foursquare check-ins from Apr. 2012 to Sep. 2013, detailing time, place, and user ID. It's transformed into $N = 266,909$ data streams, each with $T = 456$ timestamps, and places categorized into 100 types ($d = 100$).
- **Taobao**³ includes AD click logs from 1.14 million Taobao customers across 12,973 categories. For simplicity, AD commodities were grouped into $d = 150$ categories. The extracted click streams of $N = 728,745$ customers, each representing the category of the last click every ten minutes over three days, totaling $T = 432$ timestamps. For numerical domains, we obtain the labeled price of each AD commodities in Taobao, referred to as **Taobao-Price**.

We also synthesized four types of datasets and run experiments on them. The results can be found in Appendix B.8.

- 6.1.2 Targets.** We used four different target streams $\tilde{f} = (\tilde{f}_1, \tilde{f}_2, \dots, \tilde{f}_T)$ for frequency and mean with the following time-varying patterns.

¹<https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/>

²<https://sites.google.com/site/yangdingqi/home/foursquare-dataset>

³<https://tianchi.aliyun.com/dataset/dataDetail?dataId=56>

Table 3: Average manipulation gap of attacks against adaptive LDP-IDS. t_a is the total number of attack timestamps.

	LBD	LBA	LPD	LPA
Input Uniform Attack (Sec. 4.4.1)	$\left(G_{I,t}\left(m, n, \frac{\epsilon}{4}\right), G_{I,t}\left(m, n, \frac{\epsilon}{2^{w+1}}\right)\right)$	$G_{I,t}\left(m, n, \frac{\epsilon}{2^w}\right)$	$\left(G_{I,t}\left(\frac{m}{4}, \frac{n}{4}, \epsilon\right), G_{I,t}\left(\frac{m}{2^{w+1}}, \frac{n}{2^{w+1}}, \epsilon\right)\right)$	$G_{I,t}\left(\frac{m}{2^w}, \frac{n}{2^w}, \epsilon\right)$
Output Uniform Attack (Sec. 4.4.1)	$\left(G_{O,t}\left(m, n, \frac{\epsilon}{4}\right), G_{O,t}\left(m, n, \frac{\epsilon}{2^{w+1}}\right)\right)$	$G_{O,t}\left(m, n, \frac{\epsilon}{2^w}\right)$	$\left(G_{O,t}\left(\frac{m}{4}, \frac{n}{4}, \epsilon\right), G_{O,t}\left(\frac{m}{2^{w+1}}, \frac{n}{2^{w+1}}, \epsilon\right)\right)$	$G_{O,t}\left(\frac{m}{2^w}, \frac{n}{2^w}, \epsilon\right)$
Input Sampling Attack (Sec. 4.4.1)	$\left(\sum_i G_{I,t}^S\left(m, n, \frac{\epsilon}{4}, i\right) / t_a\right)$	$\left(\sum_i G_{I,t}^S\left(m, n, \frac{\epsilon}{2}, i\right) / t_a\right)$	$\left(\sum_i G_{I,t}^S\left(\frac{m}{4}, \frac{n}{4}, \epsilon, i\right) / t_a\right)$	$\left(\sum_i G_{I,t}^S\left(\frac{m}{2}, \frac{n}{2}, \epsilon, i\right) / t_a\right)$
Output Sampling Attack (Sec. 4.4.1)	$\left(\sum_i G_{O,t}^S\left(m, n, \frac{\epsilon}{4}, i\right) / t_a\right)$	$\left(\sum_i G_{O,t}^S\left(m, n, \frac{\epsilon}{2}, i\right) / t_a\right)$	$\left(\sum_i G_{O,t}^S\left(\frac{m}{4}, \frac{n}{4}, \epsilon, i\right) / t_a\right)$	$\left(\sum_i G_{O,t}^S\left(\frac{m}{2}, \frac{n}{2}, \epsilon, i\right) / t_a\right)$
Input Adaptive Attack (Sec. 5.1)	$\left(G_{I,t}\left(m, n, \frac{\epsilon}{4}\right), G_{I,t}\left(m, n, \frac{\epsilon}{2^{w+1}}\right)\right)$	$\left(G_{I,t}\left(m, n, \frac{\epsilon}{2}\right), G_{I,t}\left(m, n, \frac{\epsilon}{2^w}\right)\right)$	$\left(G_{I,t}\left(\frac{m}{4}, \frac{n}{4}, \epsilon\right), G_{I,t}\left(\frac{m}{2^{w+1}}, \frac{n}{2^{w+1}}, \epsilon\right)\right)$	$\left(G_{I,t}\left(\frac{m}{2}, \frac{n}{2}, \epsilon\right), G_{I,t}\left(\frac{m}{2^w}, \frac{n}{2^w}, \epsilon\right)\right)$
Output Adaptive Attack (Sec. 5.1)	$\left(G_{O,t}\left(m, n, \frac{\epsilon}{4}\right), G_{O,t}\left(m, n, \frac{\epsilon}{2^{w+1}}\right)\right)$	$\left(G_{O,t}\left(m, n, \frac{\epsilon}{2}\right), G_{O,t}\left(m, n, \frac{\epsilon}{2^w}\right)\right)$	$\left(G_{O,t}\left(\frac{m}{4}, \frac{n}{4}, \epsilon\right), G_{O,t}\left(\frac{m}{2^{w+1}}, \frac{n}{2^{w+1}}, \epsilon\right)\right)$	$\left(G_{O,t}\left(\frac{m}{2}, \frac{n}{2}, \epsilon\right), G_{O,t}\left(\frac{m}{2^w}, \frac{n}{2^w}, \epsilon\right)\right)$

- **Uniform:** \tilde{f}_t at each time t was set as a uniform distribution for frequency estimation and a constant for mean-value estimation to simulate a stream with no fluctuation.
- **Pulse:** To simulate huge fluctuation stream, for frequency estimation, at each time t , we selected one item $k \in [d]$ and set its frequency $\tilde{f}_t[k] = 1$ and others $\tilde{f}_t[k'] = 0$ ($k' \neq k$). We set the target as an extreme value for mean-value estimation.
- **Gaussian:** Gaussian distribution is a common distribution that always appears in natural and social sciences. We set \tilde{f}_t at each time t as a discrete Gaussian distribution $N(0, \sigma^2 t)$, where $\sigma = 0.5$. For mean-value estimation, we set the target as the value sampled from the Gaussian distribution.
- **Sigmoid:** To simulate frequency increasing pattern, for each \tilde{f}_t , we chose one item $k \in [d]$ and set its frequency as $\tilde{f}_t[k] = 2 \cdot \text{Sigmoid}(0.01t) - 1$. For mean estimation, we set the target increasing according to Sigmoid pattern.

6.1.3 Metrics. We measured the attack performance using the time and dimension averaged MSE between manipulated statistical stream \tilde{f} and the target one f , i.e., $MSE = \frac{1}{dT} \sum_{t=1}^T \sum_{k=1}^d (\tilde{f}_t[k] - f_t[k])^2$, aligning with Eq. (4). A smaller MSE means better attack effectiveness. We also used the *Success Rate* = $\frac{\# \text{success}}{\# \text{total}}$ to evaluate the effectiveness of DMAs, where #success is the number of DMAs that successfully induce the LDP protocol into beneficial strategies for the attack, and #total is the total number of launched DMAs.

6.1.4 Parameter Settings. We set default $\epsilon = 1$ and $w = 20$ and used Ada as the default FO. We evaluated attacks against other FOs with larger domain in Sec. 6.2.3. We chose the default estimated user number n^e based on a common observation that online reports tend to publish round numbers instead of precise values [4]. We selected 1000 fake users per dataset for f^e information. Here, we mainly consider a partial-knowledge attacker and compare different levels of knowledge in Sec. 6.2.3. Given the difficulty of achieving special targets like Pulse, we adjusted $\beta = m/(m+n)$ variably for different targets. For budget-division attacks with OPMA, sufficient fake users are necessary due to Corollary 4.7, setting β at 0.2 for uniform and Gaussian, and 0.3 for pulse and sigmoid per Eq. (12). IPMA inherently negates the need for sufficient fake users as per Eq. (8) due to Corollaries 4.2 and 4.3. Attacks on population-division are based on Corollaries 4.3 and 4.8, without requiring for sufficient fake users. Reasons for different β settings are detailed in Sec. 6.2.2.

6.1.5 Compared Algorithms. Due to no existing study, we compared our proposed Adaptive Attack with the baselines by applying them to the following algorithms in the stream setting.

- **w-event Level:** LDP-IDS [23], DSAT_w [37] and RescueDP [36];
- **Event Level:** PeGaus [38] and ToPL [21];
- **User Level:** FAST [39] and CGM [22].

We assume the same type of attack for DMAs and PMAs (e.g., IDMA with IPMA). We abbreviated attacks (e.g., IUA for Input Uniform Attack) and denoted attacking format as “X-Y” (e.g., LBD-IUA means attacking LBD with IUA).

6.2 Experimental Results

We present our findings on the sufficient conditions for effective attacks and how different factors impact the attack’s performance.

6.2.1 Overall Results. Fig. 3 shows the attack effectiveness of all proposed attacks with different targets while varying the fake user ratio β . As shown, for target streams with minor fluctuations like Uniform (Fig. 3(a)), Adaptive and Sampling Attacks outperform Uniform Attacks, which is consistent with the analysis in Sec. 5.1.2. For those with significant fluctuations like Pulse (Fig. 3(b)), Adaptive and Uniform Attacks outperform Sampling Attacks. Generally, we can observe that, Output Attacks (OUA, OSA and OAA) outperform Input Attacks (IUA, ISA and IAA). Adaptive Attacks (IAA and OAA) surpass Uniform Attacks (IUA and OUA) and Sampling Attacks (ISA and OSA). Among all, OAA achieves the best performance. A larger β (i.e., more fake users) generally enhances the attack effectiveness and DMA’s success rate. However, for OSA with the Pulse target, the effectiveness reduces when β is larger since the improvement of DMA’s success rate causes LDP-IDS to choose approximation more, which is contrary to the way to achieve Pulse target, i.e., requiring more publication. Appendix B.1 shows more results on different datasets.

6.2.2 Sufficient Conditions for Attacks. We examined the minimum number of fake users for achieving targets in IPMA and OPMA. Given f , ϵ , and the dataset, the minimum number of fake users could be derived by Eqs. (8) and (12). Then, the minimum β for attacks can be obtained. Figs. 5(a) and 5(b) show the minimum β required for successful IPMA and OPMA attacks respectively, during the first 400 timestamps for each dataset. We can observe that OPMA requires less fake users when achieving the same attack effect.

6.2.3 Impact of Other Factors

Impact of ϵ . Fig. 4(a) shows the attack effectiveness with different privacy budget ϵ . Attacks improve when ϵ is larger. Success rates vary with ϵ . Large ϵ typically boosts the success rate of Uniform Attacks since a larger ϵ reduces the noise injected into dissimilarity calculation, which allows DMAs to succeed with a higher probability. The success rate for Sampling Attacks either drops or remains

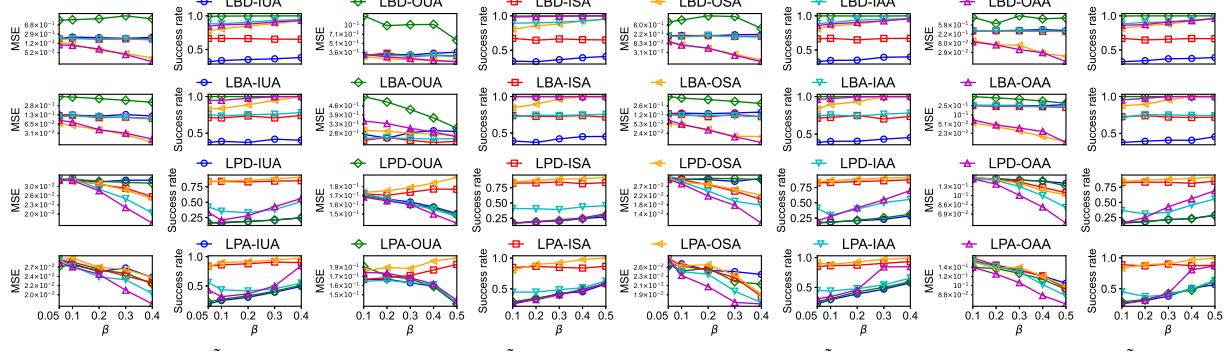


Figure 3: Attacking effectiveness against LBD, LBA, LPD and LPA, varying fake user ratio β (Taxi with different targets).

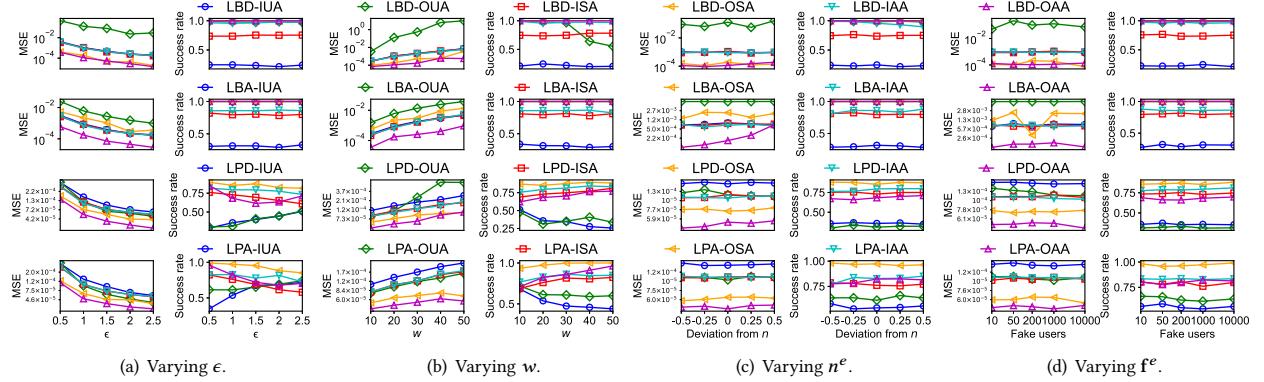


Figure 4: Attack effectiveness w.r.t. different parameters (Taobao with Sigmoid target).

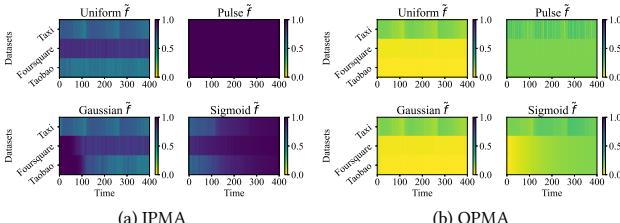


Figure 5: The minimum β needed for IPMA/OPMA with varying targets \tilde{f} on different datasets (Ada FO with $\epsilon = 0.5$).

stable. This is because Sampling Attacks always need to minimize dis via DMA to make it smaller than err but a larger ϵ also makes err smaller, leading worse successful rate for DMA. Appendix B.1 gives more results on different datasets and different targets.

Impact of w . Fig. 4(b) shows the attack effectiveness on varying sliding window size w . Overall, the attack effectiveness improves with smaller w . Also, Output Adaptive Attack always achieves better performance. Different attacks show different relations between their success rates and w . For Uniform Attacks, it decreases with w as large w increases the result variance in DMA. For Sampling Attacks, the success rate is basically unchanged since the budget or population allocated at the sampling timestamp is independent of w . Appendix B.1 shows more results.

Impact of Attacker's knowledge. Figs. 4(c) and 4(d) demonstrate the effects of varying n^e and f^e on the attacks. We varied n^e by $\pm 50\%$ and $\pm 25\%$ from n to assess the impact. For f^e , we assume the attacker can access different numbers of fake users for f^e estimation.

Generally, more accurate estimation of n^e and f^e lead to a smaller manipulation gap and higher success rate. Appendix B.1 gives more results.

Impact of dimension d . Fig. 6 illustrates the attack effectiveness on Ada, with varying domain size d . We sorted Taobao categories and varied d by grouping item IDs into different numbers of categories (from 2^2 to 2^{12}). Results indicate that the attack enhances as the domain size increases, highlighting the effectiveness of our attacks in large domains. Results on different FOs (kRR and OUE), are in Appendix B.2.

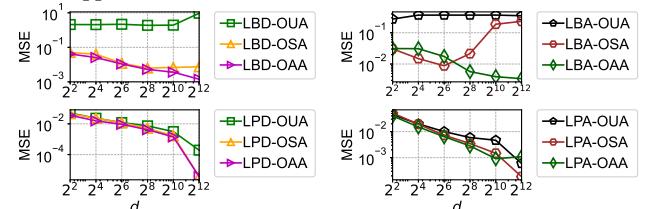


Figure 6: Effectiveness w.r.t d (Taobao with Gaussian target).

Impact of Attack Mismatch. Table 4 illustrates the mismatch scenarios where the attacker launches an unmatched attacking algorithm against an LDP protocol, e.g., using an LBA-based adaptive attack on an aggregator operating with LBD. Mismatched attacks mainly result in the wrong calculation of $G_{I,t}$ or $G_{O,t}$, leading the aggregator to choose a strategy less beneficial for attacks. Each entry (X, Y) in Table 4 means using OAA designed for LDP protocol in column Y to attack row X . As shown, despite the matched attack gains the smallest gap, the mismatch often has a slight impact

if they adopt the same budget or population division framework. Otherwise, mismatched attacks with different division frameworks often lead to much poorer attack performances since the calculation of $G_{I,t}$ or $G_{O,t}$ is quite different in the two division frameworks.

Table 4: Impact of Mismatched Attacks (Gaussian target).

MSE	Attack Methods				MSE	Attack Methods			
	OAA-BD	OAA-BA	OAA-PD	OAA-PA		OAA-BD	OAA-BA	OAA-PD	OAA-PA
LBD Agg.	LBD 0.0178	0.0213	7.9034	7.4355	LBD Agg.	0.0011	0.0013	2.3891	2.9994
LBA	0.4736	0.0152	3.537	3.1310	LBA	0.0029	0.0009	1.1676	1.2000
LPD	0.0021	0.0016	0.0004	0.0012	LPD	0.0004	0.0004	0.0002	0.0004
LPA	0.0083	0.0038	0.0011	0.0003	LPA	0.0017	0.0017	0.0004	0.0002

(a) Foursquare.

(b) Taobao.

Different Levels of Knowledge. Fig. 7 compares the performance of the full-knowledge (FK), partial-knowledge (PK) and the man-in-the-middle attack (MITM). In these scenarios, the PK attackers obtain 1% or 25% of the knowledge about frequencies, denoted by PK(0.01) and PK(0.25). MITM attackers intercept 1% or 25% of the communication between users and aggregator and recover the frequencies with FO, denoted by MITM(0.01) and MITM(0.25). Results show that FK attackers perform better than PK attackers. The performance of PK attackers with 25% information is almost as good as that of FK attackers. MITM attackers perform the worst because recovering frequencies with FO often introduce more noise. More results are in Appendix B.3.

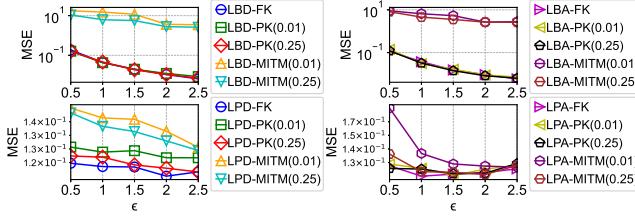


Figure 7: Attack effectiveness using OAA with different levels of knowledge, varying ϵ (Taxi with Sigmoid target).

6.2.4 Attacking Baseline Methods. Fig. 8 shows the performance of our attacks against baselines, LBU, LPU and LSP. For LBU and LPU, the Uniform Attack was used. LSP was targeted with the Sampling Attack. The results are much similar to those of attacking adaptive methods, with a larger ϵ and β and a smaller w meaning a more successful attack. More results can be found in Appendix B.4.

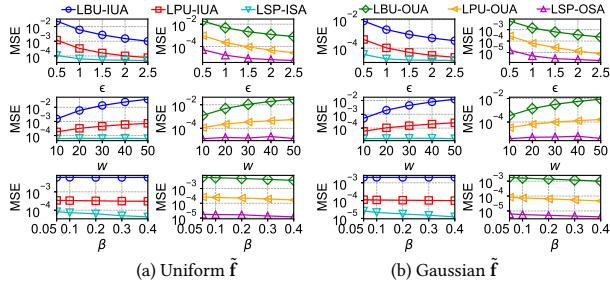


Figure 8: Attack effectiveness on baselines, varying ϵ , w and β (Foursquare).

6.2.5 Attacking Mean Estimation. We attacked HM-based mean estimation in the streaming setting. In the experiments, we used Taobao-Price and the Uniform target, and set the parameters as $\epsilon = 1$, $w = 20$, and $\beta = 0.1$. Results are shown in Fig. 9, demonstrating attack effects across different ϵ , w , and β values. Still, the Adaptive Attack generally performs the best. Larger ϵ improves the attack which is align with the analysis in [32], whereas increasing

w reduces effectiveness by decreasing the privacy budget and user population at each timestamp. A larger β allows more fake user manipulation, enhancing attack effectiveness. More results are shown in Appendix B.5.

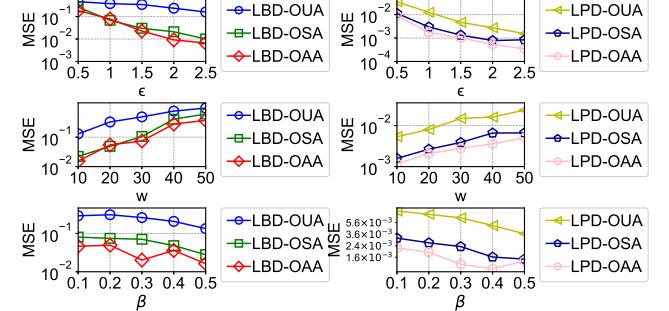


Figure 9: Attack effectiveness on HM-based LDP-IDS, varying ϵ , w and β (Taobao-Price with Uniform target).

6.2.6 Attacking Other LDP Algorithms. As discussed in Sec. 5.2, our attacking methods apply to other streaming LDPs adapted in [23], where they are categorized into budget-division (LB-*) and population-division (LP-*). Only Uniform Attack work on PeGaSus, and only Sampling Attack on FAST and RescueDP. However, DSAT_w is susceptible to all our adaptive proposed attacks. Fig. 10 shows the results. For comparison, we binarized the Taxi, Taobao, and Foursquare datasets and set β at 0.1. Results show output attacks generally outperform input ones, with Adaptive Attacks notably more effective on DSAT_w.

For other stream LDPs over numerical domain, we normalized Taxi-Longitude and Taobao-Price datasets into $[-\frac{1}{2}, \frac{1}{2}]$ for CGM and $[0, 1]$ for ToPL. We set the default value $\epsilon = 1$, $\beta = 0.1$, and $\delta = 10^{-5}$ (for CGM). Fig. 11 shows the attack effectiveness of Output Uniform Attack against CGM with user-level LDP and ToPL with event-level LDP respectively. The results demonstrate that more fake users are beneficial to the attack. More results can be found in Appendix B.6.

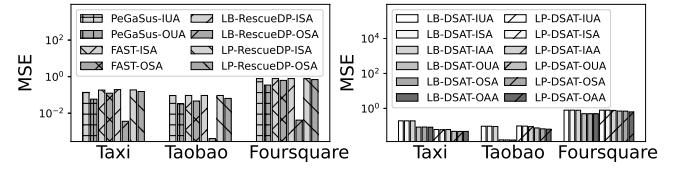


Figure 10: Effectiveness on other stream algs. (Gaussian target).

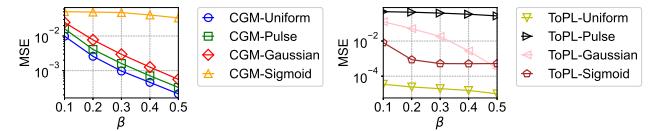
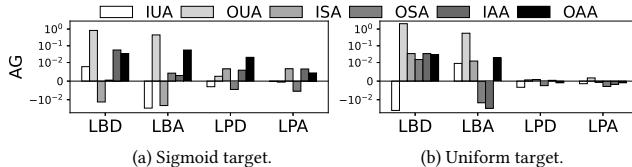
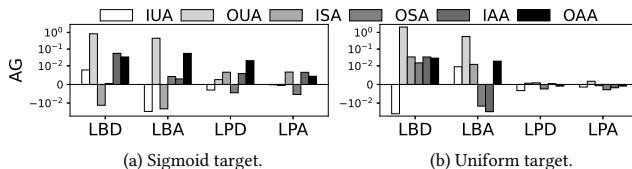


Figure 11: Attack effectiveness against stream LDPs over numerical domain (Taxi-Longitude).

7 Discussion on Possible Defense

Inspired by [32, 43], we propose a potential countermeasure. We first propose our defense for FOs. The first step is sampling. Assuming a total of $m + n$ users, the aggregator samples subsets of received data, each with the size of $r(m + n)$ where r is a fraction between 0 and 1. It then estimates item frequencies for each subset,

Figure 12: Accuracy Gain (Taxi, $\epsilon = 1$, $r=0.5$).Figure 13: Accuracy Gain (Taxi, $\epsilon = 1$, $r=0.5$).

obtaining different \hat{f} . Most \hat{f} , due to a small proportion of fake users, cluster near the true frequency distribution, but some may appear abnormal with higher fake user concentrations. We then employ Isolation Forest [44], an anomaly detection method, to score each \hat{f} for anomalies. Larger scores indicate less anomalous data, from which the aggregator determines the publication frequency. Based on the defense on FO above, we propose our defense on LDP-IDS. We notice that LDP-IDS invokes FO twice at each timestamp, producing two estimates for the same distribution. However, the proposed attack may break the similarities between two statistics, causing two estimates inconsistent. Thus, the attack can be detected by comparing two estimates. Here we use Kolmogorov-Smirnov test to detect it. Once the attack is detected and LDP-IDS chooses the publication strategy, the defense for FOs above will be invoked.

Experiments. Fig. 13 demonstrates the performance of our proposed defense mechanism on Taxi, setting $\epsilon = 1$ by default. The defense performance is measured by the Accuracy Gain (AG) of the LDP protocols brought by the defense. Specifically, AG is defined as the reduction in estimation error (measured by MSE, i.e., $MSE_{\text{before}}(\hat{f}, f) - MSE_{\text{after}}(\hat{f}, f)$) between the released statistics \hat{f} from the genuine statistics f , after applying the defense. A positive AG indicates successful mitigation. Results in Fig. 13 show our defense mechanism can generally mitigate the attack, more effective for Sigmoid target than Uniform target. It is because Sigmoid target manipulates the dataset from the original more than Uniform target, resulting in more anomalies detected and leaving more space for mitigation. More results are in Appendix B.7.

8 Related Work

LDP over data streams. Local differential privacy (LDP), a decentralized variant of Differential privacy (DP) [45, 46], has emerged as a popular privacy-preserving paradigm for large-scale data collection and analysis without relying a trusted aggregator. It has been applied in various analytic tasks like frequency estimation [4, 7–12], mean/variance estimation [13], key-value data collection [14], frequent itemset mining [15–18], graph data mining [19, 20], as well as streaming data analysis [21–23, 25, 47–50].

This paper focuses on the LDP analysis over data streams. According to the granularity of protection, related studies can be categorized into event-level DP [38], user-level DP [39, 51], and w -event level DP [24, 33]. Among them, w -event level strikes a balance between and can be easily extended to the other two, thus gaining

much attention [24]. Wang *et al.* [21] proposed ToPL composed of a threshold optimizer and perturber for outputting streaming data at event-level LDP. It actually publishes at each timestamp and then performs post-processing. Bao *et al.* [22] presented CGM that adds correlated noise to ensure approximate user-level LDP. CGM can also be seen as publishing at each timestamp with different budget. Ren *et al.* [23] proposed LDP-IDS focusing on w -event LDP, which includes two extensible budget-division and population-division frameworks. Based on the frameworks, most existing CDP algorithms on streams can be extended into the LDP setting, and roughly summarized into three phases discussed in Sec. 2.3.

Poisoning attacks to LDP protocols. Research shows the LDP aggregator's vulnerability to perturbed data, enabling data poisoning attacks. Cheu *et al.* [52] explored untargeted attacks manipulating the L_p norm distance in estimated frequencies pre- and post-attack. Cao *et al.* [30] and Wu *et al.* [31] developed targeted attacks on LDP for frequency estimation, heavy-hitter identification, and key-value data, respectively. Li *et al.* [32] examined fine-grained attacks on mean and variance estimation. Zheng *et al.* [53] disrupted LDP-based crowdsensing with bi-level optimization. Tong *et al.* [41] presented poisoning attacks against LDP frequent itemset mining protocols. However, all these focus on static settings and cannot directly work for fine-grained manipulation in streaming scenarios.

9 Conclusion

We conduct a comprehensive study on fine-grained data poisoning attacks to LDP protocols for data streams. By summarizing existing streaming LDP algorithms into three phases with poisoning surfaces, we introduce a unified attack framework with three coordinated attack modules, each with theory-driven designs. Applying the attack framework, we integrate these modules and propose detailed adaptive attacking methods to the state-of-the-art online adaptive LDP algorithms. Furthermore, we explore their potential against other algorithms under different LDP models and analytic tasks. Besides theoretical analysis, extensive experiments demonstrate the effectiveness, sufficient conditions, and parameter impacts of the proposed attacks. Finally, we discuss possible countermeasures with experiment validations. Our work highlights the security of streaming LDP protocols and appeals to further research.

References

- [1] J. Duchi, M. Jordan, and M. Wainwright, “Local privacy and statistical minimax rates,” in *Proc. IEEE FOCS*, 2013, pp. 429–438.
- [2] ——, “Privacy aware learning,” *Journal of the ACM (JACM)*, vol. 61, no. 6, pp. 1–57, 2014.
- [3] S. Kasiviswanathan, H. Lee, N. S. Raskhodnikova, and A. Smith, “What can we learn privately?” *SIAM J. Computing*, vol. 40, no. 3, pp. 793–826, 2011.
- [4] U. Erlingsson, A. Korolova, and V. Pihur, “Rappor: Randomized aggregatable privacy-preserving ordinal response,” in *Proc. ACM CCS*, 2014, pp. 1054–1067.
- [5] B. Ding, J. Kulkarni, and S. Yekhanin, “Collecting telemetry data privately,” in *Proc. NeurIPS*, 2017, pp. 3574–3583.
- [6] differential privacy team at Apple, “Learning with privacy at scale,” 2017. [Online]. Available: <https://machinelearning.apple.com/research/learning-with-privacy-at-scale>
- [7] H. Arcolezi, J. Couchot, B. Al Bouna, and X. Xiao, “Random sampling plus fake data: Multidimensional frequency estimates with local differential privacy,” in *Proc. of ACM CIKM*, 2021, pp. 47–57.
- [8] P. Kairouz, S. Oh, and P. Viswanath, “Extremal mechanisms for local differential privacy,” *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 492–542, 2016.
- [9] T. Wang, J. Blocki, N. Li, and S. Jha, “Locally differentially private protocols for frequency estimation,” in *Proc. USENIX Security*, 2017, pp. 729–745.
- [10] H. H. Arcolezi, C. Pinzón, C. Palamidessi, and S. Gambs, “Frequency estimation of evolving data under local differential privacy,” *arXiv preprint arXiv:2210.00262*, 2022.

- [11] H. H. Arcolezi, J.-F. Couchot, B. Boua, and X. Xiao, "Improving the utility of locally differentially private protocols for longitudinal and multidimensional frequency estimates," digital communications and networks (2022)," 2022.
- [12] O. Ohri menko, A. Wirth, and H. Wu, "Randomize the future: Asymptotically optimal locally private frequency estimation protocol for longitudinal data," in *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2022, pp. 237–249.
- [13] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. Hui, H. Shin, J. Shin, and G. Yu, "Collecting and analyzing multidimensional data with local differential privacy," in *Proc. IEEE ICDE*, 2019, pp. 638–649.
- [14] Q. Ye, H. Hu, X. Meng, and H. Zheng, "Privkv: Key-value data collection with local differential privacy," in *Proc. IEEE SP*, 2019, pp. 317–331.
- [15] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren, "Heavy hitter estimation over set-valued data with local differential privacy," in *Proc. ACM CCS*, 2016, pp. 192–203.
- [16] J. Li, W. Gan, Y. Gui, Y. Wu, and P. S. Yu, "Frequent itemset mining with local differential privacy," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 1146–1155.
- [17] S. Wang, L. Huang, Y. Nie, P. Wang, H. Xu, and W. Yang, "Privset: Set-valued data analyses with locale differential privacy," in *IEEE INFOCOM 2018–IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1088–1096.
- [18] T. Wang, N. Li, and S. Jha, "Locally differentially private frequent itemset mining," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 127–143.
- [19] J. Imola, T. Murakami, and K. Chaudhuri, "Locally differentially private analysis of graph statistics," in *30th USENIX security symposium (USENIX Security 21)*, 2021, pp. 983–1000.
- [20] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, "Towards locally differentially private generic graph metric estimation," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1922–1925.
- [21] T. Wang, J.-Q. Chen, Z. Zhang, D. Su, Y. Cheng, Z. Li, N. Li, and S. Jha, "Continuous release of data streams under both centralized and local differential privacy," in *Proc. ACM CCS*, 2021, pp. 1237–1253.
- [22] E. Bao, Y. Yang, X. Xiao, and B. Ding, "Cgm: An enhanced mechanism for streaming data collection with local differential privacy," *Proc. of VLDB Endow.*, vol. 14, no. 11, pp. 2258–2270, 2021.
- [23] X. Ren, L. Shi, W. Yu, S. Yang, C. Zhao, and Z. Xu, "Ldp-ids: Local differential privacy for infinite data streams," in *Proceedings of the 2022 international conference on management of data*, 2022, pp. 1064–1077.
- [24] C. Schäler, T. Hüttner, and M. Schäler, "Benchmarking the utility of w-event differential privacy mechanisms—when baselines become mighty competitors," *Proceedings of the VLDB Endowment*, vol. 16, no. 8, pp. 1830–1842, 2023.
- [25] Y. Li, X. Lee, B. Peng, T. Palpanas, and J. Xue, "Privsketch: A private sketch-based frequency estimation protocol for data streams," *arXiv preprint arXiv:2306.12144*, 2023.
- [26] M. Joseph, A. Roth, J. Ullman, and B. Waggoner, "Local differential privacy for evolving data," *Proc. NeurIPS*, vol. 31, pp. 2375–2384, 2018.
- [27] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, "Amplification by shuffling: From local to central differential privacy via anonymity," *ArXiv*, vol. abs/1811.12469, 2019.
- [28] J. Zhang, C. Zhang, G. Li, and C. Chai, "Pace: Poisoning attacks on learned cardinality estimation," *Proc. ACM Manag. Data*, vol. 2, no. 1, Mar. 2024. [Online]. Available: <https://doi.org/10.1145/3639292>
- [29] E. M. Kornaropoulos, S. Ren, and R. Tamassia, "The price of tailoring the index to your data: Poisoning attacks on learned index structures," in *Proceedings of the 2022 International Conference on Management of Data*, ser. SIGMOD '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1331–1344. [Online]. Available: <https://doi.org/10.1145/3514221.3517867>
- [30] X. Cao, J. Jia, and N. Z. Gong, "Data poisoning attacks to local differential privacy protocols," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 947–964.
- [31] Y. Wu, X. Cao, J. Jia, and N. Z. Gong, "Poisoning attacks to local differential privacy protocols for {Key-Value} data," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 519–536.
- [32] X. Li, N. Li, W. Sun, N. Z. Gong, and H. Li, "Fine-grained poisoning attack to local differential privacy protocols for mean and variance estimation," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 1739–1756.
- [33] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias, "Differentially private event sequences over infinite streams," *Proc. VLDB Endow.*, vol. 7, pp. 1155–1166, 2014.
- [34] Z. Zhang, T. Wang, N. Li, S. He, and J. Chen, "Calm: Consistent adaptive local marginal for marginal release under local differential privacy," in *Proc. ACM CCS*, 2018, pp. 212–229.
- [35] J. Duchi, M. Wainwright, and M. Jordan, "Minimax optimal procedures for locally private estimation," *J. Am. Stat. Assoc.*, vol. 113, pp. 182 – 201, 2016.
- [36] Q. Wang, Y. Zhang, X. Lu, Z. Wang, Z. Qin, and K. Ren, "Rescuedp: Real-time spatio-temporal crowd-sourced data publishing with differential privacy," *Proc. IEEE INFOCOM*, pp. 1–9, 2016.
- [37] H. Li, L. Xiong, X. Jiang, and J. Liu, "Differentially private histogram publication for dynamic datasets: An adaptive sampling approach," in *Proc. of ACM CIKM*, 2015, p. 1001–1010.
- [38] Y. Chen, A. Machanavajjhala, M. Hay, and G. Miklau, "Pegasus: Data-adaptive differentially private stream processing," *Proc. ACM CCS*, pp. 1375–1388, 2017.
- [39] L. Fan and L. Xiong, "An adaptive approach to real-time aggregate monitoring with differential privacy," *IEEE Trans. on Knowl. Data Eng.*, vol. 26, pp. 2094–2106, 2014.
- [40] K. Thomas, D. McCoy, C. Grier, A. Kolcz, and V. Paxson, "{Trafficking} fraudulent accounts: The role of the underground market in twitter spam and abuse," in *22nd USENIX Security Symposium (USENIX Security 13)*, 2013, pp. 195–210.
- [41] W. Tong, H. Chen, J. Niu, and S. Zhong, "Data poisoning attacks to locally differentially private frequent itemset mining protocols," in *Proc. ACM CCS*, 2024.
- [42] Q. Huangfu and J. J. Hall, "Parallelizing the dual revised simplex method," *Mathematical Programming Computation*, vol. 10, no. 1, pp. 119–142, 2018.
- [43] X. Cao, J. Jia, and N. Z. Gong, "Provably secure federated learning against malicious clients," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 8, 2021, pp. 6885–6893.
- [44] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.
- [45] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, pp. 211–407, 2014.
- [46] C. Dwork, "Differential privacy," in *Proc. ICALP*, 2006, pp. 1–12.
- [47] Q. Xue, Q. Ye, H. Hu, Y. Zhu, and J. Wang, "Ddrm: A continual frequency estimation mechanism with local differential privacy," *IEEE Transactions on Knowledge and Data Engineering*, 2022.
- [48] X. Li, Y. Cao, and M. Yoshikawa, "Locally private streaming data release with shuffling and subsampling," in *2023 IEEE 39th International Conference on Data Engineering Workshops (ICDEW)*. IEEE, 2023, pp. 125–131.
- [49] W. Gao and S. Zhou, "Privacy-preserving for dynamic real-time published data streams based on local differential privacy," *IEEE Internet of Things Journal*, 2023.
- [50] Q. Ye, H. Hu, N. Li, X. Meng, H. Zheng, and H. Yan, "Beyond value perturbation: Local differential privacy in the temporal setting," in *IEEE INFOCOM 2021–IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.
- [51] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum, "Differential privacy under continual observation," in *Proc. ACM STOC*, 2010, pp. 715–724.
- [52] A. Cheu, A. Smith, and J. Ullman, "Manipulation attacks in local differential privacy," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 883–900.
- [53] Z. Zheng, Z. Li, C. Huang, S. Long, M. Li, and X. Shen, "Data poisoning attacks and defenses to ldp-based privacy-preserving crowdsensing," *IEEE Transactions on Dependable and Secure Computing*, 2024.
- [54] Z. Li, T. Wang, M. Lopuhää-Zwakenberg, N. Li, and B. Škoric, "Estimating numerical distributions under local differential privacy," in *Proc. ACM SIGMOD*, 2020, pp. 621–635.

A Proofs of Theorems and Corollaries

A.1 Proof of Theorem 4.1

We consider to manipulate an FO with the privacy budget of ϵ . Suppose that the solution to the above optimization problem is $(m^*[1], m^*[2], \dots, m^*[d])$. Then, the expected manipulation gap for IPMA $G_{I,t}(m, n, \epsilon)$ can be calculated as follows:

$$\begin{aligned} G_{I,t}(m, n, \epsilon) &= \mathbb{E}\left(\frac{1}{d} \sum_{k=1}^d (\hat{f}_t[k] - \tilde{f}_t[k])^2\right) \\ &= \frac{1}{d} \sum_{k=1}^d \left(\text{Var}(\hat{f}_t[k]) + (\mathbb{E}(\hat{f}_t[k]) - \tilde{f}_t[k])^2 \right) \\ &= \frac{1}{d} \sum_{k=1}^d \left(\frac{m^*[k] + n \cdot f_t[k]}{n+m} - \tilde{f}_t[k] \right)^2 + \text{Var}(n+m, \epsilon) \quad (17) \end{aligned}$$

where $\text{Var}(n+m, \epsilon) = \frac{1}{d} \sum_{k=1}^d \text{Var}(\hat{f}_t[k])$. The solution to the optimization problem is independent of ϵ , affecting only the term $\text{Var}(n+m, \epsilon)$ in terms of attack impact.

A.2 Proof of Corollary 4.2

The optimization problem is unrelated to ϵ , so $m^*[k]$ is independent of ϵ . In $G_{I,t}$, only $\text{Var}(n+m, \epsilon)$ varies with ϵ : a larger ϵ reduces Var , decreasing $G_{I,t}(m, n, \epsilon)$.

A.3 Proof of Corollary 4.3

Suppose m and n become $\alpha > 1$ times larger, and the solution to the optimization problem before expanding m and n (denoted as problem A) is $(m^*[1], m^*[2], \dots, m^*[d])$. For the optimization problem after expanding m and n (called problem B), we can replace $m[k]$ with $m'[k]$, where $m'[k] = m[k]/\alpha$. Thus, problem B can be transformed into a problem similar to problem A. Solving $m'[k]$ of problem B is equivalent to solving $m[k]$ of problem A. So, considering that the solution to problem A is $(m^*[1], \dots, m^*[d])$, $(m'[1], \dots, m'[d])$ can be solved as $(m^*[1], \dots, m^*[d])$. Thus, the solution to problem B is $(\alpha m^*[1], \alpha m^*[2], \dots, \alpha m^*[d])$. The first term of $G_{I,t}(m, n, \epsilon)$ after expanding m and n would be equal to the first term of $G_{I,t}(m, n, \epsilon)$ before expanding m and n . However, Var becomes smaller after expanding m and n . Hence, if $n+m$ is larger, $G_{I,t}(m, n, \epsilon)$ will be smaller.

A.4 Proof of Corollary 4.4

When m satisfies Eq. (8), it is easy to prove that one feasible solution of Eq. (6) is $m[k] = (m+n)\tilde{f}_t[k] - nf_t[k]$, i.e., $m[k]$ satisfies $\sum_{k=1}^d m[k] = m$ and $0 \leq m[k] \leq m$. Also, when $m[k] = (m+n)\tilde{f}_t[k] - nf_t[k]$,

$$\begin{aligned} &\frac{1}{d} \sum_{k=1}^d (\hat{f}_t[k] - \tilde{f}_t[k])^2 \\ &= \frac{1}{d} \sum_{k=1}^d \left(\frac{m[k] + n \cdot f_t[k]}{n+m} - \tilde{f}_t[k] \right)^2 = 0, \quad (18) \end{aligned}$$

which reaches the minimum of quadratic sum. Considering that Eq. (6) is a convex optimization problem which has only one global optimal solution and $\frac{1}{d} \sum_{k=1}^d (\hat{f}_t[k] - \tilde{f}_t[k])^2$ reaches the minimum when $m[k] = (m+n)\tilde{f}_t[k] - nf_t[k]$, $m[k] = (m+n)\tilde{f}_t[k] - nf_t[k]$ is the optimal solution of Eq. (6). Substitute $m^*[k] = (m+n)\tilde{f}_t[k] - nf_t[k]$ into Eq. (7), $G_{I,t}$ can be calculated as $G_{I,t}^*(m, n, \epsilon) = \text{Var}(n+m, \epsilon)$.

A.5 Proof of Theorem 4.5

Suppose the attacker targets a publication strategy FO whose privacy budget is ϵ and the solution to the above optimization problem is $(m^*[1], m^*[2], \dots, m^*[d])$. Thus, the manipulation gap for OPMA $G_{O,t}(m, n, \epsilon)$ can be calculated as follows:

$$\begin{aligned} G_{O,t}(m, n, \epsilon) &= \mathbb{E}\left(\frac{1}{d} \sum_{k=1}^d (\hat{f}_t[k] - \tilde{f}_t[k])^2\right) \\ &= \frac{1}{d} \sum_{k=1}^d \left(\frac{nf_t[k](p-q) + m^*[k] - mq}{(m+n)(p-q)} - \tilde{f}_t[k] \right)^2 \\ &\quad + \frac{1}{d} \sum_{k=1}^d \text{Var}\left(\frac{\sum_{j=1}^n \mathbb{I}_S^{(k)}(y_j) + \sum_{j=1}^m \mathbb{I}_S^{(k)}(z_j) - q(m+n)}{(m+n)(p-q)}\right). \quad (19) \end{aligned}$$

For the second term, $\sum_{j=1}^m \mathbb{I}_S^{(k)}(z_j)$ is treated as a constant as the optimization problem is solved as $(m^*[1], m^*[2], \dots, m^*[d])$. So,

$$\begin{aligned} G_{O,t}(m, n, \epsilon) &= \frac{1}{d} \sum_{k=1}^d \left(\frac{nf_t[k](p-q) + m^*[k] - mq}{(m+n)(p-q)} - \tilde{f}_t[k] \right)^2 \\ &\quad + \left(\frac{n}{m+n} \right)^2 \text{Var}(n, \epsilon). \quad (20) \end{aligned}$$

A.6 Proof of Corollary 4.6

When m satisfies Eq. (12), it is easy to prove that one feasible solution of $\min \frac{1}{d} \sum_{k=1}^d |\mathbb{E}(\hat{f}_t[k]) - \tilde{f}_t[k]|$ is $m[k] = (p-q)[(m+n)\tilde{f}_t[k] - nf_t[k]] + mq$, i.e., $m[k]$ satisfies $\sum_{k=1}^d m[k] = m$ and $0 \leq m[k] \leq m$ when attacking kRR or $0 \leq m[k] \leq m$ when attacking OUE. Also, when $m[k] = (p-q)[(m+n)\tilde{f}_t[k] - nf_t[k]] + mq$,

$$\begin{aligned} &\frac{1}{d} \sum_{k=1}^d |\mathbb{E}(\hat{f}_t[k]) - \tilde{f}_t[k]| \\ &= \frac{1}{d} \sum_{k=1}^d \left(\frac{nf_t[k](p-q) + m[k] - mq}{(m+n)(p-q)} - \tilde{f}_t[k] \right)^2 = 0, \quad (21) \end{aligned}$$

which reaches the minimum of sum of absolute values. Note that, the optimization problem (i.e., $\min \frac{1}{d} \sum_{k=1}^d |\mathbb{E}(\hat{f}_t[k]) - \tilde{f}_t[k]|$) is convex with the only optimal solution $m[k] = (p-q)[(m+n)\tilde{f}_t[k] - nf_t[k]] + mq$. So, we substitute $m^*[k] = (p-q)[(m+n)\tilde{f}_t[k] - nf_t[k]] + mq$ into Eq. (11), we can obtain that $G_{O,t}^*(m, n, \epsilon) = \left(\frac{n}{m+n}\right)^2 \text{Var}(n, \epsilon) = \frac{n}{m+n} \text{Var}(n+m, \epsilon)$.

A.7 Proof of Corollary 4.7

According to Eq. (13), when ϵ is larger, Var becomes smaller and $G_{O,t}^*(m, n, \epsilon)$ becomes smaller.

A.8 Proof of Corollary 4.8

The proof of Corollary 4.8 is similar to that of Corollary 4.3.

A.9 Proof of Theorem 4.9

We define $n[k] = n^e \mathbf{f}_t^e[k]$ and $b_k = n[k]/(m + n^e) - \hat{\mathbf{f}}_{t-1}[k]$ with $k^* = \arg \max_{k \in [1, \dots, d]} (b_k)$, aiming to maximize the formula.

$$\begin{aligned} \frac{1}{d} \sum_{k=1}^d (\mathbf{f}[k] - \hat{\mathbf{f}}_{t-1}[k])^2 &= \frac{1}{d} \sum_{k=1}^d \left(\frac{m[k]}{n^e + m} + b_k \right)^2 \\ &= \frac{1}{d(n^e + m)^2} \sum_{k=1}^d m[k]^2 + \frac{2}{d(n^e + m)} \sum_{k=1}^d m[k] b_k + \frac{1}{d} \sum_{k=1}^d b_k^2 \quad (22) \end{aligned}$$

For the first term, the maximum value is taken if one of $m[k]$ ($m[i]$) is m and the other $m[k]$ (e.g., $k \neq i$) is 0. For the second term, the maximum value is taken if $m[k^*] = m$ and other $m[k]$ ($k \neq k^*$) are 0. And the third term is a constant. So the solution to this problem is $m[k^*] = m$ and $m[k] = 0$ ($k \neq k^*$).

A.10 Proof of Theorem 4.10

It's assumed that the attacker targets a FO with a privacy budget of ϵ . Real users have outputs (y_1, y_2, \dots, y_n) , and fake users have outputs (z_1, z_2, \dots, z_m) . Let $m[k] = \mathbb{E}(\sum_{j=1}^m \mathbb{I}_{S(z_j)}^{(k)})$. So for $\forall k \in [1, \dots, d]$,

$$\begin{aligned} \mathbb{E}\left(\left(\bar{\mathbf{f}}_t[k] - \hat{\mathbf{f}}_{t-1}[k]\right)^2\right) &= \left(\frac{n\mathbf{f}_t[k](p-q) + m[k] - mq}{(m+n)(p-q)} - \hat{\mathbf{f}}_{t-1}[k]\right)^2 \\ &+ \text{Var}\left(\frac{\sum_{j=1}^n \mathbb{I}_{S(y_j)}^{(k)} + \sum_{j=1}^m \mathbb{I}_{S(z_j)}^{(k)} - q(m+n)}{(m+n)(p-q)}\right) \quad (23) \end{aligned}$$

Here, for the second term, (z_1, z_2, \dots, z_m) are random variables, whose variance are hard to determine, while (y_1, y_2, \dots, y_n) are outputs of local perturbation, whose variance has been shown in [9]. Thus, Let $Y_k = \frac{\sum_{j=1}^n \mathbb{I}_{S(y_j)}^{(k)}}{(m+n)(p-q)}$ and $Z_k = \frac{\sum_{j=1}^m \mathbb{I}_{S(z_j)}^{(k)}}{(m+n)(p-q)}$,

$$\begin{aligned} \text{Var}\left(\frac{\sum_{j=1}^n \mathbb{I}_{S(y_j)}^{(k)} + \sum_{j=1}^m \mathbb{I}_{S(z_j)}^{(k)} - q(m+n)}{(m+n)(p-q)}\right) \\ = \text{Var}(Y_k) + \text{Var}(Z_k) + 2\text{Cov}(Y_k, Z_k) \\ = \text{Var}(Y_k) + E(Z_k^2) - (E(Z_k))^2 + 2(E(Y_k Z_k) - E(Y_k) E(Z_k)), \quad (24) \end{aligned}$$

where $\mathbb{E}(Z_k^2) \geq 0$ and $\mathbb{E}(Y_k Z_k) \geq 0$. Thus, we have

$$\begin{aligned} \mathbb{E}\left(\left(\bar{\mathbf{f}}_t[k] - \hat{\mathbf{f}}_{t-1}[k]\right)^2\right) \\ \geq \left(\frac{n\mathbf{f}_t[k](p-q) + m[k] - mq}{(m+n)(p-q)} - \hat{\mathbf{f}}_{t-1}[k]\right)^2 \\ + \text{Var}(Y_k) - (E(Z_k))^2 - 2E(Y_k) E(Z_k). \quad (25) \end{aligned}$$

where $\mathbb{E}(Z_k) = \frac{m[k]}{(m+n)(p-q)}$, $\mathbb{E}(Y_k) = \frac{\sum_{j=1}^n \mathbb{I}_{S(y_j)}^{(k)}}{(m+n)(p-q)} = \frac{A_k + (m+n)q}{(m+n)(p-q)}$, $A_k = n\mathbf{f}_t[k](p-q) - mq$. Thus,

$$\begin{aligned} &\mathbb{E}\left(\left(\bar{\mathbf{f}}_t[k] - \hat{\mathbf{f}}_{t-1}[k]\right)^2\right) \\ &\geq \left(\frac{A_k + m[k]}{(m+n)(p-q)} - \hat{\mathbf{f}}_{t-1}[k]\right)^2 + \frac{n^2}{(m+n)^2} \text{Var}\left(\frac{\sum_{j=1}^n \mathbb{I}_{S(y_j)}^{(k)}}{n(p-q)}\right) \\ &- \left(\frac{m[k]}{(m+n)(p-q)}\right)^2 - 2\frac{m[k](A_k + (m+n)q)}{(m+n)^2(p-q)^2} \\ &= \frac{A_k^2 - 2m[k](nq + mq)}{(p-q)^2(m+n)^2} + \frac{n^2}{(m+n)^2} \text{Var}\left(\frac{\sum_{j=1}^n \mathbb{I}_{S(y_j)}^{(k)} - nq}{n(p-q)}\right) \\ &- 2\frac{A_k + m[k]}{(m+n)(p-q)} \hat{\mathbf{f}}_{t-1}[k] + (\hat{\mathbf{f}}_{t-1}[k])^2 \quad (26) \end{aligned}$$

where $\sum_{k=1}^d \text{Var}\left(\frac{\sum_{j=1}^n \mathbb{I}_{S(y_j)}^{(k)} - nq}{n(p-q)}\right) = d \text{Var}(n, \epsilon)$. After summing the inequality from $k = 1$ to d , disregarding constants like $d \text{Var}(n, \epsilon)$ and focusing on $m[k]$ terms, maximizing \bar{dis} becomes an optimization problem to maximize its lower bound.

$$\max \sum_{k=1}^d \frac{-2\hat{\mathbf{f}}_{t-1}[k](m+n)(p-q) - 2(m+n)q}{(p-q)^2(m+n)^2} m[k] \quad (27)$$

Also, $0 \leq m[k] \leq m$ and for kRR, $\sum_{k=1}^d m[k] = m$. To solve this problem, we only need calculate the coefficient in front of $m[k]$ (i.e. find the minimum $\hat{\mathbf{f}}_{t-1}[k]$). Assume the index of the maximum coefficient (the minimum $\hat{\mathbf{f}}_{t-1}[k]$) is $k^* \in [d]$. Then, the solution is $m[k^*] = m$, $m[k] = 0$ ($k \neq k^*$).

B Experimental Results

B.1 Results on Real-World Datasets

The average manipulation gap against adaptive methods for real-world datasets with varying β , ϵ , w , n^e , and \mathbf{f}^e are shown in Fig. 20, Fig. 21, Fig. 22, Fig. 23, and Fig. 24, respectively.

B.2 Result on Different FOs, Varying d

Fig. 14 shows attack effectiveness against Ada, kRR and OUE on Taobao dataset with Gaussian target, varying d .

B.3 Result on Different Levels of Knowledge

Fig. 15 shows attack effectiveness using OAA on Taxi dataset with different levels of knowledge, varying ϵ .

B.4 Result on Baseline Methods

Results for baseline methods are shown in Fig. 25.

B.5 Results on Mean Estimation

Figs. 16 and 17 shows attack effectiveness on HM-based LDP-IDS, Taobao-Price and Taxi-Longitude datasets, varying ϵ , w and β .

B.6 Results on Stream LDPs over Numerical domain

Fig. 18 presents attack effectiveness on stream LDPs over numerical domain on Taxi-Longitude and Taobao-Price datasets.

B.7 Results on Defense

Fig. 19 shows accuracy gain on different r .

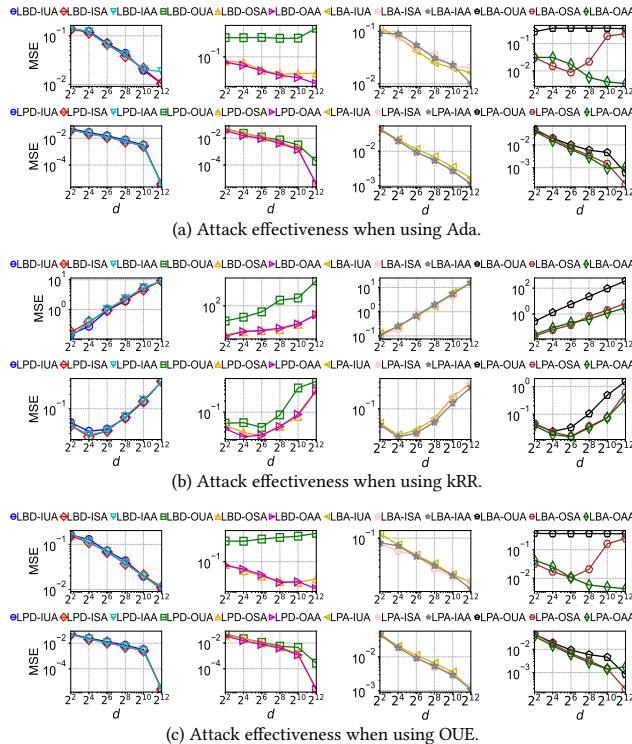


Figure 14: Attack effectiveness, varying d (Taobao with Gaussian target).

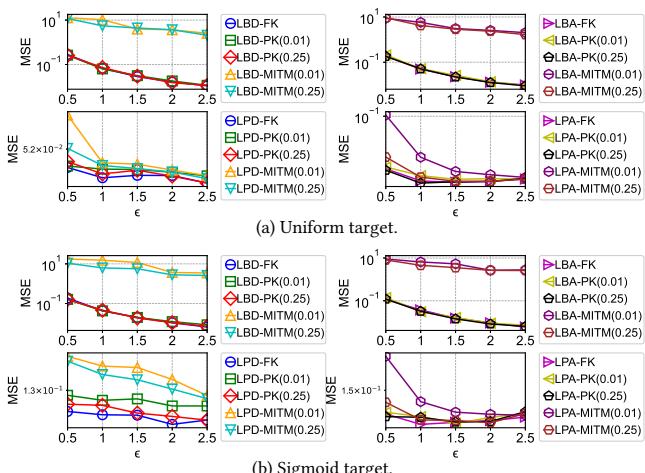


Figure 15: Attack effectiveness using OAA with different levels of knowledge, varying ϵ (Taxi).

B.8 Results on Synthetic Datasets

We created binary streaming datasets using various sequence models. Starting with a probability process model $p_t = f(t)$, time T , and user count N , we generated a sequence (p_1, p_2, \dots, p_T) . At each timestamp t , we randomly assigned p_t fraction of N users a value of 1, with others set to 0. p_t is defined accordingly.

- LNS is a linear process $p_t = p_{t-1} + \mathcal{N}(0, Q_\sigma)$, where $p_0 = 0.5$ and $\mathcal{N}(0, Q_\sigma)$ is Gaussian noise with the standard variance $\sqrt{Q_\sigma} = 0.025$

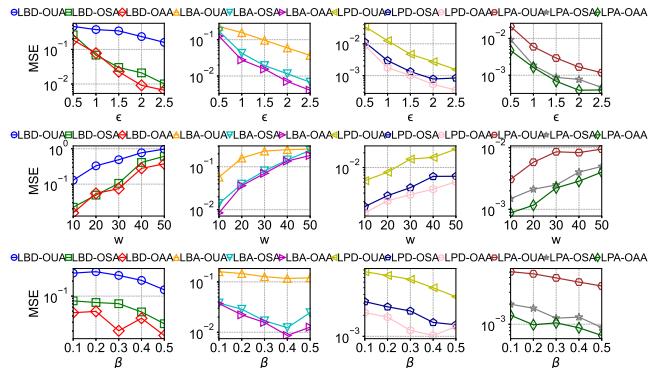


Figure 16: Attack effectiveness on HM-based LDP-IDS, varying ϵ , w and β (Taobao-Price with Uniform target).

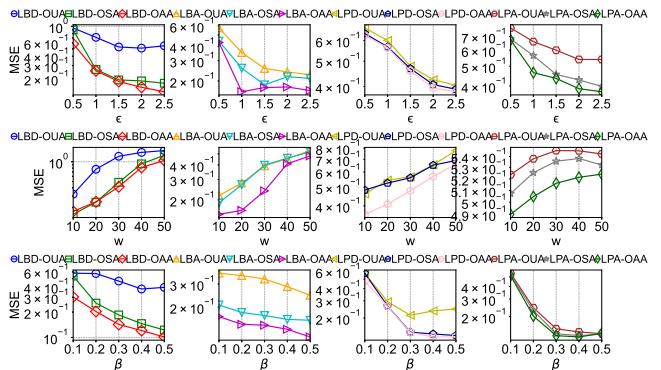


Figure 17: Attack effectiveness on HM-based LDP-IDS, varying ϵ , w and β (Taobao-Price with Uniform target).

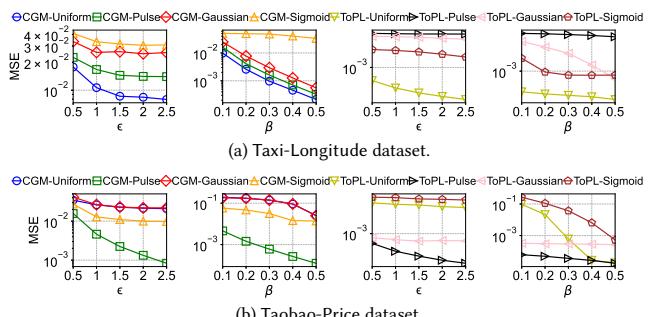
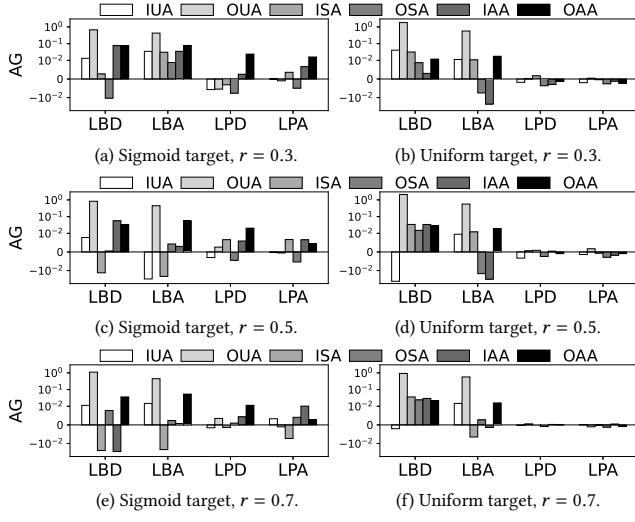


Figure 18: Attack effectiveness on stream LDPs over numerical domain.

- **Sin** is a sequence composed by a sine curve $p_t = A\sin(bt) + h$ with $A = 0.05$, $b = 0.01$ and $h = 0.5$.
 - **Log** is a series with the logistic model $p_t = A/(1 + e^{-bt})$ where $A = 0.75$ and $b = 0.01$.
 - **Pulse** is an extreme case that randomly sets p_t as 0 or 1.

Using default parameters and models, we generated synthetic binary streams from 100,000 users across 800 timestamps.

Our experimental results on these datasets are depicted in Figs. 26, 27, 28, 29, and 30, illustrating attack impacts by varying β , ϵ , w , n^e , and f^e .

Figure 19: Accuracy Gain (Taxi, $\epsilon = 1$).

C Attacking Mean Estimation over Numerical Domain

Apply HM to LDP-IDS [23]. The Hybrid Mechanism [13] is an LDP mean estimation technique merging Stochastic Rounding (SR) [35] and Piecewise Mechanism (PM) [13] for minimal error. For $\epsilon > 0.61$, it employs PM with probability $1 - e^{-\epsilon/2}$ and SR with $e^{-\epsilon/2}$. Below $\epsilon \leq 0.61$, it solely uses SR. Considering that in practice users' private values $v \in [-1, 1]$ are close to 0, the worst case variance of the perturbed value y in HM is written as

$$\text{Var}^*[y] = \begin{cases} \left(\frac{e^\epsilon + 1}{e^\epsilon - 1}\right)^2, & \text{if } \epsilon \leq 0.61 \\ \frac{1}{e^{\epsilon/2}} \left[\left(\frac{e^\epsilon + 1}{e^\epsilon - 1}\right)^2 + \frac{e^{\epsilon/2} + 3}{3(e^{\epsilon/2} - 1)^2} \right], & \text{if } \epsilon > 0.61 \end{cases}$$

The variance of the mean estimate $f = \frac{1}{n} \sum_{j=1}^n y_j$ (denote the perturbed value of j -th user as y_j) can be calculated as $\text{Var}[f] = \frac{1}{n^2} \sum_{j=1}^n \text{Var}[y_j]$. Simplifying, we use $\text{Var}[f] = \frac{1}{n} \text{Var}^*[y]$, denoted as $\text{Var}(n, \epsilon)$. The HM mechanism can then replace FOs in the LDP-IDS framework for mean estimation.

Attack HM-based LDP-IDS. [32] introduced a fine-grained poisoning attack, Output Poisoning Attack (OPA), for SR and PM to manipulate estimated means and variances. For SR, the mean estimation manipulation gap of OPA is given by $\frac{2n-2(p-q)^2S^{(2)}}{(m+n)^2(p-q)^2} + \frac{S^{(2)}}{(m+n)^2} + (\frac{n^e-n}{m+n}\mu_t + \frac{S^{(1)}-S_e^{(1)}}{m+n})^2$ where $p = \frac{e^\epsilon}{1+e^\epsilon}$, $q = 1 - p$, and $S^{(1)}, S^{(2)}$ are sums of genuine user inputs and their squares, respectively, and $S_e^{(1)}$ is the attacker-estimated $S^{(1)}$. For PM, it's $(\frac{n^e-n}{m+n}\mu_t + \frac{S^{(1)}-S_e^{(1)}}{m+n})^2 + \frac{2n(e^{\epsilon/2}+3)}{3(m+n)^2(e^{\epsilon/2}-1)^2} + \frac{(1+e^{\epsilon/2})S^{(2)}}{(m+n)^2(e^{\epsilon/2}-1)}$. OPA maintains security-privacy consistency [32] and improves as m and n scale equally, making it suitable for replacing PMAs in Adaptive Attack frameworks. For DMAs, when attackers want to minimize dissimilarity, they just need to set the target value as the last estimated mean and use OPA to attack. When attackers want to maximize dissimilarity, they can set the target as an extreme value (e.g., -1 or 1 when data is normalized to [-1, 1]) and use OPA to

attack. Manipulation gaps are calculated using the formulas above, noting that only half the users are used for mean estimation in [32], requiring adjustment by doubling m and n for exclusive mean estimation.

D Attacking Other Streaming LDP Algorithms

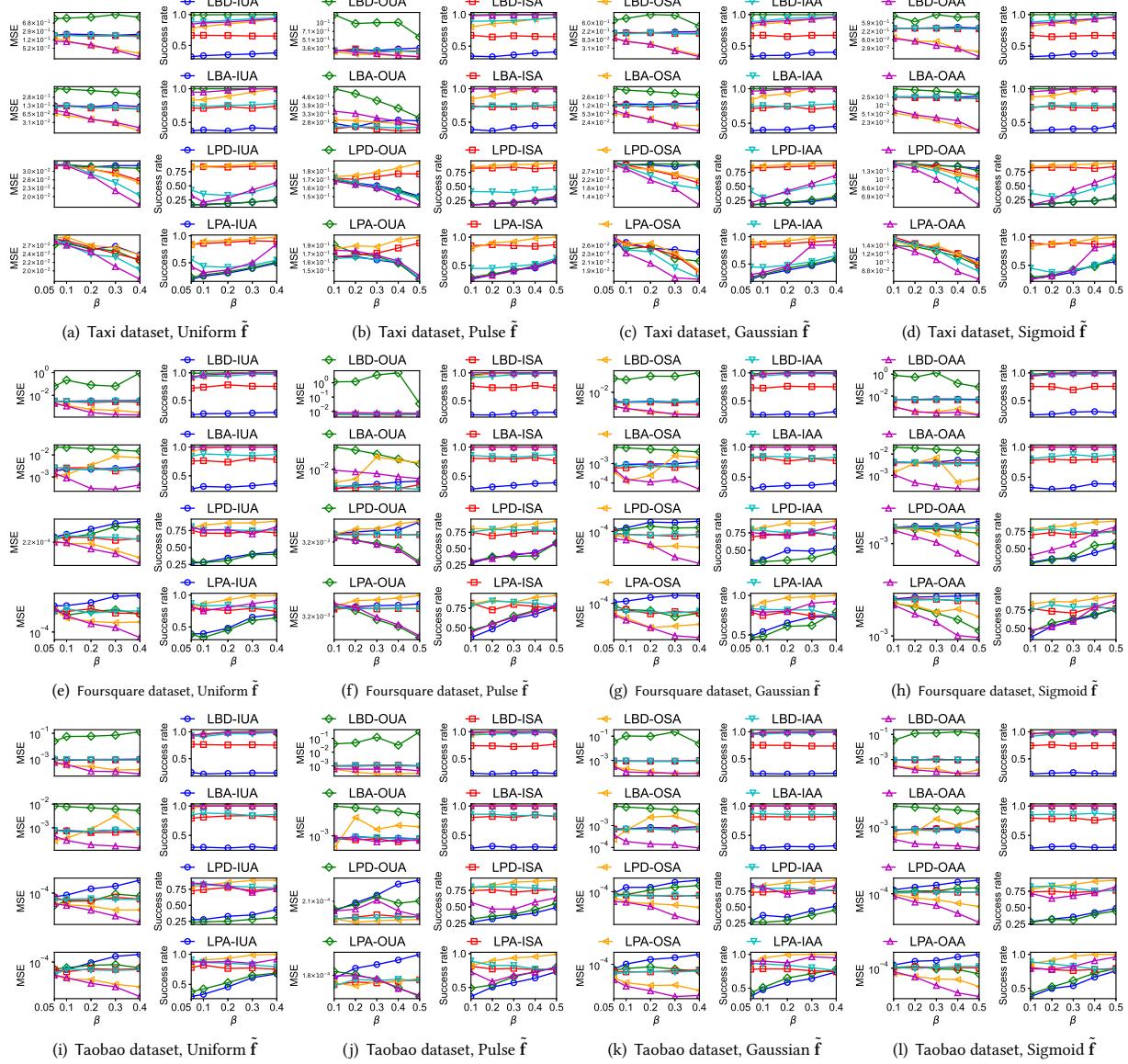
LDP protocols for data streams, which often employ methods like budget and population division for better utility, can be exploited by attackers who mimic these methodologies to optimize attack strategies for improved results, as discussed in Sec. 4.4. This approach is effective because both data estimation and our attacks benefit from larger privacy budgets or populations, enhancing estimation and attack outcomes.

Besides LDP-IDS, streaming algorithms like event-level private PeGaSus [38], user-level private FAST [39], and w -event private DSAT_w [37] and RescueDP [36] can be adapted to LDP settings. Our attack methods are also applicable to these protocols. We also consider two LDP streaming algorithms over numerical domains, CGM [22] and ToPL [21].

PeGaSus. [38] describes an event-level CDP method with three components: perturber, grouper, and smoother. The perturber adds noise independently at each timestamp, while the grouper divides the stream into partitions, and the smoother post-processes these partitions. In event-level-LDP PeGaSus, the perturber allocates ϵ LDP budget to all users with FOs at each timestamp, generating an estimate \hat{f}_t . The grouper then partitions the stream based on these estimates without additional budget, defining deviation as the squared distance within a partition. The smoother operates as in the CDP algorithm. Steps (2) and (3) are aggregator-handled, aiming for improved data stream estimation. Attackers focusing on Step (1) can manipulate the initial estimates by sending tailored data. Since PeGaSus, like LBU and LPU, performs data estimation at each timestamp, only Uniform Attacks can be adopted for attacks.

FAST. [39] describes FAST, a user-level DP algorithm adaptable to user-event LDP settings, using PID controller-based adaptive sampling and Kalman-filter-based post-processing. In these settings, FAST aggregates data using LDP protocols from all users with a reduced privacy budget of ϵ/M , generating an initial estimate \hat{f}_t with FOs, where M is the number of sampling timestamps. A Kalman filter then refines this into a posterior estimate \hat{f}_t , output at each sampling timestamp. FAST dynamically adjusts the sampling interval using the PID controller. Attacks on FAST can only target the initial estimate \hat{f}_t to influence the result. Like attacking LSP, attackers can only manipulate FAST at sampling timestamps. Thus, only Sampling Attacks are adopted.

DSAT_w is a w -event private adaptation of DSAT [37] for w -event LDP. It allocates a portion of the budget or population for distance computation (dissimilarity budget ϵ_1 or population n_1) and the remainder (publication budget ϵ_2 or population n_2) for data publication. Budget or population usage is tracked at each timestamp to avoid depletion. If depleted, DSAT_w repeats the last release $\hat{f}_t = \hat{f}_{t-1}$. If not, it uses ϵ_1/C for all users or ϵ for n_1/C sampled users for LDP reporting, estimating dissimilarity dis . This dis is compared against an adaptively controlled threshold to decide on publication or approximation. Publication decisions trigger a new reporting round for an updated estimate. Attacks on DSAT_w mirror those on LBD/LBA/LPD/LPA, exploiting the dual data collection

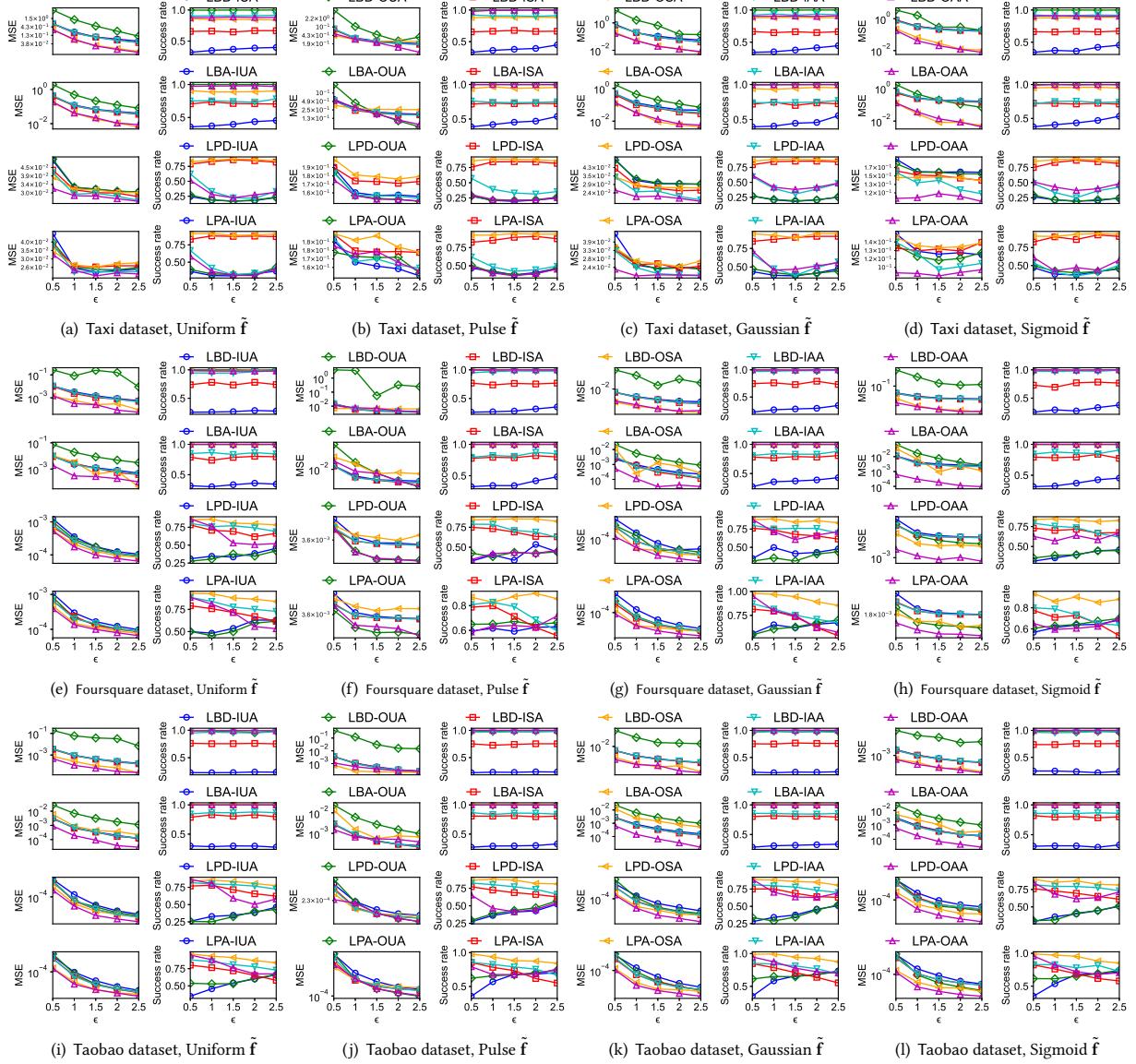
Figure 20: Attacking effectiveness for real-world datasets with varying \tilde{f} .

per timestamp. Attackers target the statistics during LDP aggregation phases. For effective manipulation, both DMAs and PMAs should be designed to influence dissimilarity and control estimates, with mechanisms to decide on publishing or approximating to aid attacks.

RescueDP. [36] is a real-time spatio-temporal data release mechanism that extends FAST [39] with w -event privacy for multi-dimensional data, adaptable to w -event LDP protocols when using FOs. Each timestamp begins with a KF-Prediction to get a prior estimate \hat{f}_t . If it's not a sampling timestamp, this estimate is immediately released as \hat{f}_t . At sampling timestamps, the aggregator calculates the budget or population to be used, then aggregates reports via LDP to derive f_t , and combines these with the prior \hat{f}_t to produce and release a posterior estimate \tilde{f}_t using a Kalman filter. Like FAST, RescueDP dynamically adjusts sampling intervals.

Attacks on RescueDP, similar to those on FAST, can only target \tilde{f}_t to influence outputs. Thus, only Sampling Attack are adopted.

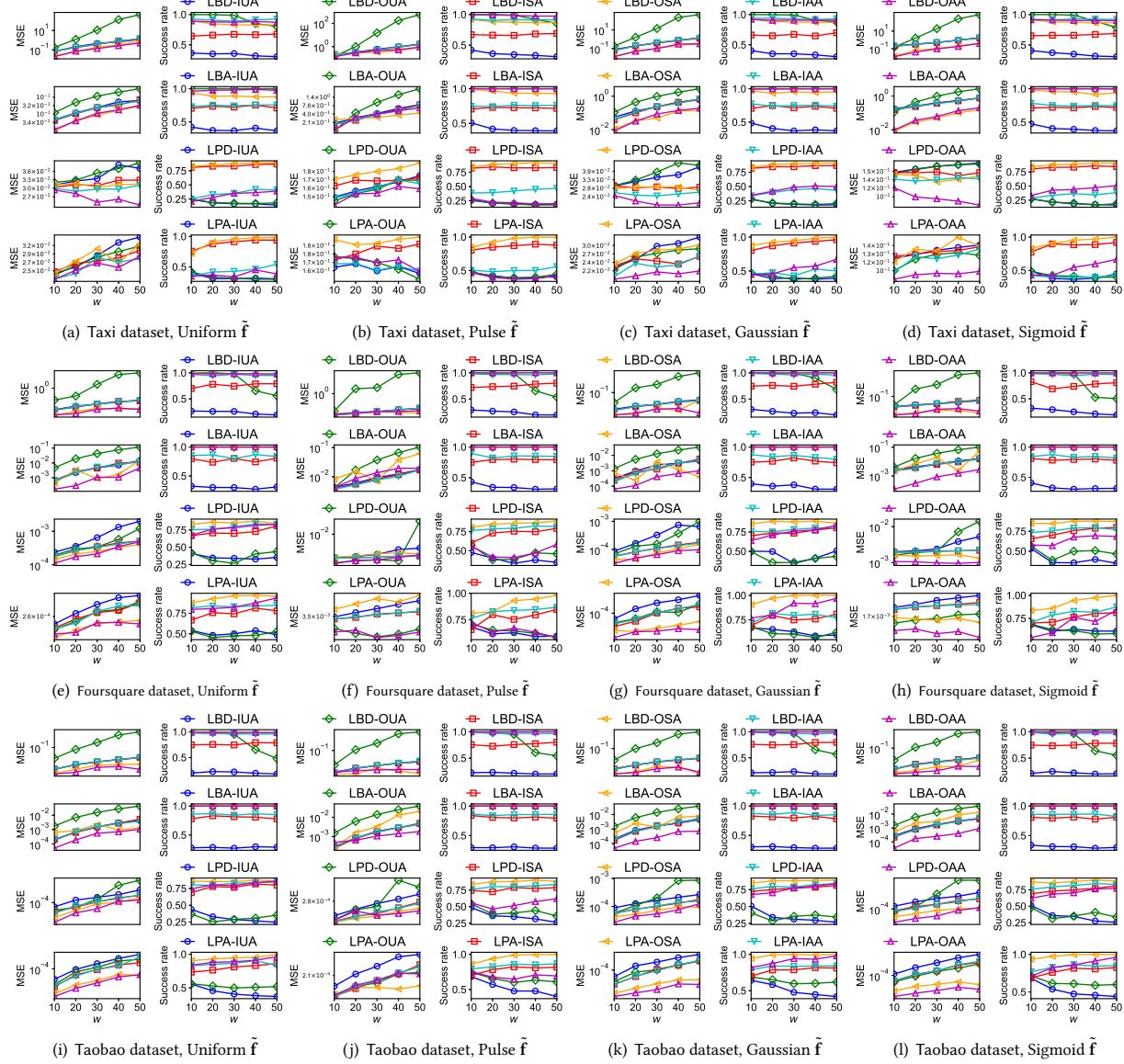
CGM. [22] proposes a novel Correlated Gaussian Mechanism (CGM) for enforcing (ϵ, δ) -LDP on streaming data collection over numerical domains. At each timestamp, CGM injects temporally correlated gaussian noise, computed through an optimization program that takes into account the given autocorrelation pattern, data value range, and utility metric, to the original data for publication. It is noted that CGM is used for processing only one user's data stream. It needs to be run n times to process n users' data streams. We consider the situation that aggregator utilizes CGM for mean value estimation, i.e., the aggregator collects every user's CGM-processed data stream and releases the mean value over all users at each timestamp. For attacks, attackers also hope to manipulate the released mean value to be close to the target mean value \tilde{f}_t at

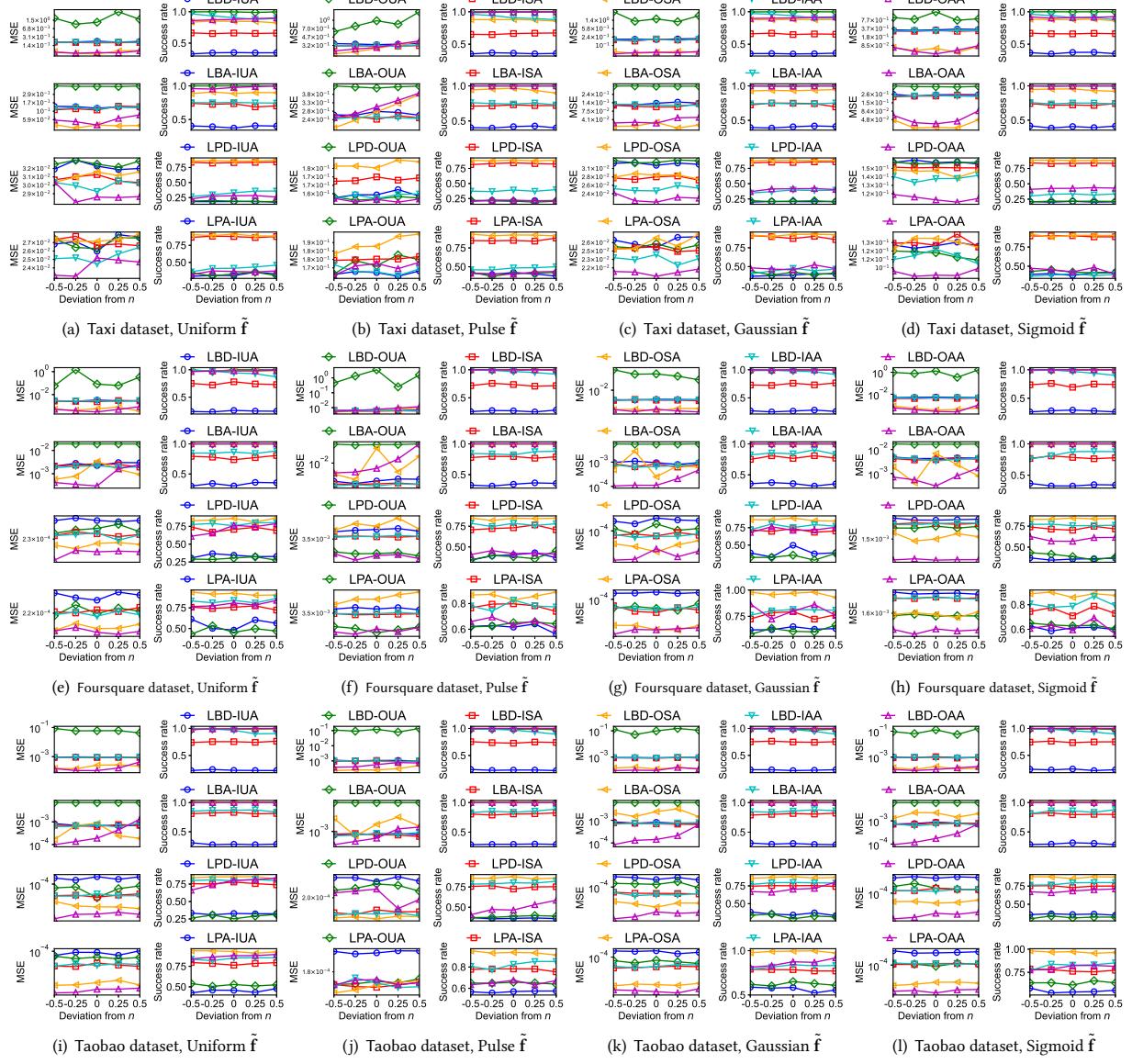
Figure 21: Attack effectiveness for real-world datasets with varying ϵ .

every timestamp. CGM publishes at every timestamp, causing only Uniform Attacks to be adopted. Assume that attackers estimate the number of users and the real mean value as n^e and f_t^e and fake users report z_j , $j \in [1, \dots, m]$ to the aggregator at each timestamp, bypassing the noise injection of CGM. Considering the expectation of gaussian noise injected to genuine users is 0, attackers only need to solve z_j , which satisfies $\frac{n^e \cdot f_t^e + \sum_{j=1}^m z_j}{n^e + m} = \tilde{f}_t$ for attack.

ToPL [21] describes ToPL method for outputting streaming data in event-level LDP setting over numerical domains. ToPL consists of two parts, Threshold Optimizer and Perturber. ToPL first utilizes Threshold Optimizer to cash a period of reports to privately find the optimal threshold θ by minimizing overall estimation errors. After obtaining θ , the sever sends it to all users. When a user reports a value, it will first be truncated to be no larger than θ . The user then

adopts the Hybrid Mechanism (HM) [13], which combines SR [35] and PM [13], as the Perturber to report the truncated value and release the estimated mean values. For attacks, considering that Threshold Optimizer adopts Square Wave (SW) [54] which utilizes Expectation Maximization algorithm to estimate distribution iteratively, we only heuristically adopt IPMA for attack. For Perturber, OPA [32], which is specifically designed for attacking mean value estimation, is used to perform each timestamp attacks on HM. ToPL publishes at every timestamp like LBU/LPU. Thus, only Uniform Attacks are adopted.


 Figure 22: Attacking effectiveness for real-world datasets with varying w .

Figure 23: Attacking effectiveness for real-world datasets with varying n^e .

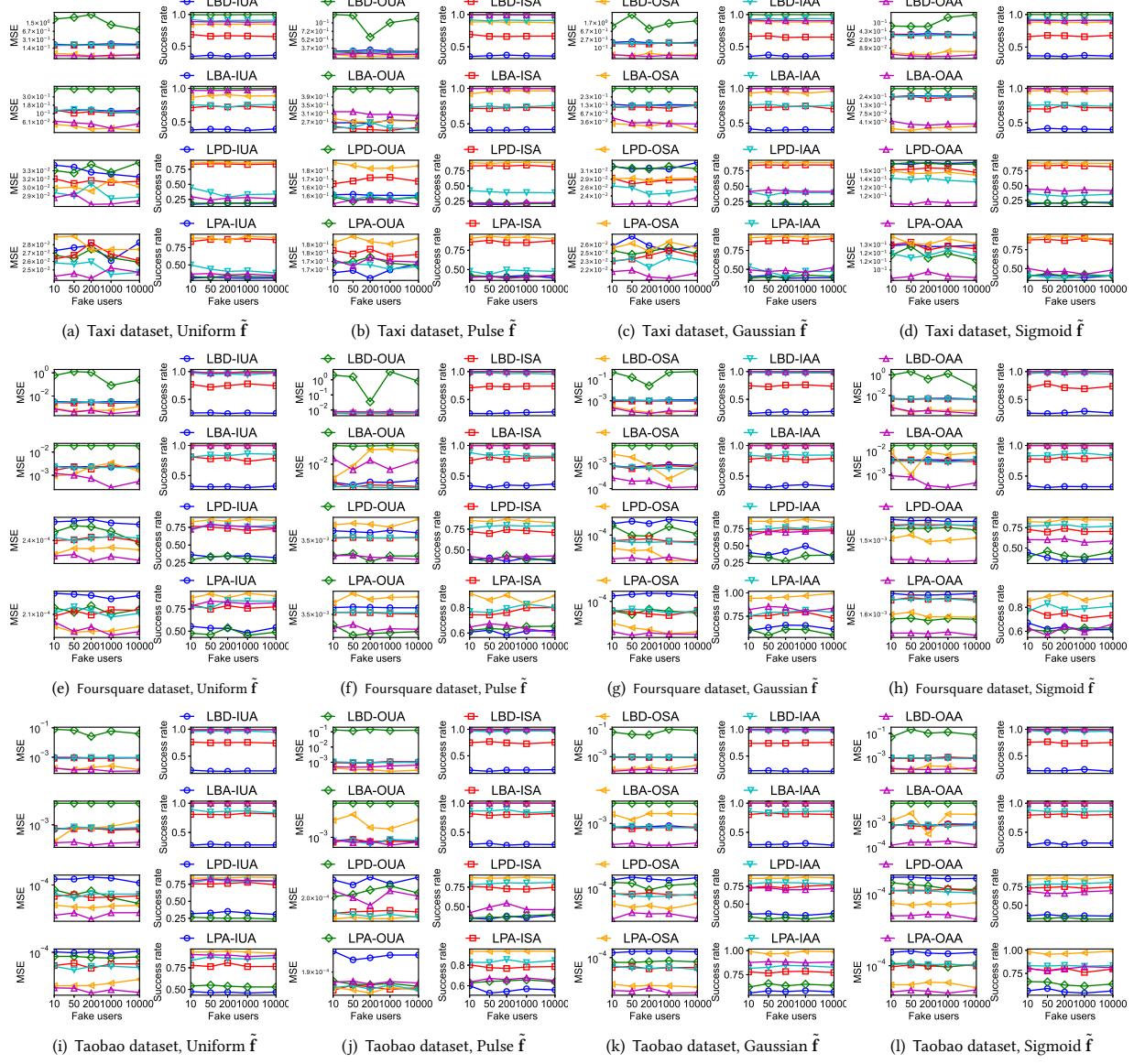
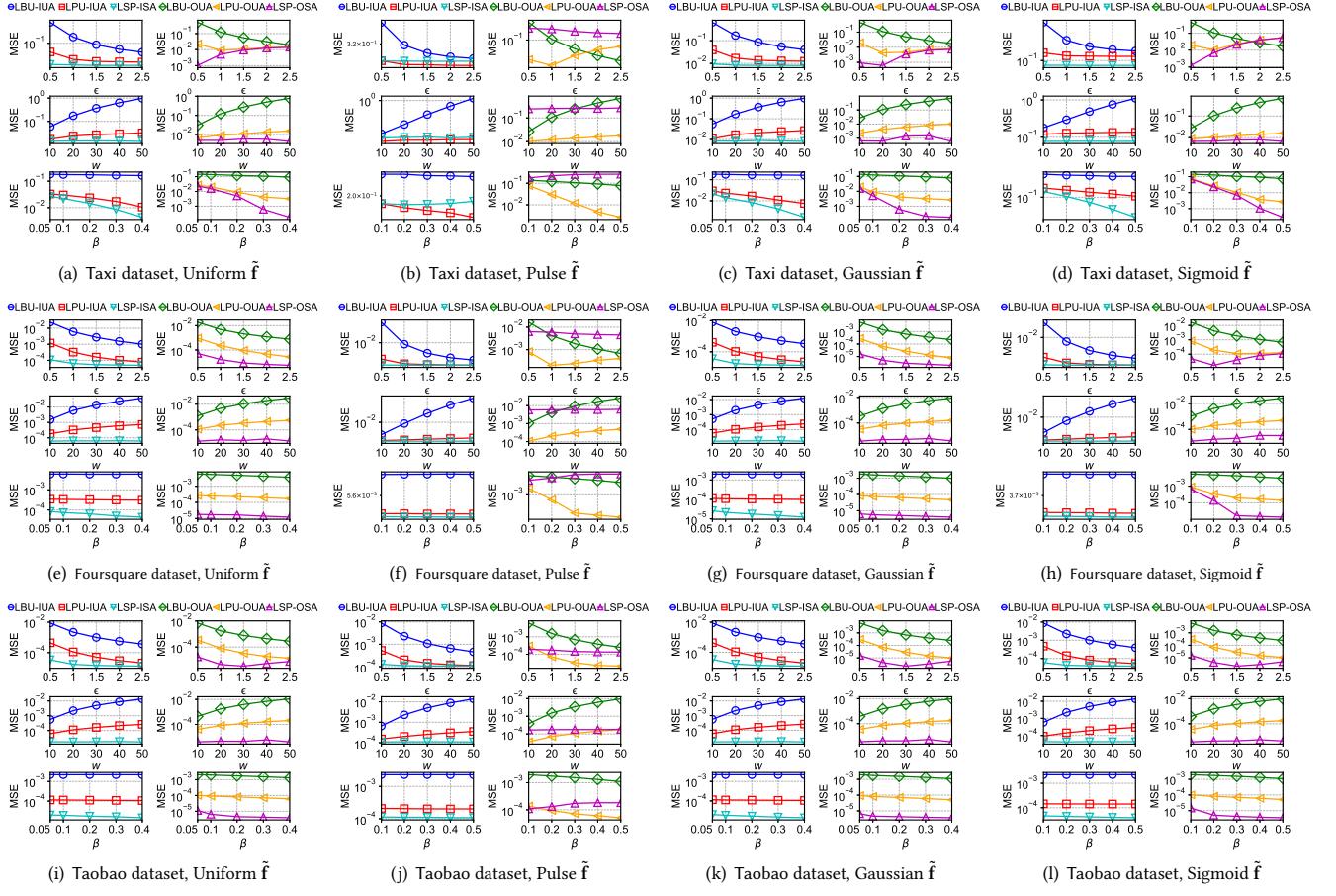
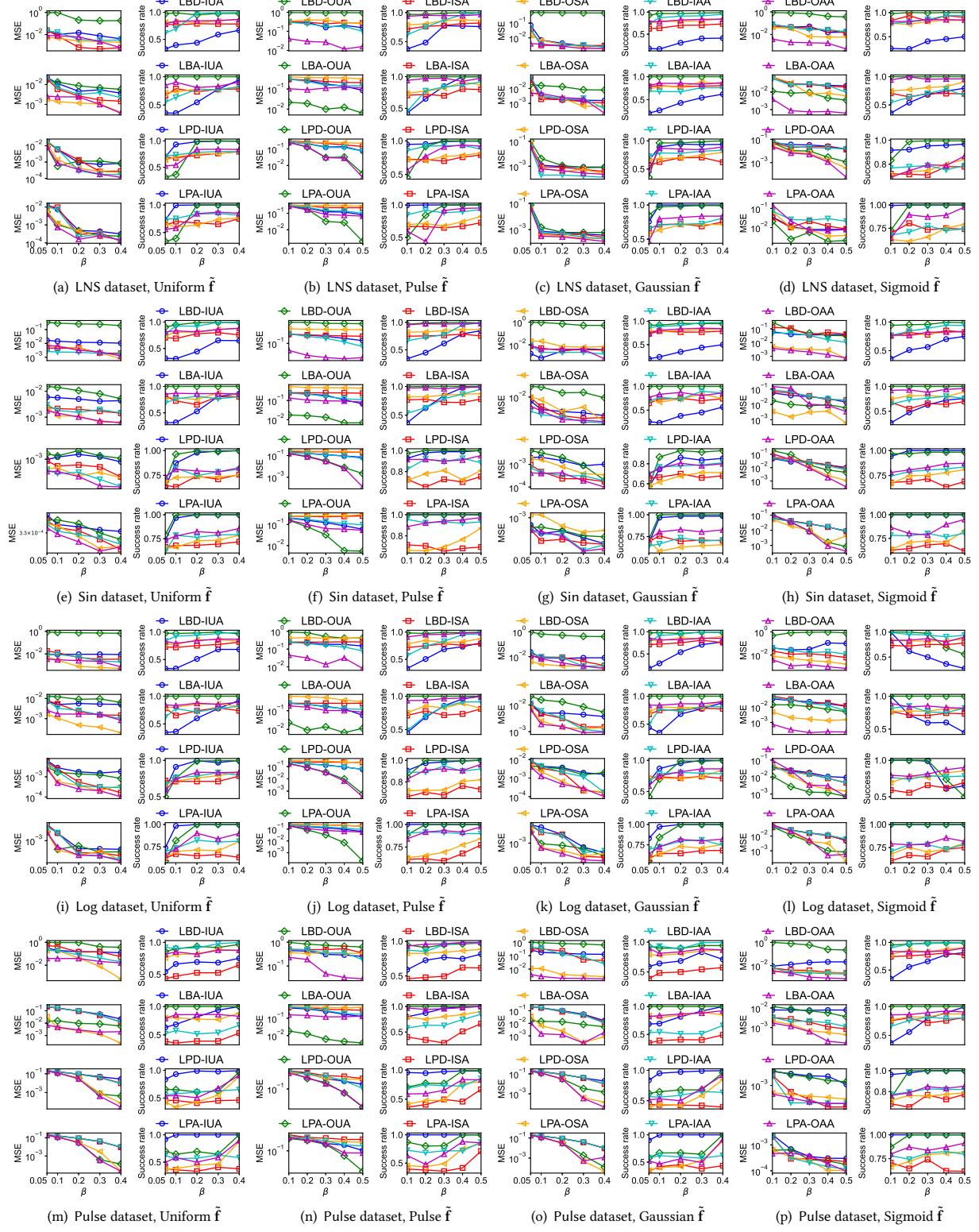
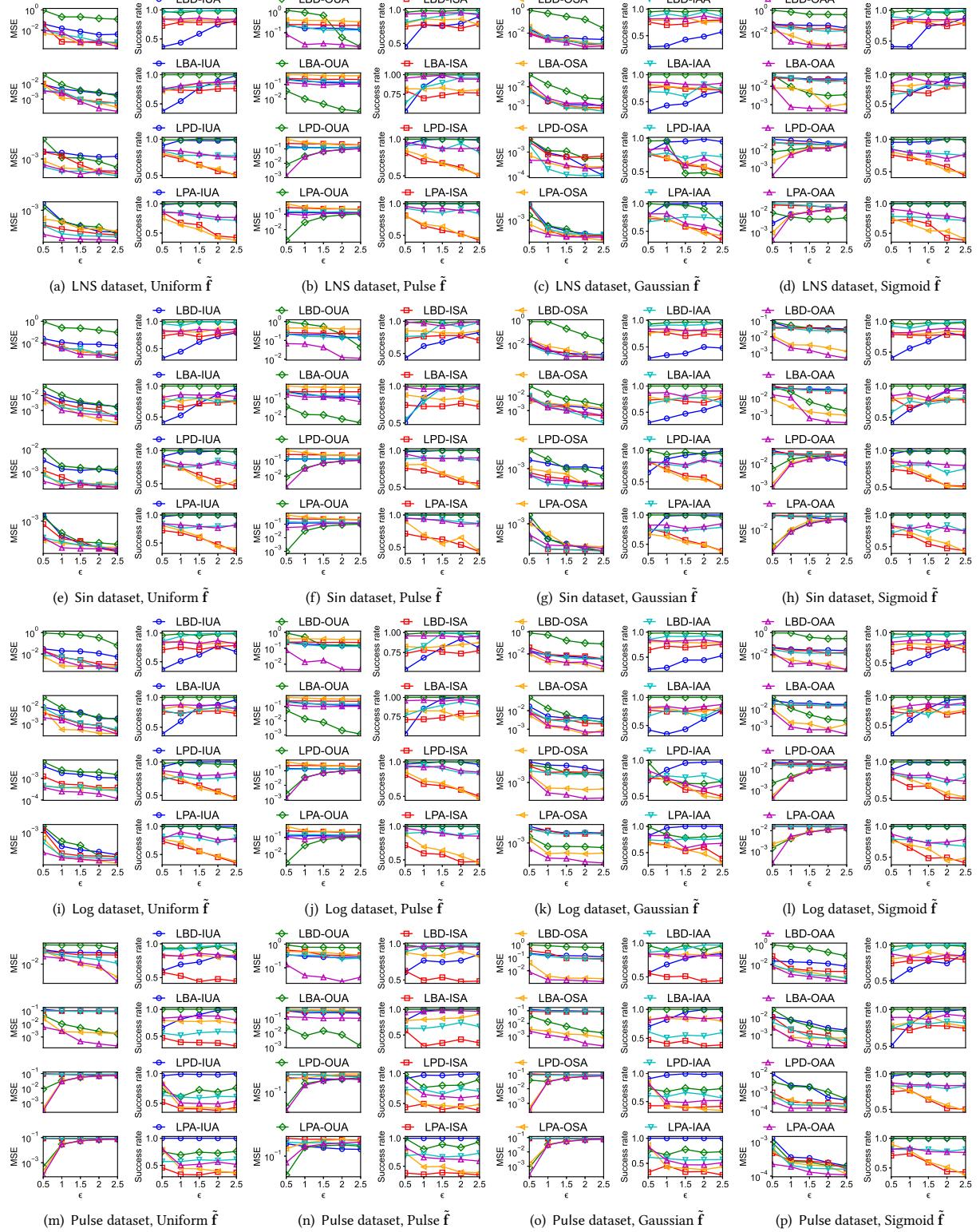
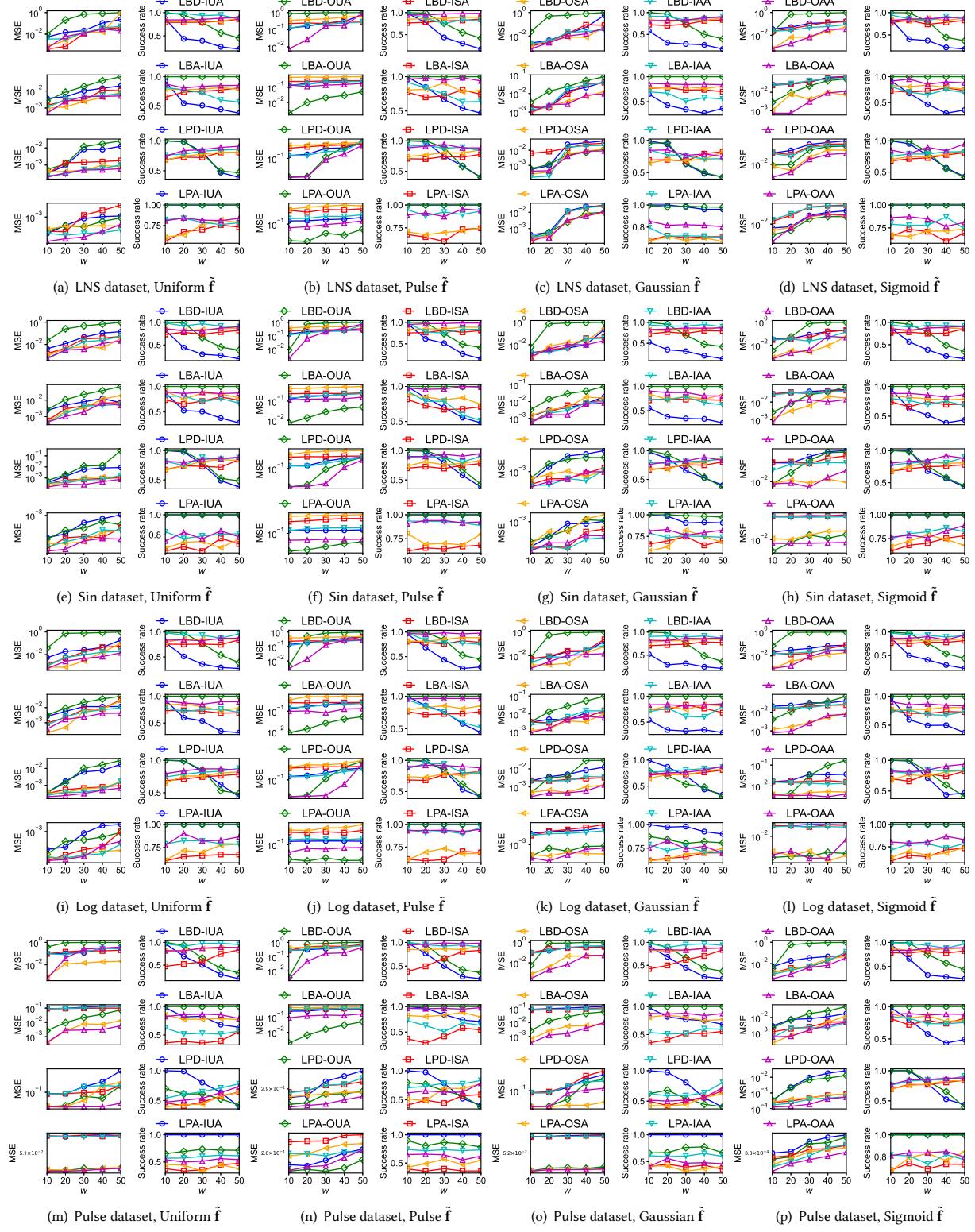


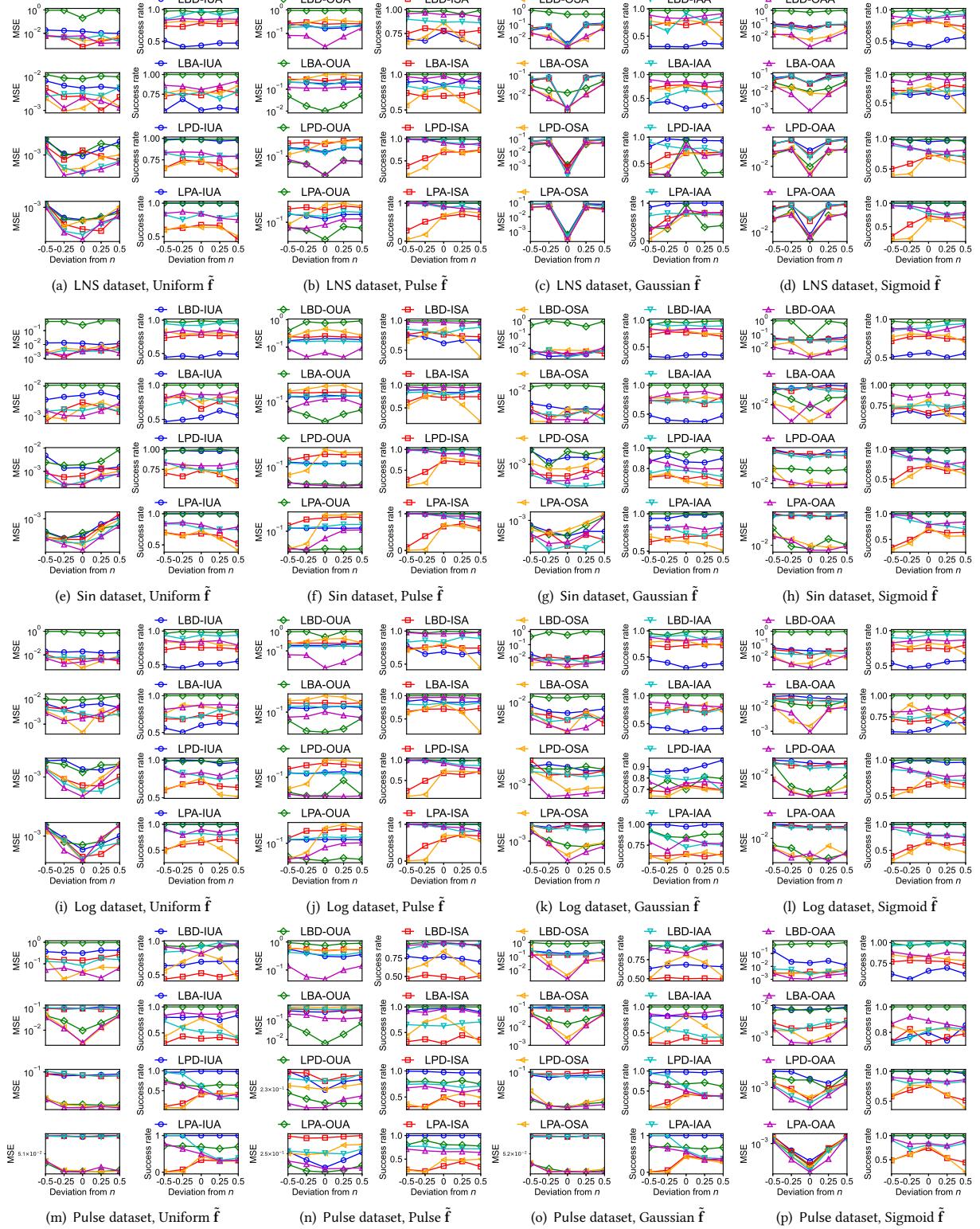
Figure 24: Attacking effectiveness for real-world datasets with varying the number of fake users for f^e calculation.

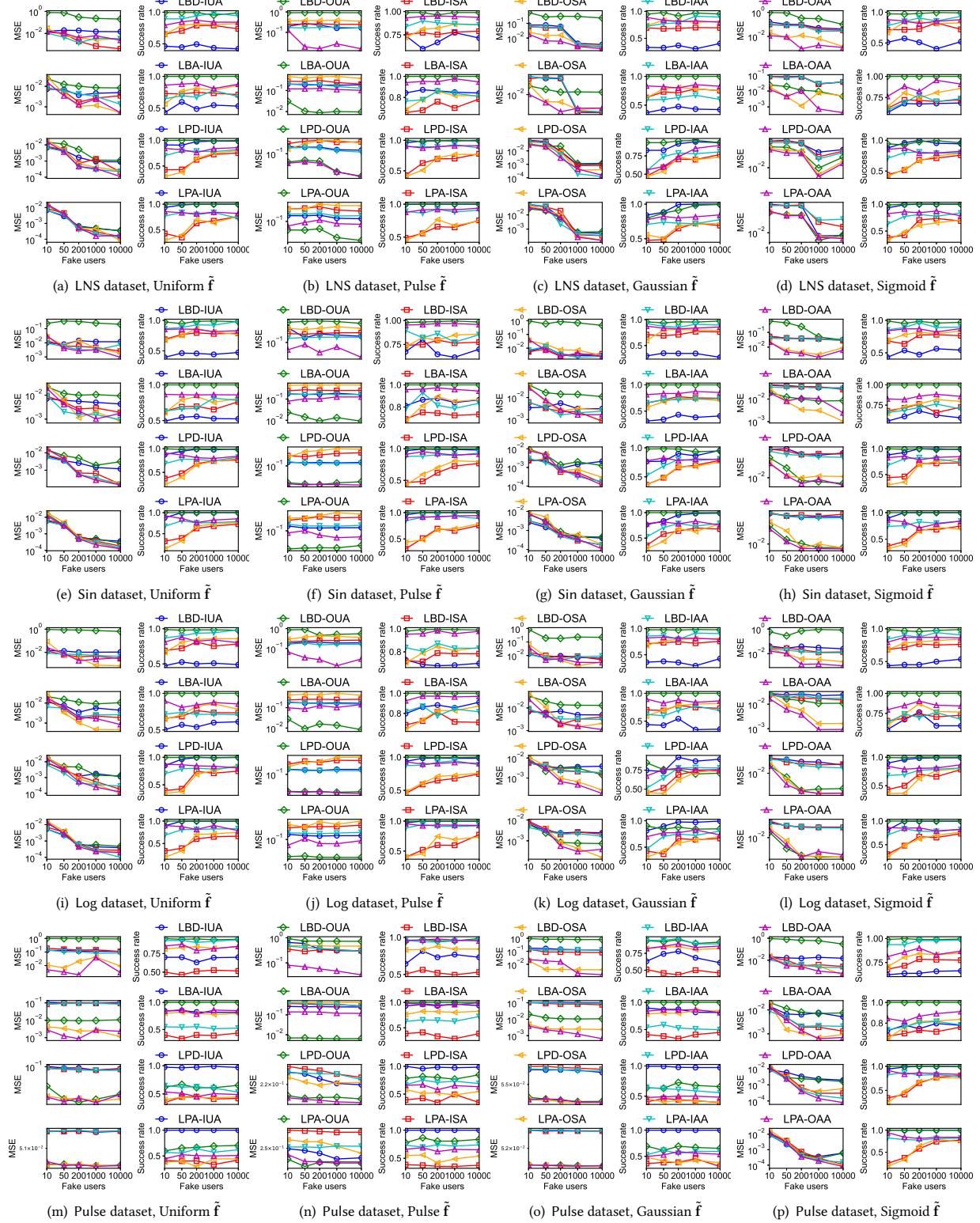
Figure 25: Attack effectiveness for real-world datasets on baseline methods, varying ϵ , w and β .


 Figure 26: Attack effectiveness for Synthetic datasets, varying β .

Figure 27: Attack effectiveness for Synthetic datasets, varying ϵ .


 Figure 28: Attack effectiveness for Synthetic datasets, varying w .

Figure 29: Attack effectiveness for Synthetic datasets, varying n^e .


 Figure 30: Attack effectiveness for Synthetic datasets, varying the number of fake users for f^e calculation.