Data Poisoning Attacks to Local Differential Privacy Protocols for Graphs [Technical Report]

Xi He[1], Kai Huang[1,§], Qingqing Ye[2], Haibo Hu[2]

[1] Faculty of Information Technology, Macau University of Science and Technology, Macau SAR

[2] The Hong Kong Polytechnic University, Hong Kong SAR

hexichn@gmail.com, kylehuangk@gmail.com, qqing.ye|haibo.hu@polyu.edu.hk

Abstract—**Graph analysis has become increasingly popular with the prevalence of big data and machine learning. Traditional graph data analysis methods often assume the existence of a trusted third party to collect and store the graph data, which does not align with real-world situations. To address this, some research has proposed utilizing Local Differential Privacy (LDP) to collect graph data or graph metrics (e.g., clustering coefficient). This line of research focuses on collecting two atomic graph metrics (the adjacency bit vectors and node degrees) from each node locally under LDP to synthesize an entire graph or generate graph metrics. However, they have not considered the security issues of LDP for graphs.**

In this paper, we bridge the gap by demonstrating that an attacker can inject fake users into LDP protocols for graphs and design data poisoning attacks to degrade the quality of graph metrics. In particular, we present three data poisoning attacks to LDP protocols for graphs. As a proof of concept, we focus on data poisoning attacks on two classical graph metrics: degree centrality and clustering coefficient. We further design two countermeasures for these data poisoning attacks. Experimental study on real-world datasets demonstrates that our attacks can largely degrade the quality of collected graph metrics, and the proposed countermeasures cannot effectively offset the effect, which calls for the development of new defenses.

Index Terms—**Graph Analysis, Local Differential Privacy, Data Poisoning Attacks**

I. Introduction

Graph analytics has become a powerful tool in various applications such as social networks and finance analysis, enabling the extraction of valuable insights from graph data. However, the sensitive and personal nature of such data raises significant privacy concerns, as exemplified by Facebook-Cambridge Analytica data scandal where personal data belonging to millions of Facebook users was collected without their consent to be used for political advertising. This scandal was mainly caused by the Facebook Graph API for third-party apps [2], [3] which allows these apps to access users' friend lists (*i.e.,* neighbor lists in a graph) without strict authorizations. Traditional privacy models for graph data, such as k-neighborhood anonymity [4], k-degree anonymity [5], and differential privacy [6], rely on a centralized trusted party to manage and release the data. Unfortunately, the centralized party, even Facebook, cannot be fully trusted to release graph data on behalf of each user. In this context, the problem of *decentralized graphs* has been studied in recent years. In such a scenario, no party has access to the whole graph, which means that (i) no party can collect the entire graph unfetteredly and derive important graph metrics, and (ii) everyone (*i.e.,* a node in the graph) has a local view (*e.g.,* number of neighbors and neighbor list) of the graph. This leads to the inability of existing of privacy-preserving graph publication and analysis approaches since the data cannot be collected in the first place.

Local differential privacy (LDP) [11] has emerged as a promising approach for decentralized graph data privacy. LDP effectively addresses privacy concerns by enabling individual users to perturb their data locally before sharing them, thus eliminating the need for a trusted central party. This approach shifts the control of privacy to the individual user level. LDP has already been successfully applied in various domains, such as frequency estimation and heavy-hitter identification, enabling privacy-preserving data collection and analysis without compromising user privacy. Recent research, including studies LDPGen [13] and LF-GDPR [62], has demonstrated the potential of LDP for graph data. These works have shown how LDP can be adapted to preserve the structural properties of graphs while providing strong privacy guarantees. For example, LDPGen proposed to generate synthetic graphs that maintain key topological features of the original graph while satisfying local differential privacy constraints. LF-GDPR introduced a framework for the estimation of privacy-preserving graph metrics under local differential privacy. In both works, users send two atomic metrics, the *adjacency bit vector* and *node degree*, to the central server (*i.e.,* data collector).

Despite the growing adoption of LDP protocols in graph analysis, there is a notable gap in the research concerning the security of these protocols. In addition, there have been studies on the security of LDP protocols for tasks such as frequency estimation and heavy-hitter identification [63], but *the security of LDP protocols tailored for graph analysis remains unexplored.*

In this paper, we bridge the gap by proposing a family of *data poisoning attacks* to LDP protocols for graph analysis. In our attacks, an attacker can manipulate some users (*also known as* fake users) to send arbitrary degree values and adjacency bit vectors to the central server, with the goal of preventing the central server from collecting accurate and useful graph metrics of arbitrary attacker-chosen users (*i.e.,* attacking tar-

[§] Kai Huang is the corresponding author.

gets/nodes in the graph). As a proof of concept, we focus on two graph analysis tasks: *degree centrality* and *clustering coefficient* estimation. Degree centrality provides insight into the connectivity and prominence of a node in the network. The clustering coefficient of a node represents the probability that two neighbors of the node are also connected to each other. The significance of our work lies in the potential consequences of such attacks. For example, consider a social network that employs an LDP protocol to estimate user popularity based on centrality measures. An attacker could manipulate the centrality measures of specific target users, making them appear more popular and influential than they actually are. This manipulation could have far-reaching effects, potentially misleading other users, influencing platform algorithms, and affecting user interactions.

Designing effective data poisoning attacks against LDP protocols for graphs introduces non-trivial challenges. First, as graph data is inherently more complex than values, LDP protocols for graphs often involve complex aggregation techniques, making it more challenging to devise attack strategies that can successfully degrade the accuracy of the collected graph metrics. Second, it is challenging to maximize the attack effectiveness given a limited number of fake users an attacker can inject. To address these challenges, we first introduce the absolute gain (denoted by $Gain$) of an attack that measures the difference of graph metrics after and before the attack. By maximizing the absolute gain, we derive the optimized attack called *Maximal Gain Attack* (MGA, see Section IV). We also propose two baseline attacks (*i.e.,* Random Value Attack (RVA) and Random Node Attack (RNA) in Section IV) to better demonstrate the effectiveness of MGA. In addition, we explore two countermeasures to defend against our attacks to highlight the need for new defenses against our attacks.

Our main contributions are summarized as follows:

- We are the first to investigate the security of LDP for graph metrics including degree centrality and clustering coefficient.
- We present three data poisoning attacks, including RVA, RNA and MGA, to LDP for graph metrics. We theoretically prove that our MGA attack can maximally distort the collected graph metrics of targeted nodes.
- We explore two countermeasures against our proposed attacks.
- Experimental study on real-world datasets demonstrates that our attacks can largely degrade the quality of collected graph metrics, and the proposed countermeasures cannot effectively offset the effect, which calls for the development of new defenses.

The remainder of this paper is organized as follows. Section II discusses related work and Section III presents the preliminaries. In Section IV, we present three data poisoning attacks, followed by their detailed implementations on degree centrality (see Section V) and clustering coefficient (see Section VI). We present two countermeasures in Section VII and experimental results in Section VIII. We conclude the paper in Section IX. Formal proofs of theorems are in the Appendix.

II. Related Work

The most germane to this work includes local differential privacy, data poisoning attacks, countermeasures against LDP protocols, and graph privacy.

Local Differential Privacy. The concept of Differential Privacy (DP) [7] was initially introduced for a setting where a trusted data curator collects data from individual users, processes it in a differentially private manner, and releases the results. This approach ensures that the influence of any single element in the dataset on the output is limited. However, in reality, third parties cannot always be trusted. Local differential privacy (LDP) [11] has been proposed to address this issue. In a local setting, there is no trusted third party. An aggregator aims to collect data from users who are willing to assist the aggregator but do not fully trust the aggregator to protect their privacy. To preserve privacy in this scenario, each user perturbs their own data before sending it to the aggregator over a secure channel.

Building on this foundation, a significant body of research has focused on applying LDP, particularly in the areas of frequency estimation and practical data analysis tasks [8]–[10], [12], [14]–[37], [62]. This research has led to the development of state-of-the-art protocols for frequency estimation, such as kRR [17], OUE [19], and OLH [19]. Although these protocols have primarily aimed to enhance data utility within the LDP framework, they have also revealed potential vulnerabilities to data poisoning attacks. Our work is orthogonal to this research.

Data Poisoning Attacks and Countermeasures against LDP Protocols. Some researchers have begun to investigate data poisoning attacks against LDP protocols, particularly those used for frequency estimation and heavy-hitter identification. These attacks can be broadly categorized into targeted [63] and untargeted attacks [64]. Targeted attacks aim to increase the estimated frequencies for items chosen by the attacker (*i.e.,* targets). Cao et al. [63] proposed three such strategies: Random perturbed-value attack (RPA), Random item attack (RIA), and Maximal gain attack (MGA). In contrast, untargeted attacks aim to disrupt the overall frequency distribution rather than targeting specific items. These attacks manipulate the global structure of the data by increasing the distance (often measured by an L_p-norm) between the original and manipulated frequency vectors. As a result, these attacks hinder the accurate recovery of the entire data distribution.

There are several countermeasures against data poisoning attacks on LDP protocols [42], [43] for machine learning models instead of graph metrics. Recently, Cao et al. [63] present three countermeasures against data poisoning attacks to LDP protocols for non-graph data, namely, *normalization*, *fake users detection*, and *conditional probability based detection*. The first one normalizes estimated item frequencies so that each estimated item frequency is nonnegative and the overall item frequency is 1. Fake users detection detects potential fake users based on their reported values and removes their impact on the final results. The last one detects fake users based on conditional probability, which is inapplicable when there are more than two target items. The former two countermeasures can take effect in some scenarios but cannot adapt to others and have limited effectiveness in reducing the impacts introduced by data poisoning attacks. To address this problem, Huang et al. [44] proposes a general countermeasure that can detect fake users in a uniform way and offset their side effects on the quality of collected data.

However, these attacks and countermeasures are not designed for LDP protocols for graph data.

Privacy-Preserving Graph Release and Processing. Researchers have developed various approaches to protect sensitive information when releasing graph data. Early approaches centered on anonymization techniques derived from k-anonymity [50], such as k-neighborhood anonymity [4], k-degree anonymity [5], and k-automorphism [51], to counter various structural attacks. However, as these methods proved vulnerable to de-anonymization [52], more robust privacy notions emerged, including L-opacity [53] and differential privacy [7]. The latter employs generative graph models like *dK-series* [54], Stochastic Kronecker Graph (SKG) [55], and Exponential Random Graph Model (ERGM) [56] to fit the original graph and produce synthetic graphs for analysis. Centralized differential privacy, in particular, has been extensively studied for various graph metrics and statistics. Researchers have developed methods to estimate the cost of minimum spanning trees, count triangles [6], and perform subgraph counting queries [57], [58] (such as k-stars, k-triangles, and k-cliques) while maintaining privacy. Other work has focused on estimating node degree distributions [59], [60] and clustering coefficients [61] under differential privacy constraints.

In addition, there has been a host of work on privacy-preserving graph processing methods such as secure shortest path queries [45], privacy-preserving shortest distance queries [46], privacy-preserving subgraph queries [47], [48], and general privacy-preserving graph queries [49]. These works assume that there is a central server that has access to the whole graph. In this paper, we focus on decentralized graphs.

Local Differential Privacy for Graph Metric Estimation. While centralized differential privacy has proven effective, recent research has explored the application of Local Differential Privacy (LDP) to graph data analysis, offering even stronger privacy guarantees. Sun et al. [58] propose a method to estimate subgraph counts in a decentralized graph, while LF-GDPR [62] introduces a novel framework to estimate graph metrics while preserving privacy under LDP constraints. This work builds upon the previous study LDPGen [13], which focused on generating synthetic graphs with LDP guarantees. LF-GDPR addresses the challenge of estimating various graph metrics, such as clustering coefficients and modularity, without compromising individual privacy. The framework provides a parameterized approach that allows for the estimation of different graph metrics by collecting two atomic graph metrics, the adjacency bit vector and node degree, from each node locally. However, the security of LDP protocols tailored for graph analysis remains largely unexplored. We are the first to investigate the security of LDP for graph metrics, including degree centrality and clustering coefficient.

TABLE I: List of key notations.

Notation	Description
$\varepsilon, \varepsilon_1, \varepsilon_2$	Privacy budgets
n, m, r	Number of genuine users, fake users, target users
$N = n + m$	Total number of users
$T = \{t_1, t_2, \ldots, t_r\}$	Target nodes
$Y = \{y_1, y_2, \ldots, y_m\}$	Crafted value from fake users
$D = \{d_1, d_2, \ldots, d_N\}$	Degree vector
$B = \{b_1, b_2, \ldots, b_N\}$	Adjacency bit vector
$M = \{B_1, B_2, \ldots, B_N\}$	Adjacency matrix
$\tilde{b}, \tilde{d}$	Perturbed adjacency bit vector and degree
$Gain$	Overall gain
$\tilde{f}_{i,b}$, $\tilde{f}_{i,a}$	Estimated graph metrics before/after attack for node i
$\Delta \tilde{f}_i = \tilde{f}_{i,a} - \tilde{f}_{i,b}$	Change in graph metrics for node i
c_i	Degree centrality for node i
cc_i	Clustering coefficient for node i
$cc_{B,i}, cc_{A,i}$	Clustering coefficient for node i before/after attack
$\Delta cc_i = cc_{A,i} - cc_{B,i}$	Change in clustering coefficient for node i
$\mathcal{R}(\cdot)$	Calibration function
β	The fraction of fake users
γ	The fraction of target users
$G = (V, E)$	G: Graph, V: Nodes, E: Edges
θ	Edge density
τ_i	Number of triangles incident to node i
$\widetilde{\theta}$	Edge density of the perturbed graph
p	Perturbation probability

III. Preliminaries

Table I lists the key notations and acronyms used in this paper. Given a privacy budget ε ($\varepsilon \geq 0$), the formal definition of ε-LDP is as follows.

Definition 1. (**Local Differential Privacy**). *An algorithm* $\mathbf{A}$ *satisfies* ε*-local differential privacy (*ε*-LDP), where* $\varepsilon \geq 0$*, if and only if for any input* v_1 *and* v_2*, we have*

$$\forall O \subseteq \operatorname{Range}(\mathbf{A}) : \frac{\Pr\left[\mathbf{A}\left(v_1\right) \in O\right]}{\Pr\left[\mathbf{A}\left(v_2\right) \in O\right]} \leq e^{\varepsilon} \tag{1}$$

where Range$(\mathbf{A})$ *denotes the set of all possible outputs of the algorithm* $\mathbf{A}$.

As a strong privacy measure, LDP has been widely applied in collecting real-value, bit vector, key-value data, and graph data. In this paper, we focus on graph data. Formally, a graph G is defined as $G = (V, E)$, with $V = \{1, 2, ..., N\}$ representing the set of nodes and $E \subseteq V \times V$ denoting the set of edges. The LDP on the graphs encompasses two primary variants: *Node Local Differential Privacy* (Node LDP) and *Edge Local Differential Privacy* (Edge LDP). Node LDP ensures that the output of a randomized algorithm does not reveal the existence of any individual node in the graph, while edge LDP guarantees that the algorithm's output does not disclose the presence of any specific edge.

Definition 2. (**Node Local Differential Privacy**). *A randomized algorithm* $\mathbf{A}$ *satisfies* ε*-node local differential privacy (*ε*-node LDP), where* $\varepsilon \geq 0$*, if and only if for any two adjacency bit vectors (*i.e., *neighbor lists)*[1] B *and* B'*, we have*

$$\forall o \in \text{Range}(\mathbf{A}) : \frac{\Pr\left[\mathbf{A}\left(B\right) = o\right]}{\Pr\left[\mathbf{A}\left(B'\right) = o\right]} \leq e^{\varepsilon} \tag{2}$$

This definition ensures that the algorithm's output remains approximately the same regardless of the presence or absence of any single node in the graph.

Definition 3. (**Edge Local Differential Privacy**). *A randomized algorithm* $\mathbf{A}$ *satisfies* ε*-edge local differential privacy (*ε*-edge LDP), where* $\varepsilon \geq 0$*, if and only if for any two adjacency bit vectors* B *and* B' *that differ only in one bit, we have*

$$\forall o \in \text{Range}(\mathbf{A}) : \frac{\Pr\left[\mathbf{A}\left(B\right) = o\right]}{\Pr\left[\mathbf{A}\left(B'\right) = o\right]} \leq e^{\varepsilon} \tag{3}$$

Observe that edge LDP is a relaxation of node LDP. It limits the definition of neighbors to any two adjacency bit vectors that differ only in one bit (*i.e.,* one edge). As edge LDP can still achieve strong indistinguishability of each edge's existence, in this paper, LDP for graphs refers to edge LDP.

Remark. Node-LDP (or Edge-LDP) differs from that of Node-DP (or Edge-DP) as defined in the work [59]. In graph differential privacy (*i.e.,* node-DP and edge-DP), the subject of discussion is a single graph G; "neighboring datasets" refer to two graphs G and G' that differ in one node (*i.e.,* node-DP) or one edge (*i.e.,* edge-DP). In contrast, in local differential privacy on graphs, each subject pertains to the private data of a single user, *i.e.,* a node in a graph.

A. Data Poisoning Attacks on Frequency Estimation

Data poisoning attacks on machine learning models [38]–[41] are a type of adversarial attack, where the attacker manipulates the training data to cause the model to learn an incorrect or biased function as the attacker desires. This can degrade the model's performance on test data. On the other hand, data poisoning attacks on frequency estimation (*i.e.,* estimating the frequency of an item) aim to increase the estimated frequencies of attacker-chosen items (*i.e.,* targets) by injecting fake users into the system. There are three main strategies for data poisoning attacks on frequency estimation under LDP: RPA, RIA, and MGA. RPA selects a value from the encoded space of the LDP protocol uniformly at random for each fake user and sends it to the central server. In contrast to RPA, RIA considers information about the target items. Specifically, RIA randomly chooses a target item from each fake user's set of target items. The LDP protocol then encodes and adds noise to the item. Finally, the perturbed value is sent to the server. The idea behind MGA is to craft the perturbed values for the fake users by solving an optimization problem, with the objective of maximizing the gain of the target item before and after the attack.

B. Local Differential Privacy for Graph Metric Estimation

LF-GDPR [62] implements a four-step process to achieve privacy-preserving graph metric estimation:

- Graph Metric Reduction: This is the initial step in the framework, designed to express the target graph metric F as a function of the adjacency matrix M and degree vector D. This process is formalized as a polynomial mapping function $F = \text{Map}(M, D)$. The key equation for this reduction is:

$$F = \sum_{l} F_l = \sum_{l} f^{\phi_l}(M^{k_l}) \cdot g^{\psi_l}(D) \tag{4}$$

  where M^{k_l} represents k_l-th power of the adjacency matrix, $f^{\phi_l}(\cdot)$ and $g^{\psi_l}(\cdot)$ are aggregation functions applied after projections on the matrix and vector respectively.
- Privacy Budget Allocation: The framework optimally divides the total privacy budget ε between the adjacency bit vector (ε_1) and node degree perturbation (ε_2): $\varepsilon_1 = \alpha\varepsilon, \varepsilon_2 = (1-\alpha)\varepsilon$ where α is determined by minimizing the expected squared error of the estimated metric.
- Local Perturbation: Each node locally perturbs its adjacency bit vector B and degree d using the allocated privacy budgets:

$$\widetilde{b}_i = \begin{cases} b_i & \text{w.p. } \frac{e^{\varepsilon_1}}{1+e^{\varepsilon_1}} \\ 1 - b_i & \text{w.p. } \frac{1}{1+e^{\varepsilon_1}} \end{cases} \tag{5}$$

  and

$$\tilde{d} = d + Lap(\frac{2}{\varepsilon_2}) \tag{6}$$

  where $Lap(\frac{2}{\varepsilon_2})$ denotes Laplace noise with scale $\frac{2}{\varepsilon_2}$.
- Aggregation and Calibration: The data collector receives the perturbed adjacency matrix $\widetilde{M}$ and degree vector $\widetilde{D}$, and uses these to estimate the target graph metric $\widetilde{F}$. This estimation is performed using the following equation:

$$\widetilde{F} = \sum_{l} \mathcal{R}(f_{\phi_l}(\widetilde{M}^{k_l})) \cdot g_{\psi_l}(\widetilde{D}) \tag{7}$$

  Here, $\mathcal{R}(\cdot)$ is a calibration function designed to suppress the aggregation bias introduced by the perturbation of $\widetilde{M}$. This calibration is necessary for $f_{\phi_l}(\widetilde{M}^{k_l})$ but not for $g_{\psi_l}(\widetilde{D})$, as $\widetilde{D}$ is already an unbiased estimation of D due to the Laplace mechanism used in its perturbation. The calibration function $\mathcal{R}$ is derived as a mapping between $f_{\phi_l}(M^{k_l})$ and $f_{\phi_l}(\widetilde{M}^{k_l})$, essentially estimating the true value of $f_{\phi_l}(M^{k_l})$ after observing the perturbed value $f_{\phi_l}(\widetilde{M}^{k_l})$. This relationship is formally expressed as: $\mathcal{R} : f_{\phi_l}(\widetilde{M}^{k_l}) \rightarrow f_{\phi_l}(M^{k_l})$.

LF-GDPR and LDPGen [13] are the only two existing works on LDP for graph metric estimation. Compared to LF-GDPR, LDPGen [13] is more complicated and ad hoc. In other words, for different tasks (*e.g.,* clustering coefficient estimation), dedicated LDP solutions must be designed from scratch. Therefore, this paper adopts the same approach as LF-GDPR to collect data, including the degree and adjacency bit vector, from each user.

[1] An adjacency bit vector is a vector representation of a node's neighbor list. Formally, a user u's neighbor list can be represented as a N-dimensional bit vector $B = (b_1, ..., b_N)$, *i.e.,* $b_i = 1, i = 1, ..., N$, if and only if there is an edge (u, u_i) in the graph; otherwise $b_i = 0$.

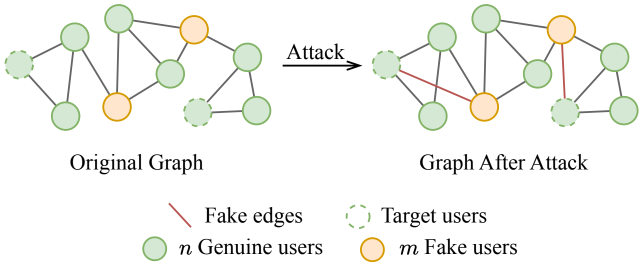


Fig. 1: Data poisoning attack.

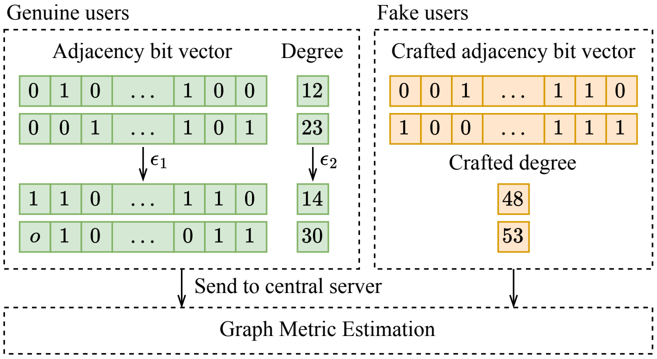


Fig. 2: Genuine users report perturbed adjacency bit vectors and degrees, while fake users report crafted data.

IV. Data Poisoning Attacks to LDP for Graphs

A. Threat Model

It is crucial to understand the potential threats of poisoning attacks to LDP, and the intentions of possible attackers. In this context, we characterize our threat model by considering the attacker's capabilities, background knowledge, and objectives.

Attacker's capability and background knowledge. We assume an attacker can control a portion of users in the LDP protocol, and these fake users can send arbitrary data to the central server. As illustrated in Fig. 1, the attacker can manipulate these controlled users to create fake connections within the network. [2] Specifically, we posit that there are n genuine users in the system, while the attacker has control of m fake users. This results in a total of $N = n+m$ users participating in the protocol. The attacker can easily obtain these fake users by compromising existing devices through malware or other means. This is a realistic assumption in many real-world scenarios where user authentication is not stringent or where devices can be infected at scale. Furthermore, attackers can also acquire fake users by purchasing them from underground markets, where large numbers of compromised accounts or devices are often available for sale.

Since the LDP protocol executes the perturbation steps locally on the user's side, the attacker has access to the implementation of these steps. Consequently, the attacker knows various parameters of the LDP protocol. In particular, the attacker is aware of the privacy budget ε_1 used for the adjacency bit vector B and the privacy budget ε_2 used for the degree. Additionally, the attacker knows the domain size for the degree and statistical characteristics of the degree, such as the average degree in the perturbed graph.

Attacker's goal. We consider the attacker's goal to be degrading the performance of estimating graph metrics for specific target nodes. In other words, the attacker aims to increase the error in the estimated graph metric values for certain target nodes. For instance, some companies may interfere with competitors to cause them to collect inaccurate data. Formally, we denote the set of r target nodes as $T = \{t_1, t_2, \cdots, t_r\}$. To increase the error in the estimated graph metric values of these target nodes, the attacker carefully crafts the perturbed values sent by fake users to the central server, as illustrated in Fig. 2. We can represent the set of crafted values from fake nodes as $Y = \{y_1, y_2, ..., y_m\}$, where y_i denotes the value crafted by the i-th fake node. In the context of LDP protocols for graphs, each y_i typically consists of two components: a bit vector representing the node's adjacency information and a value representing the node's degree.

Suppose $\tilde{f}_{t,b}$ and $\tilde{f}_{t,a}$ are the graph metrics estimated by the LDP protocol for a target node t before and after the attack, respectively. We define the gain $\Delta\tilde{f}_t$ for a target node t as

$$\Delta\tilde{f}_t = |\tilde{f}_{t,a} - \tilde{f}_{t,b}|, \forall t \in T \tag{8}$$

We then define the overall gain $Gain$ as the sum of gains across all target nodes:

$$Gain = \sum_{t \in T} \Delta\tilde{f}_t \tag{9}$$

This overall gain $Gain$ provides a measure of the total impact of the attack on the estimation of graph metrics for all target nodes. A larger value of $Gain$ indicates a more successful attack. Therefore, the attacker's objective is to maximize $Gain$, thereby maximizing the impact of the attack on the target nodes' metric estimations. Formally, the attacker aims to solve the following optimization problem:

$$\max_{Y} \quad Gain(Y) \tag{10}$$

subject to

$$Y \in \mathcal{Y} \tag{11}$$

where $Gain(Y)$ is the overall gain achieved by the set of crafted values Y, and $\mathcal{Y}$ is the set of all possible combinations of crafted values constrained by the LDP protocol and the number of fake users m.

B. Data Poisoning Attacks

We propose three attack strategies: Random value attack (RVA), Random node attack (RNA), and Maximal gain attack (MGA). These attacks aim to manipulate the estimation of graph metrics for specific target nodes by introducing carefully crafted data from fake nodes. The RVA strategy randomly assigns connections and degree values to fake nodes, RVA does not consider any information about the target nodes.

[2] The paper does not consider the removal of edges in the attacks because removing edges may render the graph disconnected.

The RNA method focuses on connecting fake nodes to one random target node and then applies normal LDP perturbation to all connections, potentially increasing the impact on specific targets while maintaining plausibility. The MGA crafts the perturbed value for each fake user to maximize the overall gain $Gain$ by solving the optimization problem presented in Equation (10). RVA and RNA serve as baseline attacks, designed to better demonstrate the effectiveness of MGA.

- **Random Value Attack (RVA)**. This strategy randomly assigns connections and degree values to fake nodes from the entire possible value space, without considering any information about the target nodes.
- **Random Node Attack (RNA)**. This strategy focuses on connecting fake nodes to target nodes. Specifically, the attacker connects each fake node to one random target node and then applies LDP perturbation to all connections.
- **Maximal Gain Attack (MGA)**. This is an optimization-based attack strategy. The attacker crafts the value for each fake user to maximize the overall gain $Gain$ by solving the optimization problem.

Note that RVA (resp. RNA) is a variation of RPA [63] (resp. RIA [63]) adapted for graphs. In particular, RPA randomly selects a value from the encoded space of the LDP protocol, which, in the context of graphs, translates to randomly assigning connections and degree values to fake nodes from the entire value space. Similarly, RIA randomly selects a target item for each fake user and perturbs that item. When applied to graphs, this corresponds to connecting fake nodes to target nodes and applying LDP perturbation to these connections.

V. Data Poisoning Attacks against Degree Centrality

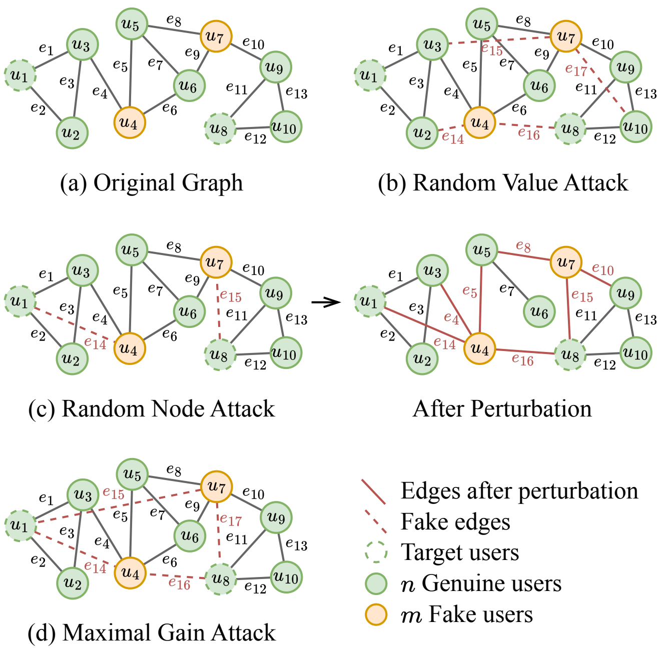
Degree centrality is a measure of a node's importance in a network based on the number of direct connections it has to other nodes. Formally, the normalized degree centrality of a node i is defined as:

$$c_i = \frac{d_i}{N-1} \tag{12}$$

where d_i is the degree of node i and N is the number of nodes. This simple yet powerful metric provides a straightforward way to quantify a node's local influence in the graph. Nodes with high degree centrality are often considered important, as they have many direct connections and potentially greater access to information or resources within the graph.

As shown in Fig. 3, we introduce three attack methods against degree centrality, namely, Random Value Attack (RVA), Random Node Attack (RNA), and Maximal Gain Attack (MGA).

Random Value Attack to Degree Centrality. To degrade the quality of degree centrality, RVA randomly manipulates the adjacency bit vector of fake nodes in the graph. In particular, it keeps the edges in the original graph unchanged and crafts some edges to be associated with the fake users. In the extreme case, all other nodes are connected with fake users, such that the degree of the fake user is $N-1$. However, the server can easily detect the fake user. To avoid detection due to an excessive number of new connections, the attack limits the number of new connections that can be added. This number is determined by calculating the average degree of all nodes after perturbation based on the adjacency bit vector.



(a) Original Graph

(b) Random Value Attack

(c) Random Node Attack

After Perturbation

(d) Maximal Gain Attack

Fig. 3: (a) Original graph; (b)−(d) Example of attacking degree centrality using (b) RVA; (c) RNA; and (d) MGA.

Fig. 3(b) illustrates an example of the Random Value Attack. Two fake nodes u_4 and u_7 are randomly connected to other nodes. In particular, node u_4 establishes new connections with nodes u_2 and u_8, while node u_7 creates new connections with nodes u_3 and u_{10}.

Random Node Attack to Degree Centrality. In this attack, each fake node randomly connects to a target node. Then, the newly created edge, along with the existing connections, is perturbed by an LDP protocol, resulting in a perturbed adjacency bit vector for the fake node.

As illustrated in Fig. 3(c), there are two fake nodes, u_4 and u_7. First, each fake node establishes a connection with a single target node. Specifically, u_4 forms a new connection with the target node u_1, while u_7 connects to the target node u_8. Then, the crafted connections together with existing edges are perturbed by the LDP protocol. As a result of this perturbation, node u_4 gets an additional connection e_{16}. In addition, two connections, e_6 and e_9, are removed from the graph.

Maximal Gain Attack to Degree Centrality. To maximize the gain, each fake node randomly connects to as many target nodes as possible. Then, the newly created edge, along with the existing connections, can be encoded into the adjacency bit vector which is directly sent to the data collector. However, this may expose the identity of a fake node, since all fake nodes may connect with the same set of nodes. Therefore, the number of additional connections for each fake node is carefully controlled. The upper limit for new connections is determined by the average degree of the graph after perturbation, which is calculated based on the ε value.

Formally, the objective of MGA can be expressed as maximizing the overall gain in degree centrality for all target nodes:

$$\max \sum_{t \in T} \Delta c_t \tag{13}$$

where Δc_t is the change in degree centrality for target node t after the attack. Given that degree centrality is simply the degree of a node, we can represent Δc_t as:

$$\Delta c_t = \frac{1}{N-1} \sum_{u \in U} x_{ut} \tag{14}$$

In this equation, x_{ut} is a binary variable that takes the value 1 if fake node u is connected to target node t, and 0 otherwise.

The most effective way to increase overall gain is to maximize the number of connections between fake nodes and target nodes, subject to the constraints imposed by the average degree limit. Each new connection from a fake node to a target node directly increases the degree of the target node by one, thus contributing to the maximization of the sum of Δc_t across all target nodes.

Fig. 3(d) is an example of Maximal Gain Attack. To maximize connections between the target nodes and fake nodes, the fake node u_4 simultaneously creates connections with two target nodes, u_1 and u_8. Similarly, the fake node u_7 also forms connections with these two target nodes.

Theorem 1. *The overall gain of MGA to degree centrality is*

$$Gain = \frac{m \cdot r}{N-1} \cdot \left(\frac{\min(r, \lfloor \bar{\tilde{d}} \rfloor)}{r} - \frac{\bar{\tilde{d}}}{N-1} \right) \tag{15}$$

where $\bar{\tilde{d}}$ is the average degree of the graph after perturbation.

VI. Data Poisoning Attacks to Clustering Coefficient

The clustering coefficient of a node in a graph measures the degree to which the node's neighbors are connected to each other. Formally, the local clustering coefficient for a node i is defined as:

$$cc_i = \frac{2\tau_i}{d_i(d_i - 1)} \tag{16}$$

where τ_i is the number of triangles connected to node i, and d_i is the degree of node i. A high clustering coefficient indicates that the node's neighbors are densely interconnected.

Random Value Attack to Clustering Coefficient. In this attack, the adjacency bit vector and degree of fake nodes are manipulated randomly. These fake nodes are connected to arbitrary other nodes in the graph. The newly created connections, combined with the existing connections, form the adjacency bit vector for the fake node. To avoid detection, the number of connections for each fake node is determined based on the average degree of real nodes after perturbation for each ε value. These connections are not perturbed after being established. The degree value for each fake node is randomly selected from the entire degree space.

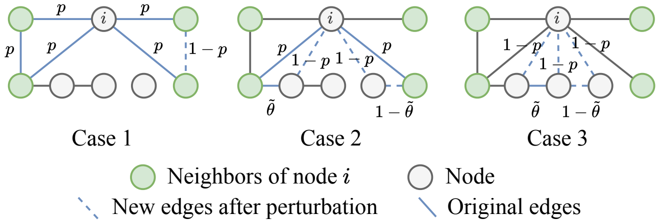


Fig. 4: Number of triangles incident to node i.

Random Node Attack to Clustering Coefficient. In this attack, each fake node randomly connects to a target node. Then, the newly created edge, along with the existing connections, is perturbed by an LDP protocol, resulting in a perturbed adjacency bit vector for the fake node. The degree of each fake node is calculated based on its connections and then perturbed according to the LDP protocol.

Maximal Gain Attack against Clustering Coefficient. For each fake node, MGA crafts its perturbed value by solving an optimization problem. The change in clustering coefficient for node i before and after the attack can be expressed as:

$$\Delta cc_i = cc_{A,i} - cc_{B,i} \tag{17}$$

where $cc_{A,i}$ and $cc_{B,i}$ represent the clustering coefficients after and before the attack, respectively. The optimization objective is to maximize the change in clustering coefficient for each target node:

$$\max \sum_{t \in T} \Delta cc_t \tag{18}$$

For the perturbed graph, we can estimate the clustering coefficient of the i-th node using the following formula:

$$cc_i = \frac{2\tau_i}{\tilde{d}_i \left(\tilde{d}_i - 1 \right)} \tag{19}$$

where τ_i is the corrected number of triangles and d_i is the perturbed degree. To estimate τ_i, we consider three cases based on the relationships of the other two nodes in the triangle, following the approach proposed in the LF-GDPR [62], as shown in Fig. 4 where p is the perturbation probability.

- Both Nodes are Neighbors (Case 1, Fig. 4). If there is an edge between the two neighboring nodes, the probability of keeping the triangle in the perturbed graph is p^3. Otherwise, the probability of forming the triangle in the perturbed graph is $p^2(1-p)$. Therefore, the total number of such triangles in the perturbed graph is $\tilde{\tau}_{i,1} = \tau_i \cdot p^3 + \left(\frac{1}{2}d(d-1) - \tau_i\right) \cdot p^2(1-p)$.
- Only One Node is a Neighbor (Case 2, Fig. 4). The number of possible triangles is $d(N-d-1)$ and the probability of retaining the triangle in the perturbed graph is $p(1-p)\tilde{\theta}$. Thus, the total number of such triangles in the perturbed graph is $\tilde{\tau}_{i,2} = d(N-d-1) \cdot p(1-p)\tilde{\theta}$.
- Neither Node is a Neighbor (Case 3, Fig. 4). The number of possible triangles is $\frac{1}{2}(N-d-1)(N-d-2)$ and the

probability of forming the triangle in the perturbed graph is $(1-p)^2\tilde{\theta}$. Therefore, the total number of such triangles in the perturbed graph equals $\tilde{\tau}_{i,3} = \frac{1}{2}(N-d-1)(N-d-2)\cdot(1-p)^2\tilde{\theta}$.

By summing up the results from the three cases, we derive the corrected number of triangles incident to node i:

$$\begin{aligned}\tau_i = \mathcal{R}(\widetilde{\tau}_i) = &\frac{1}{p^2(2p-1)}(\widetilde{\tau}_i - \frac{1}{2}\widetilde{d}_i(\widetilde{d}_i-1)p^2(1-p)\\ &- \widetilde{d}_i(N-\widetilde{d}_i-1)p(1-p)\widetilde{\theta}\\ &- \frac{1}{2}(N-\widetilde{d}_i-1)(N-\widetilde{d}_i-2)(1-p)^2\widetilde{\theta})\end{aligned} \tag{20}$$

Here, $\widetilde{\theta}$ represents the edge density of the perturbed graph, which can be calculated as:

$$\widetilde{\theta} = \frac{\sum_{i=1}^{N}\widetilde{\tau}_i}{N(N-1)} \tag{21}$$

The number of triangles for a certain node before the attack can be expressed as:

$$\begin{aligned}\tau_{B,i} = \mathcal{R}(\widetilde{\tau}_{B,i}) = &\frac{1}{p^2(2p-1)}(\widetilde{\tau}_{B,i} - \frac{1}{2}\widetilde{d}_i(\widetilde{d}_i-1)p^2(1-p)\\ &- \widetilde{d}_i(N-\widetilde{d}_i-1)p(1-p)\widetilde{\theta}\\ &- \frac{1}{2}(N-\widetilde{d}_i-1)(N-\widetilde{d}_i-2)(1-p)^2\widetilde{\theta})\end{aligned} \tag{22}$$

Since only the number of triangles changes for a node before and after the attack, while the degree remains constant, the number of triangles after the attack is

$$\begin{aligned}\tau_{A,i} = \mathcal{R}(\widetilde{\tau}_{A,i}) = &\frac{1}{p^2(2p-1)}(\widetilde{\tau}_{A,i} - \frac{1}{2}\widetilde{d}_i(\widetilde{d}_i-1)p^2(1-p)\\ &- \widetilde{d}_i(N-\widetilde{d}_i-1)p(1-p)\widetilde{\theta}\\ &- \frac{1}{2}(N-\widetilde{d}_i-1)(N-\widetilde{d}_i-2)(1-p)^2\widetilde{\theta})\end{aligned} \tag{23}$$

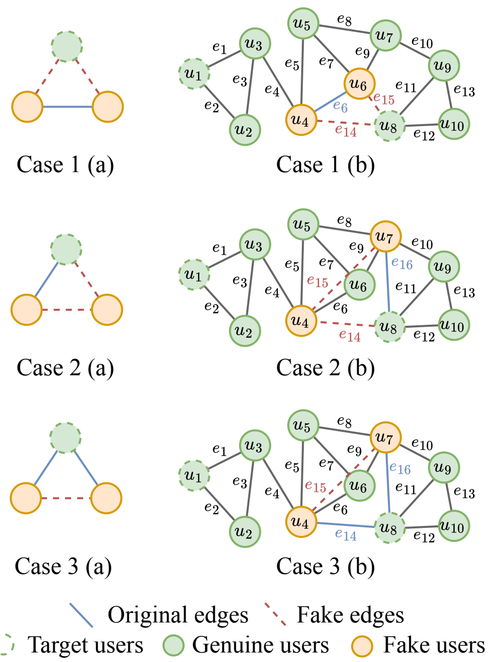
Substitute $\tau_{A,i}$ and $\tau_{B,i}$ into Δcc_i

$$\Delta cc_i = \frac{2\tau_{A,i}}{\widetilde{d}_i(\widetilde{d}_i-1)} - \frac{2\tau_{B,i}}{\widetilde{d}_i(\widetilde{d}_i-1)} \tag{24}$$

$$= \frac{2(\tau_{A,i}-\tau_{B,i})}{\widetilde{d}_i(\widetilde{d}_i-1)} \tag{25}$$

$$= \frac{2}{p^2(2p-1)}\cdot\frac{1}{\widetilde{d}_i(\widetilde{d}_i-1)}\cdot(\tilde{\tau}_{A,i}-\tilde{\tau}_{B,i}) \tag{26}$$

The first two terms in the formula, $\frac{2}{p^2(2p-1)}$ and $\frac{1}{\widetilde{d}_i(\widetilde{d}_i-1)}$, remain constant before and after the attack. Therefore, the objective is to maximize $\tilde{\tau}_{A,i}$. To achieve this, we discuss several possible connection cases between the target nodes and fake nodes, as shown in Fig. 5.



Case 1 (a)
Case 1 (b)
Case 2 (a)
Case 2 (b)
Case 3 (a)
Case 3 (b)
Original edges Fake edges
Target users Genuine users Fake users

Fig. 5: Connections between fake nodes and target nodes.

- **Neither fake node is connected to the target node (see Case 1a, Fig. 5)**. To form a triangle, each fake node should craft an edge to connect with the target. For example, as shown in Fig. 5 (Case 1b), fake nodes u_4 and u_6 are already connected to each other via edge e_6, but neither is connected to the target node. In this case, we can connect both fake nodes to the target node by creating new edges e_{14} and e_{15} to form a triangle.
- **Only one fake node is connected to the target node (Case 2a, Fig. 5)**. In this case, the attack needs to create two new edges to craft a triangle. In particular, the attack should craft an edge between the target and another fake node that is disconnected from the target. In addition, a crafted edge should be inserted to connect two fake nodes. For example, as shown in Fig. 5 (Case 2b), the target node u_8 is connected to a fake node. To create a triangle, it creates the edge e_{14} between the fake node u_4 and target u_8, and edge e_{15} between the two fake nodes.
- **Both fake nodes are connected to the target node (Case 3a, Fig. 5)**. In such a case, the attack needs to create one new edge between the two fake nodes to craft a triangle. As shown in Fig. 5 (Case 3b), the target node u_8 is connected to both fake nodes via edges e_{14} and e_{16}, but the fake nodes are not connected to each other. Therefore, the triangle can be constructed by creating the edge e_{15} between the two fake nodes.

Observe that the attack can introduce more triangles associated with the target to the graph by crafting edges as above. However, if all fake nodes connect to target nodes, they may show some patterns that will expose the identity of a fake node. In other words, the fake node can be easily detected. To maximize the gain while disguising the identities of fake nodes, we limit the number of new connections each fake node can create to the average degree calculated from the perturbed graph for each ε value. To make better use of the limited number of connections, we design a prioritized allocation mechanism where fake nodes are first connected to each other, and only then to target nodes. This strategy helps to introduce more triangles. Once created, these connections remain unperturbed. The degree of each fake node is then calculated based on its connections and subsequently perturbed according to the LDP protocol.

Theorem 2. *Overall gain of MGA to clustering coefficient is*

$$Gain = r \cdot \frac{2}{p^2(2p-1)} \cdot \frac{1}{\bar{\tilde{d}}(\bar{\tilde{d}}-1)} \cdot \frac{m}{2 \cdot p'(1-p')^2 + p'^2(1-p') + 3 \cdot (1-p')^3}$$

where $\bar{\tilde{\mathrm{d}}}$ *is the average perturbed degree and* $\mathrm{p}' = \frac{\bar{\tilde{d}}}{N-1}$ *is the probability of forming a connection.*

VII. Countermeasures

To further validate the effectiveness of the proposed data poisoning attacks, we explore two countermeasures to address these attacks. The first countermeasure aims to detect fake nodes by analyzing the frequent itemsets. The intuition is that attackers may utilize similar patterns to create multiple fake nodes, which could manifest themselves as frequent itemsets in the perturbed bit vectors. The second countermeasure compares the difference between the degree calculated from the perturbed bit vector and the directly reported degree. The idea is that the discrepancy between these two values could serve as an indicator of the presence of fake nodes. As will be seen from experiments on real-world datasets (see Section VIII), we find that these countermeasures are ultimately not effective in mitigating the impact of the attacks. The attackers are able to evade detection by the proposed methods, which highlights the need for new defenses against our data poisoning attacks.

A. Frequent Itemsets based Detection

This countermeasure is adapted from the detection method proposed by Cao et al. [63]. It is particularly effective against MGA (Maximal gain attack), as MGA tends to create connections between fake nodes and target nodes, as well as among the fake nodes themselves. This behavior pattern manifests as frequent itemsets in the bit vectors. Therefore, an intuitive countermeasure is to analyze the frequency of frequent itemsets in the bit vectors. The detection method works as follows:

1) Utilize the Apriori algorithm [65], a frequent itemset mining technique, to identify connections that appear frequently in the bit vectors, i.e., frequent itemsets.
2) If the number of frequent itemsets in a node's bit vector exceeds a predefined threshold among the collected bit vectors, that node is classified as a fake node.
3) Unlike Cao's method [63], which directly removes detected fake nodes, our approach attempts to reconstruct the previous connections. We reconstruct the previous connections of these fake nodes based on the bit vector information of the real nodes connected to the identified fake nodes.

TABLE II: Datasets.

Dataset	Number of vertices	Number of edges
$Facebook$	4,039	88,234
$Enron$	36,692	183,831
$AstroPh$	18,772	198,110
$Gplus$	107,614	12,238,285

TABLE III: Default parameter settings.

Parameter	Default setting	Description
β	0.05	The fraction of fake users
γ	0.05	The fraction of target users
ε	4	Privacy budget

However, this method has a notable drawback: it may produce false positives, as certain legitimate social patterns can also form frequent itemsets.

B. Degree based Detection

This countermeasure is particularly effective against Random Value Attack. The principle behind this method is based on the difference between the reported degree and the estimated degree calculated from the perturbed bit vector.

In RVA, the attacker sends a randomly chosen degree from the degree space, which often differs significantly from the estimated degree that can be calculated from the perturbed bit vector. In contrast, genuine nodes typically have a reported degree that follows a Laplace distribution centered around the estimated degree from their perturbed bit vector. The detection method works as follows:

1) For each node, calculate the degree based on the uploaded perturbed bit vector.
2) Compare this estimated degree with the directly reported degree. If the difference between these two values exceeds a predefined threshold, mark the node as a fake node.
3) For each detected fake node, remove its connections from the nodes it claims to be connected to, thereby restoring the degrees of genuine nodes.

The threshold for detection is set to the maximum degree calculated from the reported perturbed bit vectors plus three times the standard deviation 3σ of the Laplace distribution.

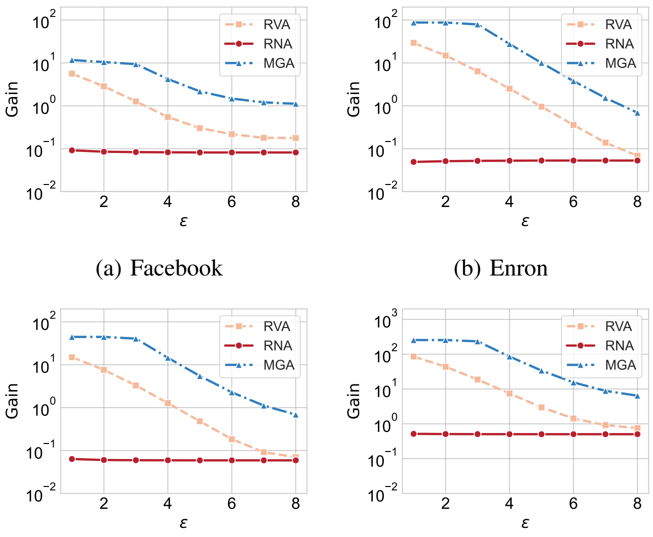
VIII. Evaluation

A. Experimental Setup

In this section, we evaluate the effectiveness of our proposed attacks. All experiments were conducted using Java 17.0.11.

Datasets. We evaluate our attacks on four real-world datasets that were widely used in the literature [62]. The statistics are in Table II.

- Facebook: an undirected social network consisting of 4,039 nodes and 88,234 edges, obtained from a survey of participants using the Facebook app.
- Enron: an undirected email communication network of 36,692 nodes and 183,831 edges. This dataset is derived



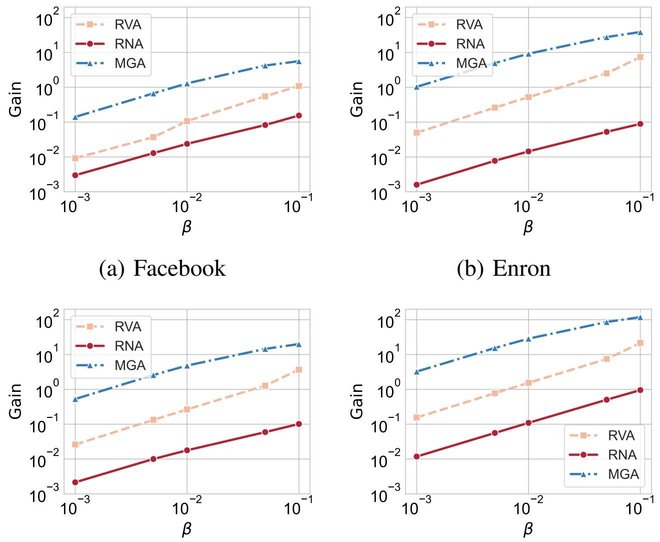
(a) Facebook

(b) Enron

(c) AstroPh

(d) Gplus

Fig. 6: Overall gains of attacks to degree centrality.



(a) Facebook

(b) Enron

(c) AstroPh

(d) Gplus

Fig. 7: Impact of β on the effectiveness of attacks to degree centrality.

from the email correspondence of Enron Corporation employees.

- AstroPh: an undirected collaboration network of 18,772 authors and 198,110 edges indicating collaborations between authors in arXiv, who submitted papers to the Astro Physical category.
- Gplus: an undirected social network of 107,614 Google+ users and 12,238,285 edges indicating shares of social circles.

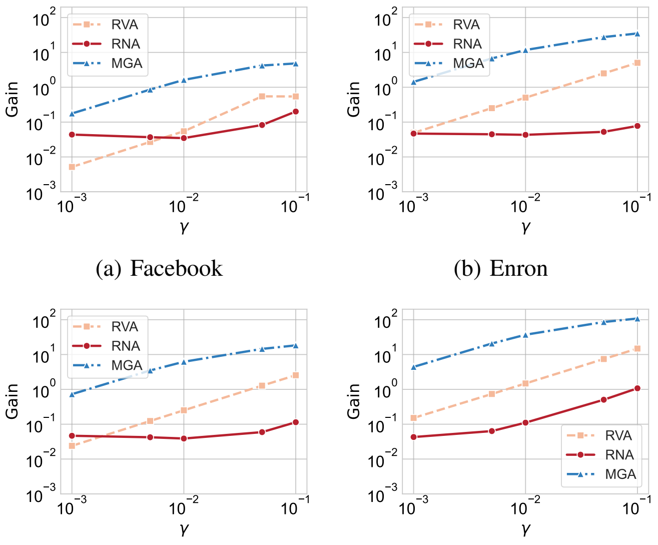
Parameter settings. For graph metric estimation, the overall gains of our attacks may depend on β (*i.e.,* the fraction of fake users), γ (*i.e.,* the fraction of target users), and ε (*i.e.,* privacy budget). Table III shows the default settings for these parameters, which will be used in our experiments unless otherwise specified. We will also analyze the impact of each parameter while fixing the remaining parameters to their default settings.

B. Results for Degree Centrality Estimation

Exp 1. Overall Results on Degree Centrality. We first evaluate the effectiveness of different attack strategies (RVA, RNA, and MGA) on degree centrality estimation under varying privacy budgets. The results are reported in Fig. 6 where the privacy budget ε ranges from 1 to 8. As can be seen in the figure, MGA and RVA show a clear inverse relationship between the privacy budget and attack effectiveness. This is because a larger privacy budget introduces a smaller number of edges allowed to be injected, which further leads to less gain. In contrast, the gain of MGA remains nearly unchanged. The reason lies in that RNA connects to only one target node, making it insensitive to changes in the privacy budget. However, no matter what the privacy budget is, MGA always outperforms RVA and RNA, as MGA can connect the limited number of injected edges to the targets.

Exp 2. Effect of β on Degree Centrality. We evaluate the impact of proportions of fake users on the effectiveness of different attack strategies (RVA, RNA, and MGA) for degree centrality estimation. The results are presented in Fig. 7 where β takes values of 0.001, 0.005, 0.01, 0.05, and 0.1. As shown in the figure, all three attack methods demonstrate a positive correlation between β and attack effectiveness. This trend can be attributed to the increased opportunities for manipulation that a higher number of fake nodes provides, which becomes more pronounced as β increases. With more fake users, attackers can create more connections and perturb the graph structure more significantly, leading to a greater impact on degree centrality estimates. Notably, across all values of β, we observe a consistent performance ranking: MGA consistently outperforms RVA, which in turn outperforms RNA. The reason is that MGA is an optimization-based method, while RNA simply adds one new connection per fake user.

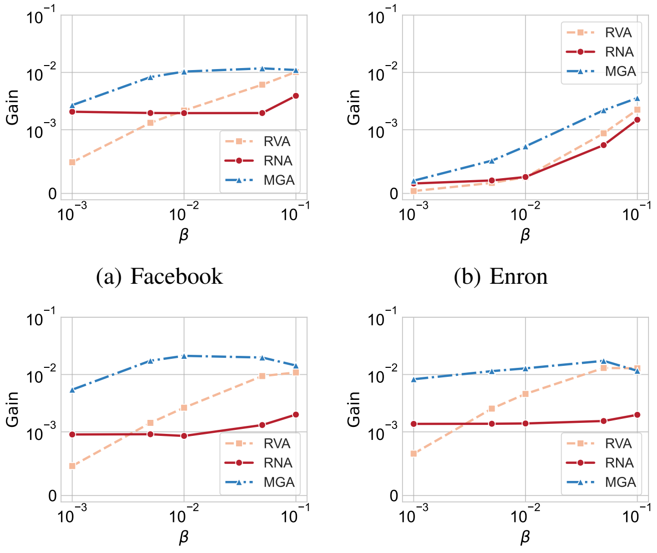
Exp 3. Effect of γ on Degree Centrality. We evaluate the impact of varying the proportion of target nodes on the effectiveness of different attack strategies (RVA, RNA, and MGA) for degree centrality estimation. The results are presented in Fig. 8 where γ takes values of 0.001, 0.005, 0.01, 0.05 and 0.1. As shown in the figure, all three attack methods demonstrate a positive correlation between γ and attack effectiveness. This phenomenon can be attributed to the expanded attack surface that a larger number of target nodes provides. With more target nodes available, attackers have a greater pool of potential connections to utilize, allowing them to more effectively alter the graph structure and influence degree centrality estimates. While RNA also shows a generally positive trend, its correlation is less pronounced. Similarly, across all values of γ, MGA consistently outperforms RVA and RNA. And in the majority of cases, RVA exhibits better performance than RNA.



(a) Facebook (b) Enron

(c) AstroPh (d) Gplus

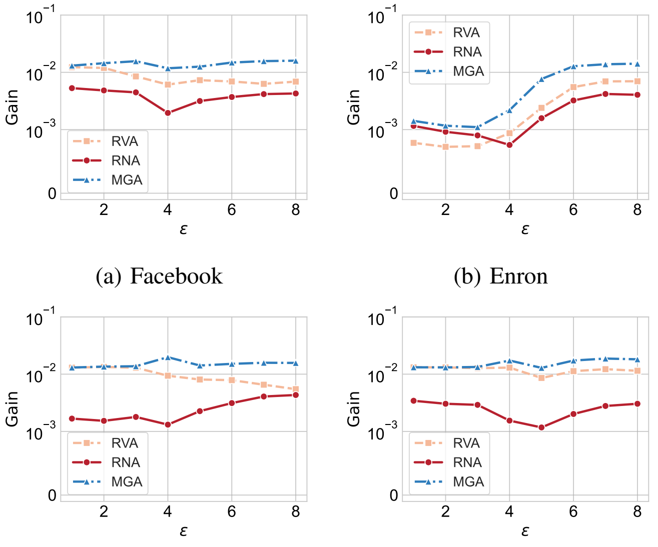
Fig. 8: Impact of γ on the effectiveness of attacks to degree centrality.



(a) Facebook (b) Enron

(c) AstroPh (d) Gplus

Fig. 10: Impact of β on the effectiveness of attacks to clustering coefficient.



(a) Facebook (b) Enron

(c) AstroPh (d) Gplus

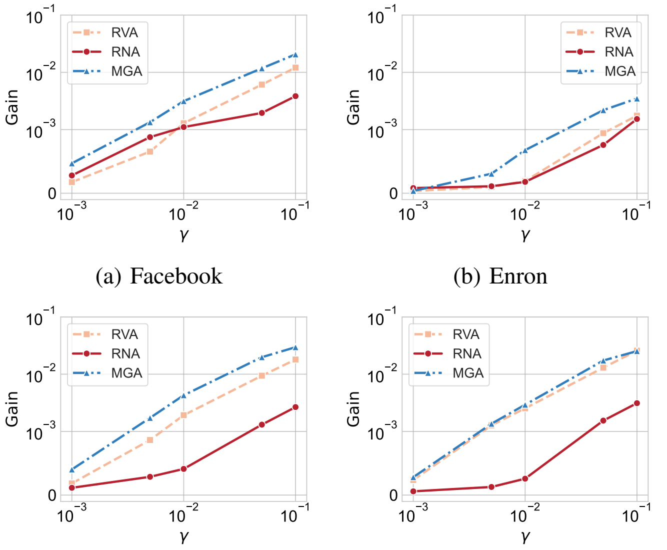
Fig. 9: Overall gains of attacks to clustering coefficient.

C. Results for Clustering Coefficient Estimation

Exp 4. Overall Results on Clustering Coefficient. We then evaluate the effectiveness of different attack strategies (RVA, RNA, and MGA) on clustering coefficient estimation under varying privacy budgets. The results are reported in Fig. 9 where the privacy budget ε also ranges from 1 to 8. The results show that the MGA consistently outperforms both RVA and RNA across all values of ε, with its performance remaining relatively stable as ε increases. This superior performance can be attributed to its optimization-based approach, which strategically crafts fake user data to maximize the overall gain. Generally, RVA performs better than RNA. The effectiveness of attacks on clustering coefficient estimation appears relatively insensitive to changes in the privacy budget ε for most datasets, with only slight fluctuations observed. This suggests that for clustering coefficient estimation, the privacy budget does not significantly alter the vulnerability of the graph to these attacks. However, the specific characteristics of social graphs can influence the impact of the privacy budget on attack effectiveness.

Exp 5. Effect of β on Clustering Coefficient. We evaluate the impact of varying proportions of fake users on the effectiveness of different attack strategies (RVA, RNA, and MGA) for clustering coefficient estimation. The results are reported in Fig. 10 where the proportion of fake users β takes values of 0.001, 0.005, 0.01, 0.05 and 0.1. As can be seen in the figure, these attacks demonstrate a positive correlation between β and attack effectiveness. This phenomenon can be attributed to the increased number of fake nodes available to the attacker. With more fake users, attackers can create a larger number of fake connections, allowing for more significant manipulation of the graph structure and, consequently, the estimated clustering coefficients. Observe that when β reaches a certain threshold (around 0.05-0.1 depending on the dataset), MGA's performance begins to plateau and becomes similar to RVA. This occurs because, at this point, the fake nodes in MGA have already connected to all target nodes. This plateau suggests an upper limit to the effectiveness of MGA when the proportion of fake users becomes sufficiently large.

Exp 6. Effect of γ on Clustering Coefficient. We evaluate the impact of varying the proportion of target nodes on the effectiveness of different attack strategies (RVA, RNA, and MGA) for clustering coefficient estimation. The results are reported in Fig. 11 where the proportion of target nodes γ



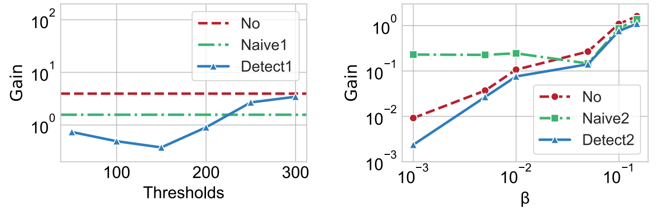
(a) Facebook

(b) Enron

(c) AstroPh

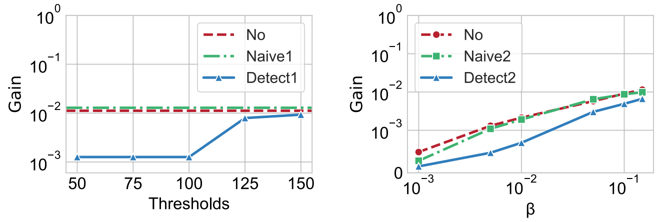
(d) Gplus

Fig. 11: Impact of γ on the effectiveness of attacks to clustering coefficient.



(a) Countermeasure against MGA (b) Countermeasure against RVA

Fig. 12: Countermeasures against attacks to degree centrality.



(a) Countermeasure against MGA (b) Countermeasure against RVA

Fig. 13: Countermeasures against attacks to clustering coefficient.

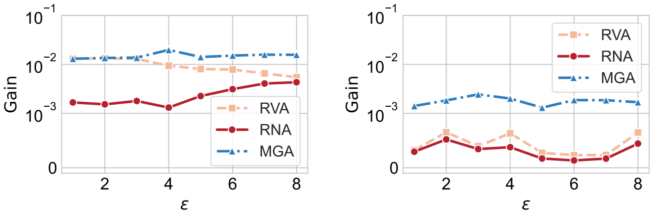
takes values of 0.001, 0.005, 0.01, 0.05, and 0.1. As can be seen in the figure, all three attack methods demonstrate a positive correlation between γ and attack effectiveness. This trend can be attributed to the fact that more target nodes likely imply a greater number of interconnections among these nodes, providing more opportunities for manipulation. MGA consistently outperforms RVA and RNA across all γ values and datasets, and RVA shows the second-best performance. We also observe that the relative performance of the attacks and their sensitivity to γ varies somewhat across different social graph datasets. This suggests that the underlying structure of the social graph influences the effectiveness of these attacks.

D. Results for Countermeasures

Exp 7. Countermeasures for Degree Centrality. We evaluate the effectiveness of two countermeasures against different attack strategies on degree centrality estimation. The results are reported in Fig. 12. To better evaluate the effectiveness of frequent itemsets based detection (denoted by Detect1) in defending against MGA on degree centrality, we implement a naive method (denoted by Naive1) that identifies the top 3% of highest-degree nodes as attackers. In Fig. 12(a), we vary the detection threshold for the frequent itemsets based method against MGA, with thresholds of 50, 100, 150, 200, 250, and 300. As can be seen in the figure, there is a U-shaped relationship between the detection threshold and the attack gain. At low thresholds, many genuine nodes are mistakenly identified as fake, leading to a decrease in gain. As the threshold increases, fewer nodes are flagged as fake, consequently increasing the attack's impact. This suggests an optimal threshold to maximize the effectiveness of this countermeasure. Note that Detect1 is generally more effective than Naive1, especially when the threshold is not significantly larger than the ratio of fake users.

To better evaluate the effectiveness of our degree based detection method (denoted by Detect2) in defending against RVA on degree centrality, we implement a naive method (denoted by Naive2) that identifies the nodes with degrees in either the top or bottom 3% of the degree distribution as fake nodes. In Fig. 12(b), we evaluate degree-based detection against RVA under different proportions of fake nodes, with β values of 0.001, 0.005, 0.01, 0.05, 0.1, and 0.15. As shown in the figure, varying β has minimal effect on the performance of the detection method. This is because the random nature of RVA leads to a high false-negative rate when identifying fake nodes, resulting in many fake nodes being missed by the detection algorithm regardless of their proportion. Compared to Detect2, Naive2 exhibits worse efficiency, generating even higher gains than the original results without detection. This is because it tends to classify many genuine nodes as fake.

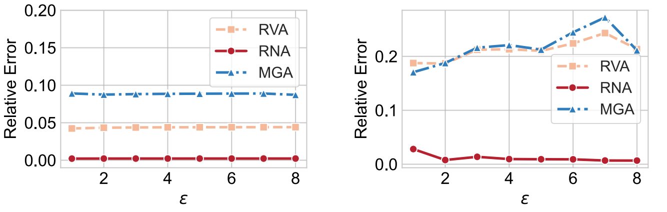
Exp 8. Countermeasures for Clustering Coefficient. We also evaluate the effectiveness of two detection methods against different attack strategies on clustering coefficient estimation. In Fig. 13(a), we vary the detection threshold for the frequent itemsets based method against MGA, using threshold values of 50, 75, 100, 125, and 150. As can be seen in the figure, the overall gain initially remains constant as the detection threshold increases, before gradually rising. This trend can be attributed to the threshold's impact on the number of detected fake nodes. At lower thresholds, the method effectively identifies and mitigates the impact of fake nodes. However, as the threshold increases, fewer nodes are flagged as fake, consequently allowing the attack's impact to grow. In Fig. 13(b), we evaluate degree-based detection against RVA under different proportions of fake nodes, with β



(a) LF-GDPR

(b) LDPGen

Fig. 14: Evaluation of attacks on LF-GDPR and LDPGen for clustering coefficient.



(a) LF-GDPR

(b) LDPGen

Fig. 15: Evaluation of attacks on LF-GDPR and LDPGen for modularity.

values of 0.001, 0.005, 0.01, 0.05, 0.1, and 0.15. As shown in the figure, the overall gain after applying the detection method is lower compared to the attack without detection, indicating that the countermeasure does provide some protection against RVA. However, the effectiveness of the detection remains relatively consistent across different β values, with only minimal variations observed as β changes. This suggests that while the degree-based detection method can reduce the impact of RVA, its performance is not significantly influenced by the proportion of fake nodes in the network. Similarly, while both countermeasures show some effectiveness in mitigating attacks, neither completely neutralizes them. This highlights the need for new defenses against the proposed attacks. Note that our countermeasures demonstrate greater effectiveness than naive solutions (e.g., Detect1 vs. Naive1 and Detect2 vs. Naive2).

E. Results for Attacks on LF-GDPR and LDPGen

Exp 9. Evaluation of Attacks on LF-GDPR and LDPGen. We finally evaluate the performance of three attack strategies against LF-GDPR and LDPGen, focusing on their impact on clustering coefficient and modularity metrics [13], [62]. The results are shown in Fig. 14 and 15. For both metrics, all three attacks prove effective across various privacy budgets ε, with MGA generally achieving the best performance, followed by RVA and RNA. These findings confirm the efficacy of our proposed attack strategies.

IX. Conclusion

In this work, we have demonstrated the susceptibility of local differential privacy (LDP) protocols for graph data collection to data poisoning attacks. Despite the privacy guarantees offered by LDP, we have shown that an attacker can inject fake users into the protocols and design targeted data poisoning attacks to significantly degrade the quality of collected graph metrics, such as degree centrality and clustering coefficient. In particular, we presented three novel data poisoning attacks, namely, RVA, RNA, and MGA, and theoretically proved that our MGA attack can maximally distort the collected graph metrics for targeted nodes. Our experimental evaluation on real-world datasets confirmed the effectiveness of these attacks in degrading the quality of the collected graph metrics. Furthermore, we explored two countermeasures against these data poisoning attacks, but found that they were unable to effectively offset the impact of the attacks. This highlights the pressing need for the development of more robust defenses to secure LDP protocols for graph data collection and analysis. Our work can serve as a call to the research community to prioritize the development of secure and reliable graph data collection and analysis methods that can withstand adversarial manipulation.

References

[1] Technical Report. Available at: https://github.com/hahahumble/DPA2Graphs/blob/main/TechnicalReport.pdf

[2] B. Stephanie, Facebook Scandal a 'Game Changer' in Data Privacy Regulation, Bloomberg, Apr. 8, 2018. [Online]. Available: https://www.bloomberg.com/news/articles/2018-04-07/facebookscandal-a-game-changer-in-dataprivacy-regulation

[3] Facebook, 2020. [Online]. Available: https://developers.facebook.com/docs/graph-api/

[4] B. Zhou and J. Pei, "Preserving privacy in social networks against neighborhood attacks," in *Proc. IEEE 24th Int. Conf. Data Eng.*, 2008, pp. 506–515.

[5] K. Liu and E. Terzi, "Towards identity anonymization on graphs," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2008, pp. 93-106.

[6] K. Nissim, S. Raskhodnikova, and A. Smith, ”Smooth sensitivity and sampling in private data analysis," in *Proc. 39th Annu. ACM Symp. Theory Comput.*, 2007, pp. 75-84.

[7] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptogr. Conf.*, 2006, pp. 265–284.

[8] J. Jia and N. Z. Gong, "Calibrate: Frequency estimation and heavy hitter identification with local differential privacy via incorporating prior knowledge," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2019, pp. 2008-2016.

[9] G. Cormode, T. Kulkarni, and D. Srivastava, "Marginal release under local differential privacy," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2018, pp. 131-146.

[10] B. Avent, A. Korolova, D. Zeber, T. Hovden, and B. Livshits, "BLENDER: Enabling local search with a hybrid differential privacy model," in *Proc. 26th USENIX Secur. Symp.*, 2017, pp. 747-764.

[11] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci.*, 2013, pp. 429–438.

[12] 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 Asia Conf. Comput. Commun. Secur. (ASIACCS)*, 2016, pp. 192-203.

[13] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 425–438.

[14] R. Bassily, K. Nissim, U. Stemmer, and A. G. Thakurta, "Practical locally private heavy hitters," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 2285–2293.

[15] R. Bassily and A. Smith, "Local, private, efficient protocols for succinct histograms," in *Proc. Conf. Symp. Theory Comput.*, 2015, pp. 127–135.

[16] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: Randomized aggregatable privacy-preserving ordinal response," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2014, pp. 1054–1067.

[17] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2879–2887.

[18] P. Kairouz, K. Bonawitz, and D. Ramage, "Discrete distribution estimation under local privacy," in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 2436-2444.

[19] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *Proc. USENIX Secur. Conf.*, 2017, pp. 729–745.

[20] X. Ren et al., "LoPub: High-dimensional crowdsourced data publication with local differential privacy," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 9, pp. 2151-2166, Sep. 2018.

[21] T. Wang, N. Li, and S. Jha, "Locally differentially private frequent itemset mining," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, 2018, pp. 127-143.

[22] T. Wang et al., "Answering multi-dimensional analytical queries under local differential privacy,"' in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2019, pp. 159-176.

[23] T. Wang, M. Lopuhaä-Zwakenberg, Z. Li, B. Skoric, and N. Li, "Locally differentially private frequency estimation with consistency," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2020, pp. 1-16.

[24] T. Wang, N. Li, and S. Jha, "Locally differentially private heavy hitter identification," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 2, pp. 982-993, Mar./Apr. 2021.

[25] Q. Ye et al., "PrivKVM*: Revisiting key-value statistics estimation with local differential privacy," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 17-35, Jan./Feb. 2022.

[26] 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 Asia Conf. Comput. Commun. Secur. (ASIACCS)*, 2018, pp. 212-229.

[27] Q. Ye, H. Hu, X. Meng, and H. Zheng, "PrivKV: Key-value data collection with local differential privacy," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, 2019, pp. 317-331.

[28] Q. Ye, H. Hu, N. Li, X. Meng, H. Zheng, and H. Yan, "Beyond value perturbation: Local differential privacy in the temporal setting," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2021, pp. 1-10.

[29] X. Ren et al., "LDP-IDS: Local differential privacy for infinite data streams," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2022, pp. 1064-1077.

[30] H. Wang et al., "L-SRR: Local differential privacy for location-based services with staircase randomized response," in *Proc. ACM Asia Conf. Comput. Commun. Secur. (ASIACCS)*, 2022, pp. 2809-2823.

[31] M. Zhou et al., "Locally differentially private sparse vector aggregation," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, 2022, pp. 422–439.

[32] Y. Zhang et al., "Trajectory data collection with local differential privacy," *arXiv preprint arXiv:2307.09339*, Jul. 2023.

[33] Y. Du et al., "LDPTrace: Locally differentially private trajectory synthesis," *arXiv preprint arXiv:2302.06180*, Feb. 2023.

[34] X. Li et al., "Local differentially private heavy hitter detection in data streams with bounded memory," *arXiv preprint arXiv:2311.16062*, Nov. 2023.

[35] Y. Mao, Q. Ye, H. Hu, Q. Wang, and K. Huang. PrivShape: Extracting shapes in time series under user-level local differential privacy. In *ICDE*, 2024.

[36] Collaborative sampling for partial multi-dimensional value collection under local differential privacy. Q Qian, Q Ye, H Hu, K Huang, TTL Chan, J Li *IEEE Transactions on Information Forensics and Security*, 3948-3961, 2023.

[37] Stateful switch: Optimized time series release with local differential privacy. Q Ye, H Hu, K Huang, MH Au, Q Xue *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, 1-10, 2023.

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

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

[40] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. Adversarial machine learning. In *AISec*, 2011.

[41] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks. In *AAAI*, 2021.

[42] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2018, pp. 19-35.

[43] M. Mozaffari-Kermani, S. Sur-Kolay, A. Raghunathan, and N. K. Jha, "Systematic poisoning attacks on and defenses for machine learning in healthcare," *IEEE J. Biomed. Health Inform.*, vol. 19, no. 6, pp. 1893-1905, Nov. 2015.

[44] K. Huang, G. Ouyang, Q. Ye, et al., "LDPGuard: Defenses against data poisoning attacks to local differential privacy protocols," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3195-3209, July. 2024.

[45] Das, Sudipto and Eğecioğlu, Ömer and El Abbadi, Amr. Anonymizing weighted social network graphs. *ICDE*, 2010.

[46] J. Gao, J. X. Yu, R. Jin, J. Zhou, T. Wang, and D. Yang. Neighborhood-privacy protected shortest distance computing in cloud. *SIGMOD*, 2011.

[47] N. Cao, Z. Yang, et al. Privacy-preserving query over encrypted graph-structured data in cloud computing. *ICDCS*, 2011.

[48] Huang K, Hu H, Zhou S, et al. Privacy and efficiency guaranteed social subgraph matching. *The VLDB Journal*, 2022.

[49] "FRESH: Towards Efficient Graph Queries in an Outsourced Graph." K Huang, Y Li, Q Ye, Y Tian, X Zhao, Y Cui, H Hu, X Zhou. In *ICDE*, 2024.

[50] P. Samarati, "Protecting respondents identities in microdata release," *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 6, pp. 1010-1027, Nov./Dec. 2001.

[51] L. Zou, L. Chen, and M. T. Özsu, “K-automorphism: A general framework for privacy preserving network publication,” *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 946-957, 2009.

[52] A. Narayanan and V. Shmatikov, “De-anonymizing social networks,” in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2009, pp. 173-187.

[53] S. Nobari, P. Karras, H. Pang, and S. Bressan, “L-opacity: Linkage-aware graph anonymization,” in *Proc. Int. Conf. Extending Database Technol. (EDBT)*, 2014, pp. 583-594.

[54] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao, “Sharing graphs using differentially private graph models,” in *Proc. ACM SIGCOMM Conf. Internet Meas. Conf. (IMC)*, 2011, pp. 81-98.

[55] D. Mir and R. N. Wright, “A differentially private estimator for the stochastic kronecker graph model,” in *Proc. Joint EDBT/ICDT Workshops*, 2012, pp. 167-176.

[56] W. Lu and G. Miklau, “Exponential random graph estimation under differential privacy,” in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining (KDD)*, 2014, pp. 921-930.

[57] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev, “Private analysis of graph structure,” *Proc. VLDB Endowment*, vol. 4, no. 11, pp. 1146-1157, 2011.

[58] H. Sun *et al.*, “Analyzing subgraph statistics from extended local views with decentralized differential privacy,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2019, pp. 703-717.

[59] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith, “Analyzing graphs with node differential privacy,” in *Proc. Theory Cryptogr. Conf. (TCC)*, 2013, pp. 457-476.

[60] M. Hay, C. Li, G. Miklau, and D. Jensen, “Accurate estimation of the degree distribution of private networks,” in *Proc. 9th IEEE Int. Conf. Data Mining (ICDM)*, 2009, pp. 169-178.

[61] Y. Wang, X. Wu, J. Zhu, and Y. Xiang, “On learning cluster coefficient of private networks,” *Soc. Netw. Anal. Mining*, vol. 3, no. 4, pp. 925-938, 2013.

[62] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, “LF-GDPR: A framework for estimating graph metrics with local differential privacy,” in *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 10, pp. 4905-4920, Oct. 2020.

[63] X. Cao, J. Jia, and N. Z. Gong, “Data poisoning attacks to local differential privacy protocols,” in *Proc. USENIX Secur. Conf.*, 2021, pp. 947-964.

[64] A. Cheu, A. Smith, and J. Ullman, “Manipulation attacks in local differential privacy,” in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2021, pp. 883–900.

[65] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules,” in *Proc. 20th Int. Conf. Very Large Data Bases (VLDB)*, 1994, pp. 487-499.

X. Appendix

A. *Proof of Theorem 1.*

The overall gain of MGA to degree centrality is

$$\text{Gain} = \frac{m \cdot r}{N-1} \cdot \left(\frac{\min(r, \lfloor \bar{\tilde{d}} \rfloor)}{r} - \frac{\bar{\tilde{d}}}{N-1} \right)$$

where $\bar{\tilde{d}}$ is the average degree of the graph after perturbation.

Proof. To prove this theorem, we begin by considering the probability of an existing connection between any two nodes, which is $p' = \frac{\bar{\tilde{d}}}{N-1}$. Next, we consider the maximum number of new connections that can be added per fake node. Each fake node can add at most $\min(r, \lfloor \bar{\tilde{d}} \rfloor)$ connections, but we need to subtract the existing connections. Thus, the maximum number of new connections per fake node is $\min(r, \lfloor \tilde{d} \rfloor) - p \cdot r$.

Extending this to all fake nodes, the total number of new connections that can be added is $E_{\text{new}} = m \cdot \left(\min(r, \lfloor \bar{\tilde{d}} \rfloor) - p \cdot r \right) = m \cdot \left(\min(r, \lfloor \bar{\tilde{d}} \rfloor) - \frac{r \bar{\tilde{d}}}{N-1} \right)$.

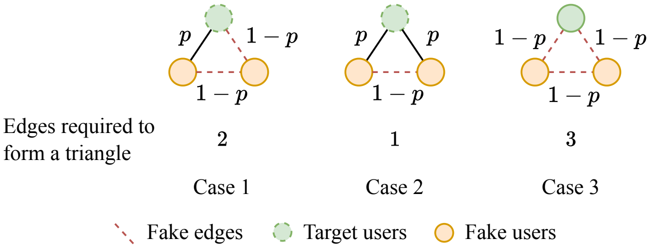


Fig. 16: Edges required to form a triangle.

We now consider the impact on degree centrality. When a single node’s degree increases by 1, its degree centrality changes by $\frac{1}{N-1}$. Therefore, the overall gain in degree centrality for all target nodes can be expressed as Gain $= \frac{E_{\text{new}}}{N-1}$. Substituting the expression for E_{new}, we have $\mathit{Gain} = \frac{m}{N-1} \cdot (\min(r, \lfloor \bar{\tilde{d}} \rfloor) - \frac{r \bar{\tilde{d}}}{N-1}) = \frac{m \cdot r}{N-1} \cdot (\frac{\min(r, \lfloor \bar{\tilde{d}} \rfloor)}{r} - \frac{\bar{\tilde{d}}}{N-1})$. □

B. *Proof of Theorem 2.*

The overall gain of MGA to the clustering coefficient is

$$\text{Gain} = r \cdot \frac{2}{p^2(2p-1)} \cdot \frac{1}{\bar{\tilde{d}}(\bar{\tilde{d}}-1)} \cdot \frac{m}{2 \cdot p'(1-p')^2 + p'^2(1-p') + 3 \cdot (1-p')^3}$$

where $\bar{\tilde{d}}$ is the average perturbed degree and $p' = \frac{\bar{\tilde{d}}}{N-1}$ is the probability of forming a connection.

Proof. We begin by considering the probability of a connection, given by $p' = \frac{\bar{\tilde{d}}}{N-1}$, where $\bar{\tilde{d}}$ is the average perturbed degree and N is the total number of nodes. To estimate the probability of forming a triangle, we should consider three possible scenarios, as shown in Fig. 16. Observe that the probability is $p'^2(1-p')$ when two edges already exist (Case 2). Similarly, the probability is $2 \cdot p'(1-p')^2$ when one edge exists (Case 1) and $3 \cdot (1-p')^3$ when no edges exist (Case 3), respectively. Given m fake nodes, each target node can have m connections with fake nodes. Consequently, the number of new triangles to be introduced is $\frac{m}{2 \cdot p'(1-p')^2 + p'^2(1-p') + 3 \cdot (1-p')^3}$.

For a target node i, the difference of clustering coefficient cc_i before and after the attack can be expressed as $\frac{2}{p^2(2p-1)} \cdot \frac{1}{\widetilde{d}_i(\widetilde{d}_i-1)} \cdot \frac{m}{2 \cdot p'(1-p')^2 + p'^2(1-p') + 3 \cdot (1-p')^3}$, where $\widetilde{d}_i$ represents the perturbed degree for node i. Since there are r target nodes, the overall gain is therefore $r \cdot \frac{2}{p^2(2p-1)} \cdot \frac{1}{\widetilde{d}(\widetilde{d}-1)} \cdot \frac{m}{2 \cdot p'(1-p')^2 + p'^2(1-p') + 3 \cdot (1-p')^3}$. □