Triangle Counting over Signed Graphs with Differential Privacy

Zening Li, Rong-Hua Li, Fusheng Jin
Beijing Institute of Technology, Beijing, China
zening-li@outlook.com, lironghuabit@126.com, jfs21cn@bit.edu.cn

***Abstract*—Triangle counting serves as a foundational operator in graph analysis. Since graph data often contain sensitive information about entities, the release of triangle counts poses privacy concerns. While recent studies have addressed privacy-preserving triangle counting, they mainly concentrate on unsigned graphs. In this paper, we investigate a new problem of developing triangle counting algorithms for signed graphs that adhere to centralized differential privacy and local differential privacy, respectively. The inclusion of edge signs and more classes of triangles leads to increased complexity and overwhelms the statistics with noise. To overcome these problems, we first propose a novel algorithm for smooth-sensitivity computation to achieve differential privacy under the centralized model. In addition, to handle large signed graphs, we devise a computationally efficient function that calculates a smooth upper bound on local sensitivity. Finally, we release the approximate triangle counts after the introduction of Laplace noise, which is calibrated to the smooth upper bound on local sensitivity. In the local model, we propose a two-phase framework tailored for balanced and unbalanced triangle counting. The first phase utilizes the Generalized Randomized Response mechanism to perturb data, followed by a novel response mechanism in the second phase. Extensive experiments conducted over real-world datasets demonstrate that our proposed methods can achieve an excellent trade-off between privacy and utility.**

I. Introduction

Many relationships in the real world can be represented by signed graphs with positive and negative edges, as a result of which signed graph analysis has attracted much attention and nurtured numerous applications [1]–[7]. However, most real-world signed graphs associated with people or human-related activities, such as social and economic networks, hold sensitive information about the relationships between individuals. These positive and negative relationships reveal intricate details about individual interactions, preferences, and social dynamics. For instance, economic transaction networks between financial institutions can be modeled to reflect lending relationships, where the signs indicate the health of these transactions, such as timely repayments or defaults. Given the private nature of the information stored in signed graphs and the pressure to allow controlled release of private data, there is a considerable interest in how to release information with assured privacy.

Differential privacy (DP) [8] has been the dominant model for the protection of individual privacy from powerful and realistic attackers. This model works well for data published as histograms or counts. However, methods that attempt to directly output a modified version of the input signed graph under DP result in the loss of its inherent properties. The major technical obstacle lies in the fact that the direct application of standard approach seems to require a lot of random modifications to the input signed graph. As an alternative, we can focus on the release of statistical properties of the signed graph under differential privacy rather than the signed graph itself. Triangle counting is a fundamental primitive in signed graph analysis. A triangle is considered balanced if it contains an odd number of positive edges and unbalanced otherwise [9] (refer to Figure 1 for illustration). Statistics based on balanced and unbalanced triangle counts reveal important structural information about signed graphs. It is valuable to release these counts, because there are many stochastic signed graph models which rely on these statistics to generate graphs, such as the Balanced Signed Chung-Lu model [9] and 2-Layer Signed Network model [5]. In addition, these metrics also support various computational tasks in signed graph analysis, such as link prediction [1] and community detection [4], [10].

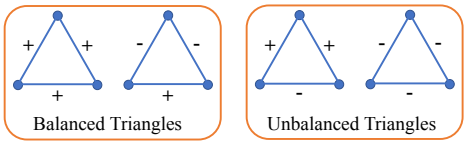


Fig. 1: Examples of balanced and unbalanced triangles

The problem then becomes how to efficiently and privately release the number of balanced and unbalanced triangles in input signed graphs. Differential privacy can be divided into centralized DP and local DP based on the underlying architecture. In the centralized model, a trusted curator holds the sensitive data and releases sanitized versions of the statistics. In contrast, local DP assumes that the data curator is untrusted. Under this premise, each user perturbs his/her data locally and transmits the perturbed data to the untrusted data curator. Numerous studies have been conducted on differentially private graph analysis algorithms, spanning both centralized [11]–[16] and local models [17]–[22]. Despite the strides made in this field, it is notable that these studies concentrated on unsigned graphs. The exploration of balanced and unbalanced triangle counting in signed graphs, the focus of this paper, remains an underexplored area in the current literature.

To fill this gap, we investigate the problem of balanced and unbalanced triangle counting under centralized DP and local DP, where both the graph structure and the sign of edges are considered private information. However, the inclusion of edge signs introduces unique challenges for privacy preservation. (i) Noise scale: a standard DP technique directly adds noise to the

statistical estimates. However, as the noise scale is proportional to the number of nodes, it causes excessive perturbation to the release statistics. (ii) Complexity of smooth sensitivity: to reduce the introduction of noise, we explore the technique of smooth sensitivity as an alternative. However, the computation of smooth sensitivity is obviously more complex in signed graphs. (iii) Scalability: beyond the computational complexity, the calculation of smooth sensitivity is extremely expensive.

To solve the mentioned issues, we propose several effective methods under centralized DP and local DP, respectively. In the centralized model, in order to reduce the introduction of noise, we adopt the smooth sensitivity framework to calculate the number of balanced and unbalanced triangles in the signed graph. More specifically, we first analyze the local sensitivity and local sensitivity at distance t of the triangle counting query function. Then, we propose a novel and efficient method for the computation of smooth sensitivity, with time complexity of $\mathcal{O}(m \cdot d_{max} + n^2)$[1]. Furthermore, to handle large signed graphs, we devise a computationally efficient function that calculates a smooth upper bound on local sensitivity, with a reduced time complexity of $\mathcal{O}(m \cdot d_{max})$. The final step is to release triangle counts after adding Laplace noise, where the noise is calibrated to the smooth upper bound on local sensitivity. In the local model, we propose a two-phase framework tailored for balanced and unbalanced triangle counting. The first phase uses the Generalized Randomized Response mechanism to perturb the data. And in the second phase, we propose a novel response mechanism with an unbiased correction to the final statistics. This mechanism injects noise into the query response of each node, which is also calibrated to the smooth upper bound on local sensitivity. At the end, to evaluate the effectiveness of our proposed methods, we conduct extensive experiments on several real-world datasets. The results show that our proposed approaches establish state-of-the-art performance and achieve a better privacy-utility trade-off than the baseline methods. To summarize, our main contributions of this paper are as follows:

- We propose innovative methods for estimating the number of balanced and unbalanced triangles under centralized and local DP. To our knowledge, this is the first exploration of privacy-preserving triangle counting in signed graphs.
- In the centralized model, we introduce a novel and efficient method for the computation of smooth sensitivity. To handle large signed graphs, we develop a computationally efficient function that calculates a smooth upper bound on local sensitivity. This approach reduces the computational complexity from $\mathcal{O}(m \cdot d_{max} + n^2)$ to $\mathcal{O}(m \cdot d_{max})$.
- In the local model, we propose a two-phase framework for triangle counting. The second phase features a novel response mechanism that enables each node to independently add noise, calibrated to the smooth upper bound on local sensitivity, to its reported statistics.
- We conduct extensive experiments across several real-world datasets. The experimental results show that our proposed methods achieve excellent privacy-utility trade-offs in both centralized and local models.

[1]Note that m denotes the number of edges, n represents the number of nodes, and d_{max} is the maximum degree.

Roadmap. In Section II, we formally define the research problem and provide an introduction to the fundamentals of DP. Section III and Section IV detail our proposed methods for the estimation of balanced and unbalanced triangle counts under centralized DP and local DP, respectively. The experimental results are presented in Section V, followed by a review of the related literature in Section VI. Finally, Section VII concludes this paper.

II. Preliminaries

In this section, we introduce the formal definition of the research problem, followed by a concise overview of differential privacy and its properties.

A. Problem Statement

We consider an undirected signed graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ denotes the set of n nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents the set of m edges. Each edge in G carries a sign "+" or "-". An edge with the sign "+" denotes a positive edge, while an edge with the sign "-" represents a negative edge. The degree of v_i is expressed as d_i, and the maximum degree is denoted as d_{max}. Let $\mathcal{G}$ be the collection of all possible signed graphs on n nodes. Any signed graph $G \in \mathcal{G}$ can also be represented as a symmetric adjacency matrix $\mathbf{A} \in \{0, +1, -1\}^{n \times n}$, where $a_{i,j} = +1$ indicates a positive edge, $a_{i,j} = -1$ denotes a negative edge, and $a_{i,j} = 0$ means no edge exists between v_i and v_j. Let $\boldsymbol{a}_i = (a_{i,1}, \ldots, a_{i,n}) \in \{0, +1, -1\}^n$ denote the adjacency vector of the node v_i, which corresponds to the i-th row of the adjacency matrix $\mathbf{A}$.

In this paper, we consider a query function $f_{\triangle} \colon \mathcal{G} \to (T^b, T^u)$ that takes a signed graph G as input and outputs the number of balanced and unbalanced triangles, denoted as T^b and T^u. Note that we assume the set of nodes $\mathcal{V}$ is public, so the identities of the nodes are non-private. Similar assumptions are widely made in previous privacy-preserving triangle counting studies [11]–[13], [20], [22]. In contrast, the set of edges $\mathcal{E}$ and the signs of edges are considered sensitive information. Our ultimate objective is to develop algorithms that can accurately estimate the balanced and unbalanced triangle counts while protecting individual privacy.

B. Differential Privacy for Signed Graphs

Differential privacy [8] has become the standard framework for data release that provides robust privacy protection in the presence of powerful and realistic adversaries. A randomized algorithm satisfies differential privacy when the distributions of its outputs are similar for any pair of neighboring databases. Therefore, the formal definition of differential privacy revolves around the concept of neighboring databases. In this paper, we use the edit distance to measure the distinction between two databases. The edit distance quantifies the minimum number of operations required to transform one object into another. We exclusively focus on edge operations, since the graph structure and edge signs are considered private information.

Centralized Differential Privacy. In the centralized model, a trusted curator first collects the private data from all users and then releases the sanitized statistical data. Next, we formally define centralized differential privacy for signed graphs.

Definition 1 (Distance between signed graphs, Neighbors)**.** *The distance between n-node signed graphs G and $\tilde{G}$, denoted as $d(G, \tilde{G})$, is the minimum number of primitive operations required to convert G to $\tilde{G}$. There are three types of primitive edit operations: (1) edge insertion to introduce a new signed edge between a pair of nodes; (2) edge deletion to remove a single edge between a pair of nodes; (3) edge substitution to alter the sign of a given edge. Signed graphs G and $\tilde{G}$ are considered neighbors if $d(G, \tilde{G}) = 1$.*

Definition 2 (Signed Edge Centrialized Differential Privacy)**.** *Given $\epsilon > 0$ and $\delta > 0$, a randomized algorithm $\mathcal{A}$ satisfies (ϵ, δ)-signed edge CDP if and only if for any two neighboring signed graphs $G, \tilde{G} \in \mathcal{G}$, and for any possible output $O \in Range(\mathcal{A})$, we have*

$$\Pr[\mathcal{A}(G) = O] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(\tilde{G}) = O] + \delta. \tag{1}$$

The parameter ϵ is called the privacy budget that controls the trade-off between privacy protection level and utility. A lower privacy budget indicates stronger privacy protection but poorer utility. The parameter δ is treated as the probability of failure and is usually chosen to be much smaller than the inverse of the number of data records. When $\delta = 0$, the algorithm satisfies ϵ-signed edge centralized differential privacy.

Local Differential Privacy. In contrast to centralized DP, local differential privacy [23] operates under the assumption of a local model, where the data curator is untrusted. In this model, each user locally perturbs his/her own data by a randomized perturbation algorithm and sends the obfuscated data to the data curator.

Definition 3 (Distance between adjacency vectors, Neighbors)**.** *The distance between two adjacency vectors $\boldsymbol{a}_i, \tilde{\boldsymbol{a}}_i \in \{0, +1, -1\}^n$, denoted as $d(\boldsymbol{a}_i, \tilde{\boldsymbol{a}}_i)$, is defined as the least number of elementary operations required to transform $\boldsymbol{a}_i$ into $\tilde{\boldsymbol{a}}_i$. These operations include (1) the insertion of a signed edge connected to v_i; (2) the removal of an edge from node v_i; (3) the alteration of the sign of an edge linked to v_i. Adjacency vectors $\boldsymbol{a}_i$ and $\tilde{\boldsymbol{a}}_i$ are neighbors when $d(\boldsymbol{a}_i, \tilde{\boldsymbol{a}}_i) = 1$.*

The primary difference between the distance metrics of adjacency vectors and signed graphs is the choice of elementary edit operations.

Definition 4 (Signed Edge Local Differential Privacy)**.** *Given $\epsilon > 0$ and $\delta > 0$, a randomized algorithm $\mathcal{A}$ satisfies (ϵ, δ)-signed edge LDP if and only if for any two neighboring adjacency vectors $\boldsymbol{a}_i$ and $\tilde{\boldsymbol{a}}_i$, and for any possible output O of $\mathcal{A}$, we have*

$$\Pr[\mathcal{A}(\boldsymbol{a}_i) = O] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(\tilde{\boldsymbol{a}}_i) = O] + \delta. \tag{2}$$

Similarly, when $\delta = 0$, $\mathcal{A}$ satisfies ϵ-signed edge local differential privacy.

C. Differential Privacy Mechanisms

Various approaches have been proposed to achieve differential privacy. One prevalent method is to introduce carefully chosen random noise to the true results. In this article, we utilize the Laplace distribution to add noise. A Laplace random variable with mean 0 and standard deviation $\sqrt{2}\lambda$ has density $h(z) = \frac{1}{2\lambda} e^{-|z|/\lambda}$, denoted as $Lap(\lambda)$. The Laplace mechanism [8], a fundamental technique that satisfies ϵ-DP, introduces independent and identically distributed noise from $Lap(GS_f/\epsilon)$ to each element of the output produced by a query function f: $\mathcal{D} \to \mathbb{R}^k$, where GS_f is the global sensitivity of f.

Definition 5 (Global Sensitivity [8])**.** *Given a query function f: $\mathcal{D} \to \mathbb{R}^k$, the global sensitivity of f is defined as*

$$GS_f = \max_{x, \tilde{x} \in \mathcal{D}: d(x, \tilde{x}) = 1} ||f(x) - f(\tilde{x})||_1, \tag{3}$$

where $|| \cdot ||_1$ represents the ℓ_1 norm.

The amount of noise added by the Laplace mechanism depends on GS_f and the privacy budget ϵ. It is crucial to note that GS_f is an inherent characteristic of the function f, independent of any input data. However, for the query functions considered in this article, this mechanism introduces a substantial amount of noise to the results, which can compromise the performance. In [11], the authors propose a local measure of sensitivity.

Definition 6 (Local Sensitivity [11])**.** *The local sensitivity of a function f: $\mathcal{D} \to \mathbb{R}^k$ on a database $x \in \mathcal{D}$ is*

$$LS_f(x) = \max_{\tilde{x} \in \mathcal{D}: d(x, \tilde{x}) = 1} ||f(x) - f(\tilde{x})||_1. \tag{4}$$

From the definition provided, it is evident that the local sensitivity depends not only on the query function f, but also on the concrete instance x. For many problems, $LS_f(x)$ tends to be smaller than GS_f [11], [12], [14], [24]. Unfortunately, local sensitivity does not satisfy the requirement of differential privacy, because $LS_f(x)$ itself contains information about the database. To solve this problem, Nissim et al. [11] employ a β-smooth upper bound on local sensitivity, rather than the local sensitivity itself, for noise calibration. Specifically, for $\beta > 0$, a function $S_{f,\beta}$: $\mathcal{D} \to \mathbb{R}$ is a β-smooth upper bound on local sensitivity of f, if $\forall x \in \mathcal{D}$, $S_{f,\beta}(x) \geq LS_f(x)$ and $\forall x, \tilde{x} \in \mathcal{D}$ with $d(x, \tilde{x}) = 1$, $S_{f,\beta}(x) \leq e^{\beta} S_{f,\beta}(\tilde{x})$. $LS_f(x)$ may have multiple smooth bounds, and the smooth sensitivity is the smallest one that meets the condition.

Definition 7 (Smooth Sensitivity [11])**.** *For any query function f: $\mathcal{D} \to \mathbb{R}^k$ and $\beta > 0$, the β-smooth sensitivity of f at $x \in \mathcal{D}$ is defined as follows:*

$$S^*_{f,\beta}(x) = \max_{\tilde{x} \in \mathcal{D}} (e^{-\beta d(x, \tilde{x})} \cdot LS_f(\tilde{x})). \tag{5}$$

To calculate the smooth sensitivity efficiently, we construct a function known as the local sensitivity at distance t. This function, denoted as $LS_f(x, t)$, can be calculated as follows:

$$LS_f(x, t) = \max_{\tilde{x} \in \mathcal{D}: d(x, \tilde{x}) \leq t} LS_f(\tilde{x}). \tag{6}$$

Here, $LS_f(x, t)$ is the maximum local sensitivity $LS_f(\tilde{x})$ of all databases $\tilde{x}$ where the maximum distance between $\tilde{x}$ and x is t. In other words, we permit at most t modifications to

the database x before computing its local sensitivity. Then the β-smooth sensitivity $S^*_{f,\beta}(x)$ can be expressed in terms of $LS_f(x,t)$ as

$$S^*_{f,\beta}(x) = \max_{t=0,1,\ldots,n} e^{-\beta t} LS_f(x,t). \tag{7}$$

Theorem 1 (Noise Calibration to Smooth Bound [11])**.** *Given a query function $f\colon \mathcal{D} \to \mathbb{R}^k$, suppose $S_{f,\beta}(x)$ is a β-smooth upper bound on local sensitivity of f, where $\beta = \frac{\epsilon}{4(k+\ln(2/\delta))}$. Then the algorithm $\mathcal{A}_{f,\epsilon}(x) = f(x) + (z_1, \ldots, z_k)$ satisfies (ϵ, δ)-DP, where the z_i are drawn i.i.d. from $Lap(2S_{f,\beta}(x)/\epsilon)$.*

Generalized Randomized Response (GRR) [25] is another perturbation mechanism that satisfies ϵ-LDP. Specifically, each user locally perturbs their private discrete value $x \in \mathcal{D}$ via the GRR mechanism. This mechanism outputs the true value x with probability $\frac{e^\epsilon}{e^\epsilon+k-1}$, or an arbitrary value $\tilde{x} \neq x$ with probability $\frac{1}{e^\epsilon+k-1}$, where $k = |\mathcal{D}|$ is the domain size.

Moreover, differential privacy has some important properties that assist in the development of sophisticated algorithms.

Proposition 1 (Sequential Composition [26])**.** *Given a sequence of computations $\mathcal{A}_1, \ldots, \mathcal{A}_k$, if each $\mathcal{A}_i$ satisfies (ϵ_i, δ_i)-DP, then their sequential execution on the same dataset satisfies $(\sum_i \epsilon_i, \sum_i \delta_i)$-DP.*

Proposition 2 (Post-processing [26])**.** *Suppose $\mathcal{A}$ satisfies (ϵ, δ)-DP, and let $\mathcal{B}$ be an arbitrary deterministic mapping function. The composed algorithm $\mathcal{B}(\mathcal{A}(\cdot))$ satisfies (ϵ, δ)-DP.*

III. The Proposed Centralized Method

In this section, we propose novel approaches to estimate the number of balanced and unbalanced triangles in signed graphs under centralized DP. Our methods not only protect individual privacy but also maintain excellent query performance. First of all, we introduce techniques to compute the local sensitivity and the local sensitivity at distance t of the triangle counting query function, which is the foundation for the implementation of differential privacy. Then, we propose efficient algorithms for the calculation of smooth upper bounds on local sensitivity, a critical factor in reducing the impact of data perturbations. Among these bounds, smooth sensitivity is the smallest upper bound. However, its calculation becomes impractical for large signed graphs due to substantial computational demands. In such cases, we select a computationally feasible smooth upper bound as an alternative to smooth sensitivity. Thus, we devise a computationally efficient function that calculates the smooth upper bound on local sensitivity of the triangle counts. Finally, we add noise proportional to the smooth upper bound on local sensitivity into the response of the triangle counting query. For small signed graphs, the method based on smooth sensitivity offers superior utility than that based on the smooth upper bound on local sensitivity. In contrast, for large signed graphs, the method based on the smooth upper bound not only ensures computational efficiency but also achieves comparable utility to the smooth-sensitivity-based method.

In the remainder of this paper, we use $[n]$ to represent the set $\{1, 2, \ldots, n\}$ and $[a, b]$ to denote $\{a, a+1, \ldots, b\}$.

Algorithm 1: Edge-Scan for Wedge Counting

Input: signed graph G represented as adjacency vectors $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$
Output: $\{(w^+_{i,j}, w^-_{i,j})\}_{i,j\in[n];i<j}$

```
Initialize w^+_{i,j} ← 0, w^-_{i,j} ← 0 for each node pair (v_i, v_j);
for each node v_i ∈ V do
    for each pair of edges (v_i, v_j), (v_i, v_k) incident to v_i do
        /* suppose j < k */
        if a_{i,j} · a_{i,k} = 1 then
            w^+_{j,k} ← w^+_{j,k} + 1;
        else
            w^-_{j,k} ← w^-_{j,k} + 1;
return {(w^+_{i,j}, w^-_{i,j})}_{i,j∈[n];i<j}
```

A. The Calculation of Local Sensitivity

This section elucidates the approaches employed to compute the local sensitivity and the local sensitivity at distance t of the function $f_\triangle$. This function takes a signed graph G as input and yields the number of balanced triangles T^b and unbalanced triangles T^u.

Local Sensitivity. The local sensitivity of the triangle counting query function $f_\triangle$, denoted as $LS_\triangle(G)$, is defined as follows:

$$LS_\triangle(G) = \max_{\widetilde{G}\in\mathcal{G}:d(G,\widetilde{G})\leq 1} |\tilde{T}^b - T^b| + |\tilde{T}^u - T^u|. \tag{8}$$

To efficiently compute the local sensitivity, we introduce a specialized definition of local sensitivity, denoted as $LS_{i,j}(G)$. This metric evaluates the maximum impact on sensitivity when modifying the relationship between nodes v_i and v_j. Such modifications require the addition of a signed edge (v_i, v_j) in its absence, or the alteration of its sign or deletion of a signed edge if it exists. Let $w^+_{i,j}$ denote the number of positive wedges (two-hop paths) between nodes v_i and v_j where the product of $a_{i,k}$ and $a_{j,k}$ is positive. In contrast, $w^-_{i,j}$ represents the number of negative wedges where this product is negative. The following lemma details the calculation of the local sensitivity of $f_\triangle$.

Lemma 1. *The local sensitivity of $f_\triangle$ is given by $LS_\triangle(G) = \max_{i,j\in[n];i\neq j} LS_{i,j}(G)$, where*

$$LS_{i,j}(G) = \begin{cases} w^+_{i,j} + w^-_{i,j}, & \text{if } a_{i,j} = 0, \\ \max(w^+_{i,j} + w^-_{i,j}, 2|w^+_{i,j} - w^-_{i,j}|), & \text{if } a_{i,j} \neq 0. \end{cases} \tag{9}$$

Proof. We start by analyzing $LS_{i,j}(G)$. For the case $a_{i,j} = 0$, the addition of the signed edge between nodes v_i and v_j changes $f_\triangle(G)$ by $w^+_{i,j} + w^-_{i,j}$, irrespective of its sign. Now consider the case where $a_{i,j} \neq 0$. If the edge (v_i, v_j) is deleted, it can be easily verified that $f_\triangle$ also changes by $w^+_{i,j} + w^-_{i,j}$. If we alter the sign of edge (v_i, v_j), the number of balanced and unbalanced triangles over the edge (v_i, v_j) interchange, which results in a change in $f_\triangle$ by $2|w^+_{i,j} - w^-_{i,j}|$. Finally, $LS_\triangle(G)$ is the maximum of all $LS_{i,j}(G)$. □

Algorithm 1 describes the process of computing the number of wedges between all pairs of nodes. It iterates over each node $v_i \in \mathcal{V}$ and then enumerates all edge pairs connected to v_i to determine the number of positive and negative wedges between specific node pairs. Since the signed graphs are undirected,

it follows that $w^+_{i,j} = w^+_{j,i}$, and the same holds for $w^-_{i,j}$. It is sufficient to compute the number of wedges for the node pairs v_i and v_j whose node indexes satisfy $i < j$. Let Γ represent the number of node pairs where $w^+_{i,j} \neq 0$ or $w^-_{i,j} \neq 0$, expressed as $\Gamma = \sum_{i=0}^{n} \sum_{j=i+1}^{n} \mathbb{I}(w^+_{i,j} \neq 0 \text{ or } w^-_{i,j} \neq 0)$. Here, the symbol $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if the condition inside the parentheses is true and 0 if it is false. The time complexity of Algorithm 1 is $\mathcal{O}(m \cdot d_{max})$. And the computation of $LS_\triangle(G)$ can be accomplished in time $\mathcal{O}(\Gamma)$.

Local Sensitivity at Distance. In the rest of this section, we explain the computation of the local sensitivity at distance t of $f_\triangle$, denoted as $LS_\triangle(G,t)$. For the special scenario where $t = 0$, $LS_\triangle(G,t)$ represents the local sensitivity of $f_\triangle$ at G. When $s \geq 1$, to compute $LS_\triangle(G,t)$ efficiently, we define $LS_{i,j}(G,t)$ as the maximum of $LS_{i,j}(\cdot)$ attained on signed graphs whose distance from G are at most t, expressed as

$$LS_{i,j}(G,t) = \max_{\tilde{G} \in \mathcal{G}: d(G,\tilde{G}) \leq t)} LS_{i,j}(\tilde{G}). \tag{10}$$

Thus, $LS_\triangle(G,t) = \max_{i,j \in [n]; i<j} LS_{i,j}(G,t)$. As stated in Lemma 1, the calculation of local sensitivity for $\tilde{G}$ over the node pair (v_i, v_j) depends on whether $\tilde{a}_{i,j}$ equals 0 or not. However, we can first introduce a signed edge (v_i, v_j) if it is absent. Therefore, regardless of the existence of edge (v_i, v_j), we need to explore modification strategies that can maximize $\tilde{w}^+_{i,j} + \tilde{w}^-_{i,j}$ and $2|\tilde{w}^+_{i,j} - \tilde{w}^-_{i,j}|$ in $\tilde{G}$, respectively. For convenience, we employ $W^1_{i,j}(G,t)$ to represent the maximum of $\tilde{w}^+_{i,j} + \tilde{w}^-_{i,j}$, and $W^2_{i,j}(G,t)$ to denote the maximum of $2|\tilde{w}^+_{i,j} - \tilde{w}^-_{i,j}|$.

To comprehend $W^1_{i,j}(G,t)$ for positive t, we modify t edges in G to obtain a signed graph $\tilde{G}$ where $\tilde{w}^+_{i,j} + \tilde{w}^-_{i,j}$ can reach its maximum. Among the possible modifications, the only one that can increase $w^+_{i,j} + w^-_{i,j}$ is the addition of edges adjacent to nodes v_i or v_j. The sign of the added edge does not impact the overall change. Thus, $\tilde{G}$ should be obtained by the addition of t edges to G. The specific allocation strategy is described in Lines 5-7 of Algorithm 2. Let $b_{i,j}$ denote the number of nodes connected to exactly one of the two nodes v_i and v_j. For these nodes v_k, adding one edge, i.e., the missing edge (v_i, v_k) or (v_j, v_k), is sufficient to increase $w^+_{i,j} + w^-_{i,j}$ by 1. Conversely, for nodes v_k that are not adjacent to both v_i and v_j, the addition of two edges, i.e., (v_i, v_k) and (v_j, v_k), is required to achieve the same increase. Therefore, if the distance $t \leq b_{i,j}$, the increase of $w^+_{i,j} + w^-_{i,j}$ is at most t, as indicated in Line 5. However, a nuanced allocation approach is required when $t > b_{i,j}$. First of all, we can add $b_{i,j}$ edges so that $w^+_{i,j} + w^-_{i,j}$ increases $b_{i,j}$. Then, for the rest of $t - b_{i,j}$ modifications, we can find $\lfloor (t - b_{i,j})/2 \rfloor$ nodes v_k, where $a_{i,k} = a_{j,k} = 0$, and add the edges (v_i, v_k) and (v_j, v_k) to increase $w^+_{i,j} + w^-_{i,j}$ by $\lfloor (t - b_{i,j})/2 \rfloor$, achieving a total increase of $w^+_{i,j} + w^-_{i,j}$ by $\lfloor (t + b_{i,j})/2 \rfloor$, as outlined in Line 6. Note that the upper bound of $w^+_{i,j} + w^-_{i,j}$ is $n - 2$. This constraint ensures that the modifications adhere to the inherent structure of the graph. Next, we introduce the calculation of $W^2_{i,j}(G,t)$. Recall that $W^2_{i,j}(G,t)$ needs to be computed only if $a_{i,j} \neq 0$. However, in the case where $a_{i,j} = 0$, we can introduce an edge (v_i, v_j) beforehand. As a result, the total number of required modifications is reduced to $t - 1$, as stated in Line 8.

Different from $W^1_{i,j}(G,t)$, $W^2_{i,j}(G,t)$ denotes the maximum of $2|\tilde{w}^+_{i,j} - \tilde{w}^-_{i,j}|$ across all signed graphs $\tilde{G}$ where $d(G,\tilde{G}) \leq t$. Thus, our modifications to G should aim to maximize $2|w^+_{i,j} - w^-_{i,j}|$. The only effective modification is to operate on edges adjacent to nodes v_i or v_j. Specifically, the addition or deletion of an edge can result in a maximum increase of 2, whereas the alteration of the sign of an edge yields an increase of up to 4. This paper employs a greedy strategy to calculate $W^2_{i,j}(G,t)$, as outlined in Lines 9-16 of Algorithm 2. The approach prioritizes edge sign alterations until further modifications cease to yield increases since they offer higher potential increases.

Algorithm 2: Local Sensitivity at Distance

Input: signed graph G represented as adjacency vectors $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$, distance $t \geq 1$, $\{(w^+_{i,j}, w^-_{i,j})\}_{i,j \in [n]; i<j}$
Output: local sensitivity at distance t of $f_\triangle$: $LS_\triangle(G,t)$

1 Initialize modification volume $c \leftarrow t$, $LS_\triangle(G,t) \leftarrow 0$;
2 **for** *each node pair* (v_i, v_j)*, where* $i, j \in [n]$, $i < j$ **do**
3 $\quad t \leftarrow c$;
4 $\quad b_{i,j} \leftarrow d_i + d_j - 2(w^+_{i,j} + w^-_{i,j} + |a_{i,j}|)$;
5 $\quad$ **if** $b_{i,j} \geq t$ **then** $W^1_{i,j}(G,t) \leftarrow w^+_{i,j} + w^-_{i,j} + t$;
6 $\quad$ **else** $W^1_{i,j}(G,t) \leftarrow w^+_{i,j} + w^-_{i,j} + \lfloor (t + b_{i,j})/2 \rfloor$;
7 $\quad W^1_{i,j}(G,t) \leftarrow \min(W^1_{i,j}(G,t), n-2)$;
8 $\quad$ **if** $a_{i,j} = 0$ **then** $t \leftarrow t - 1$;
9 $\quad$ **if** $\min(w^+_{i,j}, w^-_{i,j}) \geq t$ **then**
10 $\quad\quad W^2_{i,j}(G,t) \leftarrow 2(|w^+_{i,j} - w^-_{i,j}| + 2t)$;
11 $\quad$ **else**
12 $\quad\quad W^2_{i,j}(G,t) \leftarrow 2(|w^+_{i,j} - w^-_{i,j}| + 2\min(w^+_{i,j}, w^-_{i,j}))$;
13 $\quad\quad t \leftarrow t - \min(w^+_{i,j}, w^-_{i,j})$;
14 $\quad\quad$ **if** $b_{i,j} \geq t$ **then** $W^2_{i,j}(G,t) \leftarrow W^2_{i,j}(G,t) + 2t$;
15 $\quad\quad$ **else** $W^2_{i,j}(G,t) \leftarrow W^2_{i,j}(G,t) + 2\lfloor (t + b_{i,j})/2 \rfloor$;
16 $\quad W^2_{i,j}(G,t) \leftarrow \min(W^2_{i,j}(G,t), 2n-4)$;
17 $\quad LS_{i,j}(G,t) \leftarrow \max(W^1_{i,j}(G,t), W^2_{i,j}(G,t))$;
18 $\quad LS_\triangle(G,t) \leftarrow \max(LS_\triangle(G,t), LS_{i,j}(G,t))$;
19 **return** $LS_\triangle(G,t)$

In the context of different nodes v_i and v_j, it is essential to account for two scenarios due to the presence of an absolute value: either $w^+_{i,j} \geq w^-_{i,j}$ or $w^+_{i,j} < w^-_{i,j}$. However, the analytical approach for both scenarios remains consistent. For simplicity, we assume $w^+_{i,j} \geq w^-_{i,j}$ so that the objective function reduces to $2(w^+_{i,j} - w^-_{i,j})$. If $t \leq w^-_{i,j}$, it is possible to increase the objective by $4t$, as described in Line 10. This increase can be achieved by reversing the sign of t edges, where each edge belongs to a different negative wedge, and each adjustment contributes an increment of 4 to $2(w^+_{i,j} - w^-_{i,j})$. Consider the scenario where $t > w^-_{i,j}$. We can first reverse the sign of $w^-_{i,j}$ edges to increase $2(w^+_{i,j} - w^-_{i,j})$ by $4w^-_{i,j}$. Then the only feasible modification to further increase $2(w^+_{i,j} - w^-_{i,j})$ is to add edges connected to either nodes v_i or v_j. Hence, for the rest of $t - w^-_{i,j}$ modifications, the modification procedure is similar to that employed in the calculation of $W^1_{i,j}(G,t)$, and the sole difference is that the increment becomes twice as large, as outlined in Lines 14-15. Different from $w^+_{i,j} + w^-_{i,j}$, the upper bound of $2(w^+_{i,j} - w^-_{i,j})$ is $2n - 4$. Upon the completion of calculations for $W^1_{i,j}(G,t)$ and $W^2_{i,j}(G,t)$, it becomes feasible to determine the local sensitivity at the distance t over the edge (v_i, v_j), as stated in Line 17. This procedure is then iteratively applied

to all node pairs (v_i, v_j) to compute $LS_\triangle(G,t)$. Algorithm 2 encompasses all the steps involved in the calculation of the local sensitivity at distance t of $f_\triangle$. And the time complexity of this algorithm is $\mathcal{O}(n^2)$.

B. Smooth Upper Bound on Local Sensitivity

The previous subsection details the calculation of the local sensitivity and the local sensitivity at distance t of the triangle count query function. In what follows, we introduce methods to calculate the smooth upper bounds on local sensitivity based on the above computation.

Smooth Sensitivity. Smooth sensitivity is the smallest upper bound that satisfies the criterion for a smooth upper bound on local sensitivity. Thus, the mechanism that utilizes smooth sensitivity to calibrate noise can add less noise and provide better utility. For $\beta > 0$, the smooth sensitivity of $f_\triangle$, denoted as $S^*_{\triangle,\beta}(G)$, is expressed as

$$S^*_{\triangle,\beta}(G) = \max_{t\in[0,2n-3]} e^{-\beta t} LS_\triangle(G,t). \tag{11}$$

Given that $LS_\triangle(G,t)$ remains constant for all $t \geq 2n-3$, it is sufficient to consider $t \leq 2n-3$. To compute the smooth sensitivity, one can calculate $e^{-\beta t}LS_\triangle(G,t)$ for each relevant t and select the maximal value as the smooth sensitivity of $f_\triangle$. However, the time complexity of $LS_\triangle(G,t)$, denoted by $\mathcal{O}(n^2)$, coupled with the need to evaluate each possible value of t, raises the overall time complexity to $\mathcal{O}(n^3)$. This complexity makes the method impractical for most applications.

In the calculation of smooth sensitivity, the primary source of time complexity is the computation of local sensitivity at distance, which involves extensive redundant and unproductive calculations. To address this problem, we propose an efficient method to compute the smooth sensitivity of $f_\triangle$. This method eliminates duplicate and irrelevant data based on our established criteria and retains only those elements that are critical to the smooth sensitivity computation. Then, we can determine the local sensitivity at every distance t sequentially, thereby substantially reducing computational overhead.

First of all, we prepare the requisite data for the computation of smooth sensitivity, described in Lines 3-8 of Algorithm 3. Specifically, we compute $w^+_{i,j}$, $w^-_{i,j}$ and $b_{i,j}$ for each node pair (v_i, v_j) where $i < j$. The computations for $w^+_{i,j}$ and $w^-_{i,j}$ follow Algorithm 1, and $b_{i,j}$ is determined by the formula $b_{i,j} = d_i + d_j - 2(w^+_{i,j} + w^-_{i,j} + |a_{i,j}|)$. For node pairs with $w^+_{i,j} \neq 0$ or $w^-_{i,j} \neq 0$, we construct a list $\mathcal{Q}$ consisting of triplets $(w^+_{i,j}, w^-_{i,j}, b_{i,j})$. For node pairs where both $w^+_{i,j}$ and $w^-_{i,j}$ equal zero, we select the maximum of $b_{i,j}$ and add the triplet $(0,0,b_{i,j})$ to the list $\mathcal{Q}$.

To compute local sensitivity at distance efficiently, we propose a set of rules to eliminate duplicate and irrelevant triplets in $\mathcal{Q}$. Recall that the computation of the local sensitivity at distance t mainly considers two cases: the maximum of $\tilde{w}^+_{i,j} + \tilde{w}^-_{i,j}$ and the maximum of $2|\tilde{w}^+_{i,j} - \tilde{w}^-_{i,j}|$. Let $W^1(G,t)$ be the maximal value of $\tilde{w}^+_{i,j} + \tilde{w}^-_{i,j}$ across all $\tilde{G}$ which satisfy $d(G,\tilde{G}) \leq t$, and we define $W^2(G,t)$ as the maximal value of $2|\tilde{w}^+_{i,j} - \tilde{w}^-_{i,j}|$ within the same subset of $\tilde{G}$. Thus, it follows that $W^1(G,t) = \max_{i,j\in[n];i<j} W^1_{i,j}(G,t)$, $W^2(G,t) = \max_{i,j\in[n];i<j} W^2_{i,j}(G,t)$, and $LS_\triangle(G,t) = \max(W^1(G,t), W^2(G,t))$. Note that we do not consider whether $a_{i,j}$ is equal to zero when computing $LS_\triangle(G,t)$. In the cases where $a_{i,j} = 0$, the required number of modifications decreases from t to $t-1$. Nonetheless, our algorithm still executes t modifications to G, which does not compromise individual privacy, since $LS_\triangle(G,t)$ is a monotonically non-decreasing function of t. According to Algorithm 2, $W^1_{i,j}(G,t)$ can be formulated as

$$W^1_{i,j}(G,t) = \begin{cases} w^s_{i,j} + t, & \text{if } t \leq b_{i,j}, \\ \min(w^s_{i,j} + \lfloor (t+b_{i,j})/2 \rfloor, n-2), & \text{if } t > b_{i,j}, \end{cases} \tag{12}$$

where $w^s_{i,j} = w^+_{i,j} + w^-_{i,j}$, and $W^2_{i,j}(G,t)$ can be expressed as

$$W^2_{i,j}(G,t) = \begin{cases} 2w^a_{i,j} + 4t, \text{if } t \leq w^m_{i,j}, \\ 2w^a_{i,j} + 2w^m_{i,j} + 2t, \text{if } w^m_{i,j} < t \leq w^m_{i,j} + b_{i,j}, \\ \min(2(w^a_{i,j} + w^m_{i,j} + \lfloor \frac{t + w^m_{i,j} + b_{i,j}}{2} \rfloor), 2n-4), \text{otherwise}, \end{cases} \tag{13}$$

where $w^a_{i,j} = |w^+_{i,j} - w^-_{i,j}|$ and $w^m_{i,j} = \min(w^+_{i,j}, w^-_{i,j})$. To streamline our analysis, we transform the triplet $(w^+_{i,j}, w^-_{i,j}, b_{i,j})$ into the pair $(w^s_{i,j}, b_{i,j})$ for the calculation of $W^1(G,t)$. Similarly, we convert it into a triplet $(w^a_{i,j}, w^m_{i,j}, b_{i,j})$ for the computation of $W^2(G,t)$. This conversion facilitates a more intuitive presentation of the methods involved in these calculations without additional overhead.

To compute $W^1(G,t)$, we first sort the $\Gamma+1$ pairs $(w^s_{i,j}, b_{i,j})$ in a non-increasing order by $w^s_{i,j}$. In cases where multiple pairs have the same value of $w^s_{i,j}$, only the pair with the maximum of $b_{i,j}$ is kept, denoted as $\mathcal{Q}_1$. We then process the list in the order of descending $w^s_{i,j}$. One pair (w^s_i, b_i) is maintained only if it, relative to the previously retained pair (w^s_{i-1}, b_{i-1}), fulfills the condition $2(w^s_{i-1} - w^s_i) + b_{i-1} < b_i$. The set of pairs that meet this condition is represented as $\mathcal{L}_1 = \{(w^s_1, b_1), \ldots, (w^s_{k_1}, b_{k_1})\}$. Equally pivotal in our analysis is the determination of breakpoints, utilized to select the appropriate data from the filtered list to compute $W^1(G,t)$ or $W^2(G,t)$. In what follows, we set the initial breakpoint as 0 and the final breakpoint as $2n-3$. In the computation of $W^1(G,t)$, the other breakpoints are set to $t^1_i = 2(w^s_i - w^s_{i+1}) + b_i$ for $i \in [k_1 - 1]$. The entire filtering process in computing $W^1(G,t)$ is shown in Lines 9-13. Then $W^1(G,t) = \min(w^s_i + \lfloor (t + \min(b_i, t))/2 \rfloor, n-2)$ if $t \in [t^1_{i-1}, t^1_i)$.

The calculation of $W^2(G,t)$ is more complicated, primarily since the independent variable t must be compared with two distinct quantities: $w^m_{i,j}$ and $w^m_{i,j} + b_{i,j}$. To address this problem, we maintain three different lists. First of all, we sort the $\Gamma+1$ triplets $(w^a_{i,j}, w^m_{i,j}, b_{i,j})$ in a non-increasing order based on two criteria: $w^a_{i,j}$ and $w^a_{i,j} + w^m_{i,j}$. For the first criterion, if several triplets exhibit identical $w^a_{i,j}$ values, we retain only the triplet with the largest $w^m_{i,j}$. This list is represented as $\mathcal{Q}_2$. In addition, we also utilize the first criterion to construct $\mathcal{Q}_3$ but retain the triplet with the largest $3w^m_{i,j} + b_{i,j}$ in case of same $w^a_{i,j}$ values. Similarly, for the second criterion, if multiple triplets have the same value of $w^a_{i,j} + w^m_{i,j}$, we keep the triplet with the largest $w^m_{i,j} + b_{i,j}$, and this list is denoted as $\mathcal{Q}_4$.

Then, we respectively process the sorted lists $\mathcal{Q}_2$, $\mathcal{Q}_3$, and $\mathcal{Q}_4$. The first treatment involves a traversal of the list $\mathcal{Q}_2$ in the order of decreasing $w^a_{i,j}$. Each triplet (w^a_i, w^m_i, b_i) is evaluated for retention based on the condition $w^a_{i-1} - w^a_i + w^m_{i-1} < w^m_i$, where $(w^a_{i-1}, w^m_{i-1}, b_{i-1})$ is the previous retained triplet. The collection of triplets that satisfy this criterion is represented as $\mathcal{L}_2 = \{(w^a_1, w^m_1, b_1), \ldots, (w^a_{k_2}, w^m_{k_2}, b_{k_2})\}$. The breakpoints are

Algorithm 3: The Calculation of Smooth Sensitivity

Input: signed graph G represented as adjacency vectors $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$, the parameter β
Output: smooth sensitivity $S^*_{\triangle,\beta}(G)$

1 $S^*_{\triangle,\beta}(G) \leftarrow 0$; $b_{max} \leftarrow 0$; $\mathcal{Q} \leftarrow \emptyset$;
2 $\mathcal{L}_c \leftarrow \emptyset, t^c_0 \leftarrow 0$ for $c \in [4]$;
3 $\{(w^+_{i,j}, w^-_{i,j})\}_{i,j\in[n];i<j} \leftarrow$ Execute Algorithm 1;
4 **for** *each node pair* (v_i, v_j)*, where* $i, j \in [n], i < j$ **do**
5 $\quad b_{i,j} \leftarrow d_i + d_j - 2(w^+_{i,j} + w^-_{i,j} + |a_{i,j}|)$;
6 $\quad$ **if** $w^+_{i,j} \neq 0$ *or* $w^-_{i,j} \neq 0$ **then** $\mathcal{Q}.push((w^+_{i,j}, w^-_{i,j}, b_{i,j}))$;
7 $\quad$ **else** $b_{max} \leftarrow \max(b_{max}, b_{i,j})$;
8 $\mathcal{Q}.push((0, 0, b_{max}))$;
// $w^s_{i,j} := w^+_{i,j} + w^-_{i,j}, w^a_{i,j} := |w^+_{i,j} - w^-_{i,j}|, w^m_{i,j} := \min(w^+_{i,j}, w^-_{i,j})$
9 Sort $\mathcal{Q}$ in non-increasing order by $w^s_{i,j}$. For ties, keep the pair with the largest $b_{i,j}$, denoted as $\mathcal{Q}_1$;
10 **for** *each pair* $(w^s_i, b_i) \in \mathcal{Q}_1$ **do**
11 $\quad$ **if** $2(w^s_{i-1} - w^s_i) + b_{i-1} < b_i$ **then** $\mathcal{L}_1.push((w^s_i, b_i))$;
12 $k_1 \leftarrow \mathcal{L}_1.length()$;
13 $t^1_i \leftarrow 2(w^s_i - w^s_{i+1}) + b_i$ for $i \in [k_1 - 1]$;
14 Sort $\mathcal{Q}$ in non-increasing order by $w^a_{i,j}$. For ties, keep the triplet with the largest $w^m_{i,j}$, denoted as $\mathcal{Q}_2$;
15 **for** *each triplet* $(w^a_i, w^m_i, b_i) \in \mathcal{Q}_2$ **do**
16 $\quad$ **if** $w^a_{i-1} - w^a_i + w^m_{i-1} < w^m_i$ **then** $\mathcal{L}_2.push((w^a_i, w^m_i, b_i))$;
17 $k_2 \leftarrow \mathcal{L}_2.length()$;
18 $t^2_i \leftarrow w^a_i - w^a_{i+1} + w^m_i$ for $i \in [k_2 - 1]$;
19 Sort $\mathcal{Q}$ in non-increasing order by $w^a_{i,j}$. For ties, keep the triplet with the largest $3w^m_{i,j} + b_{i,j}$, denoted as $\mathcal{Q}_3$;
20 **for** *each triplet* $(w^a_i, w^m_i, b_i) \in \mathcal{Q}_3$ **do**
21 $\quad$ **if** $2(w^a_{i-1} - w^a_i) + 3w^m_{i-1} + b_{i-1} < 3w^m_i$ **then**
22 $\quad\quad \mathcal{L}_3.push((w^a_i, w^m_i, b_i))$;
23 $k_3 \leftarrow \mathcal{L}_3.length()$;
24 $t^3_i = (2(w^a_i - w^a_{i+1}) + 3w^m_i + b_i)/3$ for $i \in [k_3 - 1]$;
25 Sort $\mathcal{Q}$ in non-increasing order by $w^a_{i,j} + w^m_{i,j}$. For ties, keep the triplet with the largest $w^m_{i,j} + b_{i,j}$, denoted as $\mathcal{Q}_4$;
26 **for** *each triplet* $(w^a_i, w^m_i, b_i) \in \mathcal{Q}_4$ **do**
27 $\quad$ **if** $2(w^a_{i-1} - w^a_i) + 3w^m_{i-1} - 3w^m_i + b_{i-1} < b_i$ **then**
28 $\quad\quad \mathcal{L}_4.push((w^a_i, w^m_i, b_i))$;
29 $k_4 \leftarrow \mathcal{L}_4.length()$;
30 $t^4_i = 2(w^a_i - w^a_{i+1}) + 3w^m_i - 2w^m_{i+1} + b_i$ for $i \in [k_4 - 1]$;
31 **for** $t = 0$ *to* $2n - 3$ **do**
32 $\quad$ Select (w^s_i, b_i) from $\mathcal{L}_1$ where $t \in [t^1_{i-1}, t^1_i)$;
33 $\quad W^1(G, t) = \min(w^s_i + \lfloor (t + \min(b_i, t))/2 \rfloor, n - 2)$;
34 $\quad$ Select $(w^{a,c}_i, w^{m,c}_i, b^c_i)$ from $\mathcal{L}_c$ where $t \in [t^c_{i-1}, t^c_i)$;
35 $\quad W^2(G, t) = \min(\max_{c\in\{2,3,4\}} 2w^{a,c}_i + 2\min(w^{m,c}_i, t) + 2\lfloor (t + \min(t, w^{m,c}_i + b^c_i))/2 \rfloor, 2n - 4)$;
36 $\quad LS_\triangle(G, t) = \max(W^1(G, t), W^2(G, t))$;
37 $\quad S^*_{\triangle,\beta}(G) = \max(S^*_{\triangle,\beta}(G), e^{-\beta t} LS_\triangle(G, t))$;
38 **return** $S^*_{\triangle,\beta}(G)$

calculated via the formula $t^2_i = w^a_i - w^a_{i+1} + w^m_i$ for $i \in [k_2 - 1]$.

Go through the list $\mathcal{Q}_3$ in the order of decreasing $w^a_{i,j}$. Each triplet (w^a_i, w^m_i, b_i) is retained only if it satisfies $2(w^a_{i-1} - w^a_i) + 3w^m_{i-1} + b_{i-1} < 3w^m_i$ when compared with the previously preserved triplet $(w^a_{i-1}, w^m_{i-1}, b_{i-1})$. The triplets that meet this condition form the list $\mathcal{L}_3 = \{(w^a_1, w^m_1, b_1), \ldots, (w^a_{k_3}, w^m_{k_3}, b_{k_3})\}$. To determine the breakpoints for $\mathcal{L}_3$, we set $t^3_i = (2(w^a_i - w^a_{i+1}) + 3w^m_i + b_i)/3$ for $i \in [k_3 - 1]$.

For the sorted triplet list $\mathcal{Q}_4$, we traverse this list in the order of decreasing $w^a_{i,j} + w^m_{i,j}$. For each (w^a_i, w^m_i, b_i), keep this triplet only if it satisfies $2(w^a_{i-1} - w^a_i) + 3w^m_{i-1} - 3w^m_i + b_{i-1} < b_i$. The set of triplets that fulfill this criterion constitutes $\mathcal{L}_4 = \{(w^a_1, w^m_1, b_1), \ldots, (w^a_{k_4}, w^m_{k_4}, b_{k_4})\}$. Let $t^4_i = 2(w^a_i - w^a_{i+1}) + 3w^m_i - 2w^m_{i+1} + b_i$ for $i \in [k_4 - 1]$. The comprehensive filtering procedure in the computation of $W^2(G, t)$ is delineated in Lines 14-30 of Algorithm 3.

To complete the calculation of $W^2(G, t)$, it is imperative to choose the relevant triplets from the three filtered lists. Given a specific distance t, similar to computing $W^1(G, t)$, we select three triplets from $\mathcal{L}_2$, $\mathcal{L}_3$, and $\mathcal{L}_4$, respectively, based on the comparison of t with their respective breakpoint sequences, denoted as $(w^{a,c}_i, w^{m,c}_i, b^c_i)$, where $c \in \{2, 3, 4\}$. Then it follows that $W^2(G, t) = \min(\max_{c\in\{2,3,4\}} 2w^{a,c}_i + 2\min(w^{m,c}_i, t) + 2\lfloor (t + \min(t, w^{m,c}_i + b^c_i))/2 \rfloor, 2n - 4)$.

In order to compute the smooth sensitivity of the function $f_\triangle$, we sequentially compute $e^{-\beta t} LS_\triangle(G, t)$ for each distance t, where $LS_\triangle(G, t) = \max(W^1(G, t), W^2(G, t))$, and select the maximal value as the smooth sensitivity. This calculation is detailed in the pseudo-code presented in Algorithm 3.

Furthermore, we have established a theorem that delineates an explicit condition under which the maximum value between $W^1(G)$ and $W^2(G)$ is equal to the smooth sensitivity, where $W^1(G) = \max_{i,j\in[n];i<j} w^s_{i,j}$ and $W^2(G) = \max_{i,j\in[n];i<j} 2w^a_{i,j}$. Under this condition, our randomized algorithm $\mathcal{A}$ introduces noise that is proportional to $\max(W^1(G), W^2(G))$, without the necessity to calculate the local sensitivity at distance t.

Theorem 2. *Given a signed graph G, if $W^1(G) \geq \frac{1}{\beta}$ and $W^2(G) \geq \frac{4}{\beta}$, then $S^*_{\triangle,\beta}(G) = \max(W^1(G), W^2(G))$.*

Proof. Let $U^1(G, t) = W^1(G) + t$ and $U^2(G, t) = W^2(G) + 4t$. Then $U^1(G, t) \geq W^1(G, t)$ and $U^2(G, t) \geq W^2(G, t)$. Incorporating these inequalities into the smooth sensitivity calculation, we derive that

$$\begin{aligned} S^*_{\triangle,\beta}(G) &= \max_{t\in[0,2n-3]} e^{-\beta t} LS_\triangle(G, t) \\ &= \max_{t\in[0,2n-3]} \{e^{-\beta t} W^1(G, t), e^{-\beta t} W^2(G, t)\} \\ &\leq \max_{t\in[0,2n-3]} \{e^{-\beta t} U^1(G, t), e^{-\beta t} U^2(G, t)\} \\ &= \max_{t\in[0,2n-3]} \{e^{-\beta t}(W^1(G) + t), e^{-\beta t}(W^2(G) + 4t)\}. \end{aligned}$$

Consider the functions $h(t) = e^{-\beta t} \cdot (W^1(G) + t)$ and $g(t) = e^{-\beta t} \cdot (W^2(G) + 4t)$. The objective is to find their maximum values over the continuous interval $[0, 2n-3]$. For the function $h(t)$, its derivative $h'(t) = e^{-\beta t}(1 - \beta(W^1(G) + t))$ indicates that $h(t)$ is strictly concave, with a unique maximum at $t = \frac{1}{\beta} - W^1(G)$. Notably, when $W^1(G) \geq \frac{1}{\beta}$, $h(t)$ exhibits a monotonically decreasing trend over the interval $[0, \infty)$. Therefore, in the case where $W^1(G) \geq \frac{1}{\beta}$, the maximum of $h(t)$ is attained at $t = 0$ and equals $W^1(G)$. A similar analysis of $g(t)$ indicates that, under the condition $W^2(G) \geq \frac{4}{\beta}$, its maximum also occurs at $t = 0$ and is equal to $W^2(G)$. As a result, if $W^1(G) \geq \frac{1}{\beta}$ and $W^2(G) \geq \frac{4}{\beta}$, it follows that $S^*_{\triangle,\beta}(G) = \max(W^1(G), W^2(G))$. □

The time complexity of smooth sensitivity computation is analyzed methodically over several phases. In the data preparation phases, the time complexity is described as $\mathcal{O}(m \cdot d_{max} + n^2)$. This complexity comes from the $\mathcal{O}(m \cdot d_{max})$ time required to

compute $w^+_{i,j}$ and $w^-_{i,j}$, and the time required to calculate $b_{i,j}$, quantified as $\mathcal{O}(n^2)$. In the phase dedicated to the elimination of duplicate and irrelevant data, the complexity remains at $\mathcal{O}(\Gamma)$, since both the creation of sorted lists via Bucket Sort and the generation of filtered lists are achievable within $\mathcal{O}(\Gamma)$. The computation of smooth sensitivity in the final phase requires $\mathcal{O}(n)$ time. Thus, the overall time complexity of our proposed method is established as $\mathcal{O}(m \cdot d_{max} + n^2)$.

It is important to note that the time complexity analysis of the smooth sensitivity computation is based on the worst-case scenario. As demonstrated by our experimental results, due to the sparsity of real-world signed networks, our approach based on smooth sensitivity can also handle signed networks with 1 million nodes and 30 million edges.

Smooth Upper Bound on $LS_\triangle$. For large signed graphs, computing the smooth sensitivity of the triangle count query is non-trivial due to the excessive time complexity. To overcome this problem, we relax the requirements for the computation of smooth sensitivity and build a computationally efficient function that computes a smooth upper bound on local sensitivity. Given an arbitrary database x and a query function f, LS_f may have multiple smooth bounds, and the following proposition provides a method to determine the smooth upper bounds on local sensitivity.

Proposition 3. *[11] Define $S_{f,\beta}(x) = \max_{t\in[0,n]} e^{-\beta t} U(x,t)$, where $U(x,t)$ satisfies*

1) $LS_f(x) \le U(x,0)$ for all x.
2) $U(x,t) \le U(\tilde{x}, t+1)$ for all $x, \tilde{x}$ such that $d(x,\tilde{x}) = 1$.

Then $S_{f,\beta}(x)$ is a β-smooth upper bound on local sensitivity.

Inspired by Theorem 2, we develop a function that not only meets the criteria for a smooth upper bound on $LS_\triangle$, but also enhances computational efficiency.

Theorem 3. *Define $U(G,t) = \max(U^1(G,t), U^2(G,t))$, where $U^1(G,t) = W^1(G) + t$ and $U^2(G,t) = W^2(G) + 4t$. Then the function $S_{\triangle,\beta}(G) = \max_{t\in[0,2n-3]} e^{-\beta t} U(G,t)$ is a β-smooth upper bound on local sensitivity of $f_\triangle$.*

Proof. Initially, we prove that $U(G,t)$ fulfills the first condition of Proposition 3. Consider the scenario where $t = 0$. It follows that $U^1(G,0) = W^1(G)$ and $U^2(G,0) = W^2(G)$. Recall that $W^1(G) = \max_{i,j\in[n];i<j}(w^+_{i,j} + w^-_{i,j})$ and $W^2(G) = \max_{i,j\in[n];i<j} 2|w^+_{i,j} - w^-_{i,j}|$, it is easy to verify that $LS_\triangle(G) \le \max(W^1(G), W^2(G))$. Therefore, we can establish that $LS_\triangle(G) \le U(G,0)$.

Next, we demonstrate that $U(G,t)$ satisfies the second condition. It is notable that a single modification to G results in, at most, 1 and 4 changes in $W^1(G)$ and $W^2(G)$, respectively. Thus, for all G and $\tilde{G}$ such that $d(G,\tilde{G}) = 1$, one can deduce that $U^1(G,t) \le U^1(\tilde{G}, t+1)$ and $U^2(G,t) \le U^2(\tilde{G}, t+1)$. This derivation concludes that $U(G,t) \le U(\tilde{G}, t+1)$. □

In fact, it is feasible to compute $S_{\triangle,\beta}(G)$ efficiently. And the function $S_{\triangle,\beta}(G)$ can also be expressed as

$$S_{\triangle,\beta}(G) = \max_{t\in[0,2n-3]} \{e^{-\beta t}(W^1(G)+t), e^{-\beta t}(W^2(G)+4t)\}. \quad (14)$$

Algorithm 4: Smooth Upper Bound on Local Sensitivity

Input: signed graph G represented as adjacency vectors $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$, the parameter β
Output: smooth upper bound on local sensitivity $S_{\triangle,\beta}(G)$

1 $W^1(G) \leftarrow 0$; $W^2(G) \leftarrow 0$;
2 $\{(w^+_{i,j}, w^-_{i,j})\}_{i,j\in[n];i<j} \leftarrow$ Execute Algorithm 1;
3 **for** *each pair* $(w^+_{i,j}, w^-_{i,j})$, *where* $w^+_{i,j} \neq 0$ *or* $w^-_{i,j} \neq 0$ **do**
4 $W^1(G) \leftarrow \max(W^1(G), w^+_{i,j} + w^-_{i,j})$;
5 $W^2(G) \leftarrow \max(W^2(G), 2|w^+_{i,j} - w^-_{i,j}|)$;
6 $t_1 = \frac{1}{\beta} - W^1(G)$; $t_2 = \frac{1}{\beta} - W^2(G)/4$;
7 **if** $t_1 \le 0$ **then** $S_{\triangle,\beta}(G) \leftarrow W^1(G)$;
8 **else**
9 $S_{\triangle,\beta}(G) \leftarrow e^{-\beta\lceil t_1\rceil}(W^1(G) + \lceil t_1\rceil)$;
10 $S_{\triangle,\beta}(G) \leftarrow \max(S_{\triangle,\beta}(G), e^{-\beta\lfloor t_1\rfloor}(W^1(G) + \lfloor t_1\rfloor))$;
11 **if** $t_2 \le 0$ **then** $S_{\triangle,\beta}(G) \leftarrow \max(S_{\triangle,\beta}(G), W^2(G))$;
12 **else**
13 $S_{\triangle,\beta}(G) \leftarrow \max(S_{\triangle,\beta}(G), e^{-\beta\lceil t_2\rceil}(W^2(G) + 4\lceil t_2\rceil))$;
14 $S_{\triangle,\beta}(G) \leftarrow \max(S_{\triangle,\beta}(G), e^{-\beta\lfloor t_2\rfloor}(W^2(G) + 4\lfloor t_2\rfloor))$;
15 **return** $S_{\triangle,\beta}(G)$

It can be straightforwardly deduced that the peak values of the functions $h(t) = e^{-\beta t}(W^1(G)+t)$ and $g(t) = e^{-\beta t}(W^2(G)+4t)$ occur at $t = \frac{1}{\beta} - W^1(G)$ and $t = \frac{1}{\beta} - W^2(G)/4$, respectively. Given the discrete nature and bounded domain of t, the smooth sensitivity $S_{\triangle,\beta}(G)$ can be computed in constant time based on the precomputed values of $W^1(G)$ and $W^2(G)$, as outlined in Lines 6-14 of Algorithm 4. The values of $W^1(G)$ and $W^2(G)$ can be computed in time $\mathcal{O}(m \cdot d_{max})$, as described in Lines 2-5. Algorithm 4 describes the process of the computation of the smooth upper bound on local sensitivity of $f_\triangle$, and the worst-case time complexity of Algorithm 4 is $\mathcal{O}(m \cdot d_{max})$. Empirical evidence from our experiments demonstrates that our method based on the smooth upper bound on local sensitivity is more efficient than the smooth-sensitivity-based method, especially for large signed graphs.

Privacy Guarantee. The algorithms used to calculate smooth upper bounds on $LS_\triangle$ can be employed in conjunction with Theorem 1 to obtain efficient differentially private algorithms for the estimation of balanced and unbalanced triangle counts.

Theorem 4. *Given $\epsilon > 0$ and $\delta > 0$, let $\beta = \frac{\epsilon}{8+4\ln(2/\delta)}$. The randomized algorithm $\mathcal{A}(G) = f_\triangle(G) + (2S_{\triangle,\beta}(G)/\epsilon) \cdot Z$ satisfies (ϵ,δ)-signed edge centralized differential privacy where Z is sampled from 2-dimensional Laplace distribution.*

IV. Triangle Counting in Local Model

Local differential privacy has been extensively accepted and adopted in academic research and industry applications due to its robust framework for privacy preservation. Unlike centralized DP, LDP provides enhanced privacy protection since data obfuscation is performed at the node level rather than by a data curator. In this section, we explore the techniques employed to compute the number of balanced and unbalanced triangles in a signed graph under local DP. Inspired by the work of Imola et al. [20], we propose a two-phase framework tailored for balanced and unbalanced triangle counting. The first phase uses Generalized Randomized Response mechanism to perturb

the adjacency vectors. Then, in the second phase, we propose an innovative response mechanism. This mechanism injects noise into the response of each node about the triangle count query, which is calibrated to a smooth upper bound on local sensitivity. This two-phase approach achieves a better trade-off between data utility and privacy preservation.

The Two-Phase Framework. In the local model, each node v_i independently maintains its own data, denoted as an adjacency vector $\boldsymbol{a}_i$. This vector contains information about the connections of node v_i and the signs of these connections. However, when the query function is $f_\triangle$, nodes cannot locally calculate and submit the obfuscated counts of balanced and unbalanced triangles due to their limited insight into the complete graph structure. Specifically, node v_i is inherently unable to perceive any triangle (v_i, v_j, v_k) because it lacks information about the existence of the edge (v_j, v_k) and its sign in the signed graph.

To address this issue, we propose a two-phase framework to privately response to the query function $f_\triangle$. In the first phase, each node $v_i \in \mathcal{V}$ independently applies the GRR mechanism to perturb the elements $a_{i,1}, \ldots, a_{i,i-1}$ in its adjacency vector. These elements represent the connections between node v_i and nodes with smaller IDs. Given the allocated privacy budget ϵ_1 for this phase, the perturbation scheme of the GRR mechanism is as follows:

$$\Pr[\hat{a}_{i,j}|a_{i,j}] = \begin{cases} \frac{e^{\epsilon_1}}{e^{\epsilon_1}+2}, & \text{if } \hat{a}_{i,j} = a_{i,j}, \\ \frac{1}{e^{\epsilon_1}+2}, & \text{if } \hat{a}_{i,j} \neq a_{i,j}. \end{cases} \tag{15}$$

The data curator then collects the distorted data from all nodes and constructs a noisy signed graph $\hat{G}$, under the premise of an undirected structure. In fact, the data curator can compute the number of balanced and unbalanced triangles directly from $\hat{G}$. However, this estimation method infuses substantial noise into the statistical query results since each edge and its associated sign are modified with a certain probability. This indicates that the state of any triangle in $\hat{G}$ is influenced by three independent random variables. As a matter of fact, for any triangle which involves a certain node, only one edge and its associated sign are unknown to that node. If the data curator publishes the noisy signed graph $\hat{G}$, nodes can obtain information about the edges they are unaware of from $\hat{G}$. Therefore, we can introduce an additional round of interaction between nodes and the data curator to reduce noise injection.

In the second phase, each node v_i computes the count of noisy triangles formed by (v_i, v_j, v_k) where the information of edge (v_j, v_k) can be obtained from $\hat{G}$. Here, a noisy triangle is classified as balanced if the product $a_{i,j} \cdot a_{i,k} \cdot \hat{a}_{j,k}$ equals 1, and as unbalanced if it is -1. This approach ensures that only one edge in each noisy triangle is obfuscated, which enhances the overall utility, albeit at the cost of increased communication overhead per node. However, the direct release of these noisy balanced and unbalanced triangle counts leads to two primary issues. The first issue is the introduction of statistical bias into the estimates of these triangle counts. This bias arises from the noise added during the adjacency vector obfuscation process, which could skew the final estimates. Secondly, and perhaps more importantly, the direct release of such data potentially jeopardizes the privacy of the edges and their respective signs in G. This concern stems from the fact that each node still uses its own real data when calculating the number of balanced and unbalanced triangles.

To address the first problem, we utilize an empirical estimation technique [20], [21], [27] to derive an unbiased estimate of $f_\triangle(G)$ from these noisy counts. More specifically, let T_i^b and T_i^u respectively represent the number of noisy balanced and unbalanced triangles computed by node v_i, where the IDs of the three nodes in each triangle satisfy $i > j > k$. In addition, node v_i is also required to calculate the number of 2-stars centered on itself under the same constraints of node IDs, denoted as s_i. The data curator then estimates the number of balanced and unbalanced triangles in G, denoted as $\bar{T}^b$ and $\bar{T}^u$ respectively, based on the data collected from each node. Let q be $1/(e^{\epsilon_1}+2)$ and these estimates are calculated as follows:

$$\bar{T}^b = \frac{1}{1-3q}\sum_{i=1}^{n}(T_i^b - q \cdot s_i), \tag{16}$$

$$\bar{T}^u = \frac{1}{1-3q}\sum_{i=1}^{n}(T_i^u - q \cdot s_i). \tag{17}$$

Lemma 2. *The computed values $\bar{T}^b$ and $\bar{T}^u$ as delineated in Equations (16) and (17), serve as unbiased estimates of the true counts of balanced and unbalanced triangles in G, respectively, i.e., $\mathbb{E}[\bar{T}^b] = T^b$ and $\mathbb{E}[\bar{T}^u] = T^u$.*

Proof. First, we prove that $\mathbb{E}[\bar{T}^b] = T^b$. For each node v_i, T_i^b denotes the number of balanced triangles (v_i, v_j, v_k) computed by v_i, where $i > j > k$ and $a_{i,j} \cdot a_{i,k} \cdot \hat{a}_{j,k} = 1$. The random variable in each triangle is $\hat{a}_{j,k}$. Thus, we have

$$\mathbb{E}[T_i^b] = \sum_{\substack{i>j>k \\ a_{i,j}\cdot a_{i,k}=1}} \mathbb{E}[\mathbb{I}(\hat{a}_{j,k}=1)] + \sum_{\substack{i>j>k \\ a_{i,j}\cdot a_{i,k}=-1}} \mathbb{E}[\mathbb{I}(\hat{a}_{j,k}=-1)],$$

where $\mathbb{I}(\cdot)$ represents the indicator function. Under the perturbation schema of GRR mechanism, it follows that $\mathbb{E}[\mathbb{I}(\hat{a}_{j,k}=1)] = q(1+(e^{\epsilon_1}-1)\mathbb{I}(a_{j,k}=1))$, which simplifies to $q \cdot e^{\epsilon_1}$ if $a_{j,k}=1$ and q otherwise. Likewise, $\mathbb{E}[\mathbb{I}(\hat{a}_{j,k}=-1)] = q(1+(e^{\epsilon_1}-1)\mathbb{I}(a_{j,k}=-1))$. Then the expectation of $\bar{T}^b$ can be expressed as

$$\begin{aligned}
\mathbb{E}[\bar{T}^b] &= \mathbb{E}[\frac{1}{1-3q}\sum_{i=1}^{n}(T_i^b - q \cdot s_i)] \\
&= \frac{1}{1-3q}\sum_{i=1}^{n}(\mathbb{E}[T_i^b] - \sum_{\substack{i>j>k \\ a_{i,j}\cdot a_{i,k}=1}} q - \sum_{\substack{i>j>k \\ a_{i,j}\cdot a_{i,k}=-1}} q) \\
&= \sum_{i=1}^{n}(\sum_{\substack{i>j>k \\ a_{i,j}\cdot a_{i,k}=1}} \mathbb{I}(a_{j,k}=1) + \sum_{\substack{i>j>k \\ a_{i,j}\cdot a_{i,k}=-1}} \mathbb{I}(a_{j,k}=-1)) \\
&= T^b.
\end{aligned}$$

The expectation analysis of the unbalanced triangle counts is similar to that of balanced triangle counts. Due to space limits, we omit the proof that $\bar{T}^u$ is an unbiased estimate of the true unbalanced triangle counts. □

To address the privacy concerns, each node needs to perturb its computed statistics via an LDP mechanism before sending them to the data curator. A simple implementation is to utilize the Laplace mechanism for the data perturbation. Nevertheless, this approach is less practical due to the high global sensitivity,

quantified as $2(n-2)$. To solve this problem, Imola et al. [20] employ graph projection to reduce sensitivity. This approach removes some neighbors from the adjacency list so that the maximum degree d_{max} is bounded by a predetermined threshold $\hat{d}_{max}$. As a result, the global sensitivity is reduced to $2(\hat{d}_{max}-1)$. However, there are two main limitations of this method. The first limitation is that the determination of the threshold $\hat{d}_{max}$ requires an additional privacy budget. In addition, since real-world graphs obey power-law degree distributions [3], [22], the direct addition of Laplace noise calibrated to $2(\hat{d}_{max}-1)$ would substantially diminish the utility of the data. To alleviate these limitations, we propose an innovative approach that employs a smooth upper bound on local sensitivity to calibrate the noise.

Given the presence of empirical estimates, the release statistics for each node v_i are adjusted to $(T_i^b - qs_i, T_i^u - qs_i)$. Thus, we need to devise a function that can efficiently calculate the smooth upper bound on local sensitivity for such statistical estimates. For node v_i, the local sensitivity of these estimates can be expressed as follows:

$$LS_{\triangle}(\boldsymbol{a}_i) = \max_{d(\boldsymbol{a}_i,\tilde{\boldsymbol{a}}_i)\leq 1} |\tilde{T}_i^b - T_i^b - q(\tilde{s}_i - s_i)| + |\tilde{T}_i^u - T_i^u - q(\tilde{s}_i - s_i)|. \tag{18}$$

Let d_i' represent the number of nodes that are connected to v_i and have smaller IDs than v_i. A specific smooth upper bound on local sensitivity is described in the subsequent theorem.

Theorem 5. *For any node v_i, let $U(\boldsymbol{a}_i, t) = \max(d_i' + t, 2(d_i' + t - 1))$. The function $S_{\triangle,\beta}(\boldsymbol{a}_i) = \max_{t\in[0,i-1-d_i']} e^{-\beta t} U(\boldsymbol{a}_i, t)$ is a β-smooth upper bound on local sensitivity for node v_i.*

Proof. For any node v_i, the statistics T_i^b, T_i^u, and s_i are computed under the premise that node IDs satisfy the condition $i > j > k$. Thus, in the subsequent proof, we restrict our focus to modifications of the elements from the 1-st to the $(i-1)$-th in $\boldsymbol{a}_i$. Moreover, because the maximum of d_i' is $i-1$, the number of modifications is restricted to $i-1-d_i'$. First of all, we prove that $U(\boldsymbol{a}_i, 0) \geq LS_{\triangle}(\boldsymbol{a}_i)$. In the scenario where the neighbor $\tilde{\boldsymbol{a}}_i$ is obtained by the addition of a signed edge connected to v_i, s_i and $\tilde{s}_i$ can be expressed as $s_i = \binom{d_i'}{2}$ and $\tilde{s}_i = \binom{\tilde{d}_i'}{2} = \binom{d_i'+1}{2}$, respectively. Then we deduce that $\tilde{s}_i - s_i = d_i'$. In addition, since we consider an additional constraint that $\hat{a}_{j,k} = 1$ or $\hat{a}_{j,k} = -1$ for the computation of T_i^b and T_i^u, it follows that $(\tilde{T}_i^b + \tilde{T}_i^u) - (T_i^b + T_i^u) \leq \tilde{s}_i - s_i$. Given that $q = 1/(e^{\epsilon_1} + 2)$, it is easily verified that $|\tilde{T}_i^b - T_i^b - q(\tilde{s}_i - s_i)| + |\tilde{T}_i^u - T_i^u - q(\tilde{s}_i - s_i)| \leq d_i'$. If $\tilde{\boldsymbol{a}}_i$ is obtained by the deletion of a signed edge attached to v_i, then $s_i - \tilde{s}_i = d_i' - 1$. This leads to $|\tilde{T}_i^b - T_i^b - q(\tilde{s}_i - s_i)| + |\tilde{T}_i^u - T_i^u - q(\tilde{s}_i - s_i)| \leq d_i' - 1$. In the case where $\tilde{\boldsymbol{a}}_i$ is obtained by the alteration of the sign of an edge linked to node v_i, s_i and $\tilde{s}_i$ remain equal. From this, we derive that $|\tilde{T}_i^b - T_i^b - q(\tilde{s}_i - s_i)| + |\tilde{T}_i^u - T_i^u - q(\tilde{s}_i - s_i)| \leq 2(d_i' - 1)$. Based on the above derivation, we can conclude that $U(\boldsymbol{a}_i, 0) \geq LS_{\triangle}(\boldsymbol{a}_i)$.

Given that any one modification to $\boldsymbol{a}_i$ results in a variation of d_i' by at most 1, we deduce the inequality $U(\boldsymbol{a}_i, t) \leq U(\tilde{\boldsymbol{a}}_i, t+1)$ holds for all pairs of $\boldsymbol{a}_i$, and $\tilde{\boldsymbol{a}}_i$, provided that $d(\boldsymbol{a}_i, \tilde{\boldsymbol{a}}_i) \leq 1$. □

For each node $v_i \in \mathcal{V}$, the smooth upper bound on local sensitivity $S_{\triangle,\beta}(\boldsymbol{a}_i)$ can be expressed as

$$S_{\triangle,\beta}(\boldsymbol{a}_i) = \max_{t\in[0,i-1-d_i']} \{e^{-\beta t}(d_i' + t), 2e^{-\beta t}(d_i' + t - 1)\}. \tag{19}$$

The calculation of $S_{\triangle,\beta}(\boldsymbol{a_i})$ is similar to that of the smooth upper bound on local sensitivity under centralized DP. For any number d_i', the maximum values of the functions $h(t) = e^{-\beta t}(d_i'+t)$ and $g(t) = 2e^{-\beta t}(d_i'+t-1)$ are achieved at $t = \frac{1}{\beta} - d_i'$ and $t = \frac{1}{\beta} - d_i' + 1$, respectively. Given the discrete constraints and the specified interval for t, the value of $S_{\triangle,\beta}(\boldsymbol{a_i})$ can be determined in constant time.

Algorithm 5: The Two-Phase Framework

Input: signed graph G represented as adjacency vectors $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$, the privacy budget ϵ_1, ϵ_2, the invalidation probability δ

Output: private estimates for balanced triangles $\hat{T}^b$ and unbalanced triangles $\hat{T}^u$

Initialize $q \leftarrow \frac{1}{e^{\epsilon_1}+2}$, $\beta \leftarrow \frac{\epsilon_2}{8+4\ln(2/\delta)}$;
/* First phase */
// Node side
for *each node* $v_i \in \mathcal{V}$ **do**
  $\hat{\boldsymbol{a}}_i \leftarrow (GRR_{\epsilon_1}(a_{i,1}), \ldots, GRR_{\epsilon_1}(a_{i,i-1}))$;
  Send $\hat{\boldsymbol{a}}_i = \{\hat{a}_{i,1}, \ldots, \hat{a}_{i,i-1}\}$ to the server;
// Server side
Construct a noisy signed graph $\hat{G}$ based on $\{\hat{\boldsymbol{a}}_1, \ldots, \hat{\boldsymbol{a}}_n\}$;
/* Second phase */
// Node side
for *each node* $v_i \in \mathcal{V}$ **do**
  $T_i^b \leftarrow |\{(v_i, v_j, v_k) : i > j > k, a_{i,j} \cdot a_{i,k} \cdot \hat{a}_{j,k} = 1\}|$;
  $T_i^u \leftarrow |\{(v_i, v_j, v_k) : i > j > k, a_{i,j} \cdot a_{i,k} \cdot \hat{a}_{j,k} = -1\}|$;
  $s_i \leftarrow |\{(v_i, v_j, v_k) : i > j > k, a_{i,j} \cdot a_{i,k} \neq 0\}|$;
  $\hat{T}_i^b \leftarrow (T_i^b - q \cdot s_i) + Lap(2S_{\triangle,\beta}(\boldsymbol{a}_i)/\epsilon_2)$;
  $\hat{T}_i^u \leftarrow (T_i^u - q \cdot s_i) + Lap(2S_{\triangle,\beta}(\boldsymbol{a}_i)/\epsilon_2)$;
  Send the perturbed data $\hat{T}_i^b, \hat{T}_i^u$ to the server;
// Server side
[s] $\hat{T}^b \leftarrow \frac{1}{1-3q}\sum_{i=1}^n \hat{T}_i^b$; $\hat{T}^u \leftarrow \frac{1}{1-3q}\sum_{i=1}^n \hat{T}_i^u$;
return $\hat{T}^b, \hat{T}^u$

Algorithm 5 outlines the overall protocol of our two-phase framework. The algorithm takes a signed graph G, the privacy budgets ϵ_1 and ϵ_2 for the first and second phases, and an invalidation probability δ as inputs. In the first phase, each node v_i applies GRR_{ϵ_1} to the elements $a_{i,1}, \ldots, a_{i,i-1}$ for smaller node IDs in his/her adjacency vector $\boldsymbol{a}_i$. Then node v_i submits the obfuscated data $\hat{\boldsymbol{a}}_i = \{\hat{a}_{i,1}, \ldots, \hat{a}_{i,i-1}\} \in \{0, +1, -1\}^{i-1}$ to the server. Finally, the server constructs a noisy signed graph $\hat{G}$ based on the data collected from all nodes.

In the second phase, each node v_i computes the statistics T_i^b, T_i^u and s_i under the assumption that the node IDs follow the order $i > j > k$. Subsequently, each node v_i introduces the Laplace noise $Lap(2S_{\triangle,\beta}(\boldsymbol{a}_i)/\epsilon_2)$ to both $T_i^b - qs_i$ and $T_i^u - qs_i$ and sends the perturbed statistics $\hat{T}_i^b$ and $\hat{T}_i^u$ to the server. The server then releases the unbiased estimates $\hat{T}^b$ and $\hat{T}^u$, formulated as $\hat{T}^b = \frac{1}{1-3q}\sum_{i=1}^n \hat{T}_i^b$ and $\hat{T}^u = \frac{1}{1-3q}\sum_{i=1}^n \hat{T}_i^u$, respectively.

Privacy Guarantee. The subsequent theorem formalizes the privacy guarantees provided by the two-phase framework.

Theorem 6. *Algorithm 5 satisfies $(\epsilon_1+\epsilon_2, \delta)$-signed edge local*

TABLE I: Dataset Statistics

DATASET	NODES	EDGES	T^b	T^u
WikiElections	7,115	100,693	458,597	148,682
Epinions	131,580	711,210	4,368,206	541,870
WikiPolitics	138,587	715,883	2,659,365	318,661
Youtube	1,134,890	2,987,624	1,626,212	1,430,174
Pokec	1,632,803	30,622,564	17,321,674	15,235,784

differential privacy.

Proof. In the first phase, since the GRR mechanism is employed to perturb the elements of adjacency vectors, it ensures ϵ_1-signed edge LDP. In the second phase, each node v_i injects Laplace noise calibrated to the smooth upper bound on local sensitivity $S_{\triangle,\beta}(\boldsymbol{a}_i)$ into the statistical query response. This step provides (ϵ_2, δ)-signed edge LDP. In accordance with the sequential composition property of differential privacy, as delineated in Proposition 1, the two-phase framework satisfies $(\epsilon_1 + \epsilon_2, \delta)$-signed edge LDP. □

V. EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the privacy-utility performance of the proposed methods under both centralized and local DP.

A. Experimental Setup

Datasets. In our experiments, we use five real-world datasets. The important statistics of these datasets are summarized in Table I. WikiElections and WikiPolitics can be obtained from the Koblenz network collection[2]. And the rest of the datasets are available from the Stanford network dataset collection[3]. WikiElections, Epinions, and WikiPolitics are all real-world signed graphs. Youtube and Pokec are social networks utilized herein to evaluate the scalability of our proposed algorithms. Following the procedure described in [3], we generate signed labels by randomly allocating 70% of the edges as positive and the remainder as negative.

Competitors and Parameter Selection. To the best of our knowledge, we are the first to study the problem of privacy-preserving triangle counting in signed graphs. Since there is no previous research on the subject, we present baseline methods and evaluate our proposed approaches on five datasets.

In the centralized model, our proposed approaches introduce Laplace noise into the query response, and the difference lies in the noise scale. The approach that calibrates the noise based on the smooth sensitivity is denoted as CentralSS, while the one that calibrates the noise based on the smooth upper bound on local sensitivity is referred to as CentralSU. Furthermore, we implement a baseline approach, CentralGS, which injects Laplace noise proportional to the global sensitivity into the response. To provide a fair comparison, the baseline approach releases true counts with a probability of δ, and the perturbed counts otherwise. All approaches satisfy (ϵ, δ)-centralized DP, which has similar semantics to ϵ-centralized DP when $1/\delta$ is at least the number of possible edges in the signed graph [12]. Thus, for all centralized methods, δ is set as $1/(10\binom{n}{2})$, where n is the number of nodes in the signed graph.

In the local model, our proposed two-phase framework is called as LocalTwoSU. To evaluate the performance of our method, we compare it with two other approaches: LocalOne and LocalTwoGS. LocalOne estimates the statistical counts from $\hat{G}$. LocalTwoGS, similar to LocalTwoSU, operates within a two-phase framework. Nevertheless, the distinction arises in the second phase, where LocalTwoGS uses graph projection to diminish global sensitivity. Subsequently, each node reports the true counts with a probability of δ and, alternatively, provides the counts distorted by Laplace noise in other instances. Note that LocalOne satisfies ϵ-local DP, while LocalTwoGS and LocalTwoSU satisfy (ϵ, δ)-local DP. For parameter selection, we set $\epsilon_1 = 0.5\epsilon$ and $\epsilon_2 = 0.5\epsilon$ for the first phase and second phase, respectively. For the invalidation probability δ, we set δ to $\frac{1}{10n}$, where n denotes the total number of nodes.

Evaluation Metric. We assess the performance of our methods and baselines for balanced and unbalanced triangle counts across five datasets. The accuracy of each method is measured by the *Relative Error* (RE) [18], [20], [21], as defined by the formula $\frac{|\hat{T}^b - T^b| + |\hat{T}^u - T^u|}{T^b + T^u}$. To ensure statistical reliability, each experiment is replicated 100 times across all datasets, with the mean outcomes subsequently reported.

Software and Hardware. We implement all methods in C++. All experiments are performed on a machine equipped with an AMD Ryzen Threadripper PRO 3995WX CPU and 256GB main memory. Our code is available online[4].

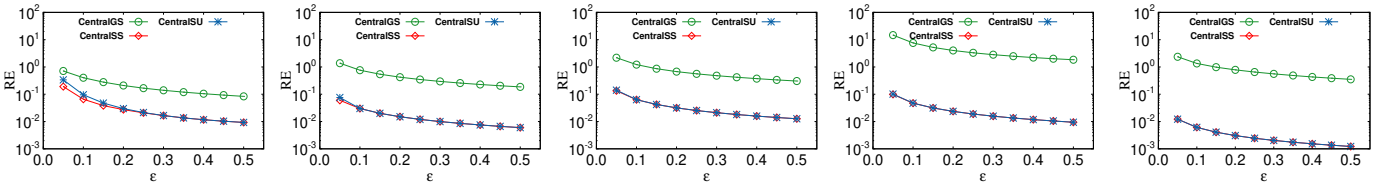
B. Experimental Results

Centralized Model. In the first set of experiments, we evaluate the performance of various differentially private methods as the privacy budget ϵ varies between 0.05 and 0.5. The relative errors of each mechanism for different privacy budgets are presented in Figure 2. The results show that our proposed approaches, CentralSS and CentralSU, both achieve good accuracy over all datasets. When the privacy budget is relatively large, for example, $\epsilon = 0.5$, their relative errors always stay below or close to 1%. With the decrease of ϵ, the accuracies drop, but they are still smaller than 15% even $\epsilon = 0.05$ except the small dataset Wikielections. Moreover, in all cases, our proposed solutions clearly outperform the benchmark method in terms of result accuracy, simply because our methods inject less noise into the true results. The improvement is remarkable since the relative errors are plotted on a logarithmic scale. In the comparison between CentralSS and CentralSU, CentralSS performs more effectively in small datasets and at small ϵ levels, while in other cases, the two methods are comparable. In the Wikielections dataset, for instance, CentralSS has better accuracy than CentralSU when $\epsilon \leq 0.25$. This phenomenon can be attributed to the fact that the smooth sensitivity is optimal over all smooth bounds. The noise added in CentralSU

[2]http://konect.cc/networks/

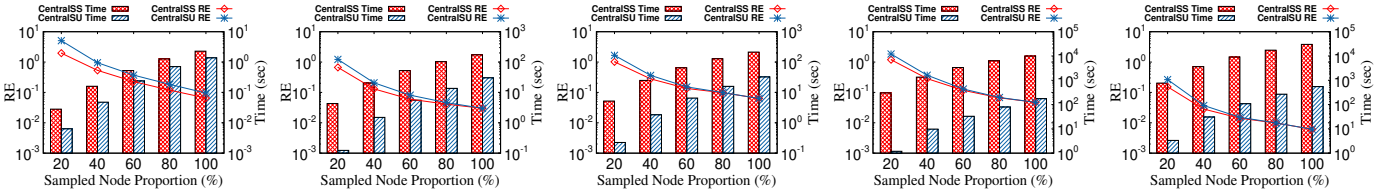
[3]http://snap.stanford.edu/data/

[4]https://github.com/Zening-Li/TC-SG



(a) WikiElections (b) Epinions (c) WikiPolitics (d) Youtube (e) Pokec

Fig. 2: Trade-offs between privacy and relative error (RE) of various mechanisms under centralized DP.



(a) WikiElections (b) Epinions (c) WikiPolitics (d) Youtube (e) Pokec

Fig. 3: Comparison of both relative error (RE) and runtime between CentralSS and CentralSU across varying sampled node proportions at $\epsilon = 0.1$.

relies on the smooth upper bound on local sensitivity $S_{\triangle,\beta}(G)$, where the stationary point of $S_{\triangle,\beta}(G)$ is subject to β (i.e.,ϵ), $W^1(G)$ and $W^2(G)$ jointly. When these influences take on small values, $S_{\triangle,\beta}(G)$ reaches its peak at increased t values, which could cause $S_{\triangle,\beta}(G)$ to exceed the smooth sensitivity.

In the second set of experiments, we assess the impact of the number of nodes n on the relative error and runtime. Specifically, we construct four subgraphs by randomly sampling 20%-80% nodes. The performance of the CentralSS and CentralSU in terms of relative error and runtime at $\epsilon = 0.1$ is summarized in Figure 3. It is observed that the relative errors of both of our proposed methods decrease with the increase in the number of nodes. This result demonstrates that we can accurately estimate triangle counts for large n in the centralized model. When the dataset size is small, such as the proportion of sampled nodes is less than or equal to 60%, the relative error of CentralSS is lower than that of CentralSU in all datasets. However, as the number of nodes increases, the runtime of both approaches increases, but CentralSU remains within a relatively acceptable runtime in all cases. For example, in the Youtube dataset, when the sampling node ratio is 100%, the runtime of CentralSU is about 175 seconds, while the runtime of CentralSS is about 10^4 seconds. Hence, we infer CentralSS is preferable for smaller datasets, while CentralSU is better suited for larger datasets.

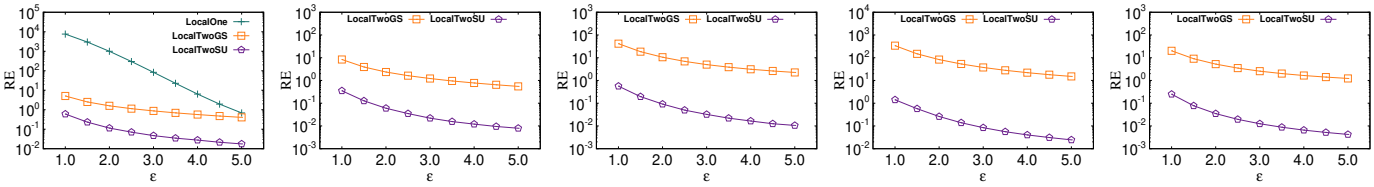
Local Model. To evaluate the effectiveness of our proposed two-phase framework, LocalTwoSU, we compare its accuracy with two other approaches, LocalOne and LocalTwoGS. We vary the privacy budget from 1.0 to 5.0, and the results of these methods are depicted in Figure 4. Note that LocalOne is only tested on the Wikielections dataset, the smallest in our study, because its $\mathcal{O}(n^3)$ time complexity renders it impractical for other datasets [20], [22]. In summary, our proposed solution, LocalTwoSU, achieves good accuracy over all datasets. While LocalOne satisfies a stricter notion of privacy, it injects too much noise, which reduces the estimation accuracy. In contrast, LocalTwoSU consistently outperforms all competitors on all datasets because it adds less noise to the final results.

Then, we evaluate the impact of empirical estimation on relative error. Figure 5 illustrates the comparative performance of LocalTwoSU versus the approach without empirical estimation across various privacy budgets. Due to the space limitation, we restrict our presentation to Wikielections and Youtube but note that similar results can also be observed on the other datasets. It is evident that LocalTwoSU clearly outperforms the method w/o empirical estimation in all cases, which indicates that the unbiased correction does reduce the estimation error.

In addition, in order to evaluate the impact of privacy budget allocation on the estimation accuracy, we adjust the allocation ratio of ϵ_1 (the privacy budget for the first phase) from 0.1 to 0.9 and analyze the relative errors produced by our method under various privacy budgets $\epsilon \in \{1.0, 3.0, 5.0\}$. The experiment results are illustrated in Figure 6. We observe that an allocation ratio between 0.4 and 0.5 consistently results in the lowest relative error, which indicates a balanced requirement for privacy budgets in both phases. Consequently, adjacency vector perturbation in the first phase and statistical estimation perturbation in the second are equally important for obtaining accurate outcomes.

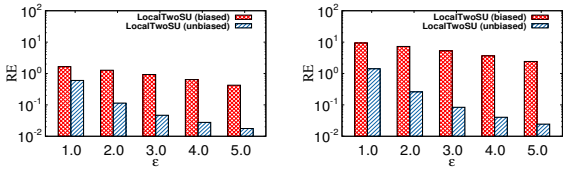
VI. Related Work

Graph Analysis under Centralized DP. Centralized DP techniques are predicated on the existence of a trusted curator. A substantial portion of graph analysis research under centralized DP has focused on the estimation of graph statistics. Nissim et al. pioneer the concept of smooth sensitivity and apply it to the estimation of minimum spanning tree costs and triangle counts [11]. This method has been further extended to other subgraph count queries, such as k-stars and k-triangles [12]. In addition, ladder functions are utilized to provide empirical improvements in accuracy with efficient time complexities [13].



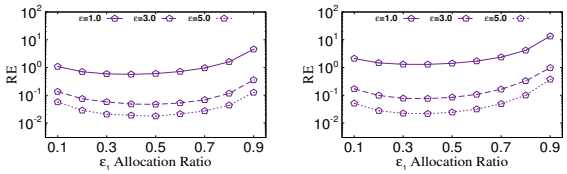
(a) WikiElections (b) Epinions (c) WikiPolitics (d) Youtube (e) Pokec

Fig. 4: Trade-offs between privacy and relative error (RE) of various mechanisms under local DP.



(a) WikiElections (b) Youtube

Fig. 5: Comparison of relative error (RE) between biased and unbiased LocalTwoSU under local DP.



(a) WikiElections (b) Youtube

Fig. 6: Impact of privacy budget allocation on LocalTwoSU performance under local DP.

Concurrently, innovative methods have been developed for the private release of degree distributions and the estimation of local clustering coefficients [14]–[16].

Apart from graph statistics, researchers have also developed various methods to address other graph-related problems under centralized DP. For instance, some studies have explored the problem of releasing specific node subsets from input graphs, such as the vertex cover [28] and densest subgraphs [29], [30]. Another research direction is to generate differentially private synthetic graphs [31], [32]. Despite these advancements, there is still a lack of discussion in the literature on the estimation of the counts of balanced and unbalanced triangles in signed graphs under centralized DP.

Graph Analysis under LDP. Different from centralized DP, local DP operates under the assumption that the data curator is untrusted. Qin et al. [17] propose a multi-phase framework for synthetic graph generation. In [19], the authors introduce the LF-GDPR framework to estimate various graph metrics in a privacy-preserving manner, such as the clustering coefficient and modularity.

Moreover, the problem of subgraph counting under local DP has attracted much attention in recent research [18], [20]–[22], [33]. Sun et al. [18] estimate the number of subgraphs under the assumption that each user permits his/her friends to see all his/her connections, but this assumption does not hold in many practical scenarios. Ye et al. [33] apply the randomized response mechanism on the adjacency matrix. However, this method introduces substantial bias to the estimation. Imola et al. estimate the number of k-stars and triangles via multiple rounds of interactions to reduce estimation errors [20] and employ edge sampling to improve communication efficiency [21]. Nevertheless, none of the current LDP mechanisms are specifically tailored to privately release the number of balanced and unbalanced triangles for signed graphs.

Signed Graph Analysis. The rapid popularity of online social media has notably increased the prevalence of signed graphs. Recent years have seen notable advancements in their analysis, as documented in a comprehensive survey [34]. The origins of signed graph analysis can be traced back to the realm of social psychology, and the structural balance theory is first presented in [35]. Subsequent research has explored diverse applications such as link prediction [1], [36], [37], balancedness analysis [5], [9], community detection [4], [10], [38], and cohesive subgraph mining [3], [6], [39]. More recently, Arya et al. [7] proposed an efficient algorithm to estimate the number of balanced and unbalanced triangles. Despite these advancements, the direct release of graph statistics compromises individual privacy, an issue our research endeavors to address.

VII. Conclusion

In this paper, we propose a series of algorithms designed for counting balanced and unbalanced triangles under centralized and local differential privacy, respectively. In the centralized model, our smooth-sensitivity-based method establishes a new benchmark for effectiveness. Meanwhile, our differentially private solution based on smooth upper bound on local sensitivity not only demonstrates superior efficiency but also provides reliable estimations for most signed graphs. In the local model, our proposed response mechanism introduces less noise to the true results, enhancing data utility. As a direction for future research, we intend to extend our algorithmic developments to cover more complex subgraph counts in signed graphs, such as cliques and 4-cycles.

References

[1] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *WWW*, 2010, pp. 641–650.
[2] J. Tang, C. Aggarwal, and H. Liu, "Recommendations in signed social networks," in *WWW*, 2016, pp. 31–40.
[3] R.-H. Li, Q. Dai, L. Qin, G. Wang, X. Xiao, J. X. Yu, and S. Qiao, "Signed clique search in signed networks: concepts and algorithms," *TKDE*, vol. 33, no. 2, pp. 710–727, 2019.
[4] R. Sun, C. Chen, X. Wang, Y. Zhang, and X. Wang, "Stable community detection in signed social networks," *TKDE*, vol. 34, no. 10, pp. 5051–5055, 2020.

[5] P. K. Pandey, B. Adhikari, M. Mazumdar, and N. Ganguly, "Modeling signed networks as 2-layer growing networks," *TKDE*, vol. 34, no. 7, pp. 3377–3390, 2020.

[6] R. Sun, C. Chen, X. Wang, W. Zhang, Y. Zhang, and X. Lin, "Efficient maximum signed biclique identification," in *ICDE*, 2023, pp. 1313–1325.

[7] A. Arya, P. K. Pandey, and A. Saxena, "Balanced and unbalanced triangle count in signed networks," *TKDE*, vol. 35, no. 12, pp. 12 491–12 496, 2023.

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

[9] T. Derr, C. Aggarwal, and J. Tang, "Signed network modeling based on structural balance theory," in *CIKM*, 2018, pp. 557–566.

[10] Y. Kang, W. Lee, Y.-C. Lee, K. Han, and S.-W. Kim, "Adversarial learning of balanced triangles for accurate community detection on signed networks," in *ICDM*, 2021, pp. 1150–1155.

[11] K. Nissim, S. Raskhodnikova, and A. Smith, "Smooth sensitivity and sampling in private data analysis," in *STOC*, 2007, pp. 75–84.

[12] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev, "Private analysis of graph structure," *TODS*, vol. 39, no. 3, pp. 1–33, 2014.

[13] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, "Private release of graph statistics using ladder functions," in *SIGMOD*, 2015, pp. 731–745.

[14] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith, "Analyzing graphs with node differential privacy," in *TCC*, 2013, pp. 457–476.

[15] W.-Y. Day, N. Li, and M. Lyu, "Publishing graph degree distribution with node differential privacy," in *SIGMOD*, 2016, pp. 123–138.

[16] X. Ding, S. Sheng, H. Zhou, X. Zhang, Z. Bao, P. Zhou, and H. Jin, "Differentially private triangle counting in large graphs," *TKDE*, 2021.

[17] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *CCS*, 2017, pp. 425–438.

[18] H. Sun, X. Xiao, I. Khalil, Y. Yang, Z. Qin, H. Wang, and T. Yu, "Analyzing subgraph statistics from extended local views with decentralized differential privacy," in *CCS*, 2019, pp. 703–717.

[19] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, "Lf-gdpr: A framework for estimating graph metrics with local differential privacy," *TKDE*, vol. 34, no. 10, pp. 4905–4920, 2020.

[20] J. Imola, T. Murakami, and K. Chaudhuri, "Locally differentially private analysis of graph statistics," in *USENIX Security*, 2021, pp. 983–1000.

[21] J. Imola, T. Murakami, and K. Chaudhuri, "Communication-efficient triangle counting under local differential privacy," in *USENIX Security*, 2022, pp. 537–554.

[22] J. Imola, T. Murakami, and K. Chaudhuri, "Differentially private triangle and 4-cycle counting in the shuffle model," in *CCS*, 2022, pp. 1505–1519.

[23] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *FOCS*, 2013, pp. 429–438.

[24] V. A. Farias, F. T. Brito, C. Flynn, J. C. Machado, S. Majumdar, and D. Srivastava, "Local dampening: Differential privacy for non-numeric queries via local sensitivity," *The VLDB Journal*, pp. 1–24, 2023.

[25] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," *NeurIPS*, vol. 27, 2014.

[26] C. Dwork, "Differential privacy: A survey of results," in *International conference on theory and applications of models of computation*, 2008, pp. 1–19.

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

[28] A. Gupta, K. Ligett, F. McSherry, A. Roth, and K. Talwar, "Differentially private combinatorial optimization," in *SODA*, 2010, pp. 1106–1125.

[29] D. Nguyen and A. Vullikanti, "Differentially private densest subgraph detection," in *ICML*, 2021, pp. 8140–8151.

[30] L. Dhulipala, Q. C. Liu, S. Raskhodnikova, J. Shi, J. Shun, and S. Yu, "Differential privacy from locally adjustable graph algorithms: k-core decomposition, low out-degree ordering, and densest subgraphs," in *FOCS*, 2022, pp. 754–765.

[31] Q. Xiao, R. Chen, and K.-L. Tan, "Differentially private network data release via structural inference," in *KDD*, 2014, pp. 911–920.

[32] Z. Jorgensen, T. Yu, and G. Cormode, "Publishing attributed social graphs with formal privacy guarantees," in *SIGMOD*, 2016, pp. 107–122.

[33] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, "Towards locally differentially private generic graph metric estimation," in *ICDE*, 2020, pp. 1922–1925.

[34] J. Tang, Y. Chang, C. Aggarwal, and H. Liu, "A survey of signed network mining in social media," *CSUR*, vol. 49, no. 3, pp. 1–37, 2016.

[35] D. Cartwright and F. Harary, "Structural balance: a generalization of heider's theory." *Psychological review*, vol. 63, no. 5, p. 277, 1956.

[36] J. Ye, H. Cheng, Z. Zhu, and M. Chen, "Predicting positive and negative links in signed social networks by transfer learning," in *WWW*, 2013, pp. 1477–1488.

[37] X. Li, H. Fang, and J. Zhang, "Rethinking the link prediction problem in signed social networks," in *AAAI*, vol. 31, no. 1, 2017.

[38] J. Zhao, R. Sun, Q. Zhu, X. Wang, and C. Chen, "Community identification in signed networks: a k-truss based model," in *CIKM*, 2020, pp. 2321–2324.

[39] R. Sun, Q. Zhu, C. Chen, X. Wang, Y. Zhang, and X. Wang, "Discovering cliques in signed networks based on balance theory," in *DASFAA*. Springer, 2020, pp. 666–674.