Acyclic Graph Pattern Counting under Local Differential Privacy

YIHUA HU, Nanyang Technological University, Singapore
KUNCAN WANG, Nanyang Technological University, Singapore
WEI DONG*, Nanyang Technological University, Singapore

Graph pattern counting serves as a cornerstone of network analysis with extensive real-world applications. Its integration with local differential privacy (LDP) has gained growing attention for protecting sensitive graph information in decentralized settings. However, existing LDP frameworks are largely ad hoc, offering solutions only for specific patterns such as triangles and stars. A general mechanism for counting arbitrary graph patterns, even for the subclass of acyclic patterns, has remained an open problem. To fill this gap, we present the first general solution for counting arbitrary acyclic patterns under LDP. We identify and tackle two fundamental challenges: generalizing pattern construction from distributed data and eliminating node duplication during the construction. To address the first challenge, we propose an LDP-tailored recursive subpattern counting framework that incrementally builds patterns across multiple communication rounds. For the second challenge, we apply a random marking technique that restricts each node to a unique position in the pattern during computation. Our mechanism achieves strong utility guarantees: for any acyclic graph pattern with k edges, we achieve an additive error of $\tilde{O}(\sqrt{N}d(G)^k)$, where N is the number of nodes and $d(G)$ is the maximum degree of the input graph G. Experiments on real-world graph datasets across multiple types of acyclic patterns demonstrate that our mechanisms achieve up to 46-2600× improvement in utility and 300-650× reduction in communication cost compared to the baseline methods.

CCS Concepts: • **Security and privacy** → **Database and storage security**; • **Information systems** → **Data management systems**.

Additional Key Words and Phrases: Differential privacy, Motif counting

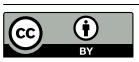
ACM Reference Format:
Yihua Hu, Kuncan Wang, and Wei Dong. 2026. Acyclic Graph Pattern Counting under Local Differential Privacy. *Proc. ACM Manag. Data* 4, 3 (SIGMOD), Article 129 (June 2026), 26 pages. https://doi.org/10.1145/3802006

1 Introduction

Graph pattern counting is fundamental for extracting meaningful insights from graph data, with applications in community detection [46], fraud detection [2], and network characterization [42]. However, directly releasing such analytical results can pose significant privacy risks, as adversaries may infer sensitive information about nodes or edges in the graph [43, 50, 55]. To address this privacy concern, *differential privacy* (DP) [18] has emerged as a gold-standard framework for privacy-preserving graph data analysis. By injecting carefully calibrated noise into the computation, DP prevents the reliable inference of any single node or edge's presence, thereby providing node- or edge-level privacy guarantees.

*Wei Dong is the corresponding author.

Authors' Contact Information: Yihua Hu, Nanyang Technological University, Singapore, Singapore, yihua001@e.ntu.edu.sg; Kuncan Wang, Nanyang Technological University, Singapore, Singapore, kuncan001@e.ntu.edu.sg; Wei Dong, Nanyang Technological University, Singapore, Singapore, wei_dong@ntu.edu.sg.



ACM 2836-6573/2026/6-ART129
https://doi.org/10.1145/3802006

Task	Query Result	Our Results			SOTA		
		Error	Comm.	Time	Error	Comm.	Time
k-line walk counting	$\leq Nd(G)^k$	$\tilde{O}(\sqrt{N}d(G)^{k-1})$	$O(M+N)$	$O(Nd(G))$	-	$O(N^2)$	$O(Nd(G))$
k-line path counting	$\leq Nd(G)^k$	$\tilde{O}(\sqrt{N}d(G)^k)$	$O(M+N)$	$O(Nd(G))$	$O(N^k)$	$O(N^2)$	$O(N^{k+1})$
k-edge acyclic pattern counting	$\leq Nd(G)^k$	$\tilde{O}(\sqrt{N}d(G)^k)$	$O(M+N)$	$O(Nd(G))$	$O(N^k)$	$O(N^2)$	$O(N^{k+1})$

Table 1. Summary of results for three pattern counting queries under edge-LDP.[1][2] Here, N is the number of nodes, M is the number of edges, $d(G)$ is the maximum degree of graph G, and k is the number of edges in the pattern. SOTA denotes the state-of-the-art method: [4] for k-line walk counting, and [48] for both k-line path counting and k-edge acyclic pattern counting. The error bound in [4] depends on ad hoc parameters and is therefore not included.

Most existing studies on DP in graphs adopt the *central differential privacy* (central-DP) model [15, 36, 45, 54], where a trusted curator accesses the entire dataset and applies the DP mechanism globally. While effective in utility (i.e., low error), this model assumes full trust in the curator, making it vulnerable to data breaches and privacy attacks. To mitigate this risk, the *local differential privacy* (LDP) [10, 37] model has gained traction, wherein each data holder privatizes their data locally before sharing it with any external party, including the analyzer and other data holders. In the graph setting, each node functions as a data holder, with access only to its incident edges. Following prior work on LDP-based graph pattern counting [19, 27, 30, 31, 47, 52, 53], we adopt the same edge-level privacy setting, referred to as edge-LDP. Consequently, any released information is well protected, offering robust privacy guarantees even under adversarial conditions.

However, realizing graph pattern counting under LDP is fundamentally challenging due to the decentralized nature of data collection. More specifically, constructing a graph pattern typically requires aggregating information distributed across multiple nodes, which is inherently complex under the LDP privacy requirement. As a result, existing works have primarily focused on simple, localized structures such as stars [30] and triangles [19, 27, 30, 31, 52, 53], where the required information can be gathered with minimal coordination between nodes. Despite this progress, no universal solution exists, even for the subclass of acyclic pattern counting queries, which serve as fundamental primitives in network analysis [33, 39, 41, 42] and relational database queries [24, 51] in the non-private setting. Furthermore, realizing acyclic pattern counting under LDP is natural and practically important when the graph is inherently distributed across multiple parties (e.g., across social platforms). In this work, we identify the following two unresolved challenges in enabling acyclic graph pattern counting under LDP.

Challenge 1. Generalizing acyclic pattern construction. Existing LDP pattern counting solutions typically exploit structure-specific properties to guide computation, making them hard to extend to more complex or varied patterns. For example, in star counting, each node independently counts the number of stars centered at itself and sends a perturbed count to the analyzer for aggregation. In triangle counting, a common approach is for each node to locally enumerate pairs of its neighbors and complete the triangle using the noisy information of the last edge. In contrast, general acyclic pattern counting should support arbitrary sizes and structures without relying on any pattern-specific assumptions.

[1] We use $\Omega(\cdot)$ to denote asymptotic lower bounds, $O(\cdot)$ for asymptotic upper bounds, and $\tilde{O}(\cdot)$ to suppress logarithmic factors. The number of pattern edges k, privacy budget ε, and error probability β are treated as constants and omitted from the asymptotic notation.

[2] All results in the introduction hold with constant probability.

Challenge 2. Eliminating node duplication. Acyclic patterns require that all nodes within a pattern instance be distinct, since repeated nodes would form cycles. While this constraint also applies to stars and triangles, duplication is easy to prevent there because all involved nodes lie within a local neighborhood, which allows a single node to ensure that no repeated nodes appear in the pattern. In contrast, general acyclic patterns may span distant nodes, making it difficult for any single node to detect and eliminate duplication.

A straightforward approach to addressing both challenges is to process data centrally on the analyzer using the standard LDP technique of *Warner's Randomized Response* (RR) [49], as recently adopted by [48]. Specifically, each node privatizes its incident edge list using RR and sends it to the analyzer, which then reconstructs a global noisy graph and performs analytics on it. However, this method yields poor utility with an error bound of $O(N^k)$, where N is the number of nodes and k is the number of edges in the target pattern. In contrast, the true pattern count is at most $Nd(G)^k$, where $d(G)$ is the maximum degree. For many real-world networks with relatively uniform degree distributions, the count typically scales as $O(Nd(G)^k)$. This gap highlights the inefficiency of the RR-based method. In the central-DP literature, algorithms often achieve error bounds of the form $\tilde{O}(\text{poly}(d(G)))$ [34]. Since LDP typically incurs an additional $O(\sqrt{N})$ overhead [6] compared to central-DP, this raises the central question of our work:

Is there a general solution for counting arbitrary acyclic graph patterns under edge-LDP with an error of $\tilde{O}(\sqrt{N}\,\text{poly}(d(G)))$*?*

This paper answers the question affirmatively. To address **Challenge 1**, we recursively reduce acyclic pattern counting to subpattern counting, where each subpattern is also acyclic, and solve it step by step. Consider path counting as an example, where the goal is to count the number of paths with k edges, namely k-line paths. In each round, a node collects from its neighbors the counts of paths of a certain length ending at those neighbors, starting from the base case of 0-line paths, each initialized with a count of 1. The node then aggregates these counts to compute the number of paths that are one edge longer and end at itself. After k rounds, each node holds the count of k-line paths ending at itself, and the global count is obtained by summing over all nodes. To address **Challenge 2**, we develop a *random marking* technique inspired by color-coding [3], which was originally proposed to reduce the runtime of non-private path and cycle detection. With random marking, each node is restricted to appear only at a designated position in the pattern during computation. This technique can be seamlessly combined with our first solution by allowing each node to participate in only one specific round. By integrating both strategies, we develop the first general solution for acyclic pattern counting under edge-LDP. Our main contributions are as follows:

(1) As our first result, we address k-line walk counting under edge-LDP, which serves as the foundation of our mechanisms for acyclic patterns. Our mechanism runs in $k-1$ rounds and produces an unbiased estimate of the true number of k-line walks, with an error of $\tilde{O}(\sqrt{N}d(G)^{k-1})$. Summing over all rounds, the mechanism incurs an overall communication cost of $O(M+N)$, a computation cost of $O(N)$ on the analyzer, and $O(d(G))$ on each node.
(2) We next address k-line path counting by applying random marking to our k-line walk counting solution. The resulting k-round mechanism also produces an unbiased estimate, with a slightly increased error of $\tilde{O}(\sqrt{N}d(G)^k)$. Moreover, it maintains the same overall communication cost of $O(M+N)$ and computation costs of $O(N)$ on the analyzer and $O(d(G))$ on each node.
(3) We further generalize our k-line path counting solution to handle arbitrary acyclic patterns with k edges. The resulting mechanism runs in at most k rounds and produces an unbiased estimate, with asymptotically the same error of $\tilde{O}(\sqrt{N}d(G)^k)$. The communication and

computation costs remain unchanged, with a total communication cost of $O(M + N)$, a computation cost of $O(N)$ on the analyzer, and $O(d(G))$ on each node.

Experimental results show that our mechanism achieves a relative error below 8% on k-line walk counting queries. For acyclic pattern counting tasks, including k-line path counting and k-edge acyclic pattern counting, our mechanisms yield up to 46–2600× improvement in utility and 300–650× reduction in communication cost over the RR-based approach.

2 Related Work

In the literature of graph pattern counting, acyclic patterns play a central role due to their wide applicability and favorable computational properties. They are commonly used in motif analysis [33, 42], where counting subtrees helps characterize branching or hierarchical structures in networks. Moreover, counts of short paths serve as important features in diffusion modeling and link prediction [39, 41]. From a complexity standpoint, there is a natural gap between cyclic and acyclic pattern counting queries in the relational database setting [1, 44]. As a result, many works focus specifically on acyclic queries, where specialized algorithms offer stronger theoretical runtime guarantees [24, 51]. Our work follows this line of reasoning by focusing exclusively on acyclic patterns.

Moving towards the privacy-preserving setting, graph pattern counting has been extensively explored under various DP models, with the central-DP model receiving the most attention. Under edge-DP, Nissim et al. [45] and Karwa et al. [36] leveraged *sensitivity-based mechanisms*; the former focuses on triangle counting, while the latter extends the approach to k-star and k-triangle counting. Zhang et al. [54] proposed the *ladder function*, which supports triangle counting, k-star counting, k-triangle counting, and k-clique counting. Johnson et al. [34] introduced *elastic sensitivity*, enabling, for the first time, the counting of arbitrary graph patterns under edge-DP. Dong and Yi [16] later proposed *residual sensitivity*, which also handles arbitrary graph pattern counting and achieves provably near-optimal error [17]. Additionally, Fichtenberger et al. [21] and Dong et al. [12] addressed edge-DP graph pattern counting in dynamic graph settings. In the context of node-DP, Kasiviswanathan et al. [38] and Blocki et al. [5] addressed arbitrary pattern counting but under assumptions on node degrees. Chen and Zhou [9] introduced the *recursive mechanism*, which significantly improves utility but suffers from high computational cost. More recently, Dong et al. [13, 14] achieved advances in both utility and efficiency for general pattern counting queries. Fang et al. [20] proposed a user-DP framework for monotonic functions that subsumes the node-DP pattern counting problem. Hu et al. [28] extended arbitrary edge-DP pattern counting mechanisms to node-DP with a multiplicative $\tilde{O}(d(G))$ error overhead.

In contrast to work under the central-DP model, which studies both edge-DP and node-DP, research under the LDP model has primarily focused on edge-LDP. Sun et al. [47] studied multi-round counting of triangles, 3-line paths, and k-cliques under an extended local view of nodes. Ye et al. [52, 53] addressed triangle counting performed in a single round. Imola et al. [30] proposed an order-optimal method for k-star counting and a higher-utility solution for triangle counting with single- and two-round variants. Building on this, Imola et al. [31] further reduced the communication cost for triangle counting in the two-round setting. Eden et al. [19] established lower bounds for triangle counting in both single- and multi-round settings. Recently, He et al. [26] presented the first multi-round LDP solution for butterfly counting on bipartite graphs, and He et al. [27] improved the utility and efficiency of triangle counting using a multi-round scheme.

Beyond the above works, Betzer et al. [4] proposed the first multi-round solution for walk counting, a fundamental problem also addressed in our work as a step toward general acyclic pattern counting. While their approach shares our strategy of walk-based decomposition, it relies

on a clipping threshold, which results in a biased estimator. They also briefly mention an idea similar to our non-clipping approach, but without providing any theoretical utility guarantees. Suppakitpaisarn et al. [48] proposed a one-round RR-based solution for general pattern counting queries, but both the error and runtime scale exponentially with respect to N, making the approach impractical for large real-world graphs.

Another DP model in the decentralized setting is the shuffle model, where only a few works have investigated graph pattern counting. Imola et al. [32] proposed the first one-round edge-DP solutions under the shuffle model for triangle and 4-cycle counting.

3 Preliminaries

3.1 Notations

We denote an unweighted, undirected graph as $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_N\}$ is the set of nodes, and $E \subseteq V \times V$ is the set of edges. Self-loops, i.e., edges of the form (u, u), are not allowed. Let $N = |V|$ and $M = |E|$ denote the number of nodes and edges in G, respectively. For a node $v_i \in V$, let $\mathcal{E}(i)$ be the set of incident edges of node v_i, and let $\mathcal{N}(i)$ denote the indices of its neighbors, where $\mathcal{N}(i) = \{j \mid (v_i, v_j) \in \mathcal{E}(i)\}$. The degree of v_i is defined as $d(v_i) = |\mathcal{N}(i)|$, and $d(G)$ denotes the maximum degree in G. If two graphs $G = (V, E)$ and $G' = (V', E')$ differ by exactly one edge $e = (v_i, v_j)$, we write $G \sim_e G'$, or simply $G \sim G'$.

3.2 Graph Pattern Counting

Given any acyclic graph pattern, we use k to denote the number of edges in the pattern. Let $Q : \mathcal{G} \to \mathbb{Z}_{\geq 0}$ be the pattern counting query that returns the number of subgraphs in an input graph G. A common type of acyclic pattern is a path, which is a sequence of distinct nodes connected by edges. In contrast, walks may include repeated nodes, making them computationally easier to count in the non-private setting: walk counting is solvable in polynomial time, whereas path counting is provably #W[1]-complete [22]. We refer to a path with k edges as a k-line path, and a walk with k edges as a k-line walk. Note that walk counting is not considered part of acyclic pattern counting, as walks can contain cycles, e.g., a 3-line walk can form a triangle.

3.3 Differential Privacy on Graphs

Most works on DP in graph analytics operate under the central-DP setting, where a trusted data curator has access to the entire graph dataset. Upon receiving a query, the curator performs analytics and ensures that the output satisfies DP. This guarantees that one cannot distinguish between two neighboring graphs based on the output, thereby preserving privacy. Formally, a randomized mechanism is said to satisfy DP if the following holds:

Definition 3.1 (Differential Privacy). A randomized mechanism $\mathcal{M} : \mathcal{G} \to \mathcal{K}$ satisfies ε-DP if, for any pair of neighboring graphs G, G' and any measurable subset $\mathcal{S} \subseteq \mathcal{K}$, it holds that

$$\Pr[\mathcal{M}(G) \in \mathcal{S}] \leq e^{\varepsilon} \Pr[\mathcal{M}(G') \in \mathcal{S}].$$

Two graphs G and G' are said to be *node-neighboring* if they differ by exactly one node and all of its incident edges. Similarly, they are *edge-neighboring* if they differ by exactly one edge. Applying these notions to the DP definition yields the concepts of *node-differential privacy* (node-DP) and *edge-differential privacy* (edge-DP) [25], respectively.

DP satisfies several desirable properties, as outlined below.

Lemma 3.2 (Post-Processing). *Let $\mathcal{M} : \mathcal{G} \to \mathcal{K}$ be a randomized mechanism that satisfies ε-DP. Then for any (possibly randomized) function $\mathcal{F} : \mathcal{K} \to \mathcal{Z}$, the mechanism $\mathcal{F} \circ \mathcal{M} : \mathcal{G} \to \mathcal{Z}$ also satisfies ε-DP.*

Lemma 3.3 (Basic Composition). *Let $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n$ be a series of randomized mechanisms, where for each $i \in \{1, 2, \ldots, n\}$, $\mathcal{M}_i : \mathcal{G} \to \mathcal{K}_i$ satisfies ε-DP. The composition mechanism $\mathcal{M}(G) = (\mathcal{M}_1(G), \ldots, \mathcal{M}_n(G))$ satisfies $n\varepsilon$-DP.*

Lemma 3.4 (Parallel Composition). *Let $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n$ be mechanisms, each satisfying ε-DP. If for any pair of neighboring graphs G and G', there exists at most one mechanism $\mathcal{M}_i$ such that $\mathcal{M}_i(G) \neq \mathcal{M}_i(G')$, then the composition mechanism $\mathcal{M}(G) = (\mathcal{M}_1(G), \ldots, \mathcal{M}_n(G))$ satisfies ε-DP.*

A common DP mechanism is the *Laplace mechanism*. Here, for any arbitrary graph query $Q : \mathcal{G} \to \mathcal{R}$, its *global sensitivity* is defined as

$$\mathrm{GS}_Q := \max_{G, G'} \|Q(G) - Q(G')\|_1,$$

where G and G' are any pair of neighboring graphs. Then, the Laplace mechanism is defined as follows:

Definition 3.5 (Laplace Mechanism [18]). Given a graph query $Q : \mathcal{G} \to \mathcal{R}$, the mechanism

$$\mathcal{M}(G) = Q(G) + \mathrm{Lap}\left(\frac{1}{\varepsilon}\mathrm{GS}_Q\right)$$

satisfies ε-DP, where $\mathrm{Lap}(\mathrm{GS}_Q/\varepsilon)$ denotes a Laplace random variable with mean 0 and scale $\mathrm{GS}_Q/\varepsilon$.

Specifically, computing global sensitivity under node-neighboring or edge-neighboring graphs leads to node-DP or edge-DP guarantees, respectively.

3.4 Local Differential Privacy

In contrast to the central-DP setting, where data analytics is performed centrally by a trusted curator, LDP studies a distributed setting, in which each party corresponds to a node with access only to its own information. When performing data analytics tasks, each node first privatizes its data locally, then sends the noisy output to an analyzer for post-processing. To guarantee LDP, we must ensure that the collection of messages received by the analyzer satisfies DP, as defined in Definition 3.1. More precisely, each node v_i has access to a subgraph $G_i = \big(\{v_i\} \cup \{v_j \mid j \in \mathcal{N}(i)\}, \mathcal{E}(i)\big)$. To perform graph analytics, each v_i invokes a local randomizer $\mathcal{R}$ on G_i, which produces a privatized output $\mathcal{X}_i = \mathcal{R}(G_i)$. The analyzer $\mathcal{A}$ then aggregates the local outputs $(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_N)$ to produce the final result. A mechanism is said to satisfy ε-LDP if the joint output $(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_N)$ satisfies ε-DP. Similar to the central-DP setting, applying node- or edge-neighboring definitions yields notions of *node-LDP* and *edge-LDP*, respectively. To date, all existing LDP graph analytics literature [19, 27, 30, 31, 47, 52, 53] has focused on the edge-LDP setting, which is also the focus of our work.

A commonly used mechanism under LDP is RR, in which each node $v_i \in V$ outputs a randomized neighbor list. The analyzer $\mathcal{A}$ can then construct a noisy graph based on the received neighboring lists and perform graph analytics on it. Since each edge is shared between two incident nodes, to avoid duplication, the information of edge (v_i, v_j) is reported only by node v_i when $i < j$. Specifically, for each node v_i, RR takes as input a binary neighbor list $\mathbf{a}_i = (a_{i,1}, \ldots, a_{i,i-1}) \in \{0, 1\}^{i-1}$, where $a_{i,j}$ is 1 if $j \in \mathcal{N}(i)$ and 0 otherwise. For each entry $a_{i,j}$, RR keeps the true bit with probability $p = e^\varepsilon/(e^\varepsilon + 1)$ and flips it with probability $q = 1/(e^\varepsilon + 1)$. This mechanism satisfies ε-edge-LDP.

Multi-round LDP mechanism. The above basic LDP model is sufficient for simple queries such as edge counting and star counting [30], where the final result can be obtained in a single round. However, more complex graph analytics tasks like triangle counting [27, 30, 31] often require collaboration between nodes and the analyzer across multiple rounds of interaction. Here, we consider a k-round LDP mechanism. In round $\ell \in \{1, \ldots, k\}$, each node v_i invokes a round-specific randomizer $\mathcal{R}^{(\ell)}$ on its local subgraph G_i, possibly incorporating messages released by the other

parties in the previous $\ell - 1$ rounds. Let $\mathcal{X}_i^{(\ell)}$ denote the privatized output of node v_i in round ℓ, and let the collection of all node messages in round ℓ be $\mathcal{X}^{(\ell)} = (\mathcal{X}_1^{(\ell)}, \ldots, \mathcal{X}_N^{(\ell)})$. The mechanism satisfies ε-LDP if the joint messages from all rounds, i.e., $(\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(k)})$, satisfy ε-DP. To ensure this, we can allocate the privacy budget across rounds such that each $\mathcal{X}^{(\ell)}$ satisfies ε/k-DP. By applying basic composition (Lemma 3.3), the entire k-round process satisfies ε-DP as required. To quantify the communication cost incurred across rounds, we sum the sizes (in bits) of all messages transmitted. For computational cost, we report the total cost incurred across all rounds by the analyzer and by a single node.

3.5 Useful Concentration Bounds

The following three lemmas provide high-probability concentration bounds that will be used in the subsequent analysis.

Lemma 3.6 (Concentration Bound of Laplace Distributions [7]). *Let $\eta_1, \eta_2, \ldots, \eta_k$ be independent random variables with $\eta_i \sim \mathrm{Lap}(b_i)$. Then, for any $\beta > 0$,*

$$\Pr\left[\left|\sum_i \eta_i\right| \geq \sqrt{8 \sum_i b_i^2 \cdot \log \frac{2}{\beta}}\right] \leq \beta.$$

Lemma 3.7 (Multiplicative Chernoff Bound [11]). *Let $X_1, \ldots, X_n$ be independent random variables taking values in $[0, 1]$. Let $X = \sum_{i=1}^n X_i$. For any $0 < \delta \leq 1$,*

$$\Pr[X \geq (1 + \delta)\,\mathbb{E}[X]] \leq \exp(-\frac{1}{3}\delta^2\,\mathbb{E}[X]).$$

Lemma 3.8 (Chebyshev's inequality [8]). *Let X be a random variable with $\mathbb{E}[X] = \mu$ and $\mathrm{Var}(X) = \sigma^2$. For any $k > 0$,*

$$\Pr[|X - \mu| \geq k\sigma] \leq \frac{1}{k^2}.$$

4 LDP Walk Counting

We begin with a special case of acyclic pattern counting: k-line path counting. As a preliminary step, we develop a mechanism for the related task of k-line walk counting. As noted in Section 3.2, walk counting can be viewed as a simpler variant of path counting. Although walk counting does not fall under acyclic graph pattern counting, it forms the foundation for our mechanisms for both path counting and general acyclic graph pattern counting. In this section, we first provide a brief overview of the existing one-round RR-based solution for walk counting, which incurs an error of $O(N^k)$. We then introduce our multi-round LDP approach, which achieves an error bound of $\tilde{O}(Nd(G)^{k-1})$ with constant probability, while improving communication and computation costs by an order of magnitude.

4.1 Overview of the RR-Based Method

Similar to many pattern counting tasks under LDP, such as triangle counting, the key challenge in answering k-line walk queries lies in capturing correlations across multiple edges. [48] adopts a straightforward one-round approach that first constructs a noisy graph $\tilde{G}$ using RR, as described in Section 3.4. Given $\tilde{G}$, for each node pair (v_i, v_j), the analyzer $\mathcal{A}$ then computes an unbiased estimator $\hat{a}_{i,j}$ for the binary existence of edge (v_i, v_j) in the original graph G. Since the existence of any walk can be unbiasedly estimated by the product of the estimators of its constituent edges, $\mathcal{A}$ enumerates all $(k + 1)$-node sequences and sums the corresponding walk estimators to compute the total count. This method guarantees an unbiased estimate with an additive error of $O(N^k)$ for

k-line walk counting, and it extends naturally to general k-edge acyclic pattern counting with the same asymptotic utility guarantee.

Redundant count removal. The above solution treats walk orientations distinctly; that is, the sequences $(v_1, \ldots, v_{k+1})$ and $(v_{k+1}, \ldots, v_1)$ are considered different walks. To eliminate redundant counts due to orientation, [48] divides the total count by the number of automorphisms [23] of the queried pattern. Due to space constraints, we defer detailed discussions on redundant count handling across pattern counting queries to Appendix A.1 of the full version [29].

Communication and computation analysis. The total communication cost is $O(N^2)$, as each node v_i sends a message of size $O(N)$ to construct the noisy graph. For computation, the analyzer $\mathcal{A}$ requires $O(N^{k+1})$ time to enumerate all possible sequences, while each node incurs $O(N)$ time to perturb its neighbor list.

4.2 Our Approach: Multi-Round Aggregation

The problem with the strawman solution is that we publish "too much" information of nodes in an overly "fine-grained" manner. Fundamentally, the more information is released with the same privacy level, the more difficult it is to achieve higher utility based on the released data. Therefore, we would like to collect the data in a more "local" and more "aggregated" manner. Given edge counting (i.e., 1-line walk counting) as an example, instead of letting each node publish its perturbed neighbor list, we release only the degree information. The analyzer then aggregates the perturbed degrees to produce the query result. With this approach, the error can be reduced from typically $\Omega(N)$ to $O(\sqrt{N})$, by simply adding Laplace noises of scale $2/\varepsilon$ to each node's degree count.

Algorithm 1: Local randomizer $\mathcal{R}^{(\ell)}$ of $\mathcal{M}_{\text{k-wk}}$

```
/* Initialization */
foreach i ∈ {1, ..., N} do
    X_i^(0) ← 1
Function R^(ℓ)(G_i, ε):
    Received X_i^(ℓ−1) for each i ∈ {1, ..., N}
    ‖X^(ℓ−1)‖_∞ := max_i |X_i^(ℓ−1)|
    X_i^(ℓ) ← Σ_{j∈N(i)} X_j^(ℓ−1) + Lap(2k/ε ‖X^(ℓ−1)‖_∞)
    Publish X_i^(ℓ)
```

This idea, also explored in [4], can be extended to support arbitrary k-line walk counting queries with multiple rounds of communication: At the beginning, we initialize $\mathcal{X}_i^{(0)} = 1$ for any $v_i \in V$, representing that the number of 0-line walk ending at each node is 1. Then, in round $\ell \in \{1, \ldots, k\}$, each node collects the number of $(\ell-1)$-line walks ending at its neighbors. By aggregating these values, each node can compute the number of ℓ-line walks that end at itself. To ensure the reported counts satisfy edge-LDP, each node must add noise to obscure the presence or absence of edges between its neighbors. Specifically, unlike [4] that limits the global sensitivity using a clipping mechanism, our approach adds Laplace noise with a scale proportional to the maximum count from the previous round. Notably, for each node, the noise scale is determined by the maximum count across all nodes rather than just its own neighborhood, thereby preventing leakage of local structural information. Besides, since each edge contributes to the computation in every round, the total privacy budget is divided across the k rounds, allocating ε/k to each. Finally, the counts

produced in round k are sent to the analyzer $\mathcal{A}$, which aggregates all node reports and returns

$$\mathcal{M}_{\text{k-wk}}(G, \varepsilon) = \sum_{i=1}^{N} \mathcal{X}_i^{(k)}$$

as the output. The detailed local randomizer $\mathcal{R}^{(\ell)}$ applied at round ℓ of this k-round local-DP mechanism $\mathcal{M}_{\text{k-wk}}$ is described in Algorithm 1.

Figure 1 illustrates the execution of the 2-round mechanism $\mathcal{M}_{\text{2-wk}}$ on graph G with a total privacy budget of $\varepsilon = 4$. Each node is assigned a per-round privacy budget of $\varepsilon/2k = 1$. In both rounds 1 and 2, each node first aggregates the outputs from its neighbors in the previous round (shown in blue), and then adds Laplace noise to privatize the result (shown in red). The final output of the mechanism, 5.7, is computed by summing the privatized values of all nodes in round 2.

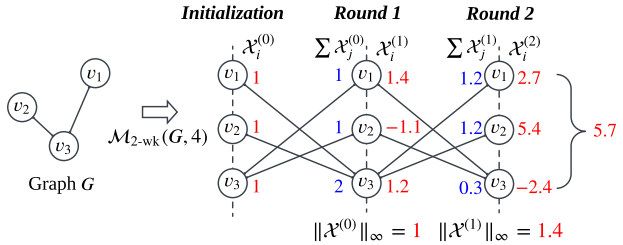


Fig. 1. Illustration of the 2-line walk counting mechanism $\mathcal{M}_{\text{2-wk}}$ applied to graph G with privacy budget $\varepsilon = 4$. For simplicity, $\sum$ denotes $\sum_{j \in \mathcal{N}(i)}$.

For privacy and utility, we establish the following guarantees:

Theorem 4.1. *For any graph G, walk length k, and privacy budget ε, $\mathcal{M}_{\text{k-wk}}$ satisfies ε-edge-LDP.*

Theorem 4.2. *For any graph G, walk length k, and parameters ε and β, the k-round mechanism $\mathcal{M}_{\text{k-wk}}$ satisfies $\mathbb{E}[\mathcal{M}_{\text{k-wk}}(G, \varepsilon)] = W_k$, and with probability at least $1 - \beta$,*

$$|\mathcal{M}_{\text{k-wk}}(G, \varepsilon) - W_k| \le k\gamma\sqrt{N}(d(G) + \gamma)^{k-1} = \tilde{O}(\sqrt{N} d(G)^{k-1}),$$

where $\gamma = 2k\sqrt{8\log(2kN/\beta)}/\varepsilon = \tilde{O}(1)$.

Proofs of all lemmas and theorems in this paper are provided in Appendix A.3 of the full version [29]. By Theorem 4.2, $\mathcal{M}_{\text{k-wk}}$ successfully reduces the error from typically $O(N^k)$ to $\tilde{O}(\sqrt{N} d(G)^{k-1})$ for general k-line walk counting queries.

Communication and computation analysis. The total communication cost of the k-round mechanism $\mathcal{M}_{\text{k-wk}}$ is $O(N^2)$. Specifically, in the first $k-1$ rounds, each node v_i broadcasts its count of $O(1)$ size to all other nodes, incurring a total cost of $O(N^2)$, considering that k is a constant. In the final round, each node sends its count to the analyzer, contributing an additional $O(N)$ cost. For the computational cost, the analyzer $\mathcal{A}$ runs in $O(N)$ time to aggregate the values received from all N nodes in round k. Each node performs $O(N)$ computation across all k rounds: in each round, it spends $O(N)$ time computing a maximum and another $O(N)$ time aggregating values from neighbors.

4.3 Optimization in Efficiency

So far, $\mathcal{M}_{\text{k-wk}}$ has achieved improved computational complexity with state-of-the-art utility. Next, we further optimize both communication and computation.

Round reduction: The number of rounds in $\mathcal{M}_{\text{k-wk}}$ can be reduced from k to $k-1$. The key observation is that, for each k-line walk, the final edge is shared between the second-to-last node and the last node. Therefore, once the second-to-last node obtains the noisy count of $(k-1)$-line walks ending at itself, it can compute the number of k-line walks where it serves as the second-to-last node, eliminating the need for round k. More precisely, in round $k-1$, each node v_i computes $\mathcal{X}_i^{(k-1)}$ as in Algorithm 1 and multiplies it by $d(v_i) + \text{Lap}(2k/\varepsilon)$. The analyzer $\mathcal{A}$ then aggregates these values to produce the final output. We will show that this modification preserves both the privacy and utility guarantees stated in Theorem 4.1 and Theorem 4.2.

Computational cost reduction: Another observation is that, in each round, each node scans through all N results from the previous round to compute a maximum, which incurs significant computational redundancy. To address this, we can shift the maximum computation to the analyzer. More precisely, at the end of each round $\ell \in \{1, \ldots, k-2\}$, each node v_i additionally sends its value $\mathcal{X}_i^{(\ell)}$ to the analyzer $\mathcal{A}$, which computes the global round maximum and broadcasts it to all nodes. This reduces the computation cost at each node from $O(N)$ to $O(d(G))$.

Communication cost reduction: The communication cost between nodes can be reduced from $O(N^2)$ to $O(M)$ under the assumption that nodes connected by an edge have direct communication channels. In this setting, at the end of each of the first $k-2$ rounds, each node sends its result only to its neighbors, while the global maximum is computed by the analyzer as discussed above. This change does not affect the node-level computation, as each node only requires the results from its neighbors and the global maximum. Consequently, in each of the first $k-2$ rounds, each edge facilitates bidirectional communication between its endpoints, leading to a total communication cost of $O(M)$ between nodes.

Algorithm 2: Local randomizer $\mathcal{R}^{(\ell)}$ of optimized $\mathcal{M}_{\text{k-wk}}$

foreach $i \in \{1, \ldots, N\}$ **do**
  $\mathcal{X}_i^{(0)} \leftarrow 1$
Function $\mathcal{R}^{(\ell)}(G_i, \varepsilon)$**:**
  Received $\|\mathcal{X}^{(\ell-1)}\|_\infty$ from $\mathcal{A}$
  Received $\mathcal{X}_j^{(\ell-1)}$ for each $j \in \mathcal{N}(i)$
  $\mathcal{X}_i^{(\ell)} \leftarrow \sum_{j \in \mathcal{N}(i)} \mathcal{X}_j^{(\ell-1)} + \text{Lap}(\frac{2k}{\varepsilon}\|\mathcal{X}^{(\ell-1)}\|_\infty)$
  if $\ell = k-1$ **then**
    $\mathcal{X}_i^{(\ell)} \leftarrow \mathcal{X}_i^{(\ell)} \cdot (d(v_i) + \text{Lap}(\frac{2k}{\varepsilon}))$
  Send out $\mathcal{X}_i^{(\ell)}$ to $\mathcal{N}(i)$ and $\mathcal{A}$

The revised local randomizer is shown in Algorithm 2. These refinements collectively reduce the total number of rounds from k to $k-1$, lower the overall communication cost from $O(N^2)$ to $O(M+N)$, and achieve an improvement from $O(N)$ to $O(d(G))$ in the per-node computational cost. Below, we show that these refinements will not affect the privacy and utility guarantee:

Theorem 4.3. *For any graph G, walk length k, and privacy budget ε, the optimized mechanism $\mathcal{M}_{\text{k-wk}}$ satisfies ε-edge-LDP.*

Theorem 4.4. *For any graph G, walk length k, and parameters ε and β, the optimized $(k-1)$-round mechanism $\mathcal{M}_{\text{k-wk}}$ satisfies $\mathbb{E}[\mathcal{M}_{\text{k-wk}}(G, \varepsilon)] = W_k$, and with probability at least $1-\beta$,*

$$|\mathcal{M}_{\text{k-wk}}(G, \varepsilon) - W_k| \leq k\gamma\sqrt{N}(d(G)+\gamma)^{k-1} = \tilde{O}(\sqrt{N}d(G)^{k-1}),$$

where $\gamma = 2k\sqrt{8\log(2kN/\beta)}/\varepsilon = \tilde{O}(1)$.

5 LDP Path Counting

Now, we extend our LDP solution for walk counting to support path counting. As discussed in Section 3.2, path counting imposes an additional constraint compared with walk counting: each node must appear at most once along the path. Even in the non-private, centralized setting, this constraint introduces significant challenges. As shown in the prior work [35], in theory, walk counting can be solved asymptotically more efficiently than path counting. Moving towards the LDP setting, the problem becomes even more challenging, as enforcing the uniqueness constraint for a path would inherently require information sharing across multiple nodes, which is fundamentally difficult to achieve under the LDP model.

To address this issue, in this section, we introduce a protocol named random marking, which can be seamlessly integrated into our walk counting mechanisms to enable private k-line path counts estimation under LDP. The resulting mechanism achieves a utility bound of $\tilde{O}(\sqrt{N}d(G)^k)$, representing only a multiplicative overhead of $\tilde{O}(d(G))$ compared to the corresponding k-line walk counting utility of $\tilde{O}(\sqrt{N}d(G)^{k-1})$.

5.1 Random Marking: Transforming Path Counting into Walk Counting

As highlighted above, detecting node repetition in a path under the LDP setting is inherently difficult due to limited global visibility. To address this challenge, we draw inspiration from color-coding [3], which colors nodes and searches for patterns with distinct node colors, and adopt a different strategy: rather than detecting repetitions retrospectively, we proactively prevent them by imposing a stricter constraint from the outset. Specifically, for each node, we require that it not only appear at most once in a walk, but also occupy a predefined position within it. Interestingly, this idea was originally introduced to improve runtime for non-private path and cycle detection, whereas we leverage it here to address the challenges in the LDP setting, where only local views are available.

To illustrate the high-level idea, we first consider the non-private setting and defer the DP-relevant operations to the next subsection. We begin with an extra marking round, in which each node v_i samples a random mark r_i uniformly from $\{0, 1, \ldots, k\}$ and sends it to its neighbors. This mark enforces that the node may appear only as the $(r_i + 1)$-th node in any k-line path being counted. For example, if a node v_j draws $r_j = 0$, it can only serve as the starting node of a valid k-line path. We then apply the $(k-1)$-round walk counting mechanism $\mathcal{M}_{\text{k-wk}}$, with an added constraint: each node v_i participates only in the r_i-th round of computation. At all other rounds $\ell \neq r_i$, the node simply outputs 0. More precisely, in each round $\ell \in \{1, \ldots, k-1\}$, nodes $v_i \in V$ with $r_i = \ell$ compute the number of ℓ-line walks ending at themselves with node marks $(0, 1, \ldots, r_i)$. This is achieved by aggregating counts from active neighbors that participated in the previous round. Additionally, nodes with mark $k-1$ complete the last line by multiplying their count by the number of their neighbors marked with k. Notably, since $\mathcal{M}_{\text{k-wk}}$ runs for only $k-1$ rounds, nodes marked with 0 or k do not participate in subsequent computation. To distinguish this from the DP mechanism, we denote the resulting non-private algorithm as $\widehat{\mathcal{M}}_{\text{k-pt}}$, where each local client executes the function $\widehat{\mathcal{R}}_i^{(\ell)}$ at round ℓ. The detailed algorithm is provided in Algorithm 3. Here, we use $\widehat{\mathcal{X}}_i^{(\ell)}$ to denote the non-private output of node v_i in round ℓ, in distinction from the DP-compliant result $\mathcal{X}_i^{(\ell)}$.

With this design, it is straightforward to ensure that each node $v_i \in V$ appears at most once in a walk, since its contribution is limited to being the $(r_i + 1)$-th node in any walk. This enforces the requirement that nodes do not repeat in a path. However, it also introduces the problem of

Algorithm 3: Local client function $\widehat{\mathcal{R}}^{(\ell)}$ of $\widehat{\mathcal{M}}_{\text{k-pt}}$

foreach $i \in \{1, \ldots, N\}$ **do**
  $\widehat{\mathcal{X}}_i^{(0)} \leftarrow 1$
Function $\widehat{\mathcal{R}}^{(\ell)}(G_i)$**:**
  if $\ell = mark$ **then**
    /* Randomly sample a mark */
    $r_i \leftarrow \mathsf{Unif}(0, k)$
    Send out r_i to $\mathcal{N}(i)$
  if $\ell = r_i$ **then**
    Received $\widehat{\mathcal{X}}_j^{(\ell-1)}$ for each $j \in \mathcal{N}(i)$
    $\widehat{\mathcal{X}}_i^{(\ell)} \leftarrow \sum_{j \in \mathcal{N}(i)} \widehat{\mathcal{X}}_j^{(\ell-1)} \mathbf{1}\{r_j = \ell - 1\}$
    if $\ell = k - 1$ **then**
      $\widehat{\mathcal{X}}_i^{(\ell)} \leftarrow \widehat{\mathcal{X}}_i^{(\ell)} \cdot \sum_{j \in \mathcal{N}(i)} \mathbf{1}\{r_j = k\}$
  else
    $\widehat{\mathcal{X}}_i^{(\ell)} \leftarrow 0$
  Send out $\widehat{\mathcal{X}}_i^{(\ell)}$ to $\mathcal{N}(i)$ and $\mathcal{A}$

underestimation. Specifically, a k-line path in the graph is counted only if all nodes along the path receive marks exactly from 0 to k. Since node marks are sampled independently, the probability that a k-line path is sampled is $(k+1)^{-(k+1)}$. Consequently, at the analyzer $\mathcal{A}$, the result is rescaled by $(k+1)^{(k+1)}$:

$$\widehat{\mathcal{M}}_{\text{k-pt}}(G) = (k+1)^{k+1} \sum_{i=1}^{N} \widehat{\mathcal{X}}_i^{(k-1)}.$$

Random marking introduces variance in $\widehat{\mathcal{M}}_{\text{k-pt}}(G)$ relative to the true k-line path count P_k. For utility analysis, we capture this effect as an additive sampling error, which we bound tightly below. Importantly, this sampling error is accounted for in the overall error of our LDP mechanism introduced in Section 5.2.

Lemma 5.1. *For any graph G, path length k, and parameter β, the k-round non-private algorithm $\widehat{\mathcal{M}}_{\text{k-pt}}$ satisfies $\mathbb{E}[\widehat{\mathcal{M}}_{\text{k-pt}}(G)] = P_k$, and with probability at least $1 - \beta$,*

$$|\widehat{\mathcal{M}}_{\text{k-pt}}(G) - P_k| \le (k+1)\sqrt{N\beta^{-1}}(d(G) + k + 1)^k = O(\sqrt{N} d(G)^k).$$

5.2 Integrating DP and Efficiency Optimization

With Algorithm 3, we successfully reduce the problem of k-line path counting to that of k-line walk counting. We can then utilize the idea of the edge-LDP mechanism $\mathcal{M}_{\text{k-wk}}$ to privatize node outputs from Algorithm 3. To begin with, publishing node marks incurs no privacy loss, as the marks are independent of any sensitive data. Before each round ℓ, the analyzer collects the outputs from active nodes, computes the global maximum $\|\mathcal{X}^{(\ell-1)}\|_\infty$, and broadcasts it to all nodes. Then in round ℓ, after aggregating the result from its neighbors in round $\ell - 1$, each active node adds Laplace noise scaled to $\|\mathcal{X}^{(\ell-1)}\|_\infty$. A key distinction from walk counting is that, under random marking, each edge affects at most one node's computation in a single round. Hence, by parallel composition (Lemma 3.2), the privacy budget ε does not need to be divided across nodes or rounds.

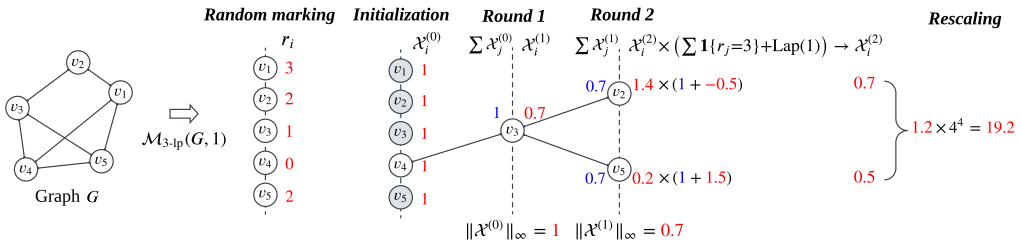


Fig. 2. Illustration of the 3-line path counting mechanism $\mathcal{M}_{3\text{-pt}}$ applied to graph G with privacy budget $\varepsilon = 1$. For simplicity, $\sum$ denotes $\sum_{j\in\mathcal{N}(i)}$. The initialization step occurs before random marking, but is shown afterward for clarity.

With the above idea, we establish the k-round edge-LDP mechanism $\mathcal{M}_{\text{k-pt}}$ for k-line path counting. Compared with the non-private local client $\widehat{\mathcal{R}}^{(\ell)}$ in Algorithm 3, here the local randomizer $\mathcal{R}^{(\ell)}$ adds noise $\text{Lap}(\|\mathcal{X}^{(\ell-1)}\|_\infty/\varepsilon)$ in Line 9 and noise $\text{Lap}(1/\varepsilon)$ to the multiplicative factor $\sum_{j\in\mathcal{N}(i)} \mathbf{1}\{r_j = k\}$ in Line 11. When evaluating the privacy of $\mathcal{M}_{\text{k-pt}}$, astute readers may consider privacy amplification, whereby privacy guarantees can be strengthened when the DP result is derived from an unknown subsample of the input dataset. However, privacy amplification does not apply here. In $\mathcal{M}_{\text{k-pt}}$, node markings are publicly revealed, making it deterministically known whether an edge will be sampled and thus violating the sampling uncertainty required for privacy amplification. Moreover, privatizing node markings to enable privacy amplification introduces new challenges, as each mark must be revealed to neighboring nodes during communication. As a result, we establish the following privacy guarantee:

Theorem 5.2. *For any graph G, path length k, and privacy budget ε, the mechanism $\mathcal{M}_{\text{k-pt}}$ satisfies ε-edge-LDP.*

For utility analysis, the total error $|\mathcal{M}_{\text{k-pt}}(G,\varepsilon) - P_k|$ consists of two parts. The first is the sampling error incurred by random marking, as shown in Lemma 5.1. For the second part, the DP noise introduced by the Laplace mechanism, we show that it slightly differs from that of k-line walk counting but admits the same asymptotic upper bound.

Lemma 5.3. *For any graph G, path length k, parameters ε and β, $\mathcal{M}_{\text{k-pt}}$ satisfies $\mathbb{E}[\mathcal{M}_{\text{k-pt}}(G,\varepsilon)] = \widehat{\mathcal{M}}_{\text{k-pt}}(G)$ under the same node marking $(r_1,\ldots,r_N)$, and with probability at least $1-\beta$,*

$$\begin{aligned}|\mathcal{M}_{\text{k-pt}}(G,\varepsilon) - \widehat{\mathcal{M}}_{\text{k-pt}}(G)| &\le k(k+1)\,\gamma\sqrt{N}(\hat{d}(G) + \gamma\sqrt{\hat{d}(G)} + \gamma)^{k-1} \\ &= \tilde{O}(\sqrt{N}d(G)^{k-1}),\end{aligned}$$

where $\gamma = (k+1)\sqrt{8\log(6N/\beta)}/\varepsilon = \tilde{O}(1)$ and $\hat{d}(G) = \max(d(G), \lceil\gamma^2\rceil)$.

Combining both parts of the error, we establish the utility bound:

Theorem 5.4. *For any graph G, path length k, parameters ε and β, the k-round mechanism $\mathcal{M}_{\text{k-pt}}$ satisfies $\mathbb{E}[\mathcal{M}_{\text{k-pt}}(G,\varepsilon)] = P_k$, and with probability at least $1-\beta$,*

$$\begin{aligned}|\mathcal{M}_{\text{k-pt}}(G,\varepsilon) - P_k| &\le (k+1)\sqrt{2N\beta^{-1}}(d(G)+k+1)^k + \\ k(k+1)\,\gamma\sqrt{N}(\hat{d}(G) + \gamma\sqrt{\hat{d}(G)} + \gamma)^{k-1} &= \tilde{O}(\sqrt{N}d(G)^k),\end{aligned}$$

where $\gamma = (k+1)\sqrt{8\log(12N/\beta)}/\varepsilon = \tilde{O}(1)$ and $\hat{d}(G) = \max(d(G), \lceil\gamma^2\rceil)$.

In summary, $\mathcal{M}_{\text{k-pt}}$ still produces an unbiased estimator of the number of k-line paths P_k, while incurring a $\tilde{O}(d(G))$ amplification in error upper bound compared to the k-line walk counting solution. Specifically, the overall error is asymptotically dominated by the sampling error, while the DP noise matches that of walk counting.

Remark. Notably, unbiased estimators are common in LDP settings, especially for RR-based mechanisms, where the final result can be regarded as a sum of independent random variables, and unbiasedness follows directly. However, our mechanism induces correlations among different noise sources, which significantly complicates proving unbiasedness.

Besides, in our mechanism, the sampling error also arises from the privacy aspect, even though it is independent of the privacy budget ε. This is because random marking is not specific to path counting; instead, it addresses an LDP-specific challenge, i.e., eliminating duplicate nodes when each node observes only limited local information. While our theoretical analysis suggests that the sampling error asymptotically dominates the DP noise by a factor of $\tilde{O}(d(G))$, our experiments show that, in practice, the gap is much smaller. This observation indicates that applying random marking offers a near "free lunch" in our mechanism. Moreover, we can optimize the overall error by mitigating this gap. Since both mechanisms $\widehat{\mathcal{M}}_{\text{k-pt}}$ and $\mathcal{M}_{\text{k-pt}}$ produce unbiased estimators, we run the LDP mechanism n_{rep} times with per run privacy budget $\varepsilon/n_{\text{rep}}$ and average the outputs. This reduces the sampling error, and while allocating a smaller per-run privacy budget increases the DP noise, averaging across runs mitigates its impact. We demonstrate the effectiveness of this optimization strategy in Section 7.3.

Optimization in efficiency. Earlier in this section, we required that nodes in rounds inconsistent with their marks report 0. In fact, this step can be omitted, since the computation results depend only on non-zero values. As a result, we instead require that nodes remain silent in such rounds. Moreover, since only nodes with mark ℓ are active in round ℓ, communication can be optimized by having the analyzer $\mathcal{A}$ send $\|X^{(\ell-1)}\|_\infty$ exclusively to those nodes; their outputs are then forwarded only to neighbors whose mark is $\ell + 1$. This requires the analyzer to know each node's mark, which can be achieved by modifying Line 6 of Algorithm 3 so that nodes also send their marks to $\mathcal{A}$.

Figure 2 shows an example of executing the optimized 3-round mechanism $\mathcal{M}_{\text{3-pt}}$ on graph G with $\varepsilon = 1$, showing only the active nodes at each step. The mechanism first initializes the number of 0-line paths ending at each node to 1. After that, each node samples its mark and becomes active only in the corresponding round. Since initialization corresponds to round 0, only outputs of nodes with mark 0 are used; outputs from all other nodes are unused and shown in gray. In the final round, the analyzer aggregates the outputs of all active nodes and rescales the sum to produce the final result 19.2.

Communication and computation analysis. The total communication cost of $\mathcal{M}_{\text{k-pt}}$ is $O(M + N)$. Communication between nodes contributes $O(M)$, as each node sends its mark and noisy result once to all its neighbors. The communication between nodes and the analyzer adds an additional $O(N)$, as each node sends its mark and participates in at most one round of receiving and sending values with the analyzer. In terms of computational cost, the analyzer runs in $O(N)$ time across $k - 2$ rounds of computing the round maximum and one round of aggregating the final result. On the node side, the runtime is $O(d(G))$, as each node participates in one round of aggregating $O(d(G))$ values from its neighbors.

6 LDP Acyclic Pattern Counting

We now move from path counting to the general problem of counting arbitrary acyclic patterns under LDP. To avoid ambiguity, from now on, we refer to the nodes in the graph pattern as vertices,

while continuing to use nodes to refer to those in G. As discussed earlier, a path inherently imposes an ordering on vertices, progressing naturally from the source to the destination. Our path counting mechanisms leverage this property by enumerating paths in a sequential, node-by-node manner. In contrast, acyclic patterns lack such directional structure and thus do not admit a natural vertex order, making it difficult to extend our previously proposed mechanism to this more general problem. To overcome this challenge, we first introduce a method to impose a vertex ordering on arbitrary acyclic patterns. Based on this ordering, we then extend the count aggregation in our LDP path counting mechanism with the idea of recursive subtree counting [51] to support general acyclic patterns. Our results show that, for any acyclic pattern with k edges, the mechanism still produces an unbiased estimation.

6.1 Ordering Vertices in Acyclic Patterns

Although general acyclic patterns lack a fixed natural orientation, they can be formulated as trees, which admit a conceptual bottom-up orientation, progressing from the leaves to the root. Based on this idea, given an acyclic pattern, we first formulate it as a tree by specifying a root, identifying the leaves, and defining parent-child relationships among vertices. Notably, the same pattern may admit multiple valid tree formulations. For example, a k-line path can be represented as (i) a linear tree with one endpoint as the root and the other as the sole leaf, or (ii) a tree folded at a midpoint with an internal root and two branches terminating at the endpoints.

After transforming the acyclic pattern into a tree $\mathcal{T}$, we define an ordering over the vertices in $\mathcal{T}$. To distinguish them from nodes in the graph, we use u to denote vertices in the tree. The ordering can be determined by a standard depth-first search algorithm, in which each parent vertex is visited after all of its children, and children are visited in lexicographic order. We denote the ordered vertices in the tree as $(u_0, u_1, \ldots, u_k)$, where the subscript indicates the order of each vertex, starting from 0. Furthermore, we use $\mathcal{L}$ to denote the set of subscripts of leaf vertices of $\mathcal{T}$. For a vertex u_ℓ, let $\mathcal{T}(u_\ell)$ denote the subtree of $\mathcal{T}$ rooted at u_ℓ. Since there is a correspondence between the tree and the pattern, any subtree $\mathcal{T}(u_\ell)$ is itself an acyclic pattern. Besides, to define the parent-child relationships, let $\mathcal{P}(\ell)$ denote the subscript of u_ℓ's parent and let $\mathcal{C}(\ell)$ denote the set of subscripts of immediate children of u_ℓ. Notably, $\mathcal{T}$ is not necessarily a binary tree; therefore, a vertex may have more than 2 children.

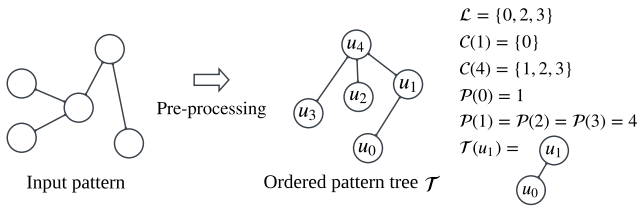


Fig. 3. An example of pre-processing a 4-edge pattern.

Our mechanism can operate on any valid tree formulation and vertex ordering, as the asymptotic error bounds, communication cost, and computational cost depend only on k and are independent of the specific tree structure. Furthermore, since the tree construction depends only on the pattern and not on the graph G, it can be performed entirely as a pre-processing step by the analyzer $\mathcal{A}$, which then publishes the relevant structural information to all nodes before any local computation. Figure 3 illustrates an example of this pre-processing step and the resulting structural information.

6.2 Non-Private Acyclic Pattern Counting

With the vertex ordering of the pattern established, we are now ready to introduce our LDP protocol for acyclic pattern counting. Similar to the case of path counting, we begin by studying the non-private solution. As mentioned earlier, the acyclic pattern is now represented as a tree, and our goal becomes counting the number of occurrences of this tree within the given graph. To achieve this, we adopt a recursive strategy [51]: the count for the entire tree is computed by first counting its subtrees. To align with our LDP setting, we need to adapt this idea to distributed settings, optimize information aggregation, calibrate noise for privacy, and analyze error correlations to derive the target bound. Specifically, we traverse the vertices of the tree in the previously defined bottom-up order, and at each vertex, we aggregate the counting information from the subtrees rooted at its children that are already computed in earlier steps. In the distributed setting, each node in G is responsible for counting the number of subtrees rooted at itself. To do so, it should collect counting information of smaller subtrees rooted at its neighbors. Building on these steps, we further employ the random marking technique to ensure that no node appears more than once in any single pattern instance.

We now present the details of the non-private algorithm. Let $\widehat{\mathcal{X}}_i^{(\ell)}$ denote the output count of node $v_i \in V$ at round ℓ. Initially, for all $\ell \in \mathcal{L}$, the count for leaf vertex u_ℓ is set to 1 at every node v_i, and each node samples a random mark $r_i \in \{0, 1, \ldots, k\}$. In each round $\ell \in \{0, 1, \ldots, k\}$, we focus on counting the subtree $\mathcal{T}(u_\ell)$ at nodes with mark ℓ. At this point, the operations performed depend on whether u_ℓ is a leaf vertex in $\mathcal{T}$:

(1) If u_ℓ is a leaf vertex, no operation is needed, as each node's count is already initialized to 1.
(2) If u_ℓ is an internal vertex with children, node v_i first collects the subtree counts for each child of u_ℓ from its neighbors. More precisely, for each $c \in C(\ell)$, node v_i computes the number of the subtree $\mathcal{T}(c)$ by summing the counts $\widehat{\mathcal{X}}_j^{(c)}$ from its neighbors v_j with mark c. We denote this sum as $\widehat{\mathcal{Y}}_i^{(c)}$. After collecting all subtree counts, v_i computes their product to obtain $\widehat{\mathcal{X}}_i^{(\ell)} = \prod_{c \in C(\ell)} \widehat{\mathcal{Y}}_i^{(c)}$, representing the count of $\mathcal{T}(u_\ell)$ rooted at v_i.

At the end of each round $\ell \notin \mathcal{L}$, nodes with mark ℓ send their counts only to neighbors with mark $\mathcal{P}(\ell)$. In the final round k, active nodes send their counts to the analyzer. The resulting mechanism runs in $k + 2 - |\mathcal{L}|$ rounds. The detailed algorithm for the local client at each node is provided in Algorithm 4.

With the above design, we proactively ensure that each counted acyclic pattern contains no repeated nodes. However, similar to the case of path counting, directly summing the node counts at the final round k leads to an underestimation of the true number of pattern instances $Q_{\mathcal{T}}$. Specifically, a subgraph $H \subseteq G$ that matches the pattern $\mathcal{T}$ is counted only if each of its $k + 1$ nodes receives a mark corresponding to the order of its mapped vertex in $\mathcal{T}$. Thus, the probability that a graph pattern instance is sampled is $(k + 1)^{-(k+1)}$. Therefore, the analyzer $\mathcal{A}$ rescales the aggregated counts by a factor of $(k + 1)^{k+1}$ and outputs:

$$\widehat{\mathcal{M}}_{\mathcal{T}}(G) = (k + 1)^{(k+1)} \sum_{i=1}^{N} \widehat{\mathcal{X}}_i^{(k)} \mathbf{1}\{r_i = k\}.$$

Lemma 6.1. *For any graph G, k-edge pattern $\mathcal{T}$, and parameter β, the $(k+2-|\mathcal{L}|)$-round non-private algorithm $\widehat{\mathcal{M}}_{\mathcal{T}}$ satisfies $\mathbb{E}[\widehat{\mathcal{M}}_{\mathcal{T}}(G)] = Q_{\mathcal{T}}$, and with probability at least $1 - \beta$,*

$$|\widehat{\mathcal{M}}_{\mathcal{T}}(G) - Q_{\mathcal{T}}| \leq (k + 1)\sqrt{N\beta^{-1}}(d(G) + k + 1)^k = O(\sqrt{N} d(G)^k).$$

Algorithm 4: Local client $\widehat{\mathcal{R}}^{(\ell)}$ of $\widehat{\mathcal{M}}_{\mathcal{T}}$

1 **foreach** $i \in \{1, \ldots, N\}$ **do**
2 **foreach** $\ell \in \mathcal{L}$ **do**
3 $\widehat{\mathcal{X}}_i^{(\ell)} \leftarrow 1$
4 **Function** $\widehat{\mathcal{R}}^{(\ell)}(G_i, \mathcal{T})$**:**
5 **if** $\ell = init$ **then**
6 $r_i \leftarrow \mathsf{Unif}(0, k)$
7 **Send out** r_i to $\mathcal{N}(i)$ and $\mathcal{A}$
8 **if** $\ell \in \mathcal{L}$ *or* $\ell \neq r_i$ **then**
9 **Skip**
10 **if** $\ell = r_i$ **then**
11 Received $\widehat{\mathcal{X}}_j^{(c)}$ for each $j \in \mathcal{N}(i)$ and $c \in C(\ell)$
12 **foreach** $c \in C(\ell)$ **do**
13 $\widehat{\mathcal{Y}}_i^{(c)} \leftarrow \sum_{j \in \mathcal{N}(i)} \widehat{\mathcal{X}}_j^{(c)} \mathbf{1}\{r_j = c\}$
14 $\widehat{\mathcal{X}}_i^{(\ell)} \leftarrow \prod_{c \in C(\ell)} \widehat{\mathcal{Y}}_i^{(c)}$
15 **Send out** $\widehat{\mathcal{X}}_i^{(\ell)}$ to $\mathcal{N}(i)$ with mark $\mathcal{P}(\ell)$ and $\mathcal{A}$

6.3 Integrating DP and Efficiency Optimization

We now discuss how to privatize $\widehat{\mathcal{M}}_{\mathcal{T}}(G)$ by applying the Laplace mechanism to each node's computation. As previously mentioned, each node sends its result to its neighbors in at most one round, and a neighbor uses the received count only if it operates in a later round. Hence, the presence of any edge can affect the computation of at most one node. By parallel composition (Lemma 3.4), the privacy budget ε can be assigned to each node v_i at its active round without division. The problem then reduces to ensuring each node's output satisfies ε-edge-DP. Since nodes with marks in $\mathcal{L}$ have a fixed output of 1, we focus only on nodes with marks outside $\mathcal{L}$. As shown in Algorithm 4, an active node v_i at round ℓ aggregates outputs from neighbors in each child round c to compute the sum $\widehat{\mathcal{Y}}_i^{(c)}$. Similar to path counting, any edge from a node with mark c can influence this sum by at most $\|\mathcal{X}^{(c)}\|_\infty$, a public value computed by the analyzer $\mathcal{A}$. Since each edge contributes to at most one such sum across all child rounds $c \in C(\ell)$, the privacy budget ε need not be divided across them. Accordingly, we modify Line 13 of Algorithm 4 by adding Laplace noise of scale $\|\mathcal{X}^{(c)}\|_\infty/\varepsilon$, resulting in the private local randomizer $\mathcal{R}^{(\ell)}$. As in the path counting case, sampling amplification does not apply here either. The resulting edge-LDP mechanism, denoted as $\mathcal{M}_{\mathcal{T}}$, satisfies the following privacy and utility guarantees:

Theorem 6.2. *For any graph G, k-edge pattern $\mathcal{T}$, and privacy budget ε, the mechanism $\mathcal{M}_{\mathcal{T}}$ satisfies ε-edge-LDP.*

For utility, the total error consists of the sampling error shown in Lemma 6.1, and the DP noise established below.

Lemma 6.3. *For any graph G, k-edge pattern $\mathcal{T}$, parameters ε and β, $\mathcal{M}_{\mathcal{T}}$ satisfies $\mathbb{E}[\mathcal{M}_{\mathcal{T}}(G, \varepsilon)] = \widehat{\mathcal{M}}_{\mathcal{T}}(G)$ under the same node marking $(r_1, \ldots, r_N)$, and with probability at least $1 - \beta$,*

$$|\mathcal{M}_{\mathcal{T}}(G, \varepsilon) - \widehat{\mathcal{M}}_{\mathcal{T}}(G)| \leq k(k+1)\gamma\sqrt{N}(\hat{d}(G) + \gamma\sqrt{\hat{d}(G) + \gamma})^{k-1}$$
$$= \tilde{O}(\sqrt{N} d(G)^{k-1}),$$

where $\gamma = (k+1)\sqrt{8\log(6kN/\beta)}/\varepsilon = \tilde{O}(1)$ *and* $\hat{d}(G) = \max(d(G), \lceil\gamma^2\rceil)$.

By combining Lemma 6.1 and Lemma 6.3, we obtain the overall utility guarantee:

Theorem 6.4. *For any graph G, k-edge pattern $\mathcal{T}$, parameters ε and β, the $(k+2-|\mathcal{L}|)$-round mechanism $\mathcal{M}_{\mathcal{T}}$ satisfies $\mathbb{E}[\mathcal{M}_{\mathcal{T}}(G,\varepsilon)] = Q_{\mathcal{T}}$, and with probability at least $1-\beta$,*

$$|\mathcal{M}_{\mathcal{T}}(G,\varepsilon) - Q_{\mathcal{T}}| \le (k+1)\sqrt{2N\beta^{-1}}(d(G)+k+1)^k +$$
$$k(k+1)\,\gamma\sqrt{N}(\hat{d}(G) + \gamma\sqrt{\hat{d}(G)} + \gamma)^{k-1} = \tilde{O}(\sqrt{N}d(G)^k),$$

where $\gamma = (k+1)\sqrt{8\log(12kN/\beta)}/\varepsilon = \tilde{O}(1)$ *and* $\hat{d}(G) = \max(d(G), \lceil\gamma^2\rceil)$.

To conclude, $\mathcal{M}_{\mathcal{T}}$ achieves the same asymptotic utility guarantee as $\mathcal{M}_{\text{k-pt}}$ while preserving unbiasedness of the output. Consequently, the same LDP error optimization strategy, i.e., averaging the LDP results across multiple runs, also applies to our general acyclic pattern counting mechanism.

Communication and computation analysis. The total communication cost of $\mathcal{M}_{\mathcal{T}}$ is $O(M+N)$, where $O(M)$ arises from inter-node communication and $O(N)$ from communication between the analyzer and nodes. The analyzer requires $O(N)$ computation time, where pattern pre-processing takes constant time. Moreover, each node performs $O(d(G))$ local computation.

Special design for k-star counting. In k-star counting, node duplication can be handled locally by each star center, allowing us to skip the random marking process. For improved performance, we propose a one-round mechanism $\mathcal{M}_{k\star}$ tailored to this case. Specifically, each node v_i first privatizes its degree by adding Laplace noise with scale $2/\varepsilon$, yielding $\tilde{d}(v_i) = d(v_i) + \text{Lap}(2/\varepsilon)$. It then computes the product $\tilde{d}(v_i)(\tilde{d}(v_i)-1)\cdots(\tilde{d}(v_i)-k+1)$ and sends the result to the analyzer $\mathcal{A}$, which aggregates all received values and outputs their sum. To conclude, $\mathcal{M}_{k\star}$ satisfies ε-LDP while achieving the SOTA utility of $\tilde{O}(\sqrt{N}d(G)^{k-1})$ [30], with the utility proof deferred to Appendix A.3 of the full version [29].

7 Experiment

We conduct experiments on four tasks under edge-LDP: k-line walk counting, k-line path counting, k-edge acyclic pattern counting, and k-star counting. For the first three tasks, we compare with the RR-based method from [48], denoted as RR_{enum}. In particular, for k-line walk counting, we additionally compare against the method introduced in [4], denoted as WalkClip$_k$. For k-star counting, we compare against LocalLap$_{k\star}$ [30].

7.1 Setup

Dataset	Nodes	Edges	Max Degree	Ave. Degree
AstroPh	18,772	198,110	504	21.1
Enron	36,692	138,831	1,383	7.6
Epinions1	75,879	405,740	3,044	10.7
Slashdot	77,360	546,487	2,541	14.1
Twitter	81,306	1,342,310	3,383	33.0
Epinions2	131,828	711,783	3,558	10.8

Table 2. Summary of datasets used in the experiments.

Datasets. We evaluate our mechanisms on six real-world graph datasets from SNAP [40], summarized in Table 2. **AstroPh** is a co-authorship network in astrophysics, **Enron** is an email communication network, **Twitter** and **Slashdot** are social networks derived from their respective platforms, and **Epinions1** and **Epinions2** represent a trust network and a social network from Epinions. All graphs were preprocessed to be undirected and free of self-loops.

Queries. For k-line walk and path counting tasks, we evaluate cases with $k \in \{4, 5, 6\}$. The corresponding walk counting queries are denoted as $Q_{4\text{-wk}}$, $Q_{5\text{-wk}}$, and $Q_{6\text{-wk}}$, and the path counting queries as $Q_{4\text{-pt}}$, $Q_{5\text{-pt}}$, and $Q_{6\text{-pt}}$. For k-edge acyclic pattern counting, we consider three representative patterns, denoted π_1, π_2, and π_3, with edge counts $k = 4$, $k = 5$, and $k = 6$, respectively. The associated queries are denoted by Q_{π_1}, Q_{π_2}, and Q_{π_3}. For the first two patterns, we explore two distinct tree structures as pre-processing results: $\mathcal{T}_1$ and $\mathcal{T}_1'$ for π_1, and $\mathcal{T}_2$ and $\mathcal{T}_2'$ for π_2. For the third pattern, we consider a single tree structure $\mathcal{T}_3$. The structural details of the patterns and their corresponding trees are illustrated in Figure 4. Furthermore, we evaluated three cases of k-star counting with $k \in \{3, 4, 5\}$, denoted as queries $Q_{3\star}$, $Q_{4\star}$, and $Q_{5\star}$. All queries are evaluated without redundant counts in the results.

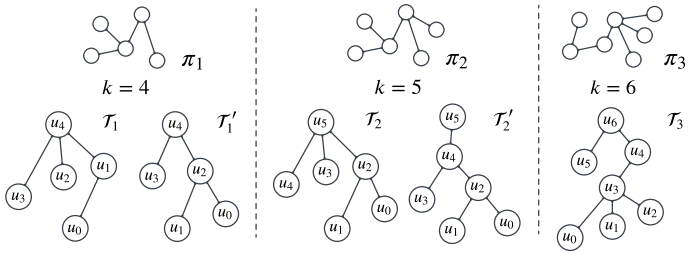


Fig. 4. Selected input patterns and their tree formulations with vertex orderings.

Experimental parameters. All experiments are conducted on a Linux server with dual 64-core AMD EPYC 9555 CPUs and 2 TB RAM. Each experiment is repeated 10 times. To evaluate utility, we discard the highest and lowest 2 results and report the average of the remaining 6 runs. Communication cost is reported as the average over all 10 runs. We set the default privacy budget to $\varepsilon = 1$. All evaluated methods, including our parameter-free mechanisms, take only the privacy budget ε and the query pattern as inputs, with no additional hyperparameters or tuning. For each query, all methods are run under identical settings to ensure fair comparison. As $d(G)$ is assumed to be unknown in the experiments, both WalkClip$_k$ and LocalLap$_{k\star}$ require an additional round to estimate the noisy maximum degree $\tilde{d}(G)$. For these two methods, we allocate a privacy budget of $\varepsilon/10$ to the estimation step, and the remaining $9\varepsilon/10$ to the main counting algorithm. Following [4], we set the clipping factor[3] in WalkClip$_k$ as $\sqrt{\tilde{d}(G)}$.

7.2 Experiment Results

Utility and communication cost. The experimental results, including relative errors and communication costs for all four tasks, are summarized in Table 3 and Table 4. Across all 72 query-dataset combinations, our mechanism consistently outperforms the baseline methods in both utility and communication cost.

[3]The clipping threshold at round ℓ is defined as this factor raised to the power of ℓ.

Query	Method	Metric	AstroPh	Enron	Epinions1	Slashdot	Twitter	Epinions2
$Q_{4\text{-wk}}$	$\mathcal{M}_{4\text{-wk}}$	Rel. err (%)	1.48	1.82	1.18	1.75	1.22	0.91
		Comm.	6.76 MB	7.01 MB	15.28 MB	17.27 MB	44.07 MB	26.72 MB
	WalkClip_4	Rel. err (%)	98.97	98.68	98.70	97.17	97.92	99.34
		Comm.	10.50 GB	40.13 GB	171.60 GB	178.36 GB	197.02 GB	515.98 GB
	RR_{enum}	Rel. err (%)	572.86	929.78	733.06	1719.88	308.38	687.89
		Comm.†	168.00 MB	641.95 MB	2745.41 MB	2853.63 MB	3152.17 MB	8255.56 MB
$Q_{5\text{-wk}}$	$\mathcal{M}_{5\text{-wk}}$	Rel. err (%)	2.27	2.30	1.72	2.80	1.49	0.88
		Comm.	10.07 MB	10.37 MB	22.63 MB	25.61 MB	65.79 MB	39.58 MB
	WalkClip_5	Rel. err (%)	99.76	99.59	99.61	98.88	99.35	99.84
		Comm.	13.13 GB	50.16 GB	214.49 GB	222.95 GB	246.27 GB	644.98 GB
	RR_{enum}	Rel. err (%)	1854.25	3269.04	2346.32	8122.89	1027.45	2208.09
$Q_{6\text{-wk}}$	$\mathcal{M}_{6\text{-wk}}$	Rel. err (%)	5.12	7.15	2.22	5.37	2.54	1.63
		Comm.	13.38 MB	13.74 MB	29.97 MB	33.95 MB	87.51 MB	52.44 MB
	WalkClip_6	Rel. err (%)	99.95	99.88	99.89	99.58	99.81	99.96
		Comm.	15.75 GB	60.19 GB	257.39 GB	267.54 GB	295.53 GB	773.98 GB
	RR_{enum}	Rel. err (%)	6870.55	13021.01	8660.78	42530.83	3771.45	8179.33
$Q_{4\text{-pt}}$	$\mathcal{M}_{4\text{-pt}}$	Rel. err (%)	5.67	11.47	5.95	14.02	8.91	8.79
		Comm.	3.55 MB	3.59 MB	7.84 MB	8.92 MB	23.35 MB	13.76 MB
	RR_{enum}	Rel. err (%)	477.46	791.34	600.68	1448.47	254.75	558.30
$Q_{5\text{-pt}}$	$\mathcal{M}_{5\text{-pt}}$	Rel. err (%)	11.77	5.95	11.55	10.23	4.13	9.55
		Comm.	3.58 MB	3.64 MB	7.96 MB	9.02 MB	23.54 MB	13.92 MB
	RR_{enum}	Rel. err (%)	1350.34	2400.53	1655.97	5877.49	733.03	1538.58
$Q_{6\text{-pt}}$	$\mathcal{M}_{6\text{-pt}}$	Rel. err (%)	6.81	19.29	15.62	9.44	6.70	6.84
		Comm.	3.60 MB	3.68 MB	8.02 MB	9.09 MB	23.59 MB	14.03 MB
	RR_{enum}	Rel. err (%)	4115.66	7936.65	4977.07	25456.53	2209.38	4634.36
Q_{π_1}	$\mathcal{M}_{\mathcal{T}_1}$	Rel. err (%)	12.10	21.90	11.71	27.54	15.16	17.38
		Comm.	3.37 MB	3.37 MB	7.36 MB	8.39 MB	22.29 MB	12.89 MB
	RR_{enum}	Rel. err (%)	256.42	198.42	178.08	368.72	65.25	166.18
Q_{π_2}	$\mathcal{M}_{\mathcal{T}_2}$	Rel. err (%)	11.53	23.27	37.30	48.14	18.64	17.24
		Comm.	3.32 MB	3.30 MB	7.23 MB	8.25 MB	21.96 MB	12.67 MB
	RR_{enum}	Rel. err (%)	448.55	200.03	209.82	619.40	70.76	178.98
Q_{π_3}	$\mathcal{M}_{\mathcal{T}_3}$	Rel. err (%)	18.74	48.23	23.81	77.08	28.20	43.45
		Comm.	3.39 MB	3.40 MB	7.44 MB	8.46 MB	22.39 MB	13.00 MB
	RR_{enum}	Rel. err (%)	877.11	273.41	205.49	547.55	54.35	232.20

Table 3. Relative error (%) and communication cost (MB/GB) for k-line walk, k-line path, and k-edge acyclic pattern counting queries across six datasets. †Communication cost of RR_{enum} remains constant across queries.

For k-line walk counting queries, our mechanism achieves a relative error of less than 8% on all datasets, with generally better performance for smaller k. In comparison, the WalkClip_k method consistently yields high relative errors due to significant clipping bias, as the clipping factor leads to the truncation of the majority of walks in large graphs. Regarding communication, results confirm that our optimization delivers substantial savings over the baselines. In summary, across all tested walk counting queries, our mechanisms achieve up to a 110× reduction in error and a 300× reduction in communication cost compared with the baseline methods.

For k-line path counting queries, the utility of our mechanism shows a slight drop compared to the k-line walk queries of the same k, due to the involvement of sampling error. Nonetheless,

Query	Method	Metric	AstroPh	Enron	Epinions1	Slashdot	Twitter	Epinions2
$Q_{3\star}$	$\mathcal{M}_{3\star}$	Rel. err (%)	0.17	0.09	0.05	0.06	0.04	0.04
		Comm.†	0.14 MB	0.28 MB	0.58 MB	0.59 MB	0.62 MB	1.00 MB
	$\text{LocalLap}_{3\star}$	Rel. err (%)	6.79	8.16	19.52	16.47	5.64	10.62
		Comm.†	0.43 MB	0.84 MB	1.74 MB	1.77 MB	1.86 MB	3.01 MB
$Q_{4\star}$	$\mathcal{M}_{4\star}$	Rel. err (%)	0.49	0.19	0.15	0.06	0.09	0.09
	$\text{LocalLap}_{4\star}$	Rel. err (%)	33.88	16.69	43.04	42.78	16.55	35.67
$Q_{5\star}$	$\mathcal{M}_{5\star}$	Rel. err (%)	0.50	0.22	0.18	0.14	0.11	0.09
	$\text{LocalLap}_{5\star}$	Rel. err (%)	38.58	24.81	82.03	53.66	26.92	37.21

Table 4. Relative error (%) and communication cost (MB) for k-star queries on six datasets. †Communication cost of both methods remains constant across star counting queries.

it offers lower communication cost and maintains a significant advantage over the baseline: we achieve up to a 2600× error reduction and a 600× communication cost improvement across all queries.

For k-edge acyclic pattern counting queries, our mechanism outperforms the baseline method by up to 46× in error and 650× in communication cost. In general, our mechanism achieves higher utility for smaller acyclic patterns. Notably, the communication cost remains stable across different values of k, demonstrating the effectiveness of our optimization strategy.

For k-star counting queries, our mechanism achieves a 3× reduction in communication cost, as it does not depend on information about the maximum degree. This enables each node to complete its interaction with the analyzer in a single round of communication. In addition to its communication efficiency, our mechanism yields up to a 710× error reduction, while consistently maintaining a relative error below 0.5% across all datasets. This demonstrates the robustness and utility of our method.

Privacy budget ε. To evaluate the impact of the privacy budget ε on performance, we assess queries $Q_{5\text{-wk}}$, $Q_{5\text{-pt}}$, Q_{π_1}, and $Q_{4\star}$ on the **AstroPh** dataset, varying ε uniformly from 0.2 to 4.0 in steps of 0.2. Relative errors are shown in Figure 5. Notably, communication cost remains independent of ε. As illustrated, all methods generally improve in utility as ε increases, with our mechanism consistently achieving order-of-magnitude lower error across the full range. WalkClip_5, however, maintains a stable error, as its performance is bottlenecked by the clipping bias rather than the DP noise. For $Q_{5\text{-pt}}$ and Q_{π_1}, the error of our mechanism stabilizes when $\varepsilon \geq 1.0$, as the sampling error, which is independent of ε, becomes dominant over the diminishing DP error.

Tree structure $\mathcal{T}$. In Section 6, we establish that the pre-processing tree structure of the input pattern does not affect the asymptotic error bounds or communication cost for k-edge acyclic pattern counting queries. To empirically validate this, we evaluate alternative tree structures $\mathcal{T}_1'$ and $\mathcal{T}_2'$ for patterns π_1 and π_2 (Figure 4), with resulting mechanisms $\mathcal{M}_{\mathcal{T}_1'}$ and $\mathcal{M}_{\mathcal{T}_2'}$ that execute in 3 and 4 rounds. Evaluation results on the **Enron**, **Epinions1**, and **Twitter** datasets are presented in Table 5.

Comparing the two tree structures for each pattern counting query, while utility differences exist, the variations remain bounded within a constant factor. This aligns with our theoretical conclusion that the asymptotic error bound is independent of the tree structure. Furthermore, the communication cost remains stable, even for π_2, where $\mathcal{M}_{\mathcal{T}_2'}$ requires an additional round compared to $\mathcal{M}_{\mathcal{T}_2}$. This is because the dominant communication overhead stems from publishing

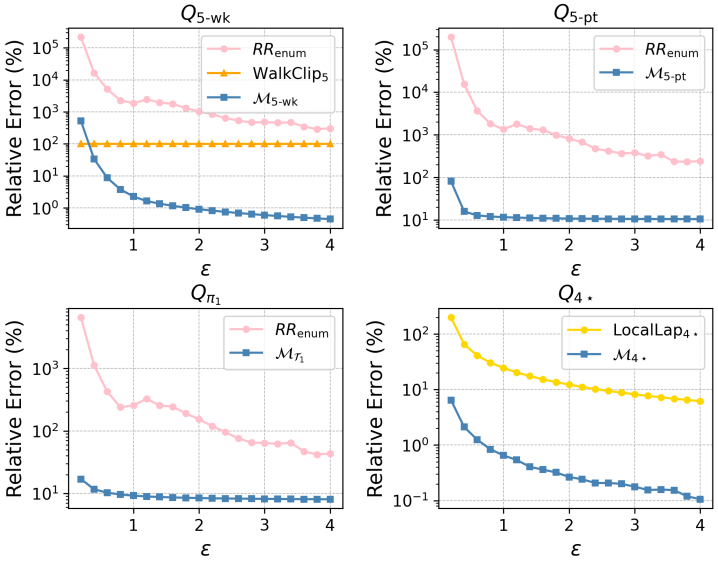


Fig. 5. Relative error (%) for $Q_{5\text{-wk}}$, $Q_{5\text{-pt}}$, Q_{π_1}, and $Q_{4\star}$ on the **AstroPh** dataset under varying ε values.

Query	Method	Metric	Enron	Epinions1	Twitter
Q_{π_1}	$\mathcal{M}_{\mathcal{T}_1}$	Rel. err (%)	21.90	11.71	15.16
		Comm.	3.37 MB	7.36 MB	22.29 MB
	$\mathcal{M}_{\mathcal{T}_1'}$	Rel. err (%)	30.66	21.40	14.69
		Comm.	3.36 MB	7.36 MB	22.30 MB
Q_{π_2}	$\mathcal{M}_{\mathcal{T}_2}$	Rel. err (%)	23.27	37.30	18.64
		Comm.	3.32 MB	7.23 MB	21.96 MB
	$\mathcal{M}_{\mathcal{T}_2'}$	Rel. err (%)	23.42	28.70	18.37
		Comm.	3.47 MB	7.60 MB	22.71 MB

Table 5. Relative error (%) and communication cost (MB) for acyclic pattern queries Q_{π_1} and Q_{π_2} using different tree structures on three datasets.

the randomized marking results, where all nodes participate, while later-round computations involve only subsets of nodes and contribute comparatively less to the total cost.

7.3 Sampling Error vs. DP Noise

In Sections 5 and 6, our theoretical analysis demonstrates that, for both k-line path and k-edge acyclic pattern counting tasks, the sampling error asymptotically dominates the overall error. This conclusion is empirically supported by the error decomposition results on all six datasets, presented in Table 6 in Appendix A.2 of the full version [29]. Notably, although our theoretical results imply an asymptotic gap of $\tilde{O}(d(G))$ between the sampling error and the DP noise, this gap is much smaller than $d(G)$ in practice.

Additionally, we empirically evaluate the LDP error optimization strategy in Section 5.2 on the **Epinions2** dataset using queries $Q_{4\text{-pt}}$, $Q_{5\text{-pt}}$, Q_{π_1}, and Q_{π_2}, varying n_{rep} from 1 to 10. Figure 6 shows the relative sampling error, DP error, and total error for Q_{π_1}. Results for all four queries are

provided in Figure 7 in Appendix A.2 of the full version [29]. As illustrated in Figure 6, sampling error decreases while DP error increases with larger n_{rep}, with total error minimized when the two are roughly balanced. This strategy yields 2.8×, 1.8×, 4.6×, and 2.0× improvements in total error at the optimal n_{rep} values for queries $Q_{4\text{-pt}}$, $Q_{5\text{-pt}}$, Q_{π_1}, and Q_{π_2}, respectively, on this dataset.

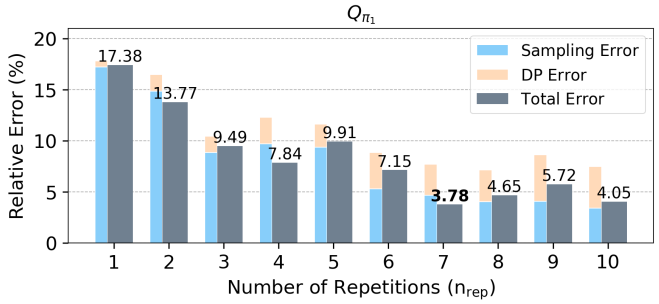


Fig. 6. Relative error decomposition for Q_{π_1} on the **Epinions2** dataset under varying numbers of repetitions (n_{rep}).

8 Future Work

A natural direction for future work is to extend our approach to cyclic patterns, which present additional challenges. Unlike acyclic patterns, cyclic patterns do not allow recursive decomposition, so their counts cannot always be derived by aggregating subpattern counts, making our multi-round aggregation strategy inapplicable. A possible workaround is to break cycles and reduce the problem to acyclic patterns with additional node constraints. For example, triangle counting can be reduced to counting 2-line paths whose endpoints are connected, but this introduces significant utility challenges, as it requires tracking both endpoints of each counted path while preserving privacy. Another promising extension is to support node-LDP, which offers stronger privacy guarantees than edge-LDP by protecting the presence of each individual node along with all its incident edges.

Acknowledgments

This research is supported by the NTU–NAP Startup Grant (024584-00001), the Singapore Ministry of Education Tier 1 Grant (RG19/25), and by the National Research Foundation, Singapore, and the Cyber Security Agency of Singapore under the National Cybersecurity R&D Programme and the CyberSG R&D Programme Office (Award CRPO-GC3-NTU-001). Any opinions, findings, conclusions, or recommendations expressed in these materials are those of the author(s) and do not reflect the views of the National Research Foundation, Singapore, the Cyber Security Agency of Singapore, or the CyberSG R&D Programme Office. We would also like to thank the anonymous reviewers who have made valuable suggestions to improve this paper.

References

[1] Mahmoud Abo Khamis, Hung Q Ngo, and Atri Rudra. 2016. FAQ: questions asked frequently. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 13–28.

[2] Leman Akoglu and Christos Faloutsos. 2013. Anomaly, event, and fraud detection in large network datasets. In *Proceedings of the sixth ACM international conference on Web search and data mining*. 773–774.

[3] Noga Alon, Raphael Yuster, and Uri Zwick. 1995. Color-coding. *Journal of the ACM (JACM)* 42, 4 (1995), 844–856.

[4] Louis Betzer, Vorapong Suppakitpaisarn, and Quentin Hillebrand. 2024. Publishing number of walks and katz centrality under local differential privacy. In *The 40th Conference on Uncertainty in Artificial Intelligence*.

[5] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. 2013. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*. 87–96.

[6] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.

[7] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. 2011. Private and Continual Release of Statistics. *ACM Transactions on Information and System Security* (2011).

[8] Pafnutii Lvovich Chebyshev. 1867. Des valeurs moyennes. *J. Math. Pures Appl* 12, 2 (1867), 177–184.

[9] Shixi Chen and Shuigeng Zhou. 2013. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 653–664.

[10] Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang. 2018. Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 international conference on management of data*. 1655–1658.

[11] Benjamin Doerr. 2019. Probabilistic tools for the analysis of randomized optimization heuristics. In *Theory of evolutionary computation: Recent developments in discrete optimization*. Springer, 1–87.

[12] Wei Dong, Zijun Chen, Qiyao Luo, Elaine Shi, and Ke Yi. 2024. Continual observation of joins under differential privacy. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–27.

[13] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. 2022. R2T: Instance-optimal Truncation for Differentially Private Query Evaluation with Foreign Keys. In *Proc. ACM SIGMOD International Conference on Management of Data*.

[14] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. 2024. Instance-optimal Truncation for Differentially Private Query Evaluation with Foreign Keys. *ACM Transactions on Database Systems* 49, 4 (2024), 1–40.

[15] Wei Dong, Qiyao Luo, and Ke Yi. 2023. Continual Observation under User-level Differential Privacy. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2190–2207.

[16] Wei Dong and Ke Yi. 2021. Residual Sensitivity for Differentially Private Multi-Way Joins. In *Proc. ACM SIGMOD International Conference on Management of Data*.

[17] Wei Dong and Ke Yi. 2022. A Nearly Instance-optimal Differentially Private Mechanism for Conjunctive Queries. In *Proc. ACM Symposium on Principles of Database Systems*.

[18] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[19] Talya Eden, Quanquan C Liu, Sofya Raskhodnikova, and Adam Smith. 2023. Triangle Counting with Local Edge Differential Privacy. In *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 52–1.

[20] Juanru Fang, Wei Dong, and Ke Yi. 2022. Shifted Inverse: A General Mechanism for Monotonic Functions under User Differential Privacy. (2022).

[21] Hendrik Fichtenberger, Monika Henzinger, and Wolfgang Ost. 2021. Differentially Private Algorithms for Graphs Under Continual Observation. In *29th Annual European Symposium on Algorithms (ESA 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[22] Jörg Flum and Martin Grohe. 2004. The parameterized complexity of counting problems. *SIAM J. Comput.* 33, 4 (2004), 892–922.

[23] Chris Godsil and Gordon F Royle. 2013. *Algebraic graph theory*. Vol. 207. Springer Science & Business Media.

[24] Georg Gottlob, Nicola Leone, and Francesco Scarcello. 2001. The complexity of acyclic conjunctive queries. *Journal of the ACM (JACM)* 48, 3 (2001), 431–498.

[25] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. 2009. Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 169–178.

[26] Yizhang He, Kai Wang, Wenjie Zhang, Xuemin Lin, Wei Ni, and Ying Zhang. 2024. Butterfly Counting over Bipartite Graphs with Local Differential Privacy. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2351–2364.

[27] Yizhang He, Kai Wang, Wenjie Zhang, Xuemin Lin, Ying Zhang, and Wei Ni. 2025. Robust Privacy-Preserving Triangle Counting under Edge Local Differential Privacy. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–26.

[28] Yihua Hu, Hao Ding, and Wei Dong. 2025. N2E: A General Framework to Reduce Node-Differential Privacy to Edge-Differential Privacy for Graph Analytics. *Proceedings of the ACM on Management of Data* 3, 6 (2025), 1–26.

[29] Yihua Hu, Kuncan Wang, and Wei Dong. 2026. Acyclic Graph Pattern Counting under Local Differential Privacy [Full Version]. (2026). https://drive.google.com/drive/folders/1L_4OA9tuXksSGel6iqcZxpzarLNc4-k8?usp=sharing

[30] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2021. Locally Differentially Private Analysis of Graph Statistics. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 983–1000.

[31] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2022. Communication-Efficient Triangle Counting under Local Differential Privacy. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 537–554.

[32] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2022. Differentially private triangle and 4-cycle counting in the shuffle model. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 1505–1519.

[33] Shalev Itzkovitz and Uri Alon. 2005. Subgraphs and network motifs in geometric networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 71, 2 (2005), 026117.

[34] Noah Johnson, Joseph P Near, and Dawn Song. 2018. Towards practical differential privacy for SQL queries. *Proceedings of the VLDB Endowment* 11, 5 (2018), 526–539.

[35] Ivan Jokić and Piet Van Mieghem. 2022. Number of paths in a graph. *arXiv preprint arXiv:2209.08840* (2022).

[36] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. 2011. Private analysis of graph structure. *Proceedings of the VLDB Endowment* 4, 11 (2011), 1146–1157.

[37] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2011. What can we learn privately? *SIAM J. Comput.* 40, 3 (2011), 793–826.

[38] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing graphs with node differential privacy. In *Theory of Cryptography Conference*. Springer, 457–476.

[39] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (1953), 39–43.

[40] Jure Leskovec and Andrej Krevl. 2014. SNAP: Stanford network analysis project.

[41] David Liben-Nowell and Jon Kleinberg. 2003. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*. 556–559.

[42] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.

[43] Arvind Narayanan and Vitaly Shmatikov. 2009. De-anonymizing social networks. In *2009 30th IEEE symposium on security and privacy*. IEEE, 173–187.

[44] Hung Q Ngo, Ely Porat, Christopher Ré, and Atri Rudra. 2018. Worst-case optimal join algorithms. *Journal of the ACM (JACM)* 65, 3 (2018), 1–40.

[45] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 75–84.

[46] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *nature* 435, 7043 (2005), 814–818.

[47] Haipei Sun, Xiaokui Xiao, Issa Khalil, Yin Yang, Zhan Qin, Hui Wang, and Ting Yu. 2019. Analyzing subgraph statistics from extended local views with decentralized differential privacy. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 703–717.

[48] Vorapong Suppakitpaisarn, Donlapark Ponnoprat, Nicha Hirankarn, and Quentin Hillebrand. 2025. Counting Graphlets of Size k under Local Differential Privacy. *arXiv preprint arXiv:2505.12954* (2025).

[49] Stanley L Warner. 1965. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American statistical association* 60, 309 (1965), 63–69.

[50] Bang Wu, Xiangwen Yang, Shirui Pan, and Xingliang Yuan. 2021. Adapting membership inference attacks to GNN for graph classification: Approaches and implications. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1421–1426.

[51] Mihalis Yannakakis. 1981. Algorithms for acyclic database schemes. In *VLDB*, Vol. 81. 82–94.

[52] Qingqing Ye, Haibo Hu, Man Ho Au, Xiaofeng Meng, and Xiaokui Xiao. 2020. LF-GDPR: A framework for estimating graph metrics with local differential privacy. *IEEE Transactions on Knowledge and Data Engineering* 34, 10 (2020), 4905–4920.

[53] Qingqing Ye, Haibo Hu, Man Ho Au, Xiaofeng Meng, and Xiaokui Xiao. 2020. Towards locally differentially private generic graph metric estimation. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1922–1925.

[54] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2015. Private release of graph statistics using ladder functions. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 731–745.

[55] Zhikun Zhang, Min Chen, Michael Backes, Yun Shen, and Yang Zhang. 2022. Inference attacks against graph neural networks. In *31st USENIX Security Symposium (USENIX Security 22)*. 4543–4560.

Received October 2025; revised January 2026; accepted February 2026