USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

Distributed Private Aggregation in Graph Neural Networks

Huanhuan Jia, Yuanbo Zhao, Kai Dong, Zhen Ling, Ming Yang, and Junzhou Luo, *Southeast University;* Xinwen Fu, *University of Massachusetts Lowell*

https://www.usenix.org/conference/usenixsecurity25/presentation/jia-huanhuan

This paper is included in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

Distributed Private Aggregation in Graph Neural Networks

Huanhuan Jia[†] *Southeast University*
Yuanbo Zhao[†] *Southeast University*
Kai Dong[*] *Southeast University*
Zhen Ling *Southeast University*
Ming Yang *Southeast University*
Junzhou Luo *Southeast University*
Xinwen Fu *University of Massachusetts Lowell*

Abstract

Graph Neural Networks (GNNs) have shown considerable promise in handling graph-structured data, yet their use is restricted in privacy-sensitive environments, especially in distributed settings. In this setting, current methods for preserving privacy in GNNs often rely on unrealistic assumptions or fail to construct effective models. In response, this paper introduces Distributed Private Aggregation (DPA), a pioneering GNN aggregation method which is built upon Secure Multi-Party Computation protocols, and is designed to ensure node-level differential privacy. We implement DPA-GNN, which to our knowledge, is the most effective privacy-preserving GNN model suitable for distributed contexts. Through extensive experiments on six real-world datasets, DPA-GNN has proven to consistently surpass existing privacy preserving GNNs, offering an optimal balance between privacy and utility.

1 INTRODUCTION

Graph Neural Networks (GNNs) have demonstrated outstanding capabilities in processing graph-structured data, but their application is limited in scenarios involving private information [6, 23, 32, 35]. In the real world, there is a vast amount of graph data, where a graph is composed of nodes (e.g., users in a social network) and edges (e.g., relationships). GNN employs a unique network structure, which aggregates the information of each node's neighbors together to represent the node itself. This process is called the GNN **aggregation**. However, node information and neighboring relationships are often private, e.g., some relationships in social networks [19] and salary information in financial systems [47]. The deployment and usage of a GNN constructed on such data poses a severe privacy risk. Consequently, privacy-preserving GNNs [11, 20, 21] have garnered widespread attention in the community.

To protect privacy, Differential Privacy (DP) [12] is a rigorous privacy protection standard that introduces controlled noise to data. The noise is added by a trustworthy server (i.e., in the centralized setting), to preserve individual privacy while still allowing for valuable data analysis. This noise is adjusted through a parameter known as the **privacy budget**. The smaller the privacy budget, the greater the noise and the higher the level of privacy. In graph-structured data, DP is applied at two levels: **edge-level DP** and **node-level DP** [36]. Edge-level DP focuses primarily on the privacy of edges between nodes [28, 48], and node-level DP is a more powerful constraint, as it must account all information of the nodes. Consider a GNN that is used for a classification downstream task, the information of a node contains its labels, features and edges (i.e., connections to its neighboring nodes). All these information are considered private in node-level DP [10, 51, 57].

To the best of our knowledge, no existing techniques can be applied to preserve **node-level privacy in distributed settings**, where an untrustworthy server needs to construct a GNN among private information of distributed users. Existing research preserves privacy defined on some certain information (e.g., either edges or features) of individual nodes by utilizing **Local Differential Privacy (LDP)** (which is a variant of DP but in distributed settings). However they still fail to achieve node-level privacy since they cannot preserve all private information of each node. This defect comes from the ineffectiveness of existing LDP mechanism in preserving frequently accessed private data. According to the *sequential composition* [15] of DP/LDP, repeated access to even different (and dependent) data of the same user requires to split privacy budget, resulting in a great increase in the amount of noise added. In constructing the GNN, the server has to access all types of node information, including features, edges, and labels. This requires over-splitting the privacy budget, ultimately degrading the performance of the GNN. Therefore, LDP cannot be applied to ensure node-level privacy in distributed settings.

We propose **D**istributed **P**rivate **A**ggregation (DPA), the first GNN aggregation method designed for distributed settings under the constraint of node-level DP. The **basic idea** of DPA

[†]These authors contribute equally to this work.
[*]Corresponding author, email: dk@seu.edu.cn.

is to make use of the Secure Multi-Party Computation (MPC) framework [9], which allows multiple parties to collaboratively compute a function without revealing their individual inputs [4, 5, 25]. DPA integrates MPC with DP to secure all node information. It operates in two phases. Initially, DPA employs an MPC protocol to compute the GNN aggregation function using encrypted private information, eliminating the need to add noise during computation. Subsequently, DPA adds calibrated noise to the decrypted aggregation results to achieve node-level DP. This offers the advantage of not requiring noise added to the original private data. However, the development of each phase presents complex challenges.

Challenge 1: Computation and Communication Constraints. The computation and communication burden associated with using MPC to directly compute the traditional GNN aggregation function is impractical. This is because the GNN aggregation function involves computation of a substantial amount of matrix multiplication, and computing matrix multiplication using MPC protocols is extremely inefficient.

Challenge 2: Privacy Budget Over-splitting. DPA requires the privacy budget to be split for each training epoch. This is necessary in the traditional training framework since DPA is invoked in each epoch and each invocation requires access to the private edge information one more time. Given that GNN training typically involves hundreds of epochs, the privacy budget must be split hundreds of times, which significantly degrades the performance of the GNN.

To address **Challenge 1**, we improve the existing aggregation function by replacing matrix multiplication with the summation of key-value pairs. We define the neighbor index of each node as a *key* and its feature and label vectors as *values*, which are sent to MPC parties via the *additive secret sharing* protocol [16]. In this protocol, every private value is split into shares and sent to multiple parties. Each computing party aggregates the received shares with matching keys to form the aggregated information within the graph. Then, distributed noise is added to this information to ensure node-level DP. After that, a collaborative process among user nodes and MPC parties can be iteratively employed to construct a GNN that can be trained according to downstream tasks.

Challenge 2 is addressed through two strategies. First, we decouple DPA from subsequent training phases, necessitating its invocation a very small number of times (e.g., 3 times for 3-layer GNNs). This addresses the problem of over-splitting, and also lowers communication overhead since distributed users and MPC parties no longer need to participate in the training process. Second, we deploy DPA on both the user side and the MPC parties, to prevent any participant from inferring edge information across multiple DPA invocations. Consequently, noise is only added at the first DPA invocation, utilizing the entire privacy budget without any split.

Building on DPA, we implement a GNN model constructed under the distributed setting, named DPA-GNN. We formally prove that DPA-GNN satisfies node-level DP. To validate its effectiveness, we conduct extensive experiments across six real-world datasets. We compare our DPA-GNN with 13 baselines (including 10 state-of-the-art methods in constructing privacy-preserving GNNs, and 3 variations of these methods). The results demonstrate that DPA-GNN consistently outperforms all baselines. We also perform parameter tuning experiments to determine the optimal settings for our proposed DPA-GNN.

We concisely summary our contributions as follows:

- We propose Distributed Private Aggregation (DPA), the first privacy-preserving GNN aggregation method that adheres to node-level DP constraints in distributed settings. By substituting matrix multiplication in traditional GNN aggregation with key-value structured addition, DPA allows for computation using MPC protocols across multiple parties.

- We implement DPA-GNN based on DPA. To the best of our knowledge, it is the best privacy-preserving GNN model that can be used in distributed settings. Compared to existing methods that rely on LDP, our DPA-GNN offers two distinct advantages. First, it does not require the assumption that node information is public or that node and edge information are independent, making it suitable for a broader range of distributed settings. Second, it protects data using cryptography from MPC protocols rather than relying solely on DP/LDP noise, thereby achieving a better privacy-utility trade-off.

- We conduct extensive experiments to demonstrate the advance of our proposed DPA-GNN. DPA-GNN is open-sourced and available at https://doi.org/10.5281/zenodo.14710401.

2 PRELIMINARIES

This section introduces Graph Neural Networks, Differential Privacy, and Secure Multi-Party Computation.

2.1 Graph Neural Networks

A graph is typically represented as $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ denotes the set of nodes and $\mathbf{E}$ represents the set of edges. Each node $V_i \in \mathbf{V}$ is associated with an ID (K_i), a feature vector ($X_i \in \mathbf{X}$), a list of edges ($A_i \in \mathbf{A}$), and a label ($Y_i \in \mathbf{Y}$). The feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ encodes the features of all nodes, where d is the dimensionality of the features, and $n = |\mathbf{V}|$ is the total number of nodes. The label matrix $\mathbf{Y} \in \{0,1\}^{n \times c}$ specifies the labels for all nodes, where c represents the number of classes. The adjacency matrix $\mathbf{A} \in \{0,1\}^{n \times n}$ captures the edge set $\mathbf{E}$, with an element A_{ij} set to 1 if and only if there is an edge $e_{ij} \in \mathbf{E}$ connecting nodes V_i and V_j. Finally, $\mathcal{N}(V_i)$ denotes the set of nodes adjacent to V_i.

Graph Neural Networks (GNNs) [42] are specialized neural network models designed to learn useful *representations* of nodes by leveraging node features and edges for various downstream tasks. Two fundamental functions are utilized in this process. The Aggregate function (*Agg*) combines neighbors' representation into this node, while the Update function (*Upd*) processes the aggregated representation along with the learnable parameter, denoted as W, to output the new representations.

A typical L-layer GNN consists of L sequential graph convolution layers. The initial representation of V_i (denoted as $H_i^{(0)}$) is X_i. Its representation of the l^{th} layer is denoted as $H_i^{(l)}$ (where $0 < l \leq L$). It is calculated as follows.

$$H_i^{(l)} = Upd\Big(Agg\Big(\{H_j^{(l-1)} : \forall V_j \in \mathcal{N}(V_i)\Big), W^{(l-1)}\Big), \quad (1)$$

where $W^{(l-1)}$ is the learnable parameter. The output is used for downstream tasks after a softmax layer.

2.2 Differential Privacy

Differential Privacy [12,13] is a formal definition on privacy. It quantifies the maximal deviation in the probability distribution functions of generating a particular output from an algorithm on any adjacent datasets, where the adjacent datasets are defined as two datasets differing by only a single record.

Definition 1. *(Differential Privacy, DP). An algorithm $\mathcal{M}$ satisfies (ε, δ)-DP if and only if for any two adjacent datasets D and D', and for the set of all possible outputs of $\mathcal{M}$ which is denoted as O, the following inequality holds:*

$$Pr[\mathcal{M}(D) \in O] \leq e^{\varepsilon} Pr[\mathcal{M}(D') \in O] + \delta. \quad (2)$$

Here, ε indicates the trade-off between privacy and utility within the algorithm, and is referred to as the *privacy budget*. A smaller ε results in enhanced privacy but reduced utility. Meanwhile, δ is informally characterized as a slight probability of privacy guarantees failing, signifying a minimal risk of surpassing the *privacy budget*. When $\delta = 0$ it satisfies pure differential privacy.

(ε, δ)-DP can be achieved by adding Gaussian noise to the output of an algorithm with variance defined as:

$$\sigma^2 = 2S^2 \log \frac{1.25}{\delta} / \varepsilon^2, \quad (3)$$

where S is the *sensitivity* of $\mathcal{M}$, which represents the greatest impact that the alteration of a single record can have on the algorithm's output. It is defined as the maximum ℓ_2 norm distance between the outputs of an algorithm f on two adjacent datasets D and D':

$$S = \max_{D,D'} \|f(D) - f(D')\|_2. \quad (4)$$

2.3 Secure Multi-Party Computation

A Secure Multi-Party Computation (MPC) [9] protocol allows multiple parties to jointly compute a function without disclosing their private inputs to each other. Secret sharing is one main implementation paradigm for MPC. In this paradigm, a secret (i.e., a private value) is divided into c shares, and at least t shares are needed to reconstruct the secret. After distributing the shares to parties, they collaboratively compute a function on the shares they have, and reconstruct the result as the output. *Additive Secret Sharing* [16] is a special type of secret sharing where $t = c$.

In *Additive Secret Sharing*, in order to split a secret s into t shares, $t-1$ numbers are randomly generated as $t-1$ shares, respectively denoted as $\langle s \rangle_1, \cdots, \langle s \rangle_{t-1}$. So the t^{th} share $\langle s \rangle_t$ is generated by calculating $s - \sum_{\lambda=1}^{n-1} \langle s \rangle_\lambda$. To reconstruct s, all parties need to exchange all the n shares and sum them up. With *Additive Secret Sharing*, parties can locally compute any linear functions of shared values [25]. For example, consider the λ^{th} party obtains its share of values $\langle x \rangle_\lambda$ and $\langle y \rangle_\lambda$. For any new secret value $z \triangleq k_1 x + k_2 y + k_3$, where k_1, k_2, k_3 are non-secret constants, the λ^{th} party can locally compute the λ^{th} share of z, denoted as $\langle z \rangle_\lambda$, as follows:

$$\langle z \rangle_\lambda = \langle k_1 x + k_2 y + k_3 \rangle_\lambda = k_1 \langle x \rangle_\lambda + k_2 \langle y \rangle_\lambda + k_3. \quad (5)$$

3 Problem and Challenges

In this section, we put forward the problem targeted by this paper, and discuss the challenges.

3.1 Problem

Target Scenario. We focus on a scenario involving an untrustworthy server tasked with training a GNN among distributed users. We consider a distributed graph $\mathbf{G} = (\mathbf{V}, \mathbf{A})$, where $\mathbf{V}$ represents the set of independent users, and $\mathbf{A}$ denotes the adjacency matrix distributed among these users. Each node contains private information, including features, labels, and edges (i.e., connections to its neighboring nodes). For example, in a real-world social network, features may encompass personal attributes such as age and interests, labels could categorize aspects like user roles or engagement levels, and edges might represent the friendships or communication links between users. Each user knows her own node information, including connections to neighboring nodes within the graph. This entails that she possesses not just her own feature vector X_i and label vector Y_i, but also a specific row of $\mathbf{A}$, referred to as A_i. Moreover, regardless of whether a user engages in activities (e.g., posting comments), the information of other users remains unaffected, as the overall graph structure is fixed, and that user has no right to force other users to follow or unfollow her [51]. We consider all features, labels, and edges maintained by the nodes as private information.

Privacy Constraints. We consider that each user requires some privacy guarantee in form of (ε, δ)-DP. However, as graph datasets are different from standard tabular datasets, the definition of (ε, δ)-DP needs to adapt to graphs. We distinguish two different privacy notions, namely the edge-level DP and node-level DP, depending on whether two adjacent graphs differ by one edge or one node.

Definition 2. *[Node-level DP (or Edge-level DP)] A randomized algorithm $\mathcal{M}$ satisfies node-level (or edge-level) (ε, δ)-DP if and only if for any two adjacent graphs G and G' that only differ by one node (or one edge), and for any possible outcome O, we have:*

$$\Pr[\mathcal{M}(G) \in O] \leq e^{\varepsilon} \Pr[\mathcal{M}(G') \in O] + \delta. \quad (6)$$

Edge-level DP only protects edges, while node-level DP also protects node features and labels. In another word, an algorithm satisfies node-level (ε, δ)-DP also satisfies edge-level (ε, δ)-DP. Our goal is to ensure node-level DP in scenarios where the model's output does not significantly differ, regardless of whether any individual node participates in training.

Trust Assumptions. We make the following assumptions. (1) The server and users are semi-honest, i.e., they follow the protocol but may attempt to infer private information. (2) The server and users are able to communicate anonymously with the users. (3) We consider additional semi-honest parties in our proposed solution so as to achieve node-level DP in building a GNN. Such parties can be distributed cloud servers, volunteers of verified nodes, etc. All these assumptions (semi-honest participants and anonymous communication) are widely adopted in recent research [4, 5, 25].

Problem Definition. Given the above scenario settings, privacy constraints and trust assumptions, how can the untrustworthy server construct an effective GNN? To the best of our knowledge, existing methods [8, 41, 51] only achieve node-level DP in centralized settings, with no solutions yet proposed for distributed scenarios.

3.2 Basic Idea and Challenges

Our initial idea is to employ MPC protocols in constructing a privacy-preserving GNN. Some very recent studies (e.g., [4, 5, 25]) establish MPC protocols to enable differentially private MPC aggregation, addressing the problems of reduced accuracy and potential privacy loss introduced by LDP. These MPC protocols ensure the confidentiality of private information through cryptographic means rather than merely balancing data privacy with utility through noise addition, thereby significantly enhancing the accuracy of statistical results while satisfying DP constraints. Following the idea of this line of approaches, we employ MPC protocols to build a GNN in distributed settings under the constraint of DP.

3.2.1 Initial Approach. We use the term **secret** from MPC to represent all private information. A straightforward workflow of utilizing MPC (with m semi-honest parties) to build a GNN is as follows. We then explain why this initial approach failed and highlight the main drawbacks.

Step 1: Share Construction. Each user node V_i generates secret shares of her feature vector X_i, adjacency list A_i and label Y_i. The label is processed into one-hot vector for secret sharing. A share of X_i (or A_i, Y_i) is denoted as $\langle X_i \rangle_\lambda$ (or $\langle A_i \rangle_\lambda$, $\langle Y_i \rangle_\lambda$), where $(\lambda = 1, ..., m)$. The λ^{th} share is then sent to the λ^{th} party.

Step 2: Secret Aggregation. Each party combines the received shares in order to construct the secret-shared feature matrix, adjacency matrix and label matrix. E.g., the λ^{th} party constructs $\langle X \rangle_\lambda$, $\langle A \rangle_\lambda$ and $\langle Y \rangle_\lambda$. The node representation H is the product of the adjacency matrix A and the feature matrix X. Thus we have:

$$H = AX = \sum_{\lambda}^{m} \langle A \rangle_\lambda \sum_{\lambda}^{m} \langle X \rangle_\lambda = \sum_{\lambda=1}^{m} SecMul\big(\langle A \rangle_\lambda \langle X \rangle_\lambda\big). \quad (7)$$

The m parties can perform a secure multiplication protocol *SecMul* [50], that each party can obtain a secret share of H, denoted as $\langle H \rangle_\lambda$.

Step 3: Secret Reconstruction. To securely publish the node representations and labels, each party independently perturbs the obtained $\langle H \rangle_\lambda$ and $\langle Y \rangle_\lambda$ to ensure that the reconstructed matrix satisfies node-level DP. One party reconstructs the perturbed node representations $\tilde{H}$ and perturbed label $\tilde{Y}$.

Step 4: Model Training. $\tilde{H}$ and $\tilde{Y}$ are passed to the next GNN layer for aggregation. The trainable parameters W are updated by minimizing the difference between the predicted labels $\hat{Y}$ and the perturbed labels $\tilde{Y}$.

Steps 2 to 4 are iteratively performed to train a GNN.

3.2.2 Challenges. The above initial approach has two key drawbacks (technical challenges).

Challenge 1. Computation and Communication Constraints. The computation and communication burden in performing an MPC protocol designed for building a GNN is far from practical. Training a GNN requires hundreds of epochs or more. In our initial approach, each epoch requires to perform a MPC multiplication protocol in Step 2. To perform such a protocol, all computing parties need to compute a so-called *Beaver triple*, which is a cryptographic element used in MPC [50] (to facilitate the division of the multiplication task among different parties without revealing their private inputs). This process also necessitates frequent communication between computing parties [33]. Consequently, our initial approach is inapplicable in training a GNN model.

Challenge 2. Privacy Budget Over-splitting. According to the *sequential composition* [15] of DP, the privacy budget must be split for each training epoch, as the graph structure is accessed during every epoch in Step 2. Additionally, within each epoch, the privacy budget needs to be further split because

participants could infer edge information by combining the results of multiple invocations of the aggregation function. For example, consider a GNN with 3 layers trained over 200 epochs. In this scenario, the privacy budget must be split $200 \times 3 = 600$ times. This leads to a substantial amount of noise being added to the reconstructed secret in Step 3, which significantly degrades the model's performance.

4 Distributed Private Aggregation

In this section, we propose Distributed Private Aggregation (DPA) to address the previous challenges. At a high level, DPA can be abstracted as the fundamental *Aggregation* function of GNN. In what follows, we detail six **modifications** to highlight the difference between DPA and our initial approach (introduced in Sec. 3.2.1).

4.1 Modifications Addressing Challenge 1

Modification 1. *Matrix addition instead of matrix multiplication.* To address Challenge 1, we aim to reduce the computational overhead associated with matrix multiplication. Each node sends its representation to its neighbors while also receiving representations from them. This allows each node to aggregate by simply summing up all received representations, effectively replacing matrix multiplication with matrix addition. Take a graph with n nodes as an example, we have:

$$H^{(l)} = AH^{(l-1)} = \sum_{i=1}^{n} \left[A_{i1}H_i^{(l-1)}, \cdots, A_{in}H_i^{(l-1)}\right]^\top, \quad (8)$$

where A_{ij} denotes the j^{th} element in the adjacency list A_i of node V_i, and $H_i^{(l-1)}$ denotes the representation of node V_i at the $(l-1)^{th}$ layer. The transposed matrix $[A_{i1}H_i, \cdots, A_{in}H_i]^\top$ is the information that node V_i should provide to her neighbor.

A new drawback caused by Modification 1. While the previous modification reduces both computation and communication overhead among the parties, it significantly increases the communication cost from users to the parties. Although the adjacency matrix is typically sparse, processing zero entries still requires heavy MPC computation and communication, since the information that two nodes are unconnected is also private and needs to be protected.

Modification 2. *Key-value instead of a sparse matrix.* To address the previous drawback, we make use of the key-value data structure to replace the feature passing matrix, as illustrated in Figure 1. Since the key-value data structure inherently protects the edges, we treat both features and edges as secrets. Let K_i represent the index of node V_i, and let H_j denote the representation of its neighboring node V_j. We construct key-representation pair, denoted as $\mathcal{H}_{ij}$, as follows:

$$\mathcal{H}_{ij}^{(l-1)} \triangleq (K_i : H_j^{(l-1)}). \quad (9)$$

Let $\{\mathcal{H}_{ij}^{(l-1)}\}$ represent the set of the key-representation pairs at the $(l-1)^{th}$ layer, as follows:

$$\left\{\mathcal{H}_{ij}^{(l-1)}\right\} \triangleq \left\{(K_i : H_j^{(l-1)}) \mid H_j^{(l-1)} \in N(K_i)\right\}, \quad (10)$$

where $N(K_i)$ represents the set of all values associated with the key K_i in the key-value pairs.

We perform addition aggregation on the key-representation pairs, calculated as follows:

$$H_i^{(l)} = KVagg\left(\left\{\mathcal{H}_{ij}^{(l-1)}\right\}\right), \quad (11)$$

where $H_i^{(l)}$ represents the aggregated representation of node V_i at the l^{th} layer, and $KVagg(\cdot)$ represents the key-value aggregation function defined as:

$$\begin{aligned} &KVagg\left(\left\{(K_i : H_j) \mid H_j \in N(K_i)\right\}\right) \\ &\triangleq \left(K_i : \textstyle\sum_{R_j \in N(K_i)} H_j\right) = \left(K_i : \sum_{j=1}^{n} A_{ij}H_j\right). \end{aligned} \quad (12)$$

This function performs an additive aggregation on the values of key-value pairs with the same key.

Compared to node representation matrix H of Modification 1, the key-representation structure does not require operations on zero elements. Therefore, secret sharing is now only applied to non-zero element, and the overhead is significantly reduced.

A new drawback caused by Modification 2. When key-value pairs are sent to untrustworthy parties, it exposes the number of neighbors (i.e., the degree, denoted as d) of each user node. Since the degree of a node is dependent of its edges, the disclosure of degrees indirectly lead to leakage of edges.

Modification 3. *Achieving DP defined on edges via dummies.* To address this drawback, we employ the selective MPC protocol [25] to achieve DP defined on node degrees by generating dummy key-value pairs. A dummy pair can be any fictitious friend added by a user in a social network. Specifically, each node independently samples a value n_k from a geometric distribution with parameter r, and generates n_k dummies for key K_i. To ensure aggregation is unaffected by dummies, each dummy's value vector is set to zero. Let t denote the size of the subset selected from m parties, and let ε_e represent the privacy budget for edges. This process satisfies edge-level ε_e-DP:

$$\varepsilon_e = \ln\left(\max\left\{\tfrac{1}{1-r}, \tfrac{1}{1-t/m} + 1 - r\right\}\right). \quad (13)$$

4.2 Modifications Addressing Challenge 2

A new drawback caused by key-value pairs. When adopting key-value pairs provided by user nodes, it makes the training process impractical in real-world graphs. The requirement

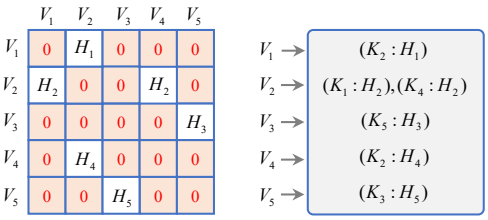


Figure 1: The feature passing matrix with 25 vectors is transformed into 6 key-value pairs.

to reuse key-value pairs at each layer for iterative training in GNNs results in significant communication overhead, and necessitates that users remain constantly online.

Modification 4. *Separating the Aggregation and Training Stages.* Target at Challenge 2 and the previous drawback, we perform the aggregation for an L-layer GNN, and use the final node representations for training. In our implementation, we make no modifications to existing frameworks and directly utilize Simple Graph Convolution (SGC) [49], a commonly adopted GNN framework. SGC inherently separates aggregation from training and avoids over-splitting the privacy budget, simplifying the GNN training process while maintaining excellent performance. We only add noise to the aggregated representations at the first layer, as the subsequent layers are protected by the *post-processing* [15] of DP. Thus, no additional privacy budget is needed for layers 2 to L.

A new drawback caused by Modification 4. Consider a party participating in secret reconstruction across two different layers, gaining access to the representations $H^{(l-1)}$ and $H^{(l)}$. This enables the party to infer the graph structure by comparing these two representations without further additional noise.

Modification 5. *Secret Reconstruction on the user side.* To maintain confidentiality, the user nodes and the computing parties jointly perform the aggregation process. The user nodes generate key-value pairs and carry out secret sharing, while the computing parties aggregate the shares corresponding to the same key. The aggregated results are then sent back to the nodes for secret reconstruction.

A new drawback caused by perturbed labels. The shared labels are perturbed and reconstructed for training. However, directly using these perturbed labels to train the GNN can cause the model to overfit to the noise, leading to poor generalization performance.

Modification 6. *Aggregating the perturbed labels.* Nodes with similar structures are more likely to have similar labels, and we utilize the graph structure to calibrate perturbed labels. Similar to the construction of the key-representation, we also construct the key-label pair, denoted as $\mathcal{Y}_{ij}$, as follows:

$$\mathcal{Y}_{ij}^{(l-1)} \triangleq (K_i : Y_j^{(l-1)}), \tag{14}$$

where $Y_j^{(l-1)}$ represents the one-hot label vector.

Let $\{\mathcal{Y}_{ij}^{(l-1)}\}$ represent a set of the key-label pairs at the $(l-1)^{th}$ layer. The aggregated labels at the l^{th} layer are:

$$Y_i^{(l)} = KVagg\Big(\{\mathcal{Y}_{ij}^{(l-1)}\}\Big). \tag{15}$$

The aggregated labels are later used to train the GNN. Note that this modification to the initial approach does not compromise the DP guarantees, as labels are protected in Step 3 in the same way as node representations.

4.3 DPA: Distributed Private Aggregation

We conclude all the above modifications, and formalize a `DPA` function. The `DPA` function is built upon three core subfunctions defined as follows.

(1) The Share Construction function (SC), which is performed on the user side, is defined as:

$$SC(\mathcal{R}_{ij}^{(l-1)}) = \langle \mathcal{R}_{ij}^{(l-1)} \rangle_\lambda, \tag{16}$$

where $\mathcal{R}_{ij} = (K_i : R_j)$ represents key-representation pair or key-label pair, where R_j denotes either node representation or label. $R_j = \sum_{\lambda=1}^{t} \langle R_j \rangle_\lambda$ represents that R_j generates t shares. $\langle \cdot \rangle_\lambda$ is the λ^{th} share and sent to the λ^{th} party.

(2) The Secret Aggregation function (SA), which is performed by the computing parties, is defined as:

$$SA\Big(\{\langle \mathcal{R}_{ij}^{(l-1)} \rangle_\lambda\}\Big) \triangleq KVagg\Big(\{\langle \mathcal{R}_{ij}^{(l-1)} \rangle_\lambda\}\Big) = \langle R_i^{(l)} \rangle_\lambda, \tag{17}$$

where $\{\langle \mathcal{R}_{ij}^{(l-1)} \rangle_\lambda\}$ denotes the set of shared key-value pairs at the $(l-1)^{th}$ layer.

(3) The Secret Reconstruction function (SR), which is performed on the user side, is defined as:

$$SR\Big(\langle R_i^{(l)} \rangle_\lambda\Big) \triangleq \sum_{\lambda=1}^{m} \langle R_i^{(l)} \rangle_\lambda = R_i^{(l)}. \tag{18}$$

The `DPA` function is defined as:

$$\texttt{DPA}\Big(\mathcal{R}_{ij}^{(l-1)}\Big) \triangleq SR\Big(SA\Big(\{SC(\mathcal{R}_{ij}^{(l-1)})\}\Big)\Big). \tag{19}$$

5 DPA-GNN

Based on `DPA`, we implement `DPA-GNN`, the first graph neural network satisfying both edge-level and node-level DP in distributed settings.

Overview. We propose five main modules for building a `DPA-GNN`, namely **Initialization**, **Share Construction (SC)**, **Secret Aggregation (SA)**, **Secret Reconstruction (SR)**, and **Model Training**. The workflow for `DPA-GNN` is shown in Figure 2. The modules SC, SA, and SR correspond to the implementation of the three functions defined in Section 4.3. The remaining part of this section provides details of each module.

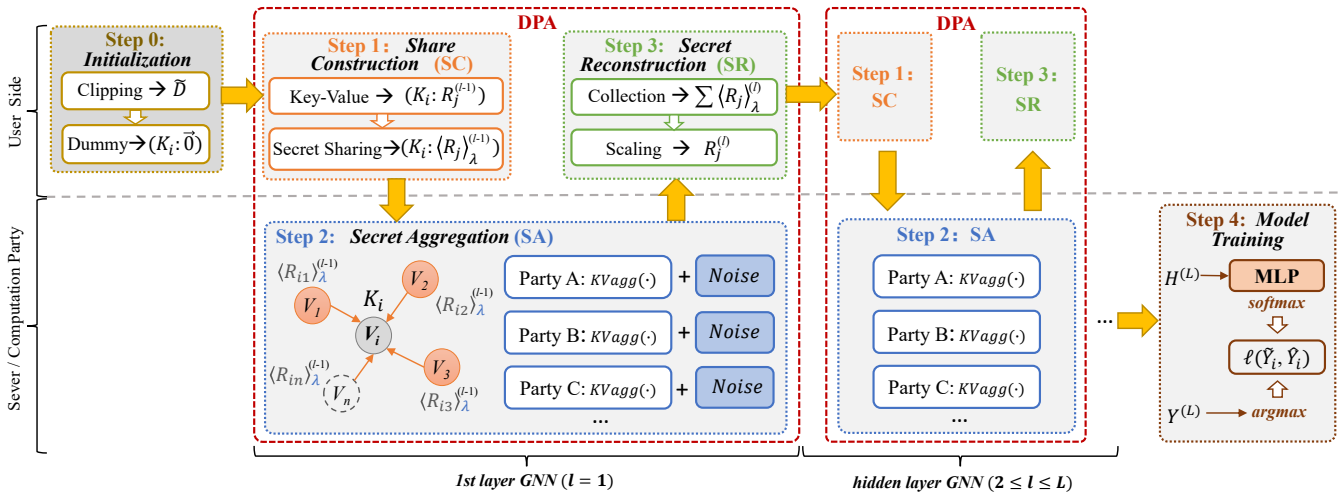


Figure 2: Workflow of DPA-GNN. Users execute the **Initialization** module for data preprocessing and the **SC** module to generate shared key-value pairs, which are then transmitted to the parties. The parties run the **SA** module to perturb and aggregate node representations and labels, which are subsequently forwarded to the **SR** module for recovery. The final output, processed through multiple layers of aggregation, is passed to the MLP for **Model Training**. Yellow arrows illustrate the data flow.

Step 0. Initialization. This module performs data preprocessing on the user side. The preprocessed data is consistently utilized in the subsequent GNN aggregation steps. It is composed of two steps.

Step 0a: Edge Clipping. In this step, the edges of each node are clipped according to the clipping rate c if the degree of this node is greater than a threshold $\tilde{D}$. This step helps limit the sensitivity and the noise to be introduced, so as to increase the accuracy of the model as in [8, 41].

Step 0b: Dummy Generation. In this step, a one-sided geometric distribution with parameter r is used to generate dummy edge pairs for each node. A dummy key-value pair generated by node V_i is in the form $(K_i : \vec{0})$, where $\vec{0}$ is an all-zero vector of length $|H_i|$ or $|Y_i|$. The dummies are generated only once for each node and reused in subsequent layers.

Step 1. Share Construction (SC). This module is performed on the user side. Since the key-value data structure inherently protects the edges, DPA-GNN treats features (or node representations) and labels as secrets. This module is composed of three steps.

Step 1a: Key-Value Construction. In this step, each node V_j contains representation H_j, label Y_j and neighbor edge pairs $e_{ij} = (K_i, K_j)$. The initial H_j is row-normalized, and Y_j is expressed as a one-hot vector, with test set labels set to $\vec{0}$. Each node constructs key-representation pair $(K_i : H_j)$ and key-label pair $(K_i : Y_j)$, collectively referred to as $\mathcal{R}_{ij}$.

Step 1b: Secret Sharing. In this step, each user node uses additive secret sharing to split $\mathcal{R}_{ij}$ into t shares (including dummy key-value pairs). This process partitions the values of the key-value pairs while preserving the original keys, generating t messages. Each message is denoted as $\langle\mathcal{R}_{ij}^{(l-1)}\rangle_\lambda$, where $\lambda \in [1, t]$ and $t < m$, as specified by Eq. (16).

Step 1c: Message Sending. In this step, each node randomly selects t out of m parties and anonymously sends $\langle\mathcal{R}_{ij}\rangle$ to the λ^{th} selected party through a shuffler [2], for all $\lambda \in [1, m]$. This step satisfies edge-level DP.

Step 2. Secret Aggregation (SA). This module performs the aggregation of node representations and labels on each party. It is composed of two steps.

Step 2a: Key-Value Aggregation. In this step, each party collects all the messages sent to him and applies the *SA* function, as defined in Eq. (17), to compute the aggregated result $\langle R_i^{(l)}\rangle_\lambda$ for each key K_i. This aggregated result represents the sum of the shared representations and labels of the node associated with that key.

Step 2b: Secret Perturbation. In this step, each party introduces Gaussian noise with a variance of σ^2/m to perturb the aggregated node representations and labels, as follows.

$$Ptb\left(\langle R_i^{(l)}\rangle_\lambda\right) = \langle R_i^{(l)}\rangle_\lambda + N_\lambda(0, \frac{\sigma^2}{m}), \tag{20}$$

where $N_\lambda(0, \sigma^2/m)$ is the Gaussian noise introduced, with a mean value of 0 and a variance of σ^2/m.

Steps 2a and 2b are conducted in the first GNN layer to ensure that the computing parties satisfy DP. In the subsequent GNN layers, only step 2a is carried out, as justified by the *post-processing* of DP [13].

Step 3. Secret Reconstruction (SR). This module reconstructs messages from the m parties on the user side. It is composed of two steps.

Step 3a: Secret Collection. In this step, the user node

receives all perturbed shares and then collects all the shares to reconstruct the secret (the node representations and labels) by calculating:

$$R_i^{(l)} = SR\Big(Ptb\Big(\big\langle R_i^{(l)} \big\rangle_\lambda\Big)\Big). \tag{21}$$

Step 3b: Secret Scaling. In this step, each user scales the reconstructed secret R_i by a factor α_i. To normalize the impact of each node's representation, we set $\alpha_i = 1/d_i$, where d_i represents the dummy degree of node V_i. Let $\widetilde{R}_i$ denotes the scaled results. We have:

$$\widetilde{R}_i^{(l)} = \big(R_i^{(l)}\big)\alpha_i. \tag{22}$$

When the scaled value $\widetilde{R}_i$ is sent back to the computing parties, it is split into shares via secret sharing to ensure its confidentiality. Therefore, $\widetilde{R}_i$ is not required to satisfy DP.

Subsequently, $\widetilde{R}_i$ is passed to the next layer, where shared key-value pairs are created, and a new round of distributed private aggregation is initiated. In this process, Gaussian noise is added only in the first GNN layer, while subsequent layers ($2 \le l \le L$) apply only the DPA function, as specified by Eq. (19), which is defined as follows:

$$R_i^{(l+1)} = \texttt{DPA}\Big(\big\{\mathcal{R}_{ij}^{(l)}\big\}\Big). \tag{23}$$

Note that the reconstructed node representation of the final layer is not scaled in order to reduce communication costs.

Step 4. Model Training. This module is performed on the server side. In this module, DPA-GNN is trained using the aggregated node representations and labels from the final GNN layer.

The secret reconstruction of the L^{th} GNN layer (the final layer) is performed on the party, obtaining $H^{(L)}$ and $Y^{(L)}$. Let $\widetilde{Y}_i$ represent the index of the maximum value obtained by applying the *argmax* function to the aggregated labels $Y^{(L)}$, as follows:

$$\widetilde{Y}_i = argmax\big(Y_i{}^{(L)}\big). \tag{24}$$

The server trains a Multi-Layer Perceptron (MLP) to update node representations and predicts node classifications using the *softmax* function. The predicted label $\hat{Y}$ is given by:

$$\hat{Y} = \text{softmax}\big(H^{(L)}W\big). \tag{25}$$

The learnable parameter W is optimized by minimizing the cross-entropy loss ℓ between $\widetilde{Y}_i$ and $\hat{Y}_i$:

$$\ell\big(\widetilde{Y}_i, \hat{Y}_i\big) = -\textstyle\sum_{i=1}^{n} \widetilde{Y}_i \log(\hat{Y}_i). \tag{26}$$

To save privacy budget and speed up model training, $H^{(L)}$ is cached for training, validation, and testing, while $Y^{(L)}$ is cached for training and validation. The labels of the test set are left clean for performance evaluation.

6 Privacy Analysis and Discussion

6.1 Privacy Analysis

DPA-GNN satisfies edge-level DP and node-level DP. ε_e, ε_h and ε_y denote the privacy budgets for edges, node representations, and labels, respectively. Detailed proofs of the following theorems are provided in Appendix A.

Theorem 1. *The Share Construction (SC) module of DPA-GNN satisfies edge-level ε_e-DP, and ensures the confidentiality of node features and labels.*

Theorem 2. *Following the SC module, the Secret Aggregation (SA) module of DPA-GNN satisfies edge-level ε_e-DP and node-level $(\varepsilon_e + \varepsilon_h + \varepsilon_y, 2\delta)$-DP.*

Theorem 3. *The secret reconstruction (SR) module of DPA-GNN satisfies edge-level DP and node-level DP.*

Theorem 4. *The model training module of DPA-GNN satisfies edge-level DP and node-level DP.*

Theorem 5. *DPA-GNN satisfies edge-level ε_e-DP and node-level $(\varepsilon_e + \varepsilon_h + \varepsilon_y, 2\delta)$-DP.*

6.2 Discussion on Colluding Nodes and Colluding Parties

The aggregation function DPA ensures that the aggregated attributes satisfy DP while protecting the confidentiality of all edges through MPC. Consequently, colluding nodes cannot compromise the privacy of other nodes and can only access the edges directly connected to themselves.

The impact of colluding parties is more complicated. DPA-GNN is designed to tolerate colluding parties. In cases where n out of the m computing parties collude, users can split the secret into at least $n+1$ shares to ensure that no single party can obtain all the secret shares and reconstruct the original secret, thereby preserving privacy. However, colluding parties can reduce the privacy level by exploiting the knowledge of the noise added by them while they cannot directly access users' private data.

7 EVALUATION

In this section, we describe the experiment setup, including datasets, baselines, and parameters, followed by the experimental results on DPA-GNN to answer the following questions:

(1) How does DPA-GNN perform on different datasets?
(2) How does DPA-GNN perform compared to existing methods in scenarios focusing on different types of protection (features, edges and labels)?
(3) When transitioning from distributed to centralized settings, how does DPA-GNN perform compared to existing centralized methods?

Table 1: Dataset statistics

Dataset	Nodes	Features	Edges	Classes
Cora [53]	2708	1433	10858	7
CiteSeer [53]	3312	3703	9430	6
LastFM [38]	7624	7842	55612	18
Facebook [37]	22470	4714	342004	4
Amazon [44]	13471	767	491772	10
Reddit [56]	140667	602	51605960	12

(4) How to set parameters of DPA-GNN?
(5) What are the communication and offline costs of DPA-GNN?

7.1 Experiment Setup

7.1.1 Environment. All experiments are conducted on a high-performance server with an Intel Xeon Gold 6430 processor (16 vCPUs), 120GB of RAM, and an NVIDIA GeForce RTX 4090 GPU. Each experiment is performed 10 times, with mean and standard deviation reported.

7.1.2 Datasets. We conduct experiments on 6 public real-world datasets. A summary of datasets is presented in Table 1. ***Cora*** [53] and ***CiteSeer*** [53] are two well-known citation networks frequently used for benchmarking. Each node represents a document, and edges denote citation relationships. ***LastFM*** [38] is a social network gathered from the music streaming service LastFM, where each node represents a user, and the edges between them denote friendships. ***Facebook*** [37] is a social network compiled from Facebook, where each node represents a verified Facebook page, and edges indicate mutual likes. ***Amazon*** [44] is a co-purchase network extracted from Amazon, and represents products as nodes and co-purchase relationships as edges. ***Reddit*** [56] is a large-scale social network, and represents posts as nodes and connects them with edges if the same user comments on both posts.

7.1.3 Baselines. We consider various baseline methods for comparison. These baselines are categorized into three types: the existing methods, the privacy-enhanced variants of existing methods and the centralized node-level DP methods. We summarize data types treated as privacy in baselines and DPA-GNN in Table 2.

Category 1. Existing privacy preserving GNN methods:

(1) ***RR*** [26], which applies the Random Response mechanism to the adjacency list for edge privacy.
(2) ***RGNN*** [3], which protects feature and label privacy using LDP, and reconstructs messages on the server side.
(3) ***LPGNN*** [40], which uses multi-bit mechanism and RR to protect node features and labels, and employs the KProp mechanism to denoise the injected noise.
(4) ***Solitude*** [31], which protects node features and edges in decentralized scenarios and calibrates introduced noise.
(5) ***Blink*** [58], which utilizes an LDP mechanism to protect edge privacy and applies Bayesian estimation to denoise.
(6) ***DPRR*** [22], which protects edge privacy by perturbing the edge list with RR and adjusts edge sampling probabilities using noisy degree information.

None of the above methods consider all types of privacy. For example, RGNN and LPGNN do not treat edges as private, while Solitude does not treat labels as private. Considering that existing methods [3, 31, 40] use RR to protect edges and labels, we extend privacy protection using RR to address the aspects these methods overlook and to better fit our scenario. We develop enhanced variants of these methods as follows and then compare them with DPA-GNN.

Category 2. Enhanced variants of existing methods:

(1) ***Solitude+RR***, which is based on Solitude [31] with node labels perturbed using RR to create Solitude+RR.
(2) ***LPGNN+RR***, which is based on LPGNN [40] with adjacency matrix perturbed using RR to create LPGNN+RR.
(3) ***RGNN+RR***, which is based on RGNN [3] with labels perturbed using RR to create RGNN+RR.

Category 3. Centralized node-level DP methods:

(1) ***DP-SGD*** [1], which trains a simple MLP using DP-SGD, ensuring node-level DP without relying on graph edges.
(2) ***GAP-NDP*** [41], which combines multi-layer aggregation perturbation with DP-SGD.
(3) ***PNPiGNNs*** [51], which combines a HeterPoisson sampling routine with symmetric multivariate Laplace noise.
(4) ***DPDGC*** [8], which introduces decoupled graph convolutions and applies DP-SGD for model training.

7.1.4 Parameter settings. The parameter settings of DPA-GNN and baselines in the experiments are as follows:

Common parameters: For all methods, we set the number of MLP layers to 2, the hidden dimension to 64, initial learning rate to 0.0005, weight decay to 0.01, dropout rate to 0.5, the privacy parameter δ to 10^{-5}, training epoch to 200 and split randomly the nodes into training, validation, and test sets with ratios of $2:1:1$.

Specific parameters: For DPA-GNN, we set the shares of secret sharing t to 2, the number of parties m to 3 (i.e., selective parameter $p = 2/3$), and dummy parameter r to 0.5. All baseline methods are implemented using parameter settings from the original studies.

7.2 Experiments and Results

7.2.1 Utility-privacy trade-off. We conduct experiments to evaluate the trade-off between utility and privacy achieved by DPA-GNN and baselines. Node feature and label vectors differ in dimension, requiring distinct aggregation layers. Let l_h and l_y represent the aggregation layers for features and labels, respectively. We set the optimal aggregation layers (l_h, l_y) to (10,8), (8,10), (10,6), (6,4), (10,2), (8,2), and the

Table 2: Data types treated as privacy in baselines and DPA-GNN.

	Category 1						Category 2	Category 3	
Baseline	RR [26]	Solitude [31]	LPGNN [40]	RGNN [3]	DPRR [22]	Blink [58]			DPA-GNN
Features		✓	✓	✓			✓	✓	✓
Edges	✓	✓			✓	✓	✓	✓	✓
Labels	✓		✓	✓			✓	✓	✓

Table 3: Performance of DPA-GNN and baselines in Category 2 across varying privacy budget settings on six datasets.

	Method	$\varepsilon=2$	$\varepsilon=4$	$\varepsilon=6$	$\varepsilon=8$	$\varepsilon=10$	$\varepsilon=2$	$\varepsilon=4$	$\varepsilon=6$	$\varepsilon=8$	$\varepsilon=10$
		Cora					CiteSeer				
ε_h=5%ε	Solitude+RR	29.0±4.40	30.3±2.21	30.9±1.57	30.5±0.99	62.8±3.00	20.7±1.48	20.7±1.67	20.2±1.19	22.6±3.31	40.8±1.11
	LPGNN+RR	21.5±7.53	16.7±9.66	18.8±9.09	30.5±6.54	59.7±5.26	15.9±2.81	17.1±4.06	17.2±2.78	19.6±1.64	38.0±3.46
	RGNN+RR	29.5±1.02	29.5±1.04	29.5±1.01	29.2±1.28	50.9±1.38	19.8±1.34	20.4±1.19	20.3±1.14	21.6±1.36	31.3±1.77
	DPA-GNN	**27.1**±3.97	**66.8**±2.07	**74.5**±1.95	**77.4**±1.94	**78.4**±1.42	**24.9**±3.05	**44.8**±2.88	**52.9**±2.73	**56.1**±1.43	**60.5**±2.23
ε_h=20%ε	Solitude+RR	28.7±5.63	29.8±1.50	29.6±1.37	30.5±1.10	37.0±2.30	20.3±1.31	19.9±2.20	20.3±2.24	21.2±1.26	30.1±2.97
	LPGNN+RR	21.8±8.17	26.5±3.72	29.2±0.00	31.3±0.00	39.2±4.01	17.8±4.78	17.9±3.82	19.0±4.04	20.4±1.41	27.2±2.26
	RGNN+RR	29.6±0.96	29.5±1.37	29.5±1.02	29.2±1.16	39.1±2.48	20.1±1.49	20.6±1.23	20.4±1.46	19.9±1.08	30.8±1.50
	DPA-GNN	**20.0**±6.94	**56.2**±5.54	**73.6**±0.91	**76.5**±2.33	**77.6**±1.18	**21.8**±3.38	**39.6**±2.25	**51.2**±1.67	**55.5**±1.46	**57.7**±1.52
		LastFM					Facebook				
ε_h=5%ε	Solitude+RR	13.9±7.32	11.7±6.37	14.0±6.67	12.9±7.43	44.1±5.18	30.1±0.81	30.6±1.13	29.9±0.95	35.5±6.50	72.0±0.98
	LPGNN+RR	8.7±7.73	9.0±7.65	11.6±6.76	17.8±4.47	38.6±12.81	25.9±5.89	28.4±2.30	28.1±2.52	38.2±7.94	60.0±7.51
	RGNN+RR	19.8±2.01	20.3±1.70	20.3±1.56	25.7±2.08	57.2±1.29	30.4±0.56	30.9±0.78	30.9±0.43	33.6±2.68	50.1±1.87
	DPA-GNN	**41.6**±8.73	**80.3**±1.46	**82.7**±0.86	**83.1**±1.06	**83.8**±1.01	**54.8**±3.28	**81.1**±0.85	**84.9**±0.46	**85.8**±0.59	**86.1**±0.59
ε_h=20%ε	Solitude+RR	14.1±6.65	13.4±6.35	18.9±2.66	17.7±3.86	28.0±5.59	29.7±1.15	29.7±0.89	30.2±1.67	34.8±4.27	58.9±3.36
	LPGNN+RR	13.4±6.94	17.7±3.96	14.8±7.72	13.7±5.67	16.4±5.37	26.5±4.39	29.3±1.17	28.2±1.97	31.0±3.88	46.4±6.78
	RGNN+RR	20.3±1.70	20.5±1.89	21.1±0.97	25.1±1.55	49.1±3.46	29.5±1.45	30.2±0.64	30.7±0.66	32.9±2.77	35.1±2.75
	DPA-GNN	**25.0**±7.46	**77.3**±1.90	**82.7**±0.52	**83.4**±0.81	**83.1**±0.51	**47.3**±2.38	**79.2**±1.23	**83.8**±0.72	**85.7**±0.51	**86.3**±0.27
		Amazon					Reddit				
ε_h=5%ε	Solitude+RR	18.7±13.20	37.1±0.71	37.3±0.44	55.0±11.72	73.2±1.13	–	–	–	–	–
	LPGNN+RR	12.0±12.66	21.7±12.67	29.7±11.67	52.6±16.47	71.7±1.60	–	–	–	–	–
	RGNN+RR	35.8±0.64	36.8±0.73	35.7±2.97	58.8±3.74	70.7±1.71	–	–	–	–	–
	DPA-GNN	**46.2**±4.45	**82.6**±0.91	**85.7**±0.52	**85.6**±0.43	**85.9**±0.74	**91.2**±0.38	**96.0**±0.15	**96.1**±0.05	**96.2**±0.07	**96.3**±0.10
ε_h=20%ε	Solitude+RR	22.6±13.09	34.6±7.73	36.8±0.75	41.6±3.52	71.7±1.67	–	–	–	–	–
	LPGNN+RR	27.9±11.04	34.5±6.21	36.7±0.00	45.3±8.29	66.3±5.95	–	–	–	–	–
	RGNN+RR	36.2±0.53	36.4±0.81	36.8±0.73	37.6±0.44	41.9±3.27	–	–	–	–	–
	DPA-GNN	**46.0**±10.37	**81.4**±1.27	**84.3**±1.21	**85.3**±0.50	**84.9**±0.54	**89.6**±0.90	**96.1**±0.09	**96.1**±0.05	**96.2**±0.12	**96.4**±0.06

optimal clipping rates c to 80%, 80%, 70%, 60%, 40%, 20% on Cora, CiteSeer, LastFM, Facebook, Amazon, and Reddit, respectively.

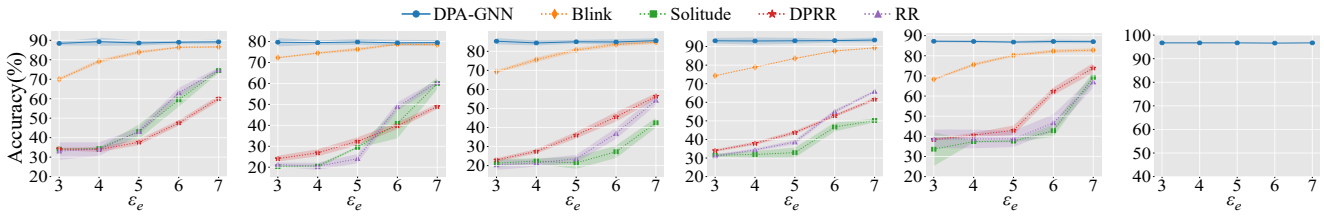
Baselines associated with LDP, including those in Category 1 and Category 2, require transforming sparse matrices into dense matrices with added noise, which imposes substantial memory requirements. Due to hardware limitations, these baselines could not be executed on the largest dataset, Reddit, thus the results of experiments [E1]–[E5] are not available.

[E1] Utility-privacy trade-off. In this experiment, we compare DPA-GNN with baselines in Category 2 (including Solitude+RR, LPGNN+RR and RGNN+RR) on six datasets under different privacy budgets. To investigate the optimal partitioning strategy under varying privacy budgets, we set the total privacy budget ε to $\{2,4,6,8,10\}$, with the node representation privacy budget ε_h set to 5%ε and 20%ε. The edge privacy budget ε_e for DPA-GNN is fixed at 1.25 (when $p=\frac{2}{3}$ and $r=0.5$). In contrast, the baselines allocate a higher proportion of the budget to ε_e, with the remaining budget assigned to the label privacy budget ε_y. The results are shown in Table 3. We observe that DPA-GNN outperforms the baselines. Additionally, our method achieves excellent classification accuracy even at $\varepsilon=4$, while the baseline results are mostly below 40%. DPA-GNN demonstrates a 10% to 30% improvement over the baseline methods when $\varepsilon=10$ and $\varepsilon_h=5\%\varepsilon$. These results indicate that DPA-GNN achieves superior performance while simultaneously protecting node features, edges, and labels.

7.2.2 Ablation studies. We conduct ablation studies by considering part but not all data types are private.

[E2] Protecting edges only. In this experiment, we compare DPA-GNN with baselines that only protect edges (including RR, Blink, Solitude, DPRR). We set the privacy budget ε_e in the range $[2,8]$, with $\varepsilon_h=\infty$, $\varepsilon_y=\infty$, and the number of layers as $l_h=2$, $l_y=2$. The results are shown in Figure 3. By using the key-value data structure and the adaptation of selective MPC [25] to protect edge information, DPA-GNN does not exhibit significant changes with the increase of ε_e. Additionally, DPA-GNN demonstrates superior accuracy compared to all four baselines. Notably, under low privacy budget conditions, it consistently outperforms the state-of-the-art method, Blink.

[E3] Protecting features and edges. In this experiment,



(a) Cora (b) CiteSeer (c) LastFM (d) Facebook (e) Amazon (f) Reddit

Figure 3: Accuracy of GNN on edge privacy budget ε_e. Only edges are treated as privacy.

we compare DPA-GNN with Solitude, the only baseline that protects features and edges. We set $\varepsilon_h \in \{0.001, 0.01, 0.1, 1, 2\}$, $\varepsilon_y = \infty$, and $\varepsilon_e \in \{4, 8\}$. The results are shown in Figure 4. DPA-GNN demonstrates comparable performance at ε_e=4 and ε_e=8, benefiting from its unique edge privacy protection mechanism. Specifically, DPA-GNN surpasses Solitude's best results, achieving improvements of 4.9%, 11.4%, 15.0%, 7.1%, and 0.1% on Cora, CiteSeer, LastFM, Facebook, and Amazon, respectively, under the privacy budget $(\varepsilon_h, \varepsilon_e) = (2, 8)$.

[E4] Protecting edges and labels. In this experiment, we compare DPA-GNN with RR, the only baseline that protects edges and labels. We set $\varepsilon_y \in \{0.1, 1, 2, 3, 4\}$, $\varepsilon_h = \infty$, and $\varepsilon_e \in \{4, 8\}$. Furthermore, when dealing with large datasets, we set $l_h = 2$. The results are shown in Figure 5. When $\varepsilon_y > 1$, DPA-GNN outperforms RR across varying privacy budgets.

[E5] Protecting features and labels. In this experiment, we compare DPA-GNN with baselines that protect features and labels (including LPGNN and RGNN). We set $\varepsilon \in \{2, 4, 6, 8, 10\}$, and the results are presented in Figure 6. DPA-GNN demonstrates performance comparable to the baselines.

7.2.3 Scenario transition. We conduct experiments to compare DPA-GNN with baselines when transitioning from distributed to centralized settings.

[E6] Performance in centralized settings. In this experiment, we compare the performance of DPA-GNN with baselines in centralized settings (including DP-SGD, GAP, PNPiGNNs, and DPDGC). These baselines utilize subgraph sampling or mini-batch training, which enables them to handle large datasets, such as Reddit. The privacy budget ε is set to $\{2, 4, 6, 8, 10\}$. As shown in Figure 7, we initially anticipated that DPA-GNN would underperform compared to the baselines, given that the centralized setting is inherently stronger than the distributed setting. However, the experimental results demonstrate that DPA-GNN achieves performance comparable to, or even surpassing, all four baselines.

7.2.4 Effect of parameters. We conduct experiments to evaluate the effect on the settings of hyper-parameters, including number of layers l_h and l_y, clipping rate c, dummy parameter r, and selective parameter p.

[E7] Number of feature and label aggregation layers. In this experiment, we compare the accuracy achieved by DPA-GNN with different numbers of feature and label aggregation layers l_h and l_y. We set l_h and l_y to values from $\{2, 4, 6, 8, 10\}$, and ε to $\{4, 8\}$, with $\varepsilon_h = 5\%\varepsilon$ and $\varepsilon_e = 1.25$. The results are shown in Figure 8. DPA-GNN effectively benefits from multiple hops, though it also experiences the over-smoothing problem [27]. In the figure, results closer to yellow indicate higher values. Based on the distribution of the color blocks, we select the optimal values for l_h and l_y.

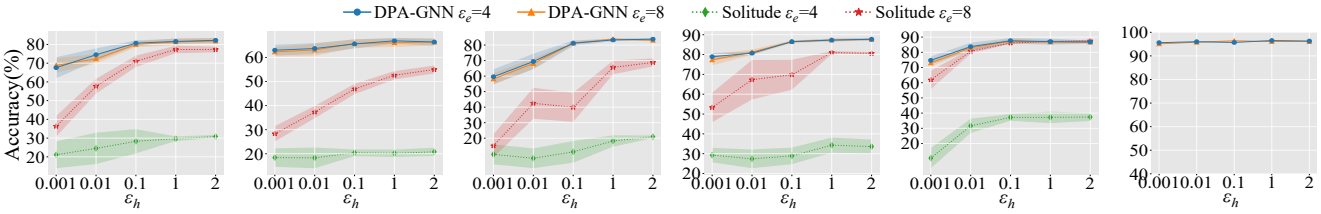
[E8] Clipping rate c. In this experiment, we calculate the classification accuracy of DPA-GNN under different clipping rates. The clipping rates indicate the percentage of nodes with the lowest degrees that are retained, while the rest with higher degrees are excluded. DPA-GNN limits the sensitivity of DP with the threshold degree. We set c to ranging from 20% to 100%, and ε to $\{4, 6, 8, 10\}$, with $\varepsilon_h = 5\%\varepsilon$ and $\varepsilon_e = 1.25$. The results are presented in Figure 9. DPA-GNN achieves its best accuracy at different clipping rates (c) for each dataset. For the six datasets, DPA-GNN achieves optimal accuracy at clipping rates of 80%, 80%, 70%, 60%, 40% and 20%, respectively.

[E9] Dummy parameter r. In this experiment, we evaluate the accuracy of DPA-GNN for different dummy parameters r in the range $[0.1, 0.9]$. We set ε to $\{4, 6, 8, 10\}$, with $\varepsilon_h = 5\%\varepsilon$ and $\varepsilon_e = 1.25$. The results are presented in Figure 10, where the bar chart illustrates the number of generated dummy edges. When $r = 0.1$, the number of dummies reaches its maximum. At $r = 0.5$, the number of dummies is twice the number of nodes, effectively balancing privacy and utility.

[E10] Selective parameter p. In this experiment, we evaluate the accuracy of DPA-GNN with varying numbers of secret shares and computing parties. We set $p = t/m$ to values from $\{2/3, 2/4, 3/4, 3/5\}$, and ε to $\{4, 8\}$, with $\varepsilon_h = 5\%\varepsilon$ and $\varepsilon_e = 1.25$. The results are shown in Table 4. The accuracy of DPA-GNN remains unaffected by variations in p, indicating that the number of computing parties and secret shares does not affect the results. However, as the number of computing parties and secret shares increases, it results in higher communication and computational costs. Therefore, we select $p = 2/3$ to maintain edge-level DP.

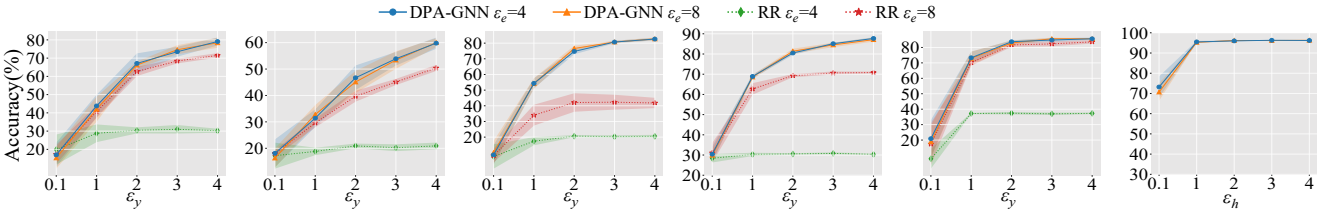
7.2.5 Discussion. We discuss the performance trade-offs associated with competing methods and large-scale datasets.

Competing methods. DPA-GNN utilizes a key-value data structure to aggregate and perturb node features and labels.



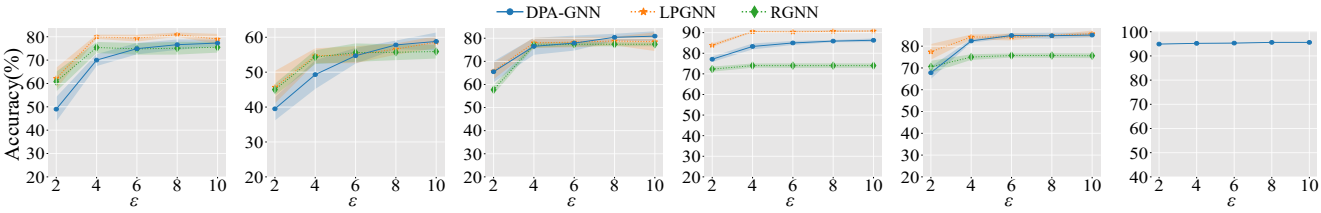
(a) Cora (b) CiteSeer (c) LastFM (d) Facebook (e) Amazon (f) Reddit

Figure 4: Accuracy of GNN on node representation privacy budget ε_h. Both features and edges are treated as privacy.



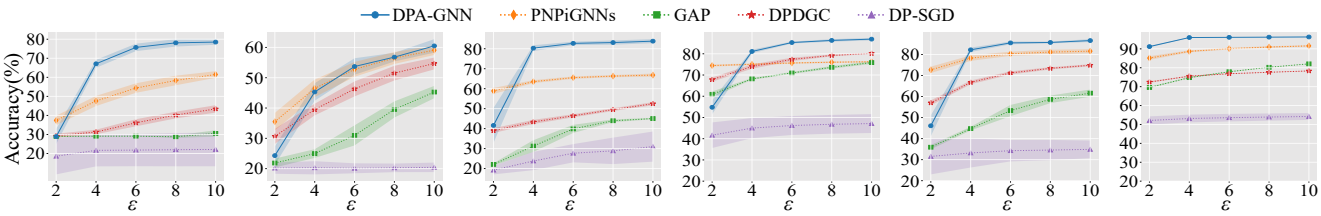
(a) Cora (b) CiteSeer (c) LastFM (d) Facebook (e) Amazon (f) Reddit

Figure 5: Accuracy of GNN on label privacy budget ε_y. Both edges and labels are treated as privacy.



(a) Cora (b) CiteSeer (c) LastFM (d) Facebook (e) Amazon (f) Reddit

Figure 6: Accuracy of GNN on total privacy budget ε. Both features and edges are treated as privacy.

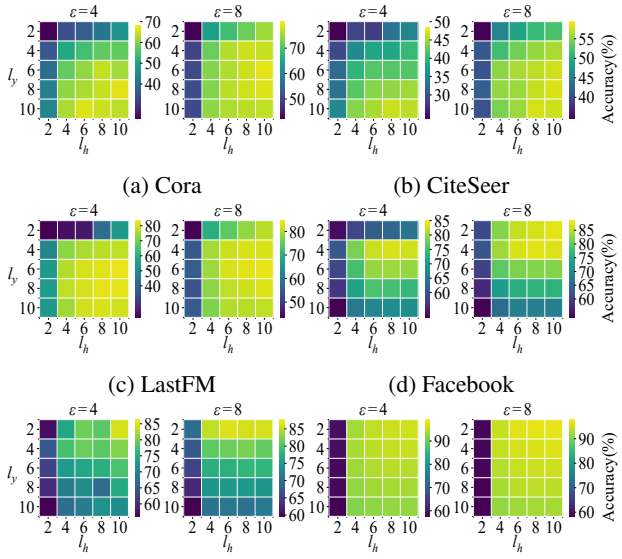


(a) Cora (b) CiteSeer (c) LastFM (d) Facebook (e) Amazon (f) Reddit

Figure 7: Comparison of DPA-GNN's accuracy with baselines when transitioning from distributed to centralized settings.

This process inherently protects the edges since anyone who obtains the data can only know one node of an edge pair. Consequently, DPA-GNN demonstrates significant advantages in scenarios where edges are considered private, as demonstrated in Experiments [E1] to [E4]. Additionally, DPA-GNN is the best choice (in comparison with competing methods, including Blink [58]) when edge privacy is a concern, particularly under a relatively low privacy budget (<6), as demonstrated in Experiment [E2]. In scenarios where edges are not treated as private, LDP methods, such as LPGNN [40], are sufficient, as demonstrated in Experiment [E5]. The reason is that all private data in this case (i.e., the graph data removing edges) becomes conventional two-dimensional tabular data.

Large-scale datasets. DPA-GNN is well-suited for large-scale graph datasets, offering significant advantages over existing LDP methods in terms of memory and GPU efficiency (as demonstrated in Experiments [E1]–[E10]). This is achieved through its inherent capability to partition large graphs using multi-party computation. In contrast, LDP methods require substantial memory resources and are constrained by hardware limitations when handling large-scale datasets such as Reddit (as demonstrated in Experiments [E1]–[E5]), since LDP con-



(a) Cora (b) CiteSeer

(c) LastFM (d) Facebook

(e) Amazon (f) Reddit

Figure 8: Effect of the feature aggregation layers l_h and label aggregation layers l_y on the accuracy of DPA-GNN.

Table 4: Effect of selective parameter p in MPC on the accuracy of DPA-GNN.

Dataset	ε	$p=t/m$ 2/3	2/4	3/4	3/5
Cora	4	66.8±2.07	65.8±3.54	66.2±3.05	65.4±4.07
	8	77.4±1.94	77.4±1.17	77.8±1.93	77.5±0.9
CiteSeer	4	44.8±2.88	45.5±2.35	46.3±2.71	44.1±2.06
	8	56.1±1.43	57.8±2.05	57.1±2.53	57.9±2.01
LastFM	4	80.3±1.46	79.7±2.18	81.0±1.24	80.8±1.56
	8	83.1±1.06	83.2±1.15	83.3±0.56	83.4±0.89
Facebook	4	81.1±0.85	81.9±0.85	82.1±1.12	82.7±0.79
	8	85.8±0.59	86.5±0.37	86.7±0.72	86.7±0.42
Amazon	4	84.4±1.05	82.9±1.30	83.8±1.14	83.1±0.89
	8	86.1±0.64	85.6±0.84	85.7±0.65	86.1±0.72
Reddit	4	96.0±0.15	96.0±0.12	96.0±0.16	96.1±0.07
	8	96.2±0.04	96.1±0.04	96.2±0.07	96.1±0.05

straints in distributed settings prevent the training server from utilizing centralized methods for dataset partitioning.

7.3 Communication and Offline Setup Costs

In this section, we analyze the communication and offline setup costs of DPA-GNN.

7.3.1 Communication costs. The communication overhead of DPA-GNN is composed of two parts. The first part comes from user-party interactions during aggregation. At each layer, each user generates key-value pairs for her neighbors (the number of which is d), splits them into t secret shares, and incurs $O(kdtC)$ communication overhead, where C is the number of feature/label dimensions and k is the number of layers. Subsequently, each of the m parties sends the shares of his aggregated result to a single server, incurring a total cost of $O(mNC)$, where N is the number of users. Since the training process is executed entirely on the single server, no additional communication is required during the training iterations.

Table 5: Comparison of communication and offline costs between DPA-GNN and initial approach.

	Communication		Offline Computation
	User-Party (per User)	Party-Party (Training)	
Initial	$O(m(N+C))$	$O(mkT(N^2+NC))$	$O(m(N+C))$
DPA-GNN	$O(kdtC)$	$O(mNC)$	$O(kdtC)$

We compare DPA-GNN with the initial approach. In the initial approach, users share their features, labels, and N-dimensional adjacency lists with m computing parties via secret sharing, costing $O(m(N+C))$. A k-layer GNN is then trained over T iterations using MPC, with a cost of $O(mkT(N^2+NC))$, where (N^2+NC) accounts for data exchanged during MPC multiplications. The results of these methods are summarized in Table 5.

For the large-scale Reddit dataset, the communication cost of a non-private method is $22KB$ for each user, and no party-party cost is required. Using additive secret sharing with a 128-bit modulus, the communication cost for DPA-GNN and initial approach are as follows. For DPA-GNN, the user-party communication cost for each user is $4.9MB$, and the party-party communication cost is $1.35GB$ for each party. For the initial approach, the user-party communication cost is $6.8MB$ for each user, and the party-party communication cost is $0.5PB = 5\times10^5 GB$ for each party.

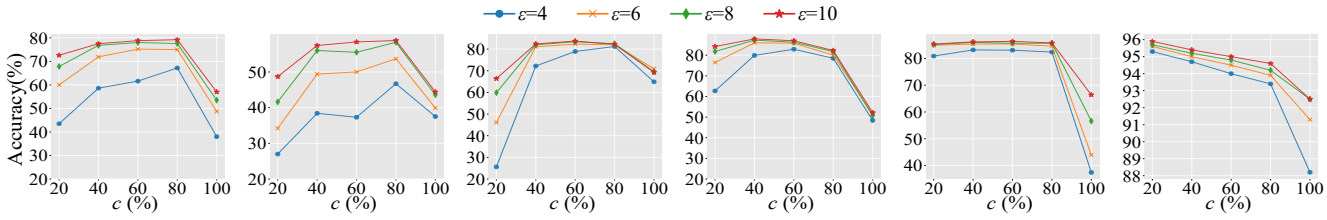
7.3.2 Offline setup costs. The offline setup costs for these methods include the process of generating data to be sent to the computing parties. Accordingly, the offline setup costs for DPA-GNN and the initial approach are $O(kdtC)$ and $O(m(N+C))$, respectively, as illustrated in Table 5.

More experimental results are provided in Appendix B.

8 Related work

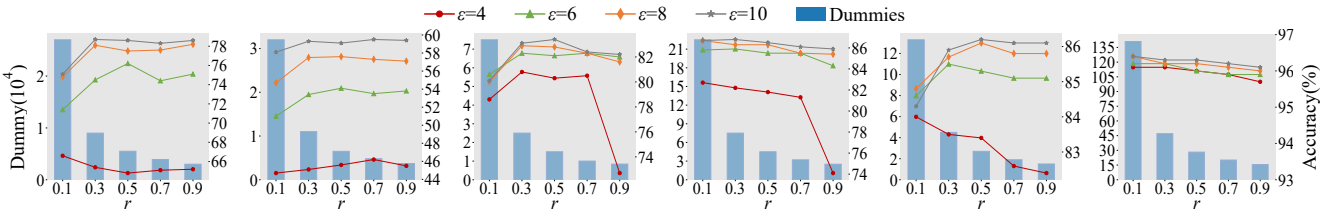
In this section, we briefly survey four lines of research: Distributed Differential Privacy, Graph Neural Networks, Centralized/Locally Differentially Private GNNs, and Other Privacy-Preserving GNNs.

Distributed Differential Privacy (DDP). This line of research processes data among multiple users or participants to satisfy differential privacy. Goryczka *et al.* [17] propose the distributed Laplace perturbation algorithm. Huang *et al.* [14, 24] introduce distributed noise generation methods to reduce the size of noise. Ruan *et al.* [39] propose a secure DPSGD protocol in secret sharing-based MPC frameworks. Humphries *et al.* [25] use selective MPC to efficiently compute statistics



(a) Cora (b) CiteSeer (c) LastFM (d) Facebook (e) Amazon (f) Reddit

Figure 9: Effect of clipping rate c on the accuracy of DPA-GNN, ε is set to 4 to 10.



(a) Cora (b) CiteSeer (c) LastFM (d) Facebook (e) Amazon (f) Reddit

Figure 10: Effect of dummy parameter r on accuracy of DPA-GNN. ε is set to 4 to 10. Bar charts represent the number of dummies.

over key-value data. [4, 5] propose DP protocols with secure multi-party computation. Cheu *et al.* [7] propose a shuffled model for distributed DP algorithm. However, DDP has not been applied to the privacy protection of GNNs.

Graph Neural Networks. GNN [42] has now become a popular model for processing graph-structured data. Researchers have proposed various GNN models, including GCN [27], GraphSAGE [19], GAT [46], GIN [52], etc. Currently, GNNs have gained impressive success in various tasks such as node classification [27], link prediction [43], and graph generation [55]. Due to their superior performance, GNNs have been widely applied in social networks [53], recommendation systems [54], drug discovery [34], and other scenarios.

Centralized/Locally Differentially Private GNNs. This line of research focuses on constructing GNNs under the constraints of centralized/local differential privacy (DP) [15]. For centralized scenarios, existing research (e.g., [8, 10, 41, 51]) typically employs the DP-SGD [1] algorithm or its variants [51] in GNNs. However, these approaches cannot ensure node-level privacy in distributed settings, as the server has access to private data of each node. For distributed scenarios, existing research only preserves edge-level privacy (e.g., [22, 58]), attribute privacy (e.g., [3, 40]), or both (e.g., [31]), but not node-level privacy. To the best of our knowledge, we are the first to guarantee node-level DP in distributed settings.

Other Privacy-Preserving GNNs. This line of research is primarily centered on generalizing privacy protection methods other than differential privacy, such as data anonymization [45], federated learning [29], and adversarial learning [30] to graph data structures. Meden *et al.* [32] combine GNN with anonymization mechanism, and provide formal guarantees for privacy protection. Olatunji *et al.* [35] achieve the transfer of model knowledge from private to public graphs. Hsieh *et al.* [23] propose a graph perturbation-based method with adversarial attacks. Chen *et al.* [6] protect the privacy of nodes in vertically partitioned data scenarios.

9 Conclusion

This paper introduces Distributed Private Aggregation (DPA), the first GNN aggregation method ensuring node-level differential privacy (DP) in distributed settings. It allows for computation using Secure Multi-Party Communication (MPC) protocols, by substituting matrix multiplication in traditional GNN aggregation with key-value structured addition. We implement a GNN based on DPA, namely DPA-GNN. Extensive experiments conducted on six datasets confirm that DPA-GNN outperforms existing methods, striking an optimal balance between privacy and utility.

Privacy protection in GNNs remains an emerging field with significant potential for future research, particularly in the context of dynamic graphs. Compared to static graphs, building privacy-preserving GNNs for dynamic graphs presents greater challenges. The temporal information in dynamic graph updates may expose additional node-specific information and reveal interaction patterns between nodes, requiring the allocation of extra privacy budgets. Furthermore, each update in a dynamic graph consumes privacy budgets for aggregation, and frequent updates inevitably over-split the privacy budget. To address these challenges, our future research will focus on exploring more efficient privacy-preserving mechanisms.

Acknowledgment

This research is supported in part by National Key R&D Program of China Grant No. 2023YFC3605804, National Natural Science Foundation of China Grant Nos. 62472089, 62232004, 62102084, 62072098 and 92467205, US National Science Foundation (NSF) Awards 1931871 and 2325451, Jiangsu Key R&D Program BE2022065-5, BE2022680 and BE2021729, Jiangsu Provincial Key Laboratory of Network and Information Security Grant No. BM2003201, Key Laboratory of Computer Network and Information Integration of Ministry of Education of China Grant No. 93K-9, and Collaborative Innovation Center of Novel Software Technology and Industrialization. Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

Ethics Considerations

Our research employs publicly available graph networks free of personal or sensitive data, mitigating privacy and ethical concerns. Using standard benchmark datasets, our study adheres to ethical research standards and avoids the risks associated with personal data processing.

Open Science

`DPA-GNN` is open-sourced and available at https://doi.org/10.5281/zenodo.14710401. This repository also includes implementations of three variants of existing methods: Solitude+RR, LPGNN+RR, and RGNN+RR.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (CCS)*, pages 308–318, 2016.

[2] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *39th Annual International Cryptology Conference (CRYPTO)*, pages 638–667, 2019.

[3] Karuna Bhaila, Wen Huang, Yongkai Wu, and Xintao Wu. Local differential privacy in graph neural networks: a reconstruction approach. *arXiv preprint arXiv:2309.08569*, 2023.

[4] Jonas Böhler and Florian Kerschbaum. Secure multiparty computation of differentially private median. In *29th USENIX Security Symposium (USENIX Security)*, pages 2147–2164, 2020.

[5] Jonas Böhler and Florian Kerschbaum. Secure multiparty computation of differentially private heavy hitters. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2361–2377, 2021.

[6] Chaochao Chen, Jun Zhou, Longfei Zheng, Huiwen Wu, Lingjuan Lyu, Jia Wu, Bingzhe Wu, Ziqi Liu, Li Wang, and Xiaolin Zheng. Vertically federated graph neural network for privacy-preserving node classification. *arXiv preprint arXiv:2005.11903*, 2020.

[7] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *38th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 375–403, 2019.

[8] Eli Chien, Wei-Ning Chen, Chao Pan, Pan Li, Ayfer Ozgur, and Olgica Milenkovic. Differentially private decoupled graph convolutions for multigranular topology protection. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2024.

[9] Ronald Cramer, Ivan Bjerre Damgård, et al. *Secure multiparty computation*. 2015.

[10] Ameya Daigavane, Gagan Madan, Aditya Sinha, Abhradeep Guha Thakurta, Gaurav Aggarwal, and Prateek Jain. Node-level differentially private graph neural networks. *arXiv preprint arXiv:2111.15521*, 2021.

[11] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. Quantifying privacy leakage in graph embedding. In *2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, pages 76–85, 2020.

[12] Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming (ICALP)*, pages 1–12, 2006.

[13] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation (TAMC)*, pages 1–19, 2008.

[14] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *24th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 486–503, 2006.

[15] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[16] Oded Goldreich, Silvio Micali, and Avi Wigderson. Completeness theorem for protocols with honest majority. In *Conference Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.

[17] Slawomir Goryczka, Li Xiong, and Vaidy Sunderam. Secure multiparty aggregation with differential privacy. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, 2013.

[18] Slawomir Goryczka, Li Xiong, and Vaidy Sunderam. Secure multiparty aggregation with differential privacy: A comparative study. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 155–163, 2013.

[19] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems (NeurIPS)*, 30, 2017.

[20] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Stealing links from graph neural networks. In *30th USENIX security symposium (USENIX security)*, pages 2669–2686, 2021.

[21] Xinlei He, Rui Wen, Yixin Wu, Michael Backes, Yun Shen, and Yang Zhang. Node-level membership inference attacks against graph neural networks. *arXiv preprint arXiv:2102.05429*, 2021.

[22] Seira Hidano and Takao Murakami. Degree-preserving randomized response for graph neural networks under local differential privacy. *arXiv preprint arXiv:2202.10209*, 2022.

[23] I-Chung Hsieh and Cheng-Te Li. Netfense: Adversarial defenses against privacy attacks on neural networks for graph data. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 35(1):796–809, 2021.

[24] Wen Huang, Ming Zhuo, Tianqing Zhu, Shijie Zhou, and Yongjian Liao. Differential privacy: Review of improving utility through cryptography-based technologies. *Concurrency and Computation: Practice and Experience*, 35(5):e7565, 2023.

[25] Thomas Humphries, Rasoul Akhavan Mahdavi, Shannon Veitch, and Florian Kerschbaum. Selective mpc: Distributed computation of differentially private key-value statistics. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1459–1472, 2022.

[26] Peter Kairouz, Keith Bonawitz, and Daniel Ramage. Discrete distribution estimation under local privacy. In *International Conference on Machine Learning (ICML)*, pages 2436–2444, 2016.

[27] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[28] Aashish Kolluri, Teodora Baluta, Bryan Hooi, and Prateek Saxena. Lpgnet: Link private graph networks for node classification. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1813–1827, 2022.

[29] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[30] Wanyu Lin, Shengxiang Ji, and Baochun Li. Adversarial attacks on link prediction algorithms based on graph neural networks. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, pages 370–380, 2020.

[31] Wanyu Lin, Baochun Li, and Cong Wang. Towards private learning on decentralized graphs with local differential privacy. *IEEE Transactions on Information Forensics and Security (TIFS)*, 17:2936–2946, 2022.

[32] Blaž Meden, Žiga Emeršič, Vitomir Štruc, and Peter Peer. k-same-net: k-anonymity with generative deep neural networks for face deidentification. *Entropy*, 20(1):60, 2018.

[33] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (S&P)*, pages 19–38, 2017.

[34] Thin Nguyen, Hang Le, Thomas P Quinn, Tri Nguyen, Thuc Duy Le, and Svetha Venkatesh. Graphdta: predicting drug–target binding affinity with graph neural networks. *Bioinformatics*, 37(8):1140–1147, 2021.

[35] Iyiola E Olatunji, Wolfgang Nejdl, and Megha Khosla. Membership inference attack on graph neural networks. In *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 11–20, 2021.

[36] Sofya Raskhodnikova and Adam Smith. Differentially private analysis of graphs. *Encyclopedia of Algorithms*, 2016.

[37] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

[38] Benedek Rozemberczki and Rik Sarkar. Characteristic functions on graphs: Birds of a feather, from statistical

descriptors to parametric models. In *Proceedings of the 29th ACM international conference on information & knowledge management (CIKM)*, pages 1325–1334, 2020.

[39] Wenqiang Ruan, Mingxin Xu, Wenjing Fang, Li Wang, Lei Wang, and Weili Han. Private, efficient, and accurate: Protecting models trained by multi-party learning with differential privacy. In *2023 IEEE Symposium on Security and Privacy (S&P)*, pages 1926–1943, 2023.

[40] Sina Sajadmanesh and Daniel Gatica-Perez. Locally private graph neural networks. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security (CCS)*, pages 2130–2145, 2021.

[41] Sina Sajadmanesh, Ali Shahin Shamsabadi, Aurélien Bellet, and Daniel Gatica-Perez. {GAP}: Differentially private graph neural networks with aggregation perturbation. In *32nd USENIX Security Symposium (USENIX Security)*, pages 3223–3240, 2023.

[42] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[43] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *Extended Semantic Web Conference (ESWC)*, pages 593–607, 2018.

[44] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[45] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems (IJUFKS)*, 10(05):557–570, 2002.

[46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[47] Jianian Wang, Sheng Zhang, Yanghua Xiao, and Rui Song. A review on graph neural network methods in financial applications. *arXiv preprint arXiv:2111.15367*, 2021.

[48] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. Linkteller: Recovering private edges from graph neural networks via influence analysis. In *2022 ieee symposium on security and privacy (S&P)*, pages 2005–2024, 2022.

[49] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[50] Zhihua Xia, Qi Gu, Wenhao Zhou, Lizhi Xiong, Jian Weng, and Naixue Xiong. Str: Secure computation on additive shares using the share-transform-reveal strategy. *IEEE Transactions on Computers (TC)*, (01):1–1, 2021.

[51] Zihang Xiang, Tianhao Wang, and Di Wang. Preserving node-level privacy in graph neural networks. In *2024 IEEE Symposium on Security and Privacy (S&P)*, pages 198–198, 2024.

[52] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[53] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning (ICML)*, pages 40–48, 2016.

[54] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 974–983, 2018.

[55] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning (ICML)*, pages 5708–5717, 2018.

[56] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2019.

[57] Qiuchen Zhang, Hong kyu Lee, Jing Ma, Jian Lou, Carl Yang, and Li Xiong. Dpar: Decoupled graph neural networks with node-level differential privacy. In *Proceedings of the ACM on Web Conference 2024*, pages 1170–1181, 2024.

[58] Xiaochen Zhu, Vincent YF Tan, and Xiaokui Xiao. Blink: Link local differential privacy in graph neural networks via bayesian estimation. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2651–2664, 2023.

A Proofs

A.1 Proof of Theorem 1

In the Step 0 (Initialization) of DPA-GNN, the confidentiality of nodes' private information is satisfied via secret sharing. During this step, each user constructs key-value pairs and anonymously sends them to the parties, preserving edge-level privacy. The proof is as follows:

Edge-level privacy must be protected in two key aspects: (1) the edges between nodes and (2) the degree of each node. The first aspect is achieved through the key-value data structure combined with anonymous communication. Each key-value pair reveals only one node index for each edge. Once the computing parties receive these key-value pairs, they cannot infer the edges between nodes because the source of the key-value pairs remains unknown due to anonymous communication. Importantly, no privacy budget is consumed during this stage.

For the second aspect, the degree of node V_i can be calculated as as the frequency of key K_i. To protect this information, the *Selective MPC* protocol [25] is employed. In *Selective MPC*, users first generate a specified number of dummy key-value pairs using a geometric distribution. The probability mass function (PMF) of geometric distribution with parameters r and z is given by:

$$G(z;r) = (1-r)^z r. \tag{27}$$

The dummy key-value pairs are generated once and reused across subsequent layers without incurring additional privacy costs. Next, each user randomly selects a subset of parties to send key-value pairs, which is equivalent to adding noise drawn from a binomial distribution. On average, each party observes a subset (t/m) of the key-value pairs. The PMF of binomial distribution with parameters a, p and z, is given by:

$$B(z;a,p) = \binom{a}{z} p^z (1-p)^{a-z}. \tag{28}$$

According to Theorem 6.1 in [25], this process satisfies ε_e-DP with

$$\varepsilon_e = ln\left(max\left\{\frac{1}{1-r}, \frac{1}{1-p}+1-r\right\}\right). \tag{29}$$

With the edge information and node degrees protected, DPA-GNN satisfies edge-level DP.

We further prove that the SC module in DPA-GNN does not disclose any node information. The node representation $\mathcal{H}_{ij}$ and label $\mathcal{Y}_{ij}$ of node V_i are split into secret shares $\langle\mathcal{H}_{ij}\rangle$ and $\langle\mathcal{Y}_{ij}\rangle$ according to Eq. (16). Through this cryptographic mechanism, the SC module ensures the confidentiality of both node features and labels.

In conclusion, the SC module of DPA-GNN guarantees the confidentiality of node representations and labels while satisfying edge-level ε_e-DP.

A.2 Proof of Theorem 2

Following the Share Construction (SC) module, the confidentiality of all node features and labels remains guaranteed in the SA module. For the first layer of DPA-GNN, the computing parties introduce distributed Gaussian noise to the secret shares of node representations and labels within the SA module, ensuring node-level DP. The proof is as follows:

For any $\delta \in (0,1)$, each of the m parties adds Gaussian noise $N(0,\sigma_h^2/m)$ with variance $\sigma_h^2/m = 2S^2\log\frac{1.25}{\delta}/(\varepsilon_h^2 m)$ to the aggregated representations. Similarly, Gaussian noise $N(0,\sigma_y^2/m)$ with variance $\sigma_y^2/m = 2S^2\log\frac{1.25}{\delta}/(\varepsilon_y^2 m)$ is added to the aggregated labels. This approach exploits the infinite divisibility property of the Gaussian distribution to ensure consistent noise addition across all parties, as discussed in [24].

Given the maximum degree $\widetilde{D}$ of the graph, DPA-GNN is defined under bounded DP with the ℓ_2 sensitivity $S = \sqrt{2(\widetilde{D}+1)}$. The node representations satisfy (ε_h,δ)-DP, where ε_h is calculated as:

$$\varepsilon_h = \frac{S}{\sigma_h}\sqrt{2\log\frac{1.25}{\delta}} = \frac{1}{\sigma_h}\sqrt{4(\widetilde{D}+1)\log\frac{1.25}{\delta}}. \tag{30}$$

The labels satisfy (ε_y,δ)-DP, where ε_y is calculated as:

$$\varepsilon_y = \frac{1}{\sigma_y}\sqrt{4(\widetilde{D}+1)\log\frac{1.25}{\delta}}. \tag{31}$$

For subsequent layers of DPA-GNN, the SA module operates on private information that already satisfies node-level DP. According to the *post-processing* [15] of DP, the SA module satisfies edge-level ε_e-DP. Furthermore, by the *sequential composition* [15] of DP, the SA module satisfies node-level $(\varepsilon_h+\varepsilon_e+\varepsilon_y, 2\delta)$-DP:

$$\varepsilon_e+\varepsilon_h+\varepsilon_y = \varepsilon_e + (\frac{1}{\sigma_h}+\frac{1}{\sigma_y})\sqrt{4(\widetilde{D}+1)\log\frac{1.25}{\delta}}. \tag{32}$$

A.3 Proof of Theorem 3

According to the *post-processing* [15] of DP, reconstructing the perturbed node representations and labels at each layer does not consume additional privacy budget. The SC module satisfies both edge-level DP and node-level DP.

A.4 Proof of Theorem 4

The private node representations and labels from the final layer are jointly used to train DPA-GNN through the MLP. As a result, no additional queries are made on the private data, ensuring that the model training module satisfies both edge-level DP and node-level DP.

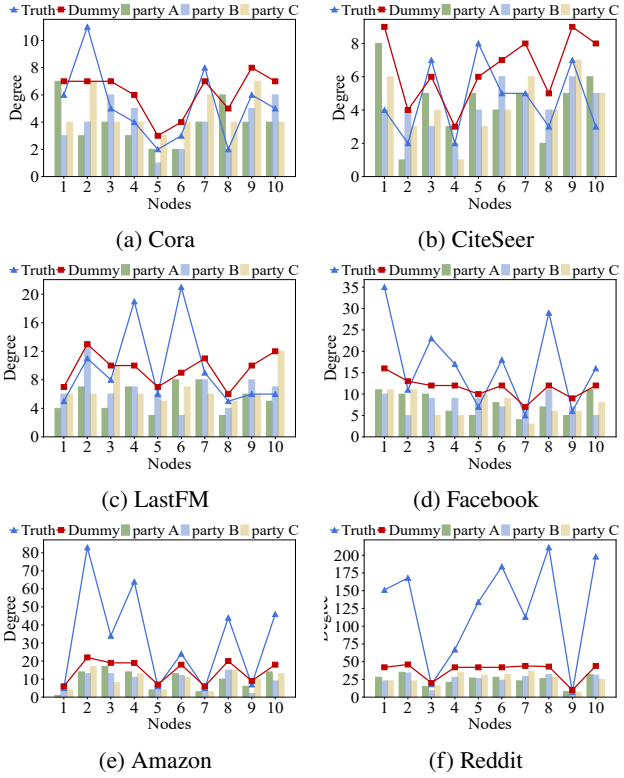
A.5 Proof of Theorem 5

Each module of DPA-GNN satisfies DP. By the *sequential composition* [15] of DP, DPA-GNN satisfies both edge-level ε_e-DP and node-level $(\varepsilon_e+\varepsilon_h+\varepsilon_y, 2\delta)$-DP.

B More Experimental Results

B.1 Degree Distribution

In this experiment, we randomly select ten nodes and present their true degrees, dummy degrees, and the degrees assigned to each party at $r = 0.5$ and $p = 2/3$. The results are displayed in Figure 11. The difference between the degrees recorded by each party and the true degrees (as observed on the user side) demonstrates the effective protection of node degrees.



(a) Cora (b) CiteSeer

(c) LastFM (d) Facebook

(e) Amazon (f) Reddit

Figure 11: The degree of 10 random selected nodes. The blue and red line represent true degrees and dummy degrees, and bar charts represent the degrees of three parties.

B.2 Effect of Scaling Factor

In this experiment, we evaluate the accuracy of DPA-GNN under varying scaling factors. Since the scaling factor α_i is calculated from the degree of the node d_i, we analyze the effects of different processed degrees, including the raw, clipped and dummy degrees. We set ε to $\{3,5,7,9,11\}$, with $\varepsilon_h = 5\%\varepsilon$ and $\varepsilon_e = 1.25$. The results, presented in Table 6, indicate that dummy degrees achieve higher accuracy on DPA-GNN compared to the other two types of degrees.

Table 6: Effect of scaling factor on the accuracy of DPA-GNN.

Dataset	α_i	3	5	7	9	11
Cora	Raw	40.3±6.37	64.4±3.42	72.3±1.93	74.7±2.16	76.3±2.46
	Clipped	46.1±5.75	66.5±4.26	74.5±1.61	76.2±0.98	78.7±1.56
	Dummy	**52.5**±2.53	**71.9**±3.02	**77.5**±1.76	**77.7**±1.02	**79.4**±1.38
CiteSeer	Raw	32.7±3.23	48.1±2.74	52.0±1.80	55.0±2.92	57.0±2.24
	Clipped	35.4±5.80	50.2±2.07	53.8±1.71	57.0±1.68	60.4±1.38
	Dummy	**36.1**±3.80	**52.7**±1.78	**56.0**±1.89	**59.0**±2.27	**61.2**±1.37
LastFM	Raw	64.1±4.93	74.3±2.34	74.1±3.35	75.0±0.79	76.8±2.18
	Clipped	67.9±2.72	76.5±2.86	78.5±2.09	79.7±1.13	80.2±1.28
	Dummy	**74.5**±3.75	**81.2**±1.40	**82.5**±0.62	**82.9**±0.93	**82.9**±1.04
Facebook	Raw	60.4±1.48	68.9±1.65	72.1±1.26	73.6±1.44	74.8±2.02
	Clipped	69.0±3.17	81.2±2.12	83.2±1.84	84.3±1.13	85.5±0.87
	Dummy	**73.7**±1.73	**84.8**±1.09	**86.1**±0.74	**86.5**±0.58	**87.0**±0.47
Amazon	Raw	64.7±1.19	71.5±0.67	74.3±0.84	74.3±1.11	74.9±0.55
	Clipped	75.5±1.44	83.6±0.67	85.4±0.72	85.7±0.76	86.2±0.47
	Dummy	**80.1**±1.63	**84.4**±0.79	**86.1**±0.60	**85.9**±0.76	**86.6**±0.66
Reddit	Raw	29.7±1.55	29.7±1.34	29.0±1.14	29.0±0.39	28.3±0.28
	Clipped	94.9±0.17	95.2±0.12	95.3±0.06	95.5±0.08	95.5±0.08
	Dummy	**95.8**±0.18	**96.1**±0.12	**96.2**±0.06	**96.3**±0.08	**96.3**±0.05

B.3 Effect of Distributed Noise

We evaluate the performance of distributed Gaussian noise and distributed Laplace noise on DPA-GNN.

(1) Distributed Gaussian noise [24]: Each of the m parties adds Gaussian noise $N(0,\sigma^2/m)$ with variance σ^2/m, which is used by DPA-GNN. The total noise is given by:

$$N(0,\sigma^2) = \sum_{\lambda=1}^{m} N_\lambda(0,\sigma^2/m). \tag{33}$$

(2) Distributed Laplace noise [18]: Each party adds Laplace noise $\sqrt{B_{m-1}}\mathcal{L}_\lambda(0,\sigma^2)$, using ℓ_1 sensitivity, where B_{m-1} follows a beta distribution. The total noise is given by:

$$\mathcal{L}(0,\sigma^2) = \sqrt{B_{m-1}}\sum_{\lambda=1}^{m}\mathcal{L}_\lambda(0,\sigma^2). \tag{34}$$

In this experiment, we compare the performance of DPA-GNN using distributed Laplace noise and Gaussian noise under identical settings. We set ε to $\{2,4,6,8,10\}$, with $\varepsilon_h = 5\%\varepsilon$ and $\varepsilon_e = 1.25$. The results are shown in Table 7. We observe that on small datasets like Cora, both types of distributed noise exhibit comparable performance. However, on larger datasets such as Facebook, distributed Gaussian noise achieves superior performance.

B.4 Evaluation of Running Time

We conduct experiments to evaluation the performance of DPA-GNN and GCN in term of running time.

- GCN [27]: GCN uses the adjacency matrix A and feature matrix X to aggregate the node representations.
- DPA-GNN: We set $\varepsilon_h = \infty, \varepsilon_y = \infty$, $t = 1$ and $m = 1$ without clipping, dummy and *ReLU* activation function.

Table 7: Effect of distributed noise on accuracy of DPA-GNN.

Dataset	Noise	2	4	6	8	10
Cora	Laplace	**31.1**±7.37	64.5±3.98	73.9±2.31	77.3±1.0	**79.1**±1.66
	Gaussian	27.1±3.97	**66.8**±2.07	**74.5**±1.95	**77.4**±1.94	78.4±1.42
CiteSeer	Laplace	**26.7**±3.33	**47.2**±1.71	**54.0**±1.08	**57.8**±1.89	60.2±1.51
	Gaussian	24.9±3.05	44.8±2.88	52.9±2.73	56.1±1.43	**60.5**±2.23
LastFM	Laplace	40.1±5.79	79.6±1.15	**82.9**±1.46	83.0±1.12	83.6±1.06
	Gaussian	**41.6**±8.73	**80.3**±1.46	82.7±0.86	**83.1**±1.06	**83.8**±1.01
Facebook	Laplace	44.4±2.7	78.9±1.48	83.7±0.87	85.4±0.7	85.9±0.59
	Gaussian	**54.8**±3.28	**81.1**±0.85	**84.9**±0.46	**85.8**±0.59	**86.1**±0.59
Amazon	Laplace	33.0±9.66	74.0±1.11	83.0±0.80	84.4±0.82	85.5±0.78
	Gaussian	**46.1**±4.98	**82.2**±1.05	**85.5**±0.66	**86.0**±0.70	**86.6**±0.68
Reddit	Laplace	22.2±2.86	92.7±0.76	95.5±0.22	96.0±0.24	95.9±0.18
	Gaussian	**91.2**±0.38	**96.0**±0.15	**96.1**±0.05	**96.2**±0.07	**96.3**±0.10

Table 8: Accuracy and running time of GCN and cleaned DPA-GNN.

Dataset	Method	Accuracy (%)	Runtime (s)
Cora	GCN	88.1±0.95	2.7±0.63
	DPA-GNN	88.1±1.06	3.2±0.04
CiteSeer	GCN	76.5±1.32	7.2±1.82
	DPA-GNN	74.8±0.94	8.3±0.22
LastFM	GCN	88.1±0.62	2.6±1.51
	DPA-GNN	86.9±1.02	11.9±0.09
Facebook	GCN	93.8±0.30	3.9±0.84
	DPA-GNN	93.2±0.26	33.5±0.15
Amazon	GCN	88.0±0.78	1.5±0.21
	DPA-GNN	85.6±0.89	20.3±0.46
Reddit	GCN	97.1±0.05	112.2±7.75
	DPA-GNN	96.7±0.09	3634.2±13.59

In this experiment, we compare the classification accuracy and running time of a two-layer GCN [27] and cleaned DPA-GNN. The results are shown in Table 8. DPA-GNN performs comparably to GCN across various datasets. However, as the dataset size increases, the running time of DPA-GNN grows significantly due to its dependence on the key-value data structure.

B.5 Effect of Feature Dimensionality Reduction

The communication costs of DPA-GNN are primarily driven by the number of feature dimensions. To further reduce these costs, we propose a straightforward dimensionality reduction technique that randomly removes feature dimensions.

We conduct experiments to evaluate the effect of feature dimensionality reduction, fixing $\varepsilon = 8$ and comparing the accuracy of DPA-GNN with different numbers of feature dimensions ranging from [10, 20, 50, 100, 200, 500] to the original size. The results are shown in Table 9. DPA-GNN achieves comparable performance to the original when the retained dimensions reach 3% to 13% of the original size, demonstrating that DPA-GNN can significantly reduce communication overhead through dimensionality reduction, while maintaining comparable performance.

Table 9: Effect of feature dimensionality reduction on accuracy of DPA-GNN.

	Dimensions					
Dataset	10	50	100	200	500	Original
Cora	51.5±4.28	74.1±2.92	76.1±1.00	76.7±2.36	77.5±1.60	77.4±1.94
CiteSeer	30.8±1.92	43.8±1.17	50.8±2.08	54.4±1.25	56.1±2.25	56.1±1.43
LastFM	66.9±2.16	80.3±0.84	81.6±0.37	82.1±1.08	83.0±0.64	83.1±1.06
Facebook	46.1±0.79	65.6±1.57	74.1±0.91	80.3±0.90	84.7±0.71	85.8±0.59
Amazon	75.5±1.18	82.8±0.60	84.8±1.03	85.3±0.39	85.6±0.32	85.6±0.43
Reddit	94.4±0.48	96.0±0.13	96.2±0.07	96.2±0.05	96.1±0.04	96.2±0.07

C Parameter Settings

The primary parameters of DPA-GNN include ε, ε_h, ε_e, ε_y, (l_h, l_y), c, r, and p, all of which collectively influence the trade-off between privacy and utility. A smaller privacy budget ε provides stronger privacy protection, where $\varepsilon = \varepsilon_h + \varepsilon_e + \varepsilon_y$. Allocating 5%$\varepsilon$ to ε_h has been shown to improve accuracy. ε_e is determined by the parameters r and p, and ε_y represents the remaining privacy budget. Aggregation operations help reduce noise [41], but excessive layers can cause over-smoothing [27]. Experiments show that for large datasets, the optimal number of feature aggregation layers l_h is in the range [4,6], with the number of label aggregation layers l_y in the range [2,4]. For small datasets, l_h is in the range [8,10], while l_y is in the range [6,10]. The clipping rate c determines the sensitivity to noise, with smaller values reducing the number of retained neighbors, thereby decreasing sensitivity. It is recommended to set the clipping rate c in the range of [0.2, 0.6] for large datasets, and in the range of [0.7, 0.9] for small datasets. The dummy parameter r is recommended to be set at 0.5, ensuring the number of dummies is approximately twice the number of nodes, which effectively balances privacy and utility. Although increasing the number of computing parties and secret shares does not directly impact the accuracy of DPA-GNN, it increases communication costs. Thus, the selective parameter p is recommended to be set to $2/3$. Based on our parameter tuning experience, we provide a recommended parameter configuration guideline, summarized in Table 10, as a reference for future DPA-GNN applications.

Table 10: Parameter setting guideline

Param.	Recommended setting
ε	range [4,10], 4(high privacy restrictions)
ε_h	recommended 5%ε
ε_e	1.25 (default), determined by the parameters r and p
ε_y	recommended range [2,6]
l_h	small datasets: range [8,10]; large datasets: range [4,6]
l_y	small datasets: range [6,10]; large datasets: range [2,4]
c	small datasets: [0.7,0.9]; large datasets: [0.2,0.6]
r	0.5 (default), recommended range [0.1,0.5]
p	2/3 (default), based on the number of parties and shares