Unveiling the Threat

Investigating Distributed and Centralized Backdoor Attacks in Federated Graph Neural Networks

Xu, Jing; Koffas, Stefanos; Picek, Stjepan

DOI
10.1145/3633206

Publication date
2024

Document Version
Final published version

Published in
Digital Threats: Research and Practice

Citation (APA)
Xu, J., Koffas, S., & Picek, S. (2024). Unveiling the Threat: Investigating Distributed and Centralized Backdoor Attacks in Federated Graph Neural Networks. *Digital Threats: Research and Practice*, *5*(2), Article 15. https://doi.org/10.1145/3633206

Unveiling the Threat: Investigating Distributed and Centralized Backdoor Attacks in Federated Graph Neural Networks

JING XU and STEFANOS KOFFAS, Delft University of Technology, The Netherlands
STJEPAN PICEK, Radboud University, The Netherlands

15

Graph neural networks (GNNs) have gained significant popularity as powerful deep learning methods for processing graph data. However, centralized GNNs face challenges in data-sensitive scenarios due to privacy concerns and regulatory restrictions. Federated learning has emerged as a promising technology that enables collaborative training of a shared global model while preserving privacy. Although federated learning has been applied to train GNNs, no research focuses on the robustness of Federated GNNs against backdoor attacks.

This article bridges this research gap by investigating two types of backdoor attacks in Federated GNNs: centralized backdoor attack (CBA) and distributed backdoor attack (DBA). Through extensive experiments, we demonstrate that DBA exhibits a higher success rate than CBA across various scenarios. To further explore the characteristics of these backdoor attacks in Federated GNNs, we evaluate their performance under different scenarios, including varying numbers of clients, trigger sizes, poisoning intensities, and trigger densities. Additionally, we explore the resilience of DBA and CBA against two defense mechanisms. Our findings reveal that both defenses cannot eliminate DBA and CBA without affecting the original task. This highlights the necessity of developing tailored defenses to mitigate the novel threat of backdoor attacks in Federated GNNs.

CCS Concepts: • **Security and privacy**; • **Computing methodologies** → **Machine learning**;

Additional Key Words and Phrases: Backdoor attacks, graph neural networks, federated learning

ACM Reference format:
Jing Xu, Stefanos Koffas, and Stjepan Picek. 2024. Unveiling the Threat: Investigating Distributed and Centralized Backdoor Attacks in Federated Graph Neural Networks. *Digit. Threat. Res. Pract.* 5, 2, Article 15 (June 2024), 29 pages.
https://doi.org/10.1145/3633206

1 INTRODUCTION

Graph Neural Networks (GNNs), which generalize traditional **Deep Neural Networks (DNNs)** to graph data, pave a new way to effectively learn representations for complex graph-structured data [46]. Due to their strong representation learning capability, GNNs have demonstrated remarkable performance in various domains, such as drug discovery [27, 49], finance [8, 42], social networks [11, 16], and recommendation systems [10, 55]. Usually, GNNs are trained through centralized training. However, because of privacy concerns, regulatory restrictions, and commercial competition, GNNs can also face challenges when centrally trained. For example, the financial institution may utilize GNN as a fraud detection model, but they can only have transaction data of its registered users (no data of other users because of privacy concerns). Thus, the model is not effective for other users.

Authors' addresses: J. Xu (Corresponding author) and S. Koffas, Delft University of Technology, Delft, The Netherlands; e-mails: j.xu-8@tudelft.nl, s.koffas@tudelft.nl; S. Picek, Radboud University, Nijmegen, The Netherlands; e-mail: stjepan.picek@ru.nl.

2576-5337/2024/6-ART15 $15.00
https://doi.org/10.1145/3633206

Similarly, in a drug discovery industry that applies GNNs, pharmaceutical research institutions can dramatically benefit from other institutions' data, but they cannot disclose their private data for commercial reasons [19].

Federated Learning (FL) is a distributed learning paradigm that works on isolated data. In FL, clients can collaboratively train a shared global model under the orchestration of a central server while keeping the data decentralized [22, 31]. As such, FL is a promising solution for training GNNs over isolated graph data, and there are already some works utilizing FL to train GNNs [19, 25, 57], which we denote as *Federated GNNs*.

Although FL has been successfully applied in diverse domains, such as computer vision [28, 29] or language processing [18, 61], there could be malicious clients among millions of clients, leading to various adversarial attacks [2, 12]. In particular, limited access to local clients' data due to privacy concerns or regulatory constraints may facilitate backdoor attacks on the global model trained in FL. A backdoor attack is a type of poisoning attack that manipulates part of the training dataset with a specific pattern (trigger) such that the model trained on the manipulated dataset will misclassify the testing dataset with the same trigger pattern [30].

Backdoor attacks on FL have been studied recently [2, 4, 48]. However, these attacks are applied in FL on Euclidean data, such as images and words. The backdoor trigger generation methods and injecting position are different between graph data and images/words [52]. In particular, Xie et al. [48] split a square-shaped trigger placed in the top left corner of an image into four parts so that four malicious clients use each part in their poisoned datasets. When the training ends, the adversary concatenates these parts to form a global trigger in the image's upper left corner that activates the backdoor. This is impossible in GNNs as the data is not Euclidean, and there is no position that we can exploit. In addition, defenses like FoolsGold [13] filter out clients that use similar updates as malicious. This can be effective for Euclidean data that use parts of the trigger in similar positions but may not be effective in GNNs. Indeed, the graph data is not Euclidean and different partial triggers vary the graph structure resulting in non-aligned updates. As well, intensive research has been conducted on backdoor attacks in GNNs [47, 52, 59]. However, these works focus on GNN models in centralized training. In FL, the malicious updates will be weakened in the aggregation function. Finally, there can be more than one malicious client, whereas in centralized GNNs, there is only one client. Thus, we should expect different behavior of backdoor attacks in Federated GNNs. Then, it is crucial to investigate if existing countermeasures that have been tested mostly with Euclidean data are still effective for backdoor attacks in Federated GNNs to understand how to deploy trustworthy AI systems.

This article conducts two backdoor attacks in FL: **Centralized Backdoor Attack (CBA)** and **Distributed Backdoor Attack (DBA)** [48]. In CBA, the attacker embeds the same global trigger in all adversarial clients, whereas in DBA, the adversary decomposes the global trigger into several local triggers and embeds them in different malicious clients. In DBA, we assume two attack scenarios—honest majority and malicious majority—to explore the impact of the percentage of malicious clients on the attack. Our work focuses on the cross-silo FL setting, and our main contributions are as follows:

— We explore two types of backdoor attacks in Federated GNNs. Based on the experiments, we find that the DBA on Federated GNNs is more effective or (at least) similar to the CBA. To the best of our knowledge, this article is the first work studying backdoor attacks in Federated GNNs.
— We conduct extensive experiments on real-world and synthetic datasets, including two molecular structure datasets, one synthetic dataset, and two social network datasets.
— We find that in the CBA, although the adversarial local model is implanted with the global trigger, the final global model can also attain promising attack performance with any local trigger. Since this phenomenon is inconsistent with the related works, we provide further experiments to explain it.
— We observe that in most cases, local triggers in DBA can achieve similar attack performance to the global trigger, which is different from the findings for the DBA in convolutional neural networks.
— We run experiments for both types of attacks, varying the trigger size, poisoning intensity, and trigger density, and show that the trigger size has more impact than the poisoning intensity.

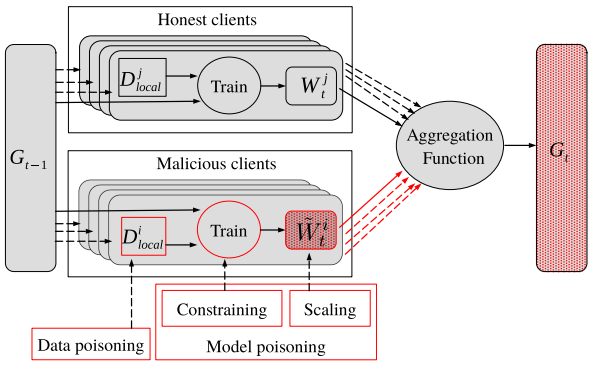


Fig. 1. Overview of backdoor attacks on FL.

— We explore the robustness of DBA and CBA against two defenses: FoolsGold and **Robust Learning Rate (RLR)**. We find that both attacks are evasive to FoolsGold, whereas CBA can even obtain a higher **Attack Success Rate (ASR)**, but the testing accuracy degrades. In addition, under RLR, the performance in the original main task is significantly degraded.

This article is an extended version of the paper with the title "More Is Better (Mostly): On the Backdoor Attacks in Federated Graph Neural Networks," published at ACSAC 2022 [51].

2 RELATED WORK

2.1 Backdoor Attacks in FL

Backdoor attacks aim to make a model misclassify inputs with triggers to preset-specific label(s). Specifically, attackers poison the model by injecting triggers into the training data, and they can activate the backdoor in the test phase. When the backdoor is not activated, the attacked model has performance similar to the normal model. Once activated, the model's output becomes the targeted label pre-specified by the attacker to achieve the malicious intent purpose (e.g., targeted misclassification). Backdoor attacks can occur in many scenarios where the training process is not fully controlled, such as using third-party datasets, using third-party platforms for training, and directly calling third-party models, thus posing a threat to the security of the model.

Backdoor attacks are common in FL systems involving multiple training dataset owners. In such attacks, the adversary $\mathcal{A}$ manipulates one or more local models to generate poisoned models, denoted as $\tilde{W}^i$, which are then aggregated into the global model G_t, thereby compromising its properties. There are two common techniques used in backdoor attacks in FL, as shown in Figure 1: (1) data poisoning, where $\mathcal{A}$ manipulates local training dataset(s) D^i_{local} used to train the local model [34, 48], and (2) model poisoning, where $\mathcal{A}$ manipulates the local training process or the trained local models themselves [2]. Regarding data poisoning backdoor attacks in FL, during the local training phase, one or more malicious clients can inject triggers into local benign datasets to produce backdoored datasets. By training on the backdoored datasets, malicious updates can be obtained. Consequently, if the server aggregates with these malicious updates, the global model will exhibit misclassification on the samples with the injected triggers. In the model poisoning backdoor attacks in FL, to enhance the effect of the attacks, the adversaries can also use the method of scaling [2] to increase their weight. In this work, we focus on data poisoning for our attacks in Federated GNNs as model poisoning requires multiplying large factors to model weights when conducting attacks, which can be detected by traditional Byzantine-robust aggregation rules such as Median [54] and Krum [5].

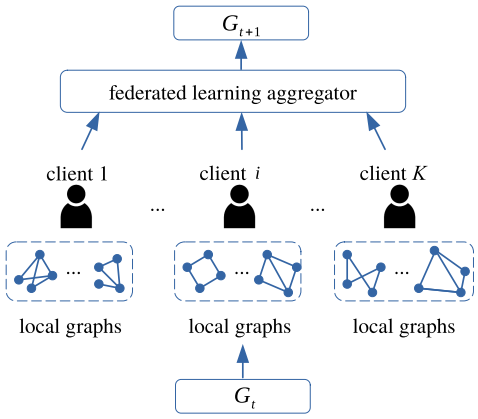


Fig. 2. Framework of federated GNNs for a graph-level task. Each client trains its local GNN model based on local graphs, and the FL aggregator aggregates the local models to obtain the global model. G_t and G_{t+1} define the global models at iterations t and $t+1$, respectively. K is the number of clients.

2.2 Backdoor Attacks in GNNs

Several recent works have conducted backdoor attacks on GNNs. Zhang et al. [59] proposed a subgraph-based backdoor attack on GNNs for the graph classification task. Xi et al. [47] presented a subgraph-based backdoor attack on GNNs that works for both node classification and graph classification tasks. Xu et al. [52] investigated the explainability of the impact of the trigger injecting position on the performance of backdoor attacks on GNNs and proposed a new backdoor attack strategy for the node classification task. Xu et al. [50] further conducted an analysis to explore and explain the varying attack performance achieved by injecting triggers into the most important or least important areas within a sample using explanation techniques. This study results in a further understanding of backdoor attacks in GNNs [50]. All current attacks are implemented in centralized training for GNNs.

2.3 FL on GNNs

FL has gained increasing attention as a training paradigm where data is distributed at remote devices, and models are collaboratively trained in a central server. FL has been widely studied in Euclidean data (e.g., images, texts, and sound); however, there are increasing studies about FL in graph data. Figure 2 illustrates the framework of federated GNNs for a graph-level task. FL on graph data was introduced in the work of Lalitha et al. [26], where each client is regarded as a node in a graph. When it comes to detecting financial crimes (e.g., fraud or money laundering), traditional machine learning tends to lead to severe over-reporting of suspicious activities. Thanks to the reasoning ability of the GNN, its advantages can be well reflected. Considering the need for privacy, Suzumura et al. [40] proposed the framework for Federated GNNs to optimize the machine learning model. Besides, other research works [21, 45, 60] have been dedicated to enhancing the security of Federated GNNs. By using secure aggregation, Jiang et al. [21] proposed a method to predict the trajectories of objects via aggregating both spatial and dynamic information without information leakage. With differential privacy, Zhou et al. [60] and Wu et al. [45] put forward a framework to train Federated GNNs for vertical FL and recommendation systems, respectively. Moreover, SpreadGNN was proposed in the work of He et al. [20] to perform FL without a server. Although there is an increasing number of works on FL for graph data, the vulnerability of Federated GNNs to backdoor attacks is still under-explored.

3 PROBLEM FORMULATION

FL is a practical choice to push machine learning to users' devices, such as smart speakers, cars, and phones. Usually, FL is designed to work with thousands or even millions of users without restrictions on eligibility [2], opening up new attack vectors. As stated in the work of Bonawitz et al. [6], training with multiple malicious clients is now considered a practical threat by the designers of FL. Because of the data privacy guarantee among the clients in the FL, local clients can modify their local training dataset without being noticed. Furthermore, existing FL frameworks do not provide a functionality to verify whether the training on local clients has been finished correctly. Consequently, one or more clients can submit their malicious models trained for the assigned task and backdoor functionality.

3.1 Preliminaries on GNNs

Before defining GNN, we first introduce the concept of a graph. Let $G = (V, E, X)$ be a graph, where V, E, X denote nodes, edges, and node features. Unlike Euclidean data (e.g., images), in graphs, one cannot define an ordering of nodes. Thus, we cannot use convolutional neural networks directly on graphs as the inputs of the matrix (e.g., adjacency and feature matrixes) of the graph will change if we label the nodes with different orderings at different times. Due to the non-Euclidean nature of the graph data, GNNs have been proposed to work on graph data and have achieved significant success in many real-world scenarios. GNNs take a graph $G = (V, E, X)$ as input and learn a representation vector (embedding) for each node $\boldsymbol{v} \in G$, $z_{\boldsymbol{v}}$, or the entire graph, z_G.

Modern GNNs follow a neighborhood aggregation strategy, where one iteratively updates the representation of a node by aggregating representations of its neighbors. After k iterations of aggregation, a node's representation captures both structure and feature information within its k-hop network neighborhood [53]. Formally, the k-th layer of a GNN is

$$z_{\boldsymbol{v}}^{(k)} = \sigma(z_{\boldsymbol{v}}^{(k-1)}, AGG(\{z_{\boldsymbol{u}}^{(k-1)}; \boldsymbol{u} \in \mathcal{N}_{\boldsymbol{v}}\})), \forall k \in [K], \tag{1}$$

where $z_{\boldsymbol{v}}^{(k)}$ is the representation of node $\boldsymbol{v}$ computed in the k-th iteration. $\mathcal{N}_{\boldsymbol{v}}$ are 1-hop neighbors of node $\boldsymbol{v}$, and the $AGG(\cdot)$ is an aggregation function that can vary for different GNN models. $z_{\boldsymbol{v}}^{(0)}$ is initialized as node feature,[1] whereas σ is an activation function.

In this article, we focus on three representation models of GNN models, which differ in the preceding neighborhood aggregation strategy. Specifically, for **Graph Convolutional Networks (GCNs)** [23], the aggregation operation in GCN is given as

$$x_{\boldsymbol{v}}^{(k)} \leftarrow \sum_{u \in \mathcal{N}_{\boldsymbol{v}} \bigcup \boldsymbol{v}} \frac{1}{\sqrt{d_{\boldsymbol{v}} d_{\boldsymbol{u}}}} z_{\boldsymbol{u}}^{(k-1)}, \tag{2}$$

where $d_{\boldsymbol{v}}$ denotes the degree (number of neighbors) of node $\boldsymbol{v}$.

Graph Attention Networks (GATs) [41] applies a non-standard neighbor aggregation scheme: weighted average via attention. Given a shared attention mechanism a, attention coefficients can be computed by

$$e_{\boldsymbol{vu}} = a(W z_{\boldsymbol{v}}^{(k-1)}, W z_{\boldsymbol{u}}^{(k-1)}) \tag{3}$$

that indicate the importance of node $\boldsymbol{u}$'s features to node $\boldsymbol{v}$. Then, the normalized coefficients can be computed by using the softmax function:

$$\alpha_{\boldsymbol{vu}} = softmax_{\boldsymbol{u}}(e_{\boldsymbol{vu}}). \tag{4}$$

Finally, the next-level feature representation of node $\boldsymbol{v}$ is

$$z_{\boldsymbol{v}}^{(k)} = \sigma\left(\frac{1}{P}\sum_{p=1}^{P}\sum_{\boldsymbol{u} \in \mathcal{N}_{\boldsymbol{v}}} \alpha_{\boldsymbol{vu}}^{p} W^{p} z_{\boldsymbol{u}}^{(k-1)}\right), \tag{5}$$

[1] Usually, we use the 1-hot degree of nodes as the initial node feature.

where α_{vu}^{p} are the normalized coefficients computed by the p-th attention mechanism a^p and W^p is the corresponding input linear transformation's weight matrix.

GraphSAGE (Graph Sample and Aggregate) [17] is an extension of the GCN framework to the inductive setting, and there are three candidate aggregator functions in the GraphSAGE algorithm: mean aggregator, LSTM aggregator, and pooling aggregator.

For the graph classification task (considered in this work), the READOUT function pools the node representations for a graph-level representation z_G:

$$z_G = READOUT(z_v; v \in V). \tag{6}$$

READOUT can be a simple permutation invariant function such as summation or a more sophisticated graph-level pooling function [56, 58].

3.2 Threat Model

Unlike traditional machine learning benchmarking datasets, graph datasets, and real-world graphs may exhibit non-i.i.d. (non-independent and identical distribution) due to factors like structure and feature heterogeneity [19]. Therefore, following the FL assumptions, we assume that graphs among K clients are non-i.i.d. distributed. The clients engaging in training can be divided into honest and malicious clients. All clients strictly follow the FL training process, but the malicious client(s) will inject graph trigger(s) into their training graphs. We also assume the server is conducting model aggregation correctly. Our primary focus is to investigate backdoor attack effectiveness on Federated GNNs, so we adopt two backdoor attack methods as defined next (the definitions of the local trigger and global trigger used in these two attacks are also given).

Definition 1 (Local Trigger and Global Trigger). The local trigger is the specific graph trigger for each malicious client in DBA. The global trigger is the combination of all local triggers.[2]

Definition 2 (Distributed Backdoor Attack (DBA)). There are multiple malicious clients, and each of them has its local trigger. Each malicious client injects its local trigger into its training dataset. All malicious clients have the same backdoor task. An adversary $\mathcal{A}$ conducts DBA by compromising at least two clients in FL.

Definition 3 (Centralized Backdoor Attack (CBA)). A global trigger consisting of local triggers is injected into one client's local training dataset. An adversary $\mathcal{A}$ conducts CBA by usually compromising only one client in FL.

Adversary's Capability. We assume the adversary $\mathcal{A}$ can corrupt M ($M \leq K$) clients to perform DBA. We perform a complete attack in every round—that is, a poisoned local dataset is used by malicious clients in every round, following the attack setting in the work of Xie et al. [48]. The adversary cannot impact the aggregation process on the central server nor the training or model updates of other clients.

Adversary's Knowledge. We assume that the adversary $\mathcal{A}$ knows the compromised clients' training dataset. In this context, the adversary can generate local triggers as described in Section 4.2. Additionally, we follow the original assumptions of FL. The number of clients participating in training, model structure, aggregation strategy, and a global model for each iteration is revealed to all clients, including malicious clients.

Adversary's Goal. Unlike some non-targeted attacks [37] aiming to deteriorate the accuracy of the model, the backdoor attacks studied in this article aim to make the global model misclassify the backdoored data samples into specific pre-determined labels (i.e., target label y_t) without affecting the accuracy on clean data.

In DBA, each malicious client injects its local trigger into its local training dataset to poison the local model. Therefore, DBA can fully leverage the power of FL in aggregating dispersed information from local models

[2]Since it is an NP-hard problem to decompose a graph into subgraphs [9], we first generate local triggers and then compose them to get the global trigger used in CBA.

to train a poisoned global model. Assuming there are M malicious clients in DBA, each has its local trigger. Each malicious client i in DBA independently implements a backdoor attack on its local model. The adversarial objective for each malicious client i is

$$w_t^{i*} = \underset{w_t^i}{\operatorname{argmin}} \left(\sum_{j \in D^i_{trigger}} \ell(w^i_{t-1}(\Phi(x^i_j, \kappa^i), y_t)) + \sum_{j \in D^i_{clean}} \ell(w^i_{t-1}(x^i_j), y^i_j) \right), \forall i \in [M], \tag{7}$$

where the poisoned training dataset $D^i_{trigger}$ and clean training dataset D^i_{clean} satisfy $D^i_{trigger} \cup D^i_{clean} = D^i_{local}$ and $D^i_{trigger} \cap D^i_{clean} = \varnothing$. D^i_{local} is the local training dataset of client i. Φ is the function that transforms the clean data with a non-target label into poisoned data using a set of trigger generation parameters κ^i. In this work, κ^i consists of trigger size s, trigger density ρ, and poisoning intensity r: $\kappa = \{s, \rho, r\}$:

- ***Trigger size*** s: The number of nodes of a local graph trigger. Here, we set the trigger size s as the γ fraction of the graph dataset's average number of nodes. Note that this does not violate our threat model (the adversary does not have access to the whole dataset), as the average number of nodes in the local dataset is similar to that of the whole dataset.
- ***Trigger density*** ρ: The complexity of a local graph trigger, which ranges from 0 to 1, and is used in the Erdős-Rényi model to generate the graph trigger.
- ***Poisoning intensity*** r: The ratio that controls the percentage of backdoored training dataset among the local training dataset.

Unlike DBA with multiple malicious clients, there is only one malicious client in CBA.[3] CBA is conducted by embedding a global trigger into a malicious client's training dataset. The global trigger is a graph consisting of local trigger graphs used in DBA, as explained further in Section 4.1. Thus, the adversarial objective of the attacker k in round t in CBA is

$$w_t^{k*} = \underset{w_t^k}{\operatorname{argmin}} \left(\sum_{j \in D^k_{trigger}} \ell(w^k_{t-1}(\Phi(x^k_j, \kappa), y_t)) + \sum_{j \in D^k_{clean}} \ell(w^k_{t-1}(x^k_j), y^k_j) \right), \tag{8}$$

where κ is the combination of κ^i. Utilizing the power of FL in message passing from local models to the global model, the global model is supposed to inherit the backdoor functionality.

4 BACKDOOR ATTACKS AGAINST FEDERATED GNNS

4.1 General Framework

This section gives a general design description of backdoor attacks against Federated GNNs. We focus on subgraph-based (data poisoning) backdoor attacks and the graph classification task. Attackers can perform DBA or CBA as shown in Figure 3. In DBA, multiple malicious clients engage in attacking, and they inject local triggers

[3]In practice, the centralized attack can poison more than one client with the same global trigger, as mentioned in the work of Bagdasaryan et al. [2]. Here, we assume there is one malicious client.

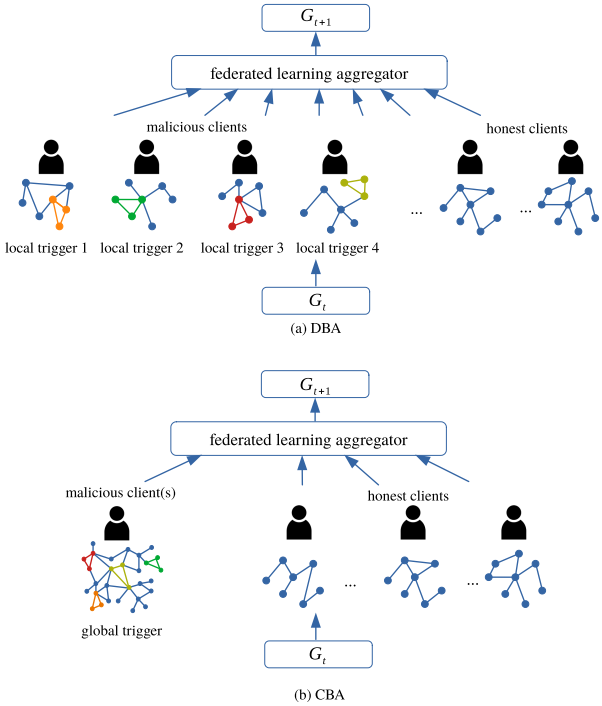


Fig. 3. Attack framework.

into corresponding malicious clients' local training datasets.[4] CBA is conducted with one malicious client whose training data is poisoned with the global trigger that consists of the local triggers used in DBA. We describe the notations used throughout the article in Table 1.

Distributed Backdoor Attack. For DBA in Federated GNNs, we assume there are M ($M \leq K$) malicious clients among K clients, as shown in Figure 3(a). Each malicious client embeds its local training dataset with a specific graph trigger to poison its local model. For instance, in Figure 3(a), each malicious client has a local trigger highlighted by a specific color (i.e., orange, green, red, yellow).[5] In this work, we did not use the same local trigger for different malicious clients in DBA as it would mean poisoning intensity for this specific local trigger is increasing, but simultaneously, the total trigger pattern activating the backdoor is reduced. We evaluated this setting by running some additional experiments, and we found the attack under this setting is not stronger than the current setting (i.e., different local triggers). Through training with these poisoned training datasets, the poisoned local models are uploaded to the server to update the global model. The final adversarial goal is to use the global trigger to attack the global model. Algorithms 1 and 2 illustrate the DBA in Federated GNNs. We first split the clients into two groups: the honest (C_h) and the malicious one (C_m) (line 2, Algorithm 1). In each round, each client updates its weights through local training (line 13, Algorithm 1), and finally, the global server aggregates local models' weights to update the global model through averaging (line 15, Algorithm 1).

[4]In the FL setting, each client has no access to other clients' local training dataset. Thus, each malicious client has its local trigger, which is not available for other malicious clients.

[5]Although we use the triangle as the graph trigger for each malicious client, in practice, the local triggers are more complex and different from each other.

Table 1. Notations Used in This Article

Notation	Description
y_t	target label
G_t	joint global model at round t
E	local epochs
K	number of clients
M	number of malicious clients
C_h, C_m	honest clients, malicious clients
D_{local}	client's local training dataset split from dataset D_{train}
D_{test}	testing dataset split from dataset D
t_{global}	global trigger
t_{local}	local trigger
w_t^k	client k's local trained model at round t
r	poisoning ratio
s	number of nodes in graph trigger
γ	ratio of the trigger size over the graph dataset's average number of nodes
ρ	edge existence probability in graph trigger
$D_{trigger}$	dataset with trigger embedded
D_{clean}	clean training dataset
$D_{backdoor}$	backdoored training dataset
B	local minibatch size
η	learning rate

The local training for every client is described in Algorithm 2. If the client is malicious (line 2, Algorithm 2), the local training dataset will be backdoored (line 4, Algorithm 2) with the local trigger (line 3, Algorithm 2). As mentioned in Section 3.2, all the local triggers form the global trigger (line 5, Algorithm 2).

We conduct experiments for the malicious majority and honest majority settings to explore the impact of different percentages of malicious clients on the ASR. We provide additional motivation for the malicious majority setting in Section 2.

Centralized Backdoor Attack. Unlike DBA conducted with multiple malicious clients, CBA performs the attack with only one malicious client. CBA is a general approach in a centralized learning scenario. For example, in image classification, the attacker poisons the training dataset with a trigger so that the model misclassifies the data sample with the same trigger into the attacker-chosen label. As shown in Figure 3(b), the malicious client embeds its training dataset with the global trigger highlighted by four colors. This global trigger consists of local triggers used in DBA, as shown in line 5 of Algorithm 2. Specifically, the attacker in CBA embeds its training data with four local patterns, together constituting a complete global pattern as the backdoor trigger.[6]

To compare the attack performance between the DBA and CBA in Federated GNNs, we need to make sure the trigger pattern in CBA is the union set of local trigger patterns in DBA. We can use two strategies: (1) first generate local triggers in DBA and then combine them to get the global trigger, or (2) first generate a global trigger in CBA and then divide it into M local triggers. We utilize the first strategy as it is an NP-hard problem to divide a graph into several subgraphs [9]. Thus, in different attack scenarios (i.e., honest majority or malicious majority attack scenarios), the CBA performance is different since the global trigger has been changed due to the different number of malicious clients.

[6]Here, the four colors are only used to denote four trigger patterns.

ALGORITHM 1: DBA in Federated GNNs

Input: Dataset D, Target label y_t
Output: Backdoored Global model G_{t+1}, global trigger t_{global}

Function `DBA()`:
  $C_h, C_m \leftarrow ClientSplit(Clients)$
  $D_{local}, D_{test} \leftarrow DataSplit(D)$
  $t_{global} \leftarrow \emptyset$
  Server executes:
  initialize G_0, $f = False$
  foreach *round* $t = 0, 1, 2, \ldots$ **do**
    foreach *client* $k \in (C_h \cup C_m)$ **do**
      $w_t^k = G_t$
      if $k \in C_m$ **then**
        $f = True$
      end
      $w_{t+1}^k \leftarrow ClientUpdate(k, w_t^k, f, t_{global})$
    end
    $G_{t+1} \leftarrow \sum_{k=1}^{K} \frac{w_{t+1}^k}{K}$
  end
End Function
return G_{t+1}, t_{global}

ALGORITHM 2: ClientUpdate

Input: Client k, Local training dataset D_{local}, Current global model w, flag f, global trigger t_{global}
Output: Updated model w

Function `ClientUpdate()`:
  if f *is* $True$ **then**
    $t_{local} \leftarrow GenerateTrigger(s, \rho)$
    $D_{local} \leftarrow BackdoorDataset(D_{local}, t_{local}, y_t)$
    $t_{global} = t_{global} \cup t_{local}$
  end
  $\mathcal{B} \leftarrow$ (split D_{local} into batches of size B)
  foreach *local epoch* i *from* 1 *to* E **do**
    foreach $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \bigtriangledown l(w, b)$
    end
  end
End Function
return w

4.2 Backdoored Data Generation

We adopt the Erdős-Rényi model [14] to generate triggers (function *GenerateTrigger* in Algorithm 2) as it is more effective than the other methods (e.g., Small World model [44] or Preferential Attachment model [3]) [59]. In particular, *GenerateTrigger* (line 3 in Algorithm 2) creates a random graph of s nodes. An edge between a pair of nodes in this graph is generated with probability ρ.

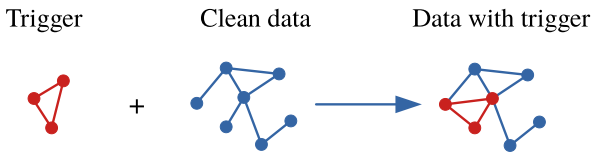


Fig. 4. Add trigger into sampled data.

Backdoored data is generated (line 4 in Algorithm 2) through the following process, as illustrated in Algorithm 3. We sample subsets of the local training datasets (with non-target labels) with proportion r, and the rest are saved as clean datasets. For each sampled data, we inject a trigger into it by sampling s (trigger size) nodes from the graph uniformly at random and replacing their connection with that in the trigger graph, as shown in Figure 4. The node features of the trigger graph are the nodes' 1-hot degree. Additionally, the attacker relabels the sampled data with an attacker-chosen target label. The backdoored data is composed of the dataset with trigger and the original clean dataset.

ALGORITHM 3: BackdoorDataset

Input: Local Training Dataset $D_{local} = \{x_i, y_i\}_{i=1}^{S}$, Target label $y_t \in [0, C)$, local trigger t_{local}
Output: Backdoored Training Dataset $D_{backdoor}$

1 **Function** `BackdoorDataset()`:
2 $D_{trigger} \leftarrow \emptyset$
3 $D_{tmp} \leftarrow sample(D_{local}, r, y \neq y_t)$
4 $D_{clean} = \{data \in D_{local} : data \notin D_{tmp}\}$
5 **foreach** $d \in D_{tmp}$ **do**
6 $x = AddTrigger(d[x], t_{local})$
7 $y = y_t$
8 $D_{trigger} = D_{trigger} \cup \{x, y\}$
9 **end**
10 **End Function**
11 $D_{backdoor} = D_{clean} \bigcup D_{trigger}$
12 **return** $D_{backdoor}$

5 EXPERIMENTS

In Table 2,[7] we summarize the settings of different experiments shown in Section 5. Molecular machine learning is a paramount application in the Federated GNNs, where many small graphs are distributed between multiple institutions [19]. Therefore, we run experiments (Exp. I and Exp. II) on two molecular datasets: NCI1 and PROTEINS_full. For these experiments, we set five clients in total because, with more clients, the local dataset of each client becomes very small, resulting in severe overfitting for the local models. Similar settings and phenomena can also be found in prior works on Federated GNNs [19]. The choice of small datasets may be a limitation of our work, but real-world cross-silo settings could involve only a few different organizations (from 2 to 100) [22]. Besides the molecular domain, substantial attention has also been given to Federated GNNs in real-world financial scenarios [43, 57]. In such scenarios, clients can be different organizations (e.g., banks), and a GNN model is trained on siloed data, leading to a cross-silo FL setting [22]. As shown in Exp. III and Exp. IV, we assume 10,

[7]Exp. I, Exp. II, Exp. III, and Exp. IV represent the experiments of the honest majority attack scenario, malicious majority attack scenario, the impact of the number of clients, and the impact of the percentage of malicious clients, respectively.

Table 2. Summary of the Experimental Setting (K: Number of Clients, M: Number of Malicious Clients)

Experiment	Dataset	K	M
Exp. I	NCI1, PROTEINS_full, TRIANGLES	5	2
Exp. II	NCI1, PROTEINS_full, TRIANGLES	5	3
Exp. III	TRIANGLES	10	4, 6
		20	8, 12
Exp. IV	TRIANGLES	100	5, 10, 15, 20
Prior work [19]	Molecules	4	0

Table 3. Dataset Statistics

Dataset	# Graphs	Avg. # Nodes	Avg. # Edges	Classes	Class Distribution
NCI1	4,110	29.87	32.30	2	2,053[0], 2,057[1]
PROTEINS_full	1,113	39.06	72.82	2	663[0], 450[1]
TRIANGLES	45,000	20.85	32.74	10	4,500[0–9]

20, and 100 clients for a synthetic dataset (i.e., TRIANGLES), which is a realistic real-world cross-silo scenario [39].

5.1 Experimental Setting

We implemented FL algorithms using the PyTorch framework. All experiments were run on a server with two Intel Xeon CPUs and one NVIDIA 1080 Ti GPU with 32 GB of RAM. Each experiment was repeated 10 times to obtain the average result.

Datasets. We run experiments on three publicly available datasets: two molecular structure datasets (NCI1 [32] and PROTEINS_full [7]) and one synthetic dataset (TRIANGLES [24]), which is a multi-class dataset. Table 3 provides more information about these datasets.

Dataset Splits. For each dataset, we randomly sample 80% of the data instances as the training dataset and the rest as the test dataset. To simulate non-i.i.d. training data and supply each participant with an unbalanced sample from each class, we further split the training dataset into K parts following the strategy in the work of Fang et al. [12] with hyperparameter 0.5 for TRIANGLES (10 classes) and hyperparameter 0.7 for other datasets (2 classes). In this work, we set trigger factors as follows: $\gamma = 0.2$, $\rho = 0.8$, and $r = 0.2$. By choosing them, we model a strong adversary that helps in evaluating the attack's behavior in the worst-case scenario.

Models and Metrics. In our experiments, we use three state-of-the-art GNN models: GCN [23], GAT [41], and GraphSAGE [17].

We use the ASR to evaluate the attack effectiveness, as shown in Algorithm 4. We embed the testing dataset with local triggers or the global trigger and then calculate the ASR of the global model on the poisoned testing dataset. We only embed the testing dataset of the non-target label with triggers to avoid the influence of the original label. The ASR measures the proportion of trigger-embedded inputs that are misclassified by the backdoored GNN into the target class y_t chosen by the adversary. The trigger-embedded inputs are

$$D_{g_t} = \left\{(G_{1,g_t}, y_1), (G_{2,g_t}, y_2), \ldots, (G_{n,g_t}, y_n)\right\}.$$

Here, g_t is the graph trigger, $\left\{G_{1,g_t}, G_{2,g_t} \ldots, G_{n,g_t}\right\}$ is the test dataset embedded with graph trigger g_t, and $y_1, y_2, \ldots, y_n$ is the label set. Formally, ASR is defined as

$$Attack\ Success\ Rate = \frac{\sum_{i=1}^{n} \mathbb{I}(G_{backdoor}(G_{i,g_t}) = y_t)}{n}, \tag{9}$$

where $\mathbb{I}$ is an indicator function and $G_{backdoor}$ refers to the backdoored global model. Here, the graph trigger g_t can be local triggers or a global trigger.

We use the **Clean Accuracy Drop (CAD)** to measure the effect of the backdoor attack on the original task. It is calculated by comparing the performance of the backdoored and clean models on a clean testing set. The accuracy drop should generally be small to keep the attack stealthy. Given the clean inputs

$$D_{clean} = \{(G_1, y_1), (G_2, y_2), \ldots, (G_n, y_n)\},$$

CAD is defined as

$$Clean\ Accuracy\ Drop = \frac{\sum_{i=1}^{n} \mathbb{I}(G_{clean}(G_i) = y_i)}{n} - \frac{\sum_{i=1}^{n} \mathbb{I}(G_{backdoor}(G_i) = y_i)}{n}, \tag{10}$$

where G_{clean} refers to the clean global model.

ALGORITHM 4: Evaluate Backdoor Attack

Input: Global Model G_t, Testing Dataset D_{test}, Target label y_t, graph trigger g_t
Output: Attack Success Rate ASR

Function `Evaluation()`:
  $D_{tmp} \leftarrow d \in D_{test}$ if $d[y] \neq y_t$
  foreach $d \in D_{tmp}$ **do**
    $x = AddTrigger(d[x], g_t)$
    $y = y_t$
    $D_{tmp} = D_{tmp} \cup \{x, y\}$
  end
End Function
$ASR = accuracy(G_t, D_{tmp})$
return ASR

5.2 Backdoor Attack Results

We evaluate multiple-shot attack [2], which means that the attackers perform attacks in multiple rounds, and the malicious updates are accumulated to achieve a successful backdoor attack. We do not evaluate the single-shot attack [2] because the multi-shot is stealthier [36]. The multi-shot attack does not require multiplying large factors to model weights when conducting the attack, whereas the single-shot needs to multiply large factors to maintain the effectiveness of backdoor attacks, which can be filtered out or detected by traditional anomaly detection-based approaches such as Krum [5]. Since our main goal is conducting backdoor attacks on FL, we chose a multiple-shot attack with a high ASR and stealthiness. As mentioned in Section 3.2, we perform a complete attack in every round, showing the difference between DBA and CBA in a shorter time [48].

To explore the impact of different percentages of malicious clients on the attack performance, we evaluate the honest majority and malicious majority attack scenarios according to the percentage of malicious clients among all clients. Specifically, we set two and three malicious clients among five clients for the honest majority and malicious majority attack scenarios, respectively.

In our experiments, we evaluate the ASR of CBA and DBA with the global trigger and local triggers. The goal is to explore the following:

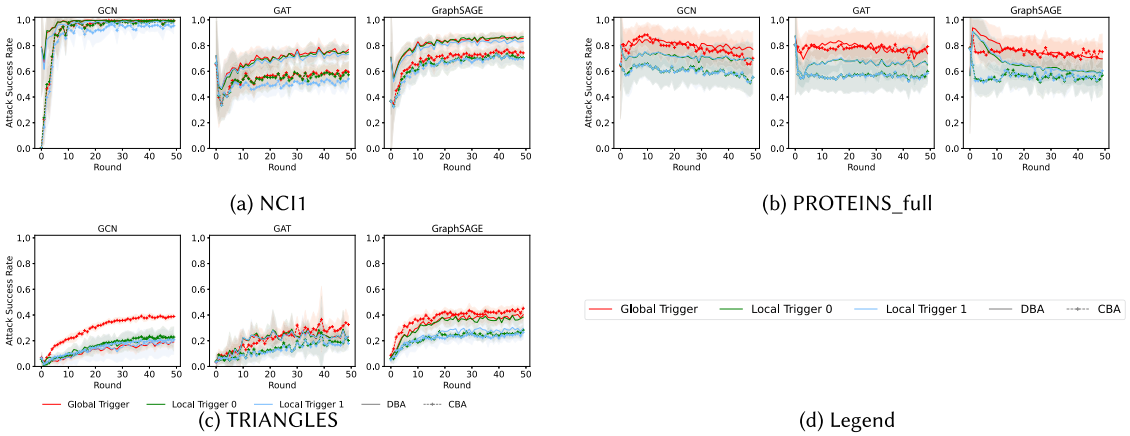


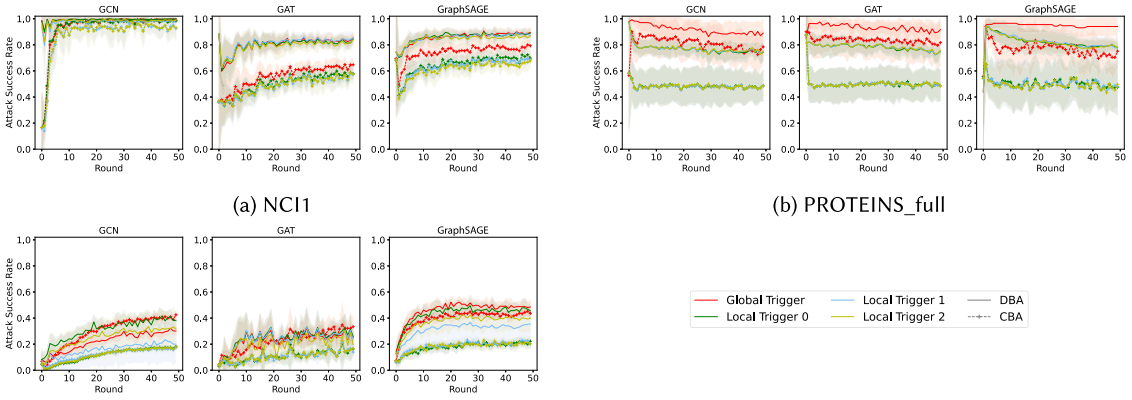
Fig. 5. Backdoor attack results in the honest majority attack scenario.

— In CBA, whether the ASR of local triggers can achieve similar performance to the global trigger even if the centralized attacker would embed a global trigger into the model.
— In DBA, whether the ASR of the global trigger is higher than all local triggers even if the global trigger never actually appears in any local training dataset, as mentioned in the work of Xie et al. [48].

Honest Majority Attack Scenario. The attack results of CBA and DBA in the honest majority attack scenario are shown in Figure 5. Notice that the DBA ASR with a specific trigger is always higher than or at least similar to that of CBA with the corresponding trigger. For example, in Figure 5(a) (the result for the GAT model), the DBA ASR with the global trigger is higher than the CBA with a global trigger. The only exception happens for GCN on TRIANGLES. We also find that the ASR of the two attacks in TRIANGLES is significantly lower than the other two datasets but still higher than random guessing. TRIANGLES is a multi-class dataset containing complex data relations. Thus, more information needs to be encoded in each model's weights for the class features compared to the other datasets. As a result, there is not enough remaining space to learn our triggers easily. In most results on NCI1 and PROTEINS_full, there is an initial drop in the ASR for both DBA and CBA, resulting from the high local learning rate of honest clients [2]. Based on the result for CBA, surprisingly, the ASR of all local triggers can be as high as the global trigger even if the centralized attacker embeds the global trigger into the model, which is inconsistent with the behavior in the work of Xie et al. [48]. We analyze it through further experiments shown later in Figure 7.

Moreover, the results for the PROTEINS_full dataset show that in DBA, the ASR of the global trigger is higher than (or at least similar to) any local trigger, even if the global trigger never actually appears in any local training dataset. This indicates that the high ASR of the global trigger does not require the same high ASR of local triggers. However, for the other two datasets (NCI1 and TRIANGLES), the ASR of the global trigger is close to all local triggers (except the result of GraphSAGE on TRIANGLES). This indicates that in some cases, the local trigger embedded in local models can successfully transfer to the global model so that once any local trigger is activated, the global model will misclassify the data sample into the attacker-chosen target label. This phenomenon is not consistent with the observations in the work of Xie et al. [48] as in Euclidean data, most locally triggered images are similar to the clean image, but any (small) change in the structure of a graph will result in a significant dissimilarity.

Malicious Majority Attack Scenario. Figure 6 illustrates the attack results in the malicious majority attack scenario. Compared with the honest majority attack scenario, in most cases, the ASR of DBA and CBA increases, as with more malicious clients, more malicious updates are uploaded to the global model, making the attack more



(a) NCI1 (b) PROTEINS_full

(c) TRIANGLES (d) Legend

Fig. 6. Backdoor attack results in the malicious majority attack scenario.

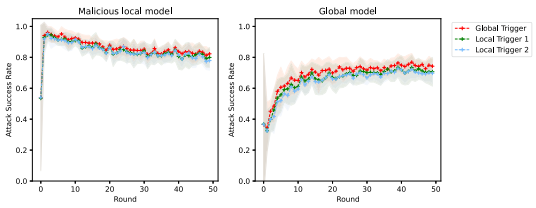
effective and persistent. Moreover, the increase in DBA is more significant than in CBA. For instance, based on the NCI1 dataset and GAT model, the DBA ASR with the global trigger in the honest majority attack scenario is 17.54% higher than CBA, whereas in the malicious majority attack scenario, the ASR difference is 20.65%. Thus, increasing the number of malicious clients is more beneficial for DBA than CBA. With more malicious clients, more local models are used to learn the trigger patterns in DBA, whereas there is only one malicious local model in CBA.

For CBA, the ASR with the global trigger is higher, whereas the attack performance with local triggers stays at a similar level or even decreases. One possible reason is that more malicious clients mean a larger global trigger, requiring more learning capacity of the model. If there is not enough learning capacity for every local trigger in the global trigger, the backdoored model can have poor attack performance with a specific local trigger but will behave well with the union set of the local triggers (i.e., the global trigger).

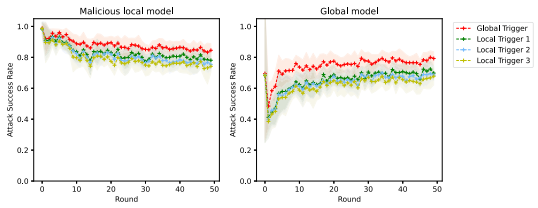
Analysis of CBA Results. In Figure 5, for CBA, the ASR of all local triggers can be as high as the global trigger, which is counterintuitive as the centralized attack only embeds the global trigger into the model. To explain these results, we further implement an experiment (NCI1 on the GraphSAGE model) where we evaluate the attack success rate of the global trigger and local triggers in both the malicious local model[8] and the global model. As shown in Figure 7, in the malicious local model, the ASR of all local triggers is already close to the global trigger, which means that the malicious local model has learned the pattern of each local trigger. After aggregation, the global model inherits the capacity of local models. Once any local trigger exists, the global model will misclassify the data sample into the attacker-chosen target label.

Still, in the work of Xie et al. [48], for the CBA, the ASR of all local triggers is significantly lower than the global trigger. There, the malicious local model learns the global trigger instead of each local trigger, so the poisoned model can only misclassify the data sample once there is a global trigger in the data. The different results in CBA between the work of Xie et al. [48] and our work can be explained since there, the local triggers composing the global trigger are located close to each other (i.e., less than three pixels distance). In our work, the location of local triggers is random since a graph is non-Euclidean data where we cannot put nodes in some order. When the local trigger graphs are further away from each other, the malicious local model in CBA can only learn the local trigger instead of the global trigger.

[8]For CBA, we assume there is one centralized attacker, so there is only one local model that will be poisoned and we define this model as the malicious local model.



(a) Honest majority attack scenario



(b) Malicious majority attack scenario

Fig. 7. Centralized backdoor attack results on the malicious local model and global model with different triggers.

Table 4. CAD of CBA and DBA in the Honest Majority Attack Scenario

Dataset	CAD (CBA% | DBA%)		
	GCN	GAT	GraphSAGE
NCI1	2.54 | 2.01	0.84 | 1.42	0.93 | 0.16
PROTEINS_full	1.81 | 4.06	0.49 | 0.46	2.31 | 2.82
TRIANGLES	0.01 | 1.32	3.71 | 2.87	3.31 | 4.45

Table 5. CAD of CBA and DBA in the Malicious Majority Attack Scenario

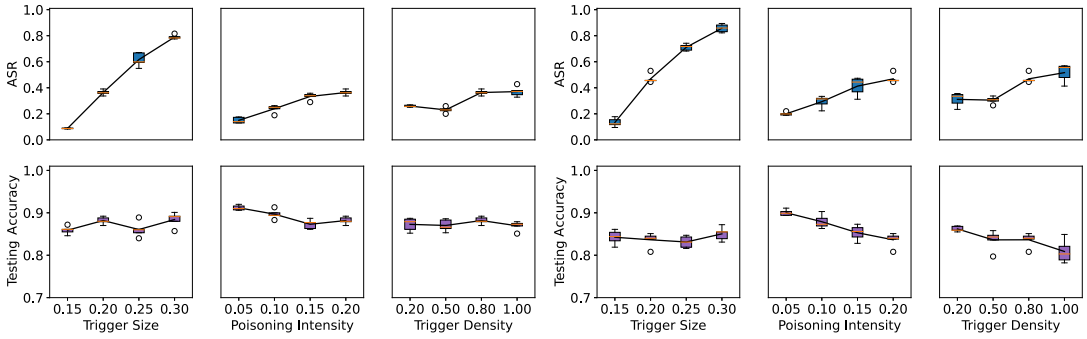
Dataset	CAD (CBA% | DBA%)		
	GCN	GAT	GraphSAGE
NCI1	4.45 | 2.74	1.03 | 1.07	1.29 | 2.22
PROTEINS_full	2.78 | 1.30	0.03 | 2.65	2.72 | 4.59
TRIANGLES	0.14 | 0.30	5.10 | 5.84	3.24 | 5.74

5.3 Clean Accuracy Drop

The goal of the backdoor attack is to make the backdoored model simultaneously fit the main task and backdoor task. Therefore, it is critical that the trained model still behaves normally on untampered data samples after training with the poisoned data. Here, we use CAD to evaluate if the backdoored model can still fit the original main task. CAD is the classification accuracy difference between global models with and without malicious clients over the clean testing dataset. CBA's and DBA's final CAD results in the honest and malicious majority attack scenarios are given in Tables 4 and 5, respectively. In most cases, both attacks have a low CAD (i.e., around 2%), and only in a few cases is there a significant CAD. These results imply that, in most cases, both attacks have a negligible impact on the original task of the model. Additionally, in some cases, DBA's CAD is significantly higher than CBA's—for example, DBA's CAD is 5.74% in the GraphSAGE model on TRIANGLES, whereas CBA's is 3.24%, as shown in Table 5. At the same poisoning intensity for each client, there are more poisoned data in DBA than in CBA, leading to worse performance in the main task. The substantial CAD in DBA can also be observed in the work of Xie et al. [48].

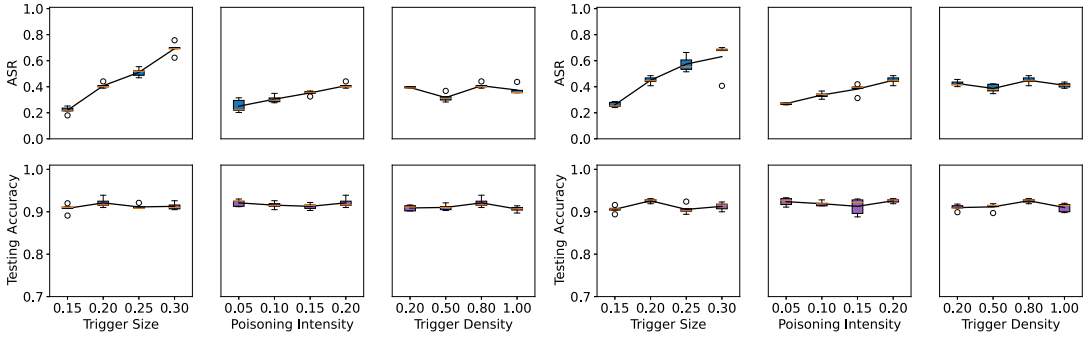
5.4 Analysis of Backdoor Hyperparameters

This section studies the backdoor hyperparameters discussed in Section 3.2. We only modify one factor for each experiment and keep other factors the same as in Section 5.1. We provide results for TRIANGLES and the GraphSAGE model as an example because those results are more stable (i.e., have the smallest standard error), and the results of other models are aligned. For each factor, we evaluate the global trigger's ASR and the test



(a) DBA in honest majority attack scenario

(b) DBA in malicious majority attack scenario



(c) CBA in honest majority attack scenario

(d) CBA in malicious majority attack scenario

Fig. 8. Results on TRIANGLES with different trigger parameters.

accuracy on the clean test dataset. We illustrate the results on TRIANGLES in two attack scenarios to analyze the effects of each factor for DBA and CBA. The results are shown in Figure 8.

Effects of Trigger Size. From the ASR results in Figure 8, for both attacks and attack scenarios, with the increase of trigger size, the ASR rises significantly—for example, the DBA's ASR increases from 0.09 to 0.80 with trigger size rising from 0.15 to 0.30 (honest majority attack scenario). There is no significant effect of trigger size on the test accuracy of the global model, implying that the trigger size has little impact on the original main task.

Effects of Poisoning Intensity. Similar to the impact of trigger size on the ASR, a higher poisoning intensity gives a higher ASR. Intuitively, a backdoor attack can perform better with more poisoned data. Nevertheless, the increase is less significant than that of different trigger sizes. Specifically, in comparison with the work of Xi et al. [47], where there is no obvious difference between the impact of poisoning intensity and trigger size, here, a larger trigger size has a more positive influence on the ASR than a larger poisoning intensity. Moreover, in DBA, the test accuracy decreases with the increasing poisoning intensity, and with more malicious clients, the drop is more significant, as shown in Figure 8(a) and (b). With higher poisoning intensity and more malicious clients, more model weights (including some for the original task) are influenced by the malicious trigger patterns, and the performance on the main task degrades more. We can also observe that with higher poisoning intensity, there is no obvious drop in the testing accuracy for CBA, as presented in Figure 8(c) and (d). Although more local data is poisoned, the other honest clients (the majority part) still guarantee the performance on the main task by aggregating local models' weights in the server.

Effects of Trigger Density. From Figure 8(b), DBA's ASR improves from 30.10% to 47.96% when the trigger density increases from 0.50 to 0.80. This is because the average complexity of the TRIANGLES dataset is 0.16 [38]. Thus, when the trigger density is set close to this value, the difference between the original graph and the trigger graph is harder to distinguish. However, the effect of the trigger density in CBA's ASR is not strong. We see a slight fluctuation as the trigger density increases, but its range is very small to be seen as a trend. In CBA, we use only one malicious client, and the weak effect of the trigger density is smoothed by the averaging operation.

In Figure 8, in most cases, there is no significant drop in the test accuracy with an increase in the trigger size and trigger density. On the contrary, in the backdoor attacks in centralized GNNs [52], as trigger size increases, the test accuracy decreases. This can be explained as, in FL, the influence of backdoor functionality on the main task is weakened by the aggregation of local models.

5.5 Attack Performance in Real-World Social Network Datasets

Two Real-World Social Network Datasets. We already used two molecular structure datasets and one synthetic dataset to explore the DBA and CBA in the Federated GNNs. However, to further explore the DBA and CBA in real-world applications and scenarios, we extend our analysis of DBA and CBA to two additional real-world social network datasets. In recent years, with the rapid development of internet technology, social network graphs have played an increasingly important role in people's lives [15]. These graphs contain valuable information regarding individuals' behavior and interactions, making them applicable in various domains such as sentiment analysis, recommendation systems, and spam detection. Federated GNNs can also be employed in these areas. Nonetheless, the vulnerability of Federated GNNs to backdoor attacks poses significant risks. For instance, in the scenario of applying the Federated GNNs on spam detection, if the attacker executes a backdoor attack on the system, the model will detect any input graph with a specific trigger as benign, potentially leading to severe safety issues. Consequently, it is crucial also to explore the attack performance of the Federated GNNs on social network datasets. What is more, these two social network datasets have different graph characteristics from the aforementioned two molecular structure datasets and one synthetic dataset. Specifically, these two datasets have a larger average number of edges—for example, the average number of edges of COLLAB is 2, 457.78, whereas that of NCI, PROTEINS_full, and TRIANGLES datasets are 32.30, 72.82, and 32.74, respectively. Through experiments on these two social network datasets, it can be verified that our attacks are effective not only on simple datasets but also on complex datasets:

— *COLLAB*: This scientific-collaboration dataset is derived from three public collaboration datasets, namely *High Energy Physics, Condensed Mater Physics*, and *Astro Physics*. It consists of ego networks[9] of different researchers from each field, with each graph labeled according to the researcher's field. The task is to determine whether the ego-collaboration graph of a researcher belongs to the High Energy, Condensed Matter, or Astro Physics field.

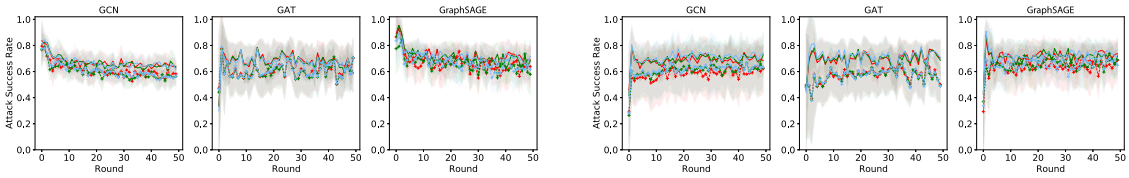
— *IMDB-BINARY*: This dataset is a movie-collaboration dataset where actor/actress and genre information of different movies on IMDB was collected. Collaboration graphs are generated based on *Action* and *Romance* genres, and ego networks are derived for each actor/actress. Similar to the COLLAB dataset, each ego network is labeled with the genre graph it belongs to. The task is to identify to which genre an ego network graph belongs. Table 6 describes the information about these two social network datasets.

Attack Results. The attack results of DBA and CBA on two social network datasets in the honest majority attack scenario are presented in Figure 9. As seen in the figure, we can observe that for both GCN and GAT models, the ASR of DBA with a specific trigger is consistently higher than that of CBA with the corresponding trigger. For instance, for GAT on IMDB-BINARY, the ASR of DBA with the global trigger is typically 10% higher

[9]An ego network is defined as a portion of a social network formed for a given individual, termed *ego*, and the other persons with whom the user has a social relationship, termed *alters* [1].

Table 6. Dataset Statistics of Two Social Network Datasets

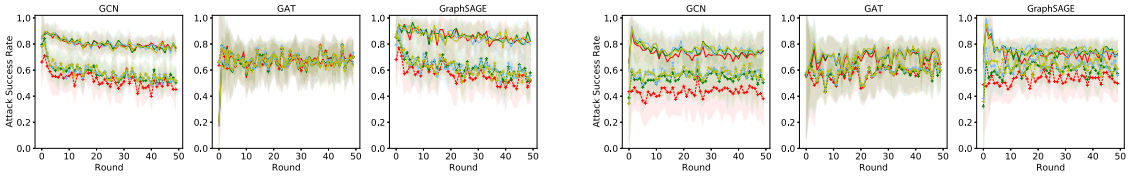
Dataset	# Graphs	Avg. # Nodes	Avg. # Edges	Classes	Class Distribution
COLLAB	5,000	74.49	2,457.78	3	2,600[0], 775[1], 1,625[2]
IMDB-BINARY	1,000	19.77	96.53	2	500[0], 500[1]



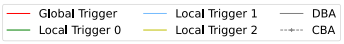
(a) COLLAB (b) IMDB-BINARY

(c) Legend

Fig. 9. Backdoor attack results in the honest majority attack scenario (social network datasets).



(a) COLLAB (b) IMDB-BINARY

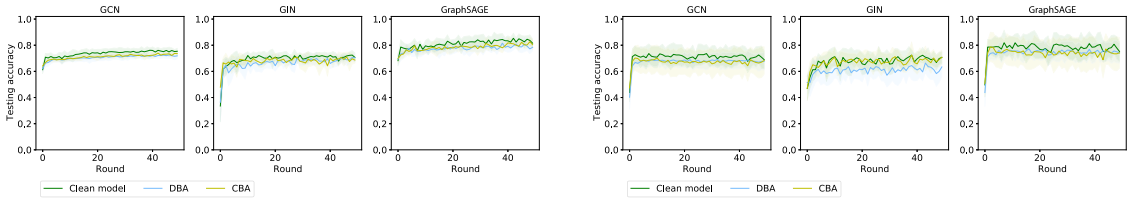


(c) Legend

Fig. 10. Backdoor attack results in the malicious majority attack scenario (social network datasets).

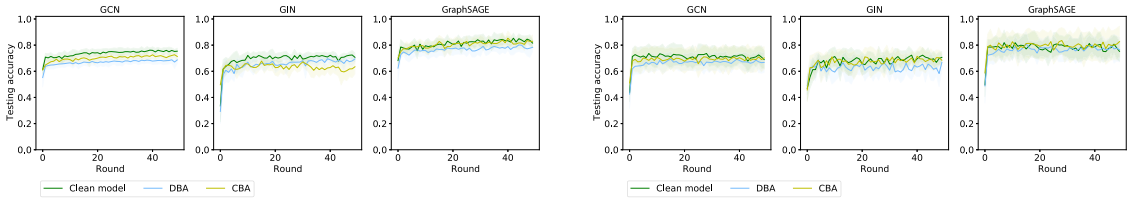
than that of CBA with the global trigger. However, for the GraphSAGE model, the ASR of the two attacks is similar. It can be explained that the GraphSAGE model has a more redundant learning capacity, making it easier for CBA to learn the global trigger pattern. As a result, the ASR of CBA is higher than the other two models and is similar to that of DBA. Similar to the results obtained from the prior datasets (i.e., NCI1, PROTEINS_full, and TRIANGLES), for CBA on the two social network datasets, the ASR of all local triggers can be as high as the global trigger even if the centralized attacker embeds the global trigger into the model. This observation highlights the vulnerability of the GNN models to even the slight structure manipulation in the graph. Furthermore, in DBA, the ASR of the global trigger is also close to all the local triggers, which further indicates that in DBA, the local trigger embedded in local models can successfully transfer to the global model. It implies that the attacker can compromise the global trigger by embedding a simple local trigger into the local models, leading to significant security concerns.

Figure 10 shows the attack results of two social network datasets in the malicious attack scenario. In comparison to the honest majority attack scenario, the ASR of DBA increases for GCN and GraphSAGE models as more malicious clients participate in the attack, more model capacity is used to learn the trigger pattern, and more malicious updates are uploaded into the global model, making it easier to achieve the attack. For example, for



(a) COLLAB (b) IMDB-BINARY

Fig. 11. Testing accuracy in the honest majority attack scenario (social network datasets).



(a) COLLAB (b) IMDB-BINARY

Fig. 12. Testing accuracy in the malicious majority attack scenario (social network datasets).

the GCN model on COLLAB, the ASR of DBA in the malicious majority attack scenario is around 15% higher than that in the honest majority attack scenario. However, the ASR of CBA notably decreases in the malicious majority attack scenario, especially for the global trigger in GCN and GraphSAGE models. One possible explanation for this observation is that in the CBA, there is only one malicious local model and the global trigger is larger in the malicious majority attack scenario than in the honest majority attack scenario, which requires more learning capacity of the model to learn every local trigger. It is likely that the backdoored model cannot learn the local trigger patterns clearly, and therefore it cannot recognize the global trigger effectively. These findings indicate that DBA is generally more effective than CBA in the malicious majority attack scenario. Moreover, the results demonstrate that the number of malicious clients participating in the attack significantly impacts the effectiveness of the backdoor attacks in the Federated GNNs. It is essential to develop more robust and secure models that can resist these backdoor attacks, especially when the attacker has access to more clients in the FL setting.

Clean Accuracy Drop. Here, we also use CAD to evaluate the impact of backdoor attacks on the original mask task on the social network datasets. The testing accuracy of DBA and CBA on the two social network datasets are shown in Figure 11 (honest majority attack scenario) and Figure 12 (malicious majority attack scenario). Specifically, the CAD results in the honest and malicious majority attack scenarios are given in Tables 7 and 8, respectively. In most cases, the CAD of both attacks is less than 5% for both datasets. Additionally, we observe that the CAD of DBA is generally higher than that of CBA, with a difference of more than 3% in most cases. This observation is consistent with the results obtained in the prior experiments.

6 DEFENSE

6.1 Potential Countermeasures

FLAME [33] is one of the state-of-the-art defenses against backdoor attacks in FL, combining the benefits of both defense types (Byzantine-robust aggregation mechanisms and differential privacy techniques) to eliminate the impact of backdoor attacks while maintaining the performance of the aggregated model on the main task. FoolsGold [13] is a robust FL aggregation algorithm that can identify attackers in FL based on the diversity of

Table 7. CAD of CBA and DBA in the Honest Majority Attack Scenario (Social Network Datasets)

Dataset	CAD (CBA% | DBA%)		
	GCN	GAT	GraphSAGE
COLLAB	2.44 | 3.46	1.75 | 2.27	2.52 | 4.40
IMDB-BINARY	3.75 | 2.89	1.38 | 8.06	4.35 | 2.97

Table 8. CAD of CBA and DBA in the Malicious Majority Attack Scenario (Social Network Datasets)

Dataset	CAD (CBA% | DBA%)		
	GCN	GAT	GraphSAGE
COLLAB	3.83 | 7.37	9.07 | 3.16	1.14 | 5.09
IMDB-BINARY	1.22 | 3.58	0.74 | 6.72	2.37 | 0.61

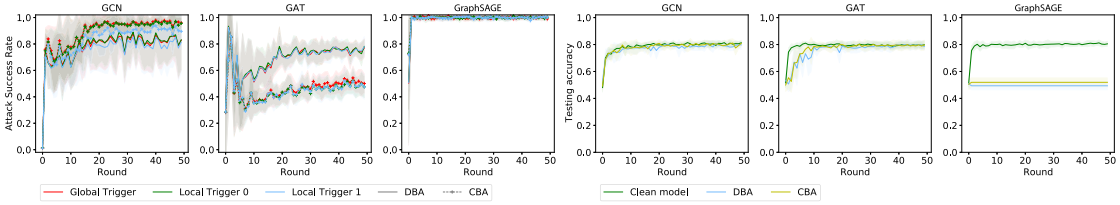
client updates. It reduces the aggregation weights of detected malicious clients while retaining the weights of other clients. One of the assumptions in this defense is that each client's training data is non-i.i.d and has a unique distribution, which fits the non-i.i.d. data distribution setting in our article. Ozdayi et al. [35] also proposed a defense mechanism against backdoor attacks in FL. The idea is to adaptively adjust the learning rate of the server's aggregation function per dimension and per round based on the sign information of agents' updates. In this work, we focus on evaluating the attack effectiveness of DBA and CBA against FLAME, FoolsGold, and RLR.

6.2 Results against FLAME

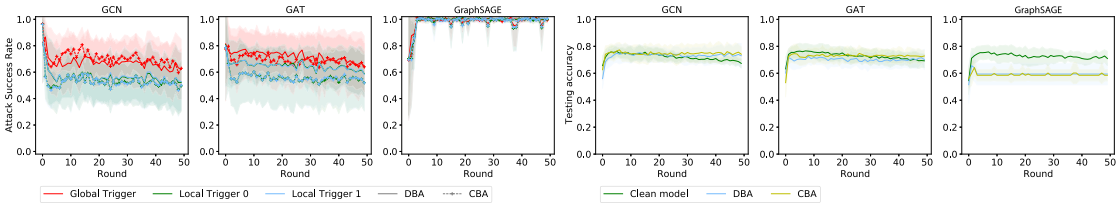
Figure 13 shows the attack performance of the NCI1, PROTEINS_full, and TRIANGLES datasets under FLAME in the honest majority attack scenario (the results in the malicious majority attack scenario are similar). One important parameter in FLAME is the Gaussian noise level σ, and σ is set to be 0.01 for Figure 13 as default. Compared with the attack performance before FLAME (see Figure 5), for GCN and GAT models, the ASR of both DBA and CBA decreases around 10% for NCI1 and PROTEINS_full datasets, whereas for the TRIANGLES dataset, the ASR even decreases to nearly 0. However, we can also observe that for the GraphSAGE model, the ASR under FLAME increases for all datasets, and the testing accuracy decreases dramatically. Given that for PROTEINS_full and NCI1 datasets the attack performance under FLAME does not degrade obviously, we further evaluate the attack performance under FLAME on PROTEINS_full and NCI1 datasets with different Gaussian noise levels, as shown in Figures 14 and 15. It can be observed that with a higher Gaussian noise level, there is more fluctuation in the attack performance under FLAME (i.e., more standard deviation). Generally, with a larger Gaussian noise level, the ASR under FLAME for these two datasets decreases while the testing accuracy also reduces significantly. Based on the experimental results against FLAME, we find that FLAME can indeed degrade the attack performance of DBA and CBA, especially for TRIANGLES datasets with GCN and GAT models. However, generally, it can only reduce the ASR of both attacks for less than 10% for PROTEINS_full and NCI1 datasets, and it does not work for the GraphSAGE model.

6.3 Results against FoolsGold

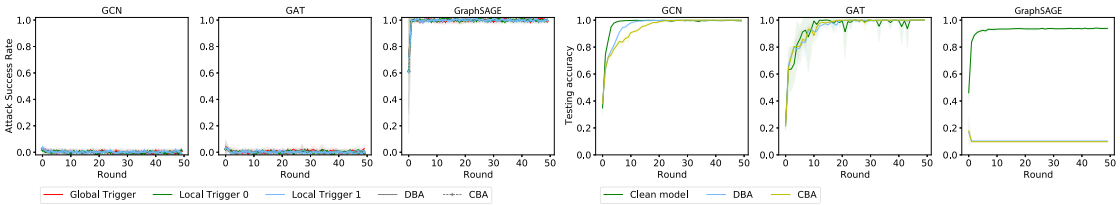
Figure 16 shows the attack performance for the TRIANGLES dataset under FoolsGold in the honest majority attack scenario (the results in the malicious majority attack scenario are similar). The results for other datasets illustrate that FoolsGold has a negligible impact on the attack performance, as shown in Appendix A.1. As



(a) NCI1

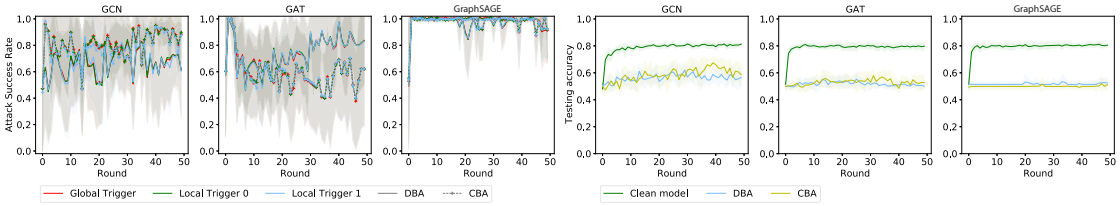


(b) PROTEINS_full

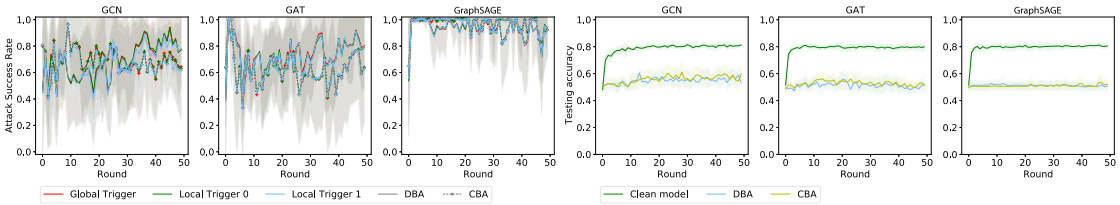


(c) TRIANGLES

Fig. 13. Backdoor attack results against FLAME (in the honest majority attack scenario).

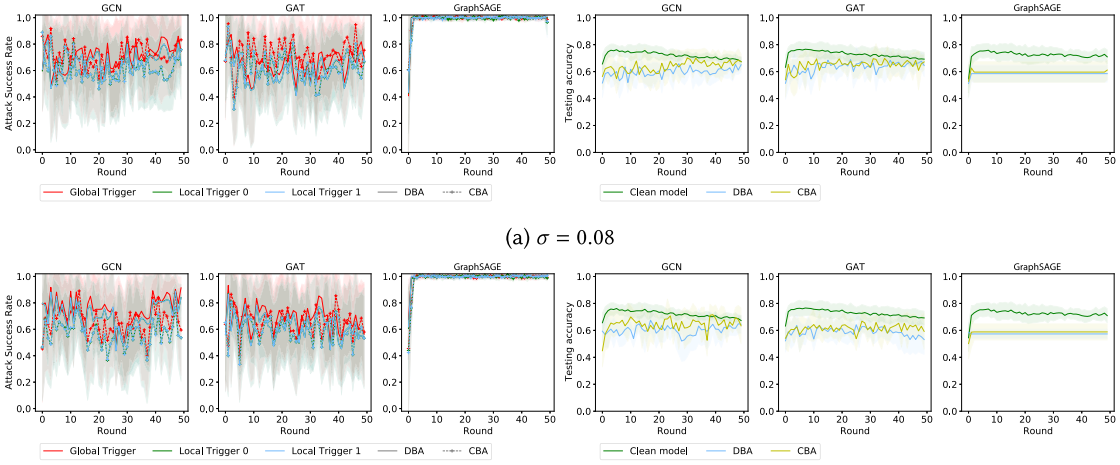


(a) $\sigma = 0.08$



(b) $\sigma = 0.10$

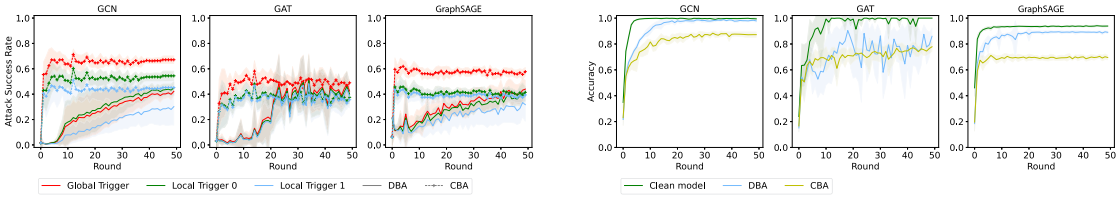
Fig. 14. Backdoor attack results against FLAME on NCI1 with different σ (in the honest majority attack scenario).



(a) $\sigma = 0.08$

(b) $\sigma = 0.10$

Fig. 15. Backdoor attack results against FLAME on PROTEINS_full with different σ (in the honest majority attack scenario).



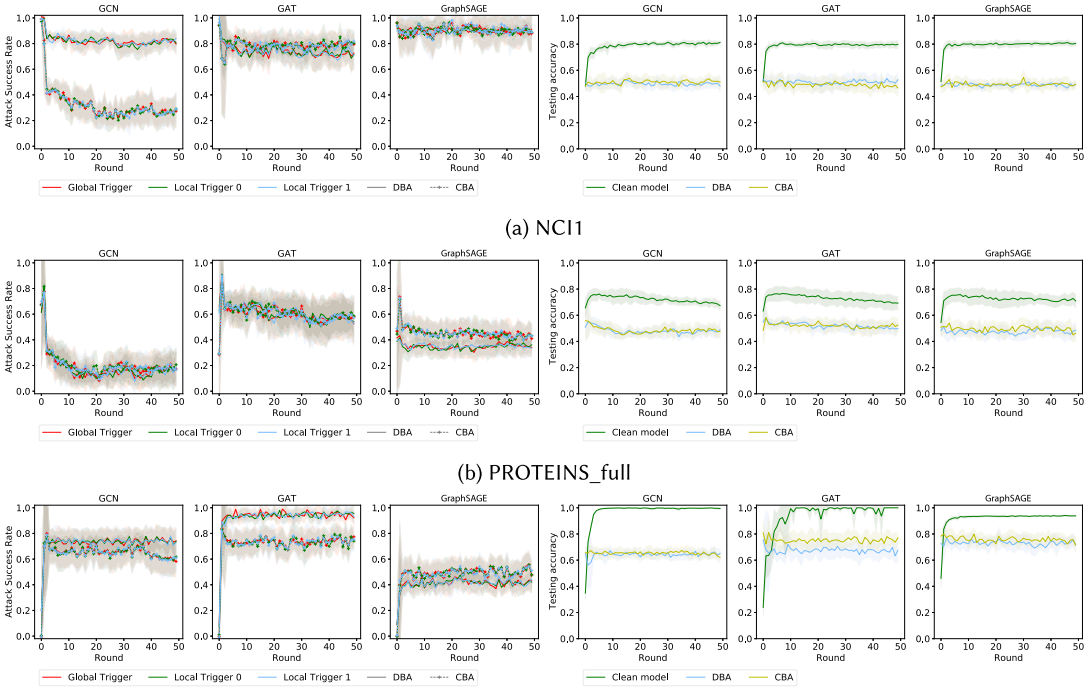
(a) Attack success rate

(b) Testing accuracy

Fig. 16. Backdoor attack results of TRIANGLES on FoolsGold for the honest majority.

illustrated in Figure 16(a), we can observe that for DBA, the ASR under FoolsGold remains consistent for the GraphSAGE model, whereas it rises along with a decrease in testing accuracy for the GAT model. Moreover, generally, under FoolsGold, there is a significant increase in CBA's ASR in all models, but the testing accuracy of CBA reduces significantly at the same time. For example, the CBA's ASR increases by about 20% for the GraphSAGE model. However, the testing accuracy of CBA on GraphSAGE has a drop of more than 20% (Figure 16(b)). Our hypothesis for this situation is that under FoolsGold, the malicious client in CBA is assigned a higher weight (recall the description of the FoolsGold mechanism from the preceding paragraph) than other clients, so malicious updates contribute more to the aggregated model. Simultaneously, the low weights on the honest clients' updates lead to the failure of the performance on the original task. We report FoolsGold's weights on every client in DBA and CBA in Appendix B and show that this hypothesis is reasonable. One possible reason is that in CBA, there is only one malicious client whose updates are likely to appear dissimilar from those of other honest clients, so FoolsGold cannot identify the malicious updates successfully.

Based on the experimental results against FoolsGold, we find that the tested defense cannot detect malicious updates successfully. One reason may be that this defense applies *cosine distance* to try to identify malicious models—that is, the distance between malicious updates is smaller between honest updates. Still, in our attacks, the malicious clients' updates could already be very dissimilar to each other, so the malicious updates are likely to be clustered into honest updates.



(a) NCI1

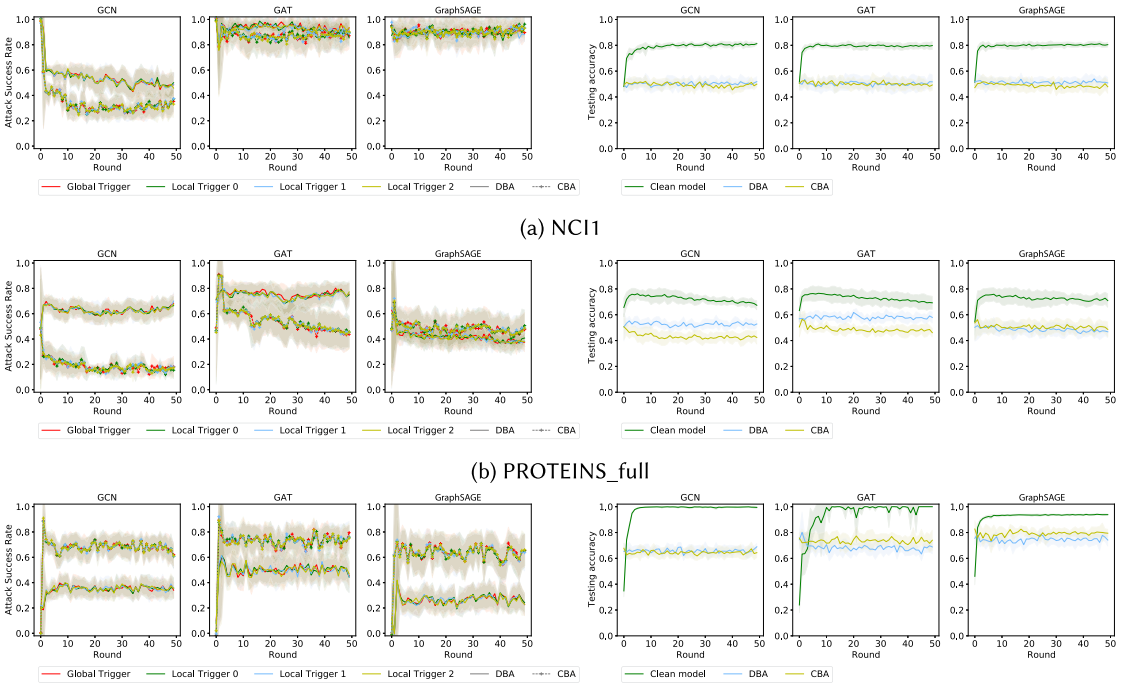
(b) PROTEINS_full

(c) TRIANGLES

Fig. 17. Backdoor attack results against RLR (in the honest majority attack scenario).

6.4 Results Against RLR

The attack performance of NCI1, PROTEINS_full, and TRIANGLES against RLR is shown in Figure 17 (honest majority attack scenario) and Figure 18 (malicious majority attack scenario) including the ASR and testing accuracy. The results reveal some interesting insights, and the ASR of DBA and CBA against RLR varies significantly across different datasets and models. In the honest majority attack scenario, compared to the attack performance before the defense, as shown in Figure 5, we can observe that the ASR of DBA and CBA decreases for the NCI1 dataset in the GCN model. For example, the ASR of CBA decreases from around 100% to 20%, whereas it increases for GAT and GraphSAGE models. However, for the PROTEINS_full dataset, the ASR of both attacks decreases dramatically (i.e., the decrease is up to 60% compared to Figure 5(b)) for all models. In contrast, for the TRIANGLES dataset, the ASR of DBA and CBA surprisingly increases after RLR. Compared to the NCI1 and PROTEINS_full datasets, the TRIANGLES dataset has fewer average nodes, which makes it more challenging to create a backdoor trigger pattern. Thus, as we can see from Figures 5 and 6, TRIANGLES has lower ASR than other datasets. Moreover, the undistinguished backdoor trigger pattern in the TRIANGLES dataset leads to the inefficiency of RLR. Another intriguing discovery is that the testing accuracy of all datasets and models reduces significantly under RLR, dropping to around 50% for NCI1 and PROTEINS_full, which is random guessing for these two datasets. The large testing accuracy drop here is not consistent with that in the work of Ozdayi et al. [35]. One possible reason behind this observation is that in the RLR defense, the malicious updates are generally detected as benign and then assigned a positive learning rate. In contrast, benign updates are detected as malicious ones and assigned a negative learning rate. Therefore, the malicious updates contribute more to the aggregated global model, leading to a higher ASR and poorer performance on the original task. Moreover, RLR only works under



(a) NCI1

(b) PROTEINS_full

(c) TRIANGLES

Fig. 18. Backdoor attack results against RLR (in the malicious majority attack scenario).

the assumption that the number of malicious clients is sufficiently below θ (learning threshold) [35]. However, in our threat model, the number of malicious clients can be higher than θ.

When there are more malicious clients (see Figure 18), the ASR of DBA decreases dramatically in most cases compared to the honest majority attack scenario, whereas the ASR of CBA remains relatively stable. For instance, the ASR of DBA in the TRIANGLES in the malicious majority attack scenario is around 36%, whereas that in the honest majority attack scenario is around 75%. It may be explained that in CBA, there is only one malicious local model, whereas in DBA, there are multiple malicious local models. Thus, with an increase in the number of malicious clients, the malicious updates in DBA increase more than in CBA. This makes it easier for RLR to detect malicious updates in DBA, resulting in a decrease in DBA's ASR. Despite the decrease of ASR in DBA under RLR in the malicious majority attack scenario, the CAD under RLR is still very high in both attacks (i.e., more than 25% in most cases). Our experimental results against RLR indicate that this defense mechanism is not effective in detecting malicious updates, and it can significantly degrade the original main task.

7 CONCLUSION AND FUTURE WORK

This article explored how CBA and DBA behave in Federated GNNs. Through extensive experiments on three datasets and three popular GNN models, we showed that generally, DBA achieves a higher ASR than CBA. We showed that in CBA, the ASR of local triggers could be as high as the global trigger even if, during training, only the global trigger is embedded in the model. The impact of the percentage of malicious clients on DBA's ASR is analyzed with correlation, where we confirm the intuition that more malicious clients lead to more successful attacks. We analyzed the critical backdoor hyperparameters to explore their impact on the attack performance and the main task. We also demonstrated that DBA and CBA are robust against two defenses for the backdoor

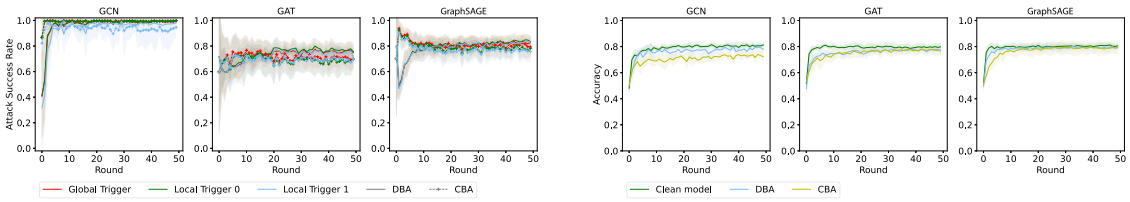
attack in FL. Interestingly, the ASR of DBA and CBA can be even higher under the defenses. We consider our work to provide new challenges when exploring adversarial attacks in Federated GNNs, a domain unexplored before our work. Furthermore, a powerful defense targeting backdoor attacks in Federated GNNs is required. The experimental setting in this work verifies the effectiveness of our method in a cross-silo FL setting and motivates further research in exploring backdoor attacks in Federated GNNs considering cross-device FL [39]. Future work will include exploring backdoor attacks in Federated GNNs for the node classification task. For example, in a social media app where each user has a local social network G^k and $\{G^k\}$ constitutes the latent entire human social network G, the developers can train a fraud detection GNN model through FL. In such a case, an attacker can conduct a backdoor attack to force the trained global model to classify a fraud node as benign.

APPENDICES

A ADDITIONAL EXPERIMENTAL RESULTS

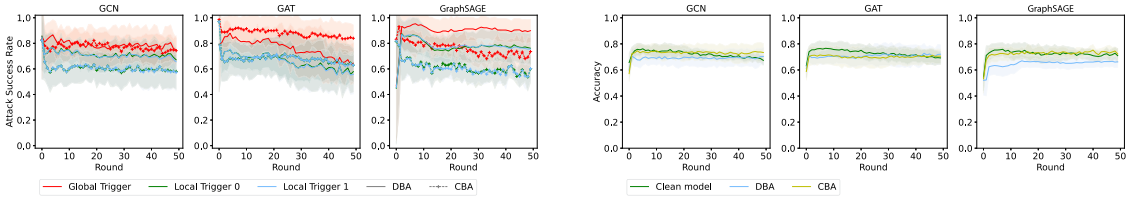
A.1 Additional Defense Results

The ASRs under FoolsGold on NCI1 and PROTEINS_full datasets (honest majority attack scenario) are shown in Figures 19 and 20, respectively. There is a slight increase in the ASR of DBA and CBA under FoolsGold, which indicates that this defense fails to identify the malicious updates and misclassifies them as benign. The graph data are not Euclidean data (e.g., images), so the slightly different subgraphs used as triggers do not induce aligned updates. As a result, cosine similarity cannot be used to detect malicious clients based on their updates. Even though there are more malicious clients in the malicious majority scenario and the probability of detecting the malicious updates should be higher, we observe the same behavior. This further verifies our hypothesis that the defense based on cosine similarity between updates is not very effective in the graph domain. The CAD under FoolsGold on these two datasets is similar to that without the defense. Thus, the defense does not affect the original task in that case.



(a) Attack success rate (b) Testing accuracy

Fig. 19. Attack success rate on NCI1 on FoolsGold (in the honest majority attack scenario).



(a) Attack success rate (b) Testing accuracy

Fig. 20. Attack success rate on PROTEINS_full on FoolsGold (in the honest majority attack scenario).

Table 9. FoolsGold Weight in DBA and CBA on TRIANGLES (Honest Majority Attack Scenario)

Attack	Attacker 1	Attacker 2 (client 2 in CBA)	Client 3	Client 4	Client 5	Attacker (sum)
DBA	1.00 ± 0.00	1.00 ± 0.00	0.82 ± 0.15	0.81 ± 0.09	0.78 ± 0.12	2.00 ± 0.00
CBA	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00

B FOOLSGOLD WEIGHTS

To verify our hypothesis (Section 6) for a reason behind the attack performance of DBA and CBA against the FoolsGold defense, we reported the FoolsGold weights on every client in the DBA and CBA on the GraphSAGE model, as shown in Table 9. Here, the FoolsGold weight for each client ranges from 0 to 1. As we can see, in CBA, the weight of the malicious client is 1, and the weights of other clients are 0, which means only the malicious updates are aggregated into the global model. Therefore, the ASR of CBA increases significantly under FoolsGold.

However, in DBA, the weights of the malicious clients are similar to the honest clients, indicating that the malicious and honest updates contribute equally to the aggregated model, as in the aggregation function without the defense (i.e., the average aggregation function). Therefore, there is no obvious difference between the attack performance of DBA before and after the defense. The reported weights in Table 9 verify that our hypothesis is valid.

REFERENCES

[1] Valerio Arnaboldi, Marco Conti, Massimiliano La Gala, Andrea Passarella, and Fabio Pezzoni. 2016. Ego network structure in online social networks and its impact on information diffusion. *Computer Communications* 76 (2016), 26–41.

[2] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *Proceedings of AISTATS*.

[3] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439 (1999), 509–512.

[4] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. 2019. Analyzing federated learning through an adversarial lens. In *Proceedings of ICML*.

[5] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems* 30 (2017), 1–11.

[6] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, et al. 2019. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046* (2019).

[7] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, andHans-Peter Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21, Suppl. 1 (2005), 47–56.

[8] Dawei Cheng, Fangzhou Yang, Sheng Xiang, and Jin Liu. 2022. Financial time series forecasting with multi-modality graph neural network. *Pattern Recognition* 121 (2022), 108218.

[9] Sanjoy Dasgupta, Christos H. Papadimitriou, and Umesh Virkumar Vazirani. 2008. *Algorithms*. McGraw-Hill Higher Education, New York, NY.

[10] Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-guided heterogeneous graph neural network for intent recommendation. In *Proceedings of the 25th ACM SIGKDD*.

[11] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *Proceedings of WWW*.

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

[13] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. 2018. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866* (2018).

[14] E. N. Gilbert. 1959. Random graphs. *Annals of Mathematical Statistics* 30, 4 (1959), 1141–1144. https://doi.org/10.1214/aoms/1177706098

[15] Sensen Guo, Xiaoyu Li, and Zhiying Mu. 2021. Adversarial machine learning on social network: A survey. *Frontiers in Physics* 9 (2021), 651.

[16] Zhiwei Guo and Heng Wang. 2020. A deep graph neural network-based mechanism for social recommendations. *IEEE Transactions on Industrial Informatics* 17, 4 (2020), 2776–2783.

[17] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of NeurIPS*.

[18] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Francois Beaufays, Sean Augenstein, Hubert Eichner, Chloe Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).
[19] Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Carl Yang, Han Xie, Lichao Sun, Lifang He, Liangwei Yang, Philip S. Yu, Yu Rong, et al. 2021. FedGraphNN: A federated learning system and benchmark for graph neural networks. *arXiv:2104.07145 [cs.LG]* (2021).
[20] Chaoyang He, Emir Ceyani, Keshav Balasubramanian, Murali Annavaram, and Salman Avestimehr. 2021. SpreadGNN: Serverless multi-task federated learning for graph neural networks. *arXiv:2106.02743 [cs.LG]* (2021).
[21] Meng Jiang, Taeho Jung, Ryan Karl, and Tong Zhao. 2020. Federated dynamic GNN with secure aggregation. *arXiv preprint arXiv:2009.07351* (2020).
[22] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurelien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zhachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977* (2019).
[23] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.
[24] Boris Knyazev, Graham W. Taylor, and Mohamed Amer. 2019. Understanding attention and generalization in graph neural networks. *Advances in Neural Information Processing Systems* 32 (2019), 1–11.
[25] Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. 2019. Peer-to-peer federated learning on graphs. *arXiv preprint arXiv:1901.11173* (2019).
[26] Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. 2019. Peer-to-peer federated learning on graphs. *arXiv preprint arXiv:1901.11173* (2019).
[27] Jaechang Lim, Seongok Ryu, Kyubyong Park, Yo Joong Choe, Jiyeon Ham, and Woo Youn Kim. 2019. Predicting drug–target interaction using a novel graph neural network with 3D structure-embedded graph representation. *Journal of Chemical Information and Modeling* 59, 9 (2019), 3981–3988.
[28] Fenglin Liu, Xian Wu, Shen Ge, Wei Fan, and Yuexian Zou. 2020. Federated learning for vision-and-language grounding problems. In *Proceedings of AAAI*.
[29] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. 2020. FedVision: An online visual object detection platform powered by federated learning. In *Proceedings of AAAI*.
[30] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. In *Proceedings of NDSS*.
[31] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera Y. Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of AISTATS*.
[32] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663* (2020).
[33] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, et al. 2022. FLAME: Taming backdoors in federated learning. In *Proceedings of USENIX Security*. 1415–1432.
[34] Thien Duc Nguyen, Phillip Rieger, Markus Miettinen, and Ahmad-Reza Sadeghi. 2020. Poisoning attacks on federated learning-based IoT intrusion detection system. In *Proceedings of DISS*.
[35] Mustafa Safa Ozdayi, Murat Kantarcioglu, and Yulia R. Gel. 2021. Defending against backdoors in federated learning with robust learning rate. In *Proceedings of AAAI*, Vol. 35. 9268–9276.
[36] Krishna Pillutla, Sham M. Kakade, and Zaid Harchaoui. 2019. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445* (2019).
[37] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. 2018. Generative adversarial perturbations. In *Proceedings of CVPR*.
[38] Pavel Pudlák, Vojtěch Rödl, and Petr Savickỳ. 1988. Graph complexity. *Acta Informatica* 25, 5 (1988), 515–535.
[39] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. 2022. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *Proceedings of SP*. IEEE, Los Alamitos, CA, 1354–1371.
[40] Toyotaro Suzumura, Yi Zhou, Natahalie Baracaldo, Guangnan Ye, Keith Houck, Ryo Kawahara, Ali Anwar, Lucia Larise Stavarache, Yuji Watanabe, Pablo Loyola, et al. 2019. Towards federated graph learning for collaborative financial crimes detection. *arXiv preprint arXiv:1909.12946* (2019).
[41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of ICLR*. https://openreview.net/forum?id=rJXMpikCZ
[42] Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. 2019. A semi-supervised graph attentive network for financial fraud detection. In *Proceedings of ICDM*. IEEE, Los Alamitos, CA.
[43] Jianian Wang, Sheng Zhang, Yanghua Xiao, and Rui Song. 2022. A review on graph neural network methods in financial applications. *Journal of Data Science* 20, 2 (2022), 111–134.
[44] Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature* 393, 6684 (1998), 440–442.
[45] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925* (2021).

[46] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.
[47] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. 2021. Graph backdoor. In *Proceedings of USENIX Security*.
[48] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. 2019. Dba: Distributed backdoor attacks against federated learning. In *Proceedings of ICLR*.
[49] Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xufong Li, Zhaojun Li, Xiaomin Lui, Kaixian Chen, Hualiang Jiang, et al. 2019. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of Medicinal Chemistry* 63, 16 (2019), 8749–8760.
[50] Jing Xu, Gorka Abad, and Stjepan Picek. 2023. Rethinking the trigger-injecting position in graph backdoor attack. *arXiv preprint arXiv:2304.02277* (2023).
[51] Jing Xu, Rui Wang, Stefanos Koffas, Kaitai Liang, and Stjepan Picek. 2022. More is better (mostly): On the backdoor attacks in federated graph neural networks. In *Proceedings of ACSAC*. ACM, New York, NY, 684–698. https://doi.org/10.1145/3564625.3567999
[52] Jing Xu, Minhui Xue, and Stjepan Picek. 2021. Explainability-based backdoor attacks against graph neural networks. In *Proceedings of WiseML*.
[53] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).
[54] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of ICML*. 5650–5659.
[55] Ruiping Yin, Kan Li, Guangquan Zhang, and Jie Lu. 2019. A deeper graph neural network for recommender systems. *Knowledge-Based Systems* 185 (2019), 105020.
[56] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. *arXiv preprint arXiv:1806.08804* (2018).
[57] Huanding Zhang, Tao Shen, Fei Wu, Mingyang Yin, Hongxia Yang, and Chao Wu. 2021. Federated graph learning—A position paper. *arXiv preprint arXiv:2105.11099* (2021).
[58] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *Proceedings of AAAI*.
[59] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. 2021. Backdoor attacks to graph neural networks. In *Proceedings of the 26th ACM SACMAT*.
[60] Jun Zhou, Chaochao Chen, Longfei Zheng, Huiwen Wu, Jia Wu, Xiaolin Zheng, Bingzhe Wu, Ziqi Liu, and Li Wang. 2021. Vertically federated graph neural network for privacy-preserving node classification. *arXiv:2005.11903 [cs.LG]* (2021).
[61] Xinghua Zhu, Jianzong Wang, Zhenhou Hong, and Jing Xiao. 2020. Empirical studies of institutional federated learning for natural language processing. In *Proceedings of EMNLP*.

Received 20 May 2023; revised 2 November 2023; accepted 10 November 2023