More is Better (Mostly): On the Backdoor Attacks in Federated Graph Neural Networks

Xu, J.; Wang, R.; Koffas, S.; Liang, K.; Picek, S.

DOI
10.1145/3564625.3567999

Publication date
2022

Document Version
Final published version

Published in
Proceedings - 38th Annual Computer Security Applications Conference, ACSAC 2022

More is Better (Mostly): On the Backdoor Attacks in Federated Graph Neural Networks

Jing Xu
j.xu-8@tudelft.nl
Delft University of Technology
Delft, The Netherlands

Rui Wang
r.wang-8@tudelft.nl
Delft University of Technology
Delft, The Netherlands

Stefanos Koffas
s.koffas@tudelft.nl
Delft University of Technology
Delft, The Netherlands

Kaitai Liang
kaitai.liang@tudelft.nl
Delft University of Technology
Delft, The Netherlands

Stjepan Picek
picek.stjepan@gmail.com
Radboud University
Nijmegen, The Netherlands

ABSTRACT

Graph Neural Networks (GNNs) are a class of deep learning-based methods for processing graph domain information. GNNs have recently become a widely used graph analysis method due to their superior ability to learn representations for complex graph data. Due to privacy concerns and regulation restrictions, centralized GNNs can be difficult to apply to data-sensitive scenarios. Federated learning (FL) is an emerging technology developed for privacy-preserving settings when several parties need to train a shared global model collaboratively. Although several research works have applied FL to train GNNs (Federated GNNs), there is no research on their robustness to backdoor attacks.

This paper bridges this gap by conducting two types of backdoor attacks in Federated GNNs: centralized backdoor attacks (CBA) and distributed backdoor attacks (DBA). Our experiments show that the DBA attack success rate is higher than CBA in almost all cases. For CBA, the attack success rate of all local triggers is similar to the global trigger, even if the training set of the adversarial party is embedded with the global trigger. To explore the properties of two backdoor attacks in Federated GNNs, we evaluate the attack performance for a different number of clients, trigger sizes, poisoning intensities, and trigger densities. Finally, we explore the robustness of DBA and CBA against two state-of-the-art defenses. We find that both attacks are robust against the investigated defenses, necessitating the need to consider backdoor attacks in Federated GNNs as a novel threat that requires custom defenses.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → *Machine learning*;

KEYWORDS

backdoor attacks, graph neural networks, federated learning

ACSAC '22, December 5–9, 2022, Austin, TX, USA

ACM ISBN 978-1-4503-9759-9/22/12.
https://doi.org/10.1145/3564625.3567999

ACM Reference Format:
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 *Annual Computer Security Applications Conference (ACSAC '22), December 5–9, 2022, Austin, TX, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3564625.3567999

1 INTRODUCTION

Graph Neural Networks, which generalize traditional deep neural networks (DNNs) to graph data, pave a new way to effectively learn representations for complex graph-structured data [45]. Due to their strong representation learning capability, GNNs have demonstrated remarkable performance in various domains, e.g., drug discovery [27, 48], finance [8, 41], social networks [12, 16], and recommendation systems [11, 52]. 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. 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 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, 54], which we denote as *Federated GNNs*.

Although FL has been successfully applied in diverse domains, e.g., computer vision [28, 29] or language processing [18, 58], there could be malicious clients among millions of clients, leading to various adversarial attacks [1, 13]. 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 recently studied [1, 3, 47]. However, these attacks are applied in federated learning on the Euclidean data, e.g., images and words. The backdoor trigger generation methods and injecting position are different between graph data and images/words [49]. In particular, in [47], the authors 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. Also, defenses like FoolsGold [14] 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. Additionally, intensive research has been conducted on backdoor attacks in GNNs [46, 49, 56]. However, these works focus on GNN models in centralized training. In federated learning, the malicious updates will be weakened in the aggregation function. Finally, there can be more than one malicious client, while 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 paper conducts two backdoor attacks in FL: centralized backdoor attacks (CBA) and distributed backdoor attacks (DBA) [47]. In CBA, the attacker embeds the same global trigger to all adversarial clients, while 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 federated learning setting and our main contributions are:

- 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 paper is the first work studying backdoor attacks in Federated GNNs.
- 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 (CNNs).
- 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.
- We explore the robustness of DBA and CBA against two state-of-the-art defenses: FLAME and FoolsGold. We find both attacks are evasive to these defenses, while CBA can even obtain a higher attack success rate, but the testing accuracy degrades.

2 BACKGROUND

2.1 Federated Learning

Federated Learning enables n clients to train a global model $\boldsymbol{w}$ collaboratively without revealing local datasets. Unlike centralized learning, where local datasets must be collected by a central server before training, FL performs training by uploading the weights of local models ($\{\boldsymbol{w}^i \mid i \in n\}$) to a parametric server. Specifically, FL aims to optimize a loss function:

$$\min_{\boldsymbol{w}} \ell(\boldsymbol{w}) = \sum_{i=1}^{n} \frac{k_i}{n} L_i(\boldsymbol{w}), L_i(\boldsymbol{w}) = \frac{1}{k_i} \sum_{j \in P_i} \ell_j(\boldsymbol{w}, x_j), \quad (1)$$

where $L_i(\boldsymbol{w})$ and k_i are the loss function and local data size of i-th client, and P_i refers to the set of data indices with size k_i.

At the t-th iteration, the training can be divided into three steps:

- *Global model download.* All clients download the global model $\boldsymbol{w}_t$ from the server.
- *Local training.* Each client updates the global model by training with their datasets: $\boldsymbol{w}_t^i \leftarrow \boldsymbol{w}_t^i - \eta \frac{\partial L(\boldsymbol{w}_t, b)}{\partial \boldsymbol{w}_t^i}$, where η and b refer to learning rate and local batch, respectively.
- *Aggregation.* After the clients upload their local models $\{\boldsymbol{w}_t^i \mid i \in n\}$, the server updates the global model by aggregating the local models. In this paper, we use the averaging aggregation function: $\boldsymbol{w}_{t+1} \leftarrow \sum_{i=1}^{n} \frac{1}{n} \boldsymbol{w}_t^i$.

2.2 Graph Neural Networks

Recently, Graph Neural Networks (GNNs) have achieved significant success in processing non-Euclidean spatial data, which are very common in many real-world scenarios. Unlike traditional neural networks, e.g., CNNs and Recurrent Neural Networks (RNNs), GNNs work on graph data. GNNs take a graph $G = (V, E, X)$ as an input, where V, E, X denote nodes, edges, and node attributes, 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 [50]. Formally, the k-th layer of a GNN is (e.g., GCN [23], GraphSAGE [17], and GAT [40]):

$$z_{\boldsymbol{v}}^{(k)} = \sigma(z_{\boldsymbol{v}}^{(k-1)}, AGG(\{z_{\boldsymbol{u}}^{(k-1)}; \boldsymbol{u} \in \mathcal{N}_{\boldsymbol{v}}\})), \forall k \in [K], \quad (2)$$

where $z_{\boldsymbol{v}}^{(k)}$ is the representation of node $\boldsymbol{v}$ computed in the k-th iteration. $\mathcal{N}_{\boldsymbol{v}}$ are 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, while σ is an activation function. For the graph classification task (considered in this work), the READOUT function pools the node representations for a graph-level representation z_G:

$$z_G = READOUT(z_{\boldsymbol{v}}; v \in V). \quad (3)$$

READOUT can be a simple permutation invariant function such as summation or a more sophisticated graph-level pooling function [53, 55].

2.3 Backdoor Attacks on Federated Learning

Backdoor attacks aim to make a model misclassify its inputs to a preset-specific label without affecting its original task. Attackers poison the model by injecting triggers into the training data that activate the backdoor in the test phase. Once activated, the model's output becomes the targeted label pre-specified by the attacker to achieve the malicious intent purpose (such as misclassification).

Backdoor attacks are common in FL systems with multiple training dataset owners. Specifically, the adversary $\mathcal{A}$ manipulates one or more local models to obtain poisoned models $\tilde{W}^i$ that are then aggregated into the global model G_t affecting its properties. There are two common techniques used in backdoor attacks in FL: 1) data poisoning where $\mathcal{A}$ manipulates local training dataset(s) D^i_{local} used to train the local model [34, 47], and 2) model poisoning where $\mathcal{A}$ manipulates the local training process or the trained local models themselves [1]. In this work, we use 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 [51] and Krum [4].

3 PROBLEM FORMULATION

3.1 Overview

FL is a practical choice to push machine learning to users' devices, e.g., smart speakers, cars, and phones. Usually, federated learning is designed to work with thousands or even millions of users without restrictions on eligibility [1], opening up new attack vectors. As stated in [5], training with multiple malicious clients is now considered a practical threat by the designers of federated learning. Because of the data privacy guarantee among the clients in the federated learning, local clients can modify their local training dataset without being noticed. Furthermore, existing federated learning 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.

3.2 Threat Model

Unlike traditional machine learning benchmarking datasets, graph datasets and real-world graphs may exhibit non-independent and identical distribution (non-i.i.d) 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. In Table 1,[1] 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 II) on two molecular datasets, i.e., NCI1 and PROTEINS_full. For these experiments, we set 5 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 two to one hundred) [22]. Besides the molecular domain, substantial attention has also been given to Federated GNNs in real-world financial scenarios [42, 54]. 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 federated learning setting [22]. As shown in Exp. III and IV, we assume 10, 20, and 100 clients for a synthetic dataset, i.e., TRIANGLES, which is a realistic real-world cross-silo scenario [38].

Table 1: 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

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 below (the definitions of the local trigger and global trigger used in these two attacks are also given).

Definition 1 (Local Trigger & 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, i.e., a poisoned local dataset is used by malicious clients in every round, following the attack setting in [47]. 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.

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

[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.

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 [36] aiming to deteriorate the accuracy of the model, the backdoor attacks studied in this paper 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 distributed backdoor attacks, 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 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:

$$w_t^{i*} = \underset{w_t^i}{\text{argmin}}\Big(\sum_{j\in D^i_{trigger}} \ell(w^i_{t-1}(\Phi(x^i_j,\kappa^i),y_t)) + \sum_{j\in D^i_{clean}} \ell(w^i_{t-1}(x^i_j),y^i_j)\Big), \forall i \in [M], \tag{4}$$

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 $D^i_{trigger} \cap D^i_{clean} = \emptyset$. 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 paper, 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 to be 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 the number 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 (ER) 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:

$$w_t^{k*} = \underset{w_t^k}{\text{argmin}}\Big(\sum_{j\in D^k_{trigger}} \ell(w^k_{t-1}(\Phi(x^k_j,\kappa),y_t)) + \sum_{j\in D^k_{clean}} \ell(w^k_{t-1}(x^k_j),y^k_j)\Big), \tag{5}$$

[3]In practice, the centralized attack can poison more than one client with the same global trigger, as mentioned in [1]. Here, we assume there is one malicious client

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

We focus on subgraph-based (data poisoning) backdoor attacks and the graph classification task. Attackers can perform DBA or CBA as shown in Figure 1. In DBA, multiple malicious clients engage in attacking, and they inject local triggers into corresponding malicious clients' local training datasets. 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 paper in Table 5 in Appendix A.

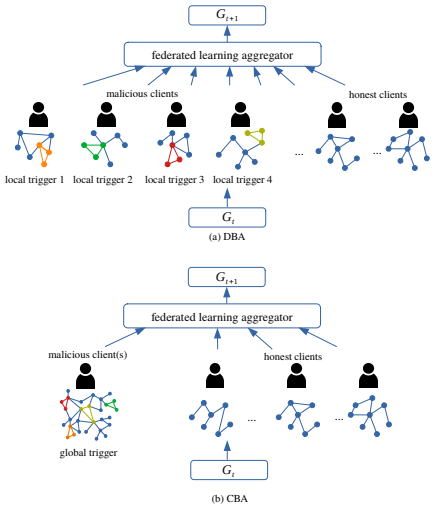


Figure 1: Attack Framework.

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 1(a). Each malicious client embeds its local training dataset with a specific graph trigger to poison its local model. For instance, in Figure 1(a), each malicious client has a local trigger highlighted by a specific color (i.e., orange, green, red, yellow).[4] In this paper, 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 distributed backdoor attack in Federated GNNs.

[4]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.

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).

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 attack success rate. We provide additional motivation for the malicious majority setting in Section 7.

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 1(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.[5]

To compare the attack performance between the distributed backdoor attack and centralized backdoor attack 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.

4.2 Backdoored Data Generation

We adopt the Erdős-Rényi (ER) model [15] to generate triggers (function *GenerateTrigger* in Algorithm 2) as it is more effective than the other methods (e.g., Small World model [43] or Preferential Attachment model [2]) [56]. 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.

Backdoored data is generated (line 4 in Algorithm 2) through the following process. 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. Additionally, the attacker re-labels the sampled data with an attacker-chosen target label. The backdoored data is composed of the dataset with trigger and the original clean dataset.

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

Algorithm 1: Distributed Backdoor Attacks in Federated GNNs

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

1 **Function** DBA():
2 $C_h, C_m \leftarrow ClientSplit(Clients)$
3 $D_{local}, D_{test} \leftarrow DataSplit(D)$
4 $t_{global} \leftarrow \emptyset$
5 **Server executes:**
6 initialize G_0, $f = False$
7 **foreach** *round* $t = 0, 1, 2, \ldots$ **do**
8 **foreach** *client* $k \in (C_h \cup C_m)$ **do**
9 $w_t^k = G_t$
10 **if** $k \in C_m$ **then**
11 $f = True$
12 **end**
13 $w_{t+1}^k \leftarrow ClientUpdate(k, w_t^k, f, t_{global})$
14 **end**
15 $G_{t+1} \leftarrow \sum_{k=1}^{K} \frac{w_{t+1}^k}{K}$
16 **end**
17 **End Function**
18 **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

1 **Function** ClientUpdate():
2 **if** f *is* $True$ **then**
3 $t_{local} \leftarrow GenerateTrigger(s, \rho)$
4 $D_{local} \leftarrow BackdoorDataset(D_{local}, t_{local}, y_t)$
5 $t_{global} = t_{global} \cup t_{local}$
6 **end**
7 $\mathcal{B} \leftarrow$ (split D_{local} into batches of size B)
8 **foreach** *local epoch i from* 1 *to E* **do**
9 **foreach** $b \in \mathcal{B}$ **do**
10 $w \leftarrow w - \eta \nabla l(w, b)$
11 **end**
12 **end**
13 **End Function**
14 **return** w

5 EXPERIMENTS

5.1 Experimental Setting

We implemented FL algorithms using the PyTorch framework. All experiments were run on a server with 2 Intel Xeon CPUs, one NVIDIA 1080 Ti GPU with 32GB RAM, and Ubuntu 20.04 LTS OS. Each experiment was repeated ten times to obtain the average result. Our code is blinded for review but will be made public.

Datasets. We run experiments on three publicly available datasets: two molecular structure datasets - NCI1 [32], PROTEINS_full [6], and one synthetic dataset - TRIANGLES [24], which is a multi-class dataset. Table 6 in Appendix B 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 [13] with hyperparameter 0.5 for TRIANGLES (10 classes) and hyperparameter 0.7 for other datasets (2 classes). In this paper, apart from Appendix C

where we analyze the effect of trigger factors, we set trigger factors as follows: $\gamma = 0.2$, $\rho = 0.8$, and $r = 0.2$. As we show in Appendix C, these hyperparameters yield an effective attack. 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 [40], and GraphSAGE [17].

We use the *attack success rate (ASR)* to evaluate the attack effectiveness. 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:

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

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.

5.2 Backdoor Attack Results

We evaluate multiple-shot attack [1], 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 [1] because the multi-shot is stealthier [35]. The multi-shot attack does not require multiplying large factors to model weights when conducting the attack, while 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 [4]. Since our main goal is conducting backdoor attacks on FL, we chose a multiple-shot attack with a high attack success rate 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 [47].

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:

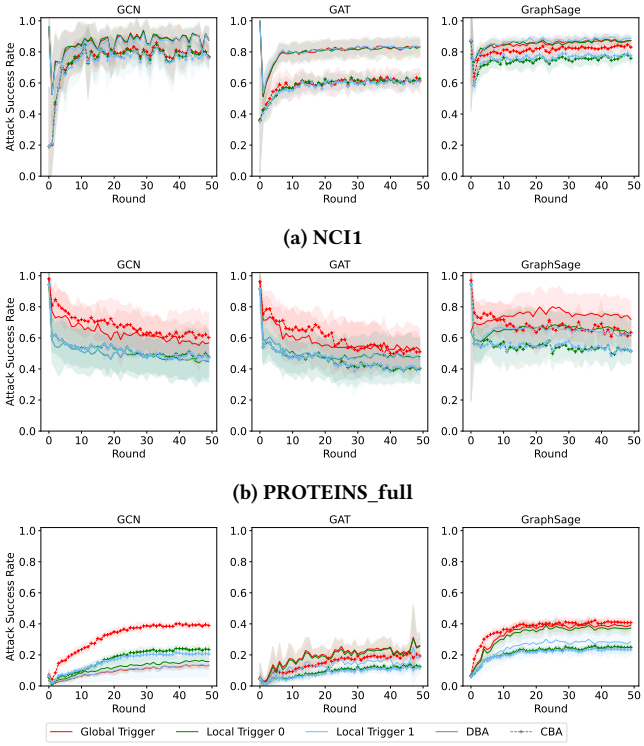
- In CBA, whether the ASR of local triggers can achieve similar performance to the global trigger even if the centralized attacker would embed a global trigger into the model.
- In DBA, whether the ASR of the global trigger is higher than all local triggers even if the global trigger never actually appears in any local training dataset, as mentioned in [47].

Honest Majority Attack Scenario. The attack results of CBA and DBA in the honest majority attack scenario are shown in Figure 2. 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 2a (the result for the GAT model), the DBA ASR with the global trigger is higher than 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. The 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 attack success rate for both DBA and CBA, resulting from the high local learning rate of honest clients [1]. *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 [47].* We analyze it through further experiments shown in Figure 6.

Moreover, the results for the PROTEINS_full dataset show that *in DBA, the attack success rate 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 attack success rate of the global trigger does not require the same high attack success rate of local triggers. However, for the other two datasets (NCI1 and TRIANGLES), the attack success rate 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 [47] 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 3 illustrates the attack results in the malicious majority attack scenario. Compared with the honest majority attack scenario, in most cases, the attack success rate of DBA and CBA increases as with more malicious clients, more malicious updates are uploaded to the global model, making the attack more effective and persistent. Moreover, the increase in DBA is more significant than in CBA. For instance, based on the NCI1 dataset and GraphSage model, the DBA ASR with the global trigger in the honest majority attack scenario is 3.85% higher than CBA, while in the malicious majority attack scenario, the ASR difference is 10.33%. 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, while there is only one malicious local model in CBA.

For CBA, the ASR with the global trigger is higher while 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

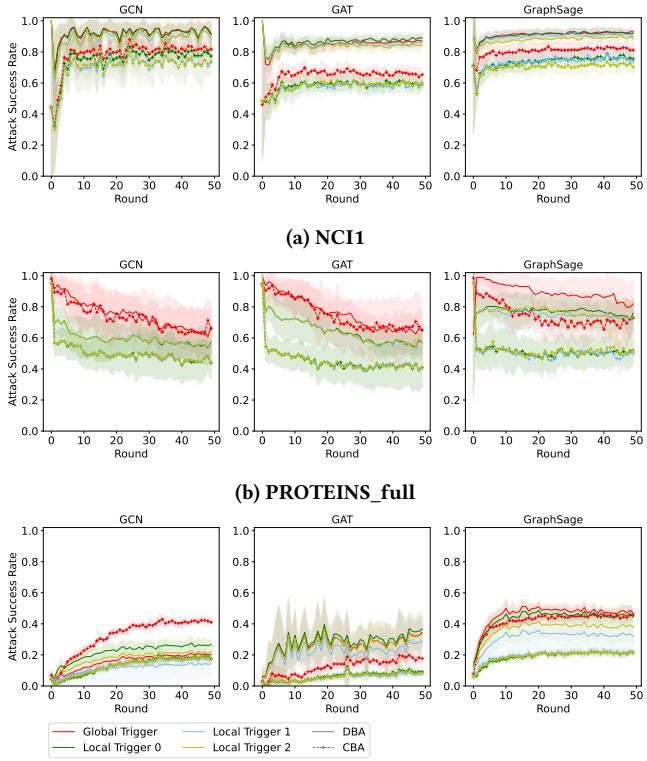


(a) NCI1

(b) PROTEINS_full

(c) TRIANGLES

Figure 2: Backdoor attack results in the honest majority attack scenario.



(a) NCI1

(b) PROTEINS_full

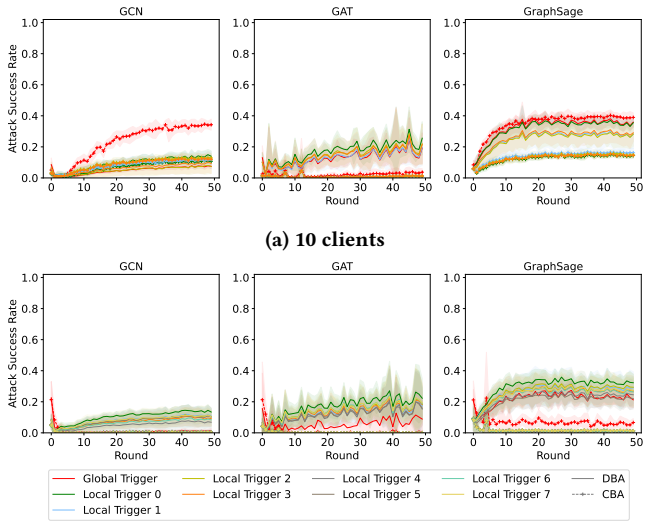
(c) TRIANGLES

Figure 3: Backdoor attack results in the malicious majority attack scenario.

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.

Impact of the Number of Clients We only set the number of clients as 5 for these graph datasets because some of these datasets, i.e., NCI1 and PROTEINS_full, are small (less than 5,000 graphs). However, to explore the impact of the number of clients on DBA and CBA, we also conduct experiments with more clients on the largest dataset - TRIANGLES. We set the number of clients as 10 and 20 and keep the ratio of malicious clients among the total clients the same as before, i.e., 0.4 and 0.6 for the honest majority and malicious majority attack scenarios, respectively. Here, we provide the results of the honest majority attack scenario, as shown in Figure 4. The results of the malicious attack scenario are given in Appendix D.1, and the phenomenon between the two attack scenarios with 10 and 20 clients is similar to that with 5 clients.

It is obvious that with the increase in the number of clients, the attack success rate of CBA decreases dramatically while the attack performance of DBA keeps steady. This is reasonable because, in CBA, there is only one malicious client whose malicious updates contribute less to the global model with more clients in total. On the contrary, in DBA, the proportion of malicious clients among total clients is the same, meaning the malicious updates contribute the same to the global model regarding the different number of



(a) 10 clients

(b) 20 clients

Figure 4: Backdoor attack results of TRIANGLES with more clients in the honest majority attack scenario.

clients. Therefore, as shown in Figures 4a and 4b, the number of clients has negligible impact to the DBA.

Impact of the Percentage of Malicious Clients Although we have analyzed the experiments with the honest majority and malicious majority scenarios, we further explore the impact of the percentage of malicious clients on the attack performance by calculating their Pearson Correlation Coefficient (PCC), as shown in Figure 14 in Appendix E, (we provide the results for the GraphSage model as the example as they are more stable, and the results of other models are aligned). Recall that M represents the number of malicious clients, and each number over the line is the corresponding PCC. As we can see, for all datasets, PCC in DBA is larger than CBA, meaning the increase in M has a more positive impact on DBA than CBA. This is intuitive as more malicious clients in DBA lead to more local models embedded with local triggers, while in CBA, it means a larger global trigger due to more local triggers. Specifically, in DBA, more malicious clients mean more model weights to learn the trigger. In CBA, there is only one attacker, and learning a larger global trigger can be out of the model's representation capability. Additionally, as we keep the poisoning intensity of DBA and CBA the same for each malicious client, there are more poisoned training data in DBA than CBA as more malicious clients are used.

We also explore the attack performance with more clients and less percentage of malicious clients on the large dataset - TRIANGLES. Figure 5 shows the attack results on TRIANGLES with 100 clients and fewer malicious clients, ranging from 5% to 20% (here, we also take the results of the GraphSage model as the example, the results of other models are presented in Appendix D.2). Table 2 illustrates the specific attack results. We can see from Figure 5 that DBA's ASR gradually increases with more malicious clients while CBA's ASR stays very low (around 2%), further verifying that the increase in M has a more positive impact on DBA than CBA. Comparing Figures 5 and 4, with 20% malicious clients, DBA can also achieve similar ASR (nearly 20%) to that of 40% malicious clients (ASR of 21%), which means with less percentage (e.g., 20%) of malicious clients, the DBA is still effective. With more clients in total, the attack performance of CBA decreases, consistent with the observation in Figure 4. Thus, adding more clients does not change our previous conclusions (with 5 clients).

Table 2: Attack success rate of CBA and DBA with less percentage of malicious clients in TRIANGLES (K=100, GraphSage).

Model	Attack Success Rate (CBA% | DBA%)			
	5%	10%	15%	20%
GCN	2.76 | 1.07	2.48 | 1.45	2.70 | 2.25	2.67 | 6.84
GAT	0.29 | 2.51	0.33 | 4.75	0.12 | 8.16	0.12 | 15.24
GraphSage	1.96 | 9.30	2.01 | 15.16	2.50 | 17.63	2.40 | 19.99

Analysis of CBA results In Figure 2, for CBA, the attack success rate 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 GraphSage model) where we evaluate the attack success rate of the global trigger and local triggers in both the malicious local model [6] and the global model. As shown in Figure 6, 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 [47], for the CBA, the attack success rate 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 [47] 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.

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 *clean accuracy drop (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 clean accuracy drop results in the honest and malicious majority attack scenarios are given in Tables 3 and 4, 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, e.g., DBA's CAD is 11.11% in the GAT model on TRIANGLES while CBA's is 1.58%. At the same poisoning intensity for each client, there are more poisoned data in DBA than CBA, leading to worse performance in the main task. The substantial clean accuracy drop in DBA can also be observed in [47].

Table 3: Clean accuracy drop of CBA and DBA in the honest majority attack scenario.

Dataset	Clean Accuracy Drop (CBA% | DBA%)		
	GCN	GAT	GraphSage
NCI1	2.73 | 3.88	0.37 | 1.75	0.91 | 0.53
PROTEINS_full	0.19 | 2.50	0.72 | 0.87	3.23 | 1.82
TRIANGLES	0.18 | 0.06	2.21 | 2.39	1.22 | 6.70

6 DEFENSES

Potential Countermeasures FLAME [33] is one of the state-of-the-art defenses against backdoor attacks in FL, combining the

[6] For the 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

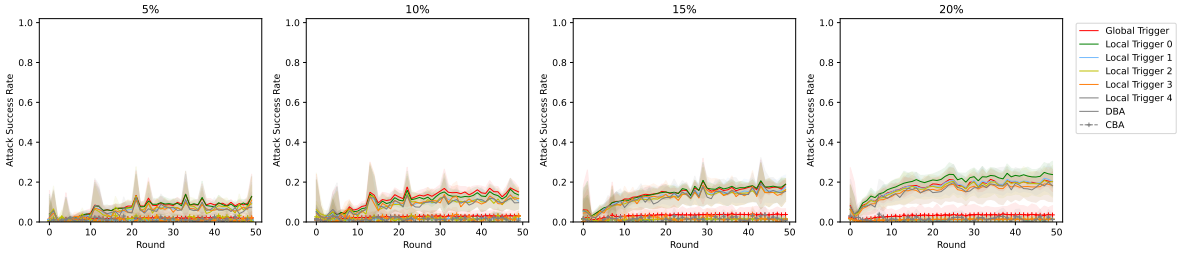
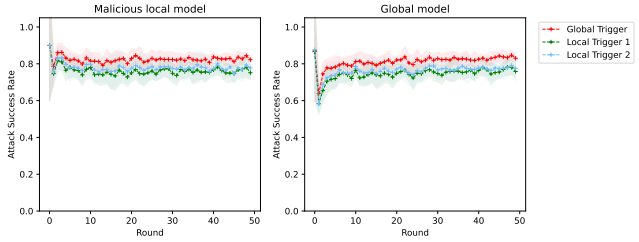
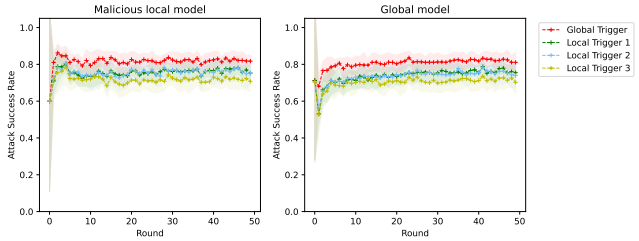


Figure 5: Backdoor attack results of TRIANGLES with less percentage of malicious clients ($K = 100$, GraphSage).



(a) Honest majority attack scenario



(b) Malicious majority attack scenario

Figure 6: Centralized backdoor attack results on the malicious local model and global model with different triggers.

Table 4: Clean accuracy drop of CBA and DBA in the malicious majority attack scenario.

Dataset	Clean Accuracy Drop (CBA% | DBA%)		
	GCN	GAT	GraphSage
NCI1	2.21 | 2.50	0.43 | 0.02	1.07 | 0.91
PROTEINS_full	2.13 | 0.65	3.44 | 6.15	0.99 | 0.85
TRIANGLES	0.22 | 0.36	1.58 | 11.11	1.62 | 9.72

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 [14] is a robust FL aggregation algorithm that can identify attackers in federated learning based on the diversity of 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 paper. Thus, we focus on evaluating the attack effectiveness of DBA and CBA against both FLAME and FoolsGold.

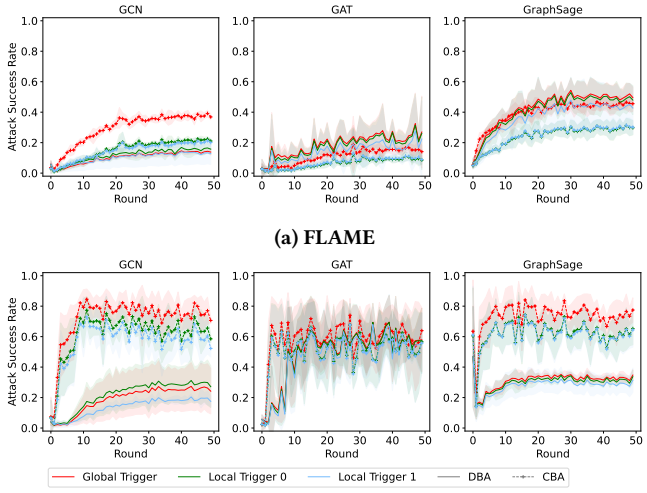
Results and Analysis Figures 7 and 8 show the attack performance for the TRIANGLES dataset under FLAME and FoolsGold in the honest majority attack scenario (the results in the malicious majority attack scenario are similar). The results for other datasets illustrate that these two defenses have a negligible impact on the attack performance, as shown in Appendix D.3. As we can see in Figure 7 and 8, generally, for both defenses, once there is an obvious increase in the ASR of an attack, the testing accuracy of the corresponding attack decreases. For example, under FLAME, the DBA's and CBA's ASR stays steady for GCN and GAT models while it increases by about 10% for the GraphSage model. However, the testing accuracy of these two attacks on GraphSage has a more obvious drop than on other models (Figure 8a).

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. 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 paragraph above) 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 reported FoolsGold's weights on every client in DBA and CBA in Appendix F and showed 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 defenses, we find that both defenses cannot detect malicious updates successfully. One reason may be that both methods apply *cosine distance* to try to identify malicious models, i.e., the distance between malicious updates is smaller than 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. It thus seems crucial to design a defense specifically for the backdoor attacks in Federated GNNs.

7 RELATED WORK

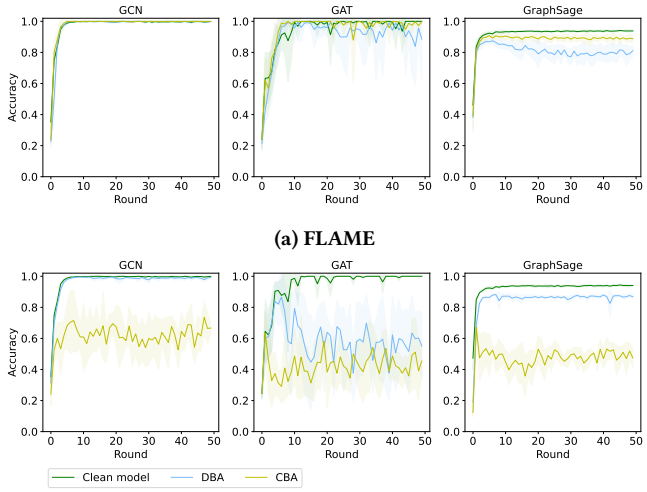
Backdoor Attacks in GNNs Several recent works have conducted backdoor attacks on GNNs. Zhang et al. proposed a subgraph-based backdoor attack on GNNs for the graph classification task [56]. Xi et al. presented a subgraph-based backdoor attack on GNNs, that works for both node classification and graph classification tasks [46]. Xu et al. investigated the explainability of the impact



(a) FLAME

(b) FoolsGold

Figure 7: Backdoor attack results of TRIANGLES on two defenses: FLAME and FoolsGold.



(a) FLAME

(b) FoolsGold

Figure 8: Testing accuracy of TRIANGLES on two defenses: FLAME and FoolsGold.

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 [49]. *All current attacks are implemented in centralized training for GNNs. No works explore the backdoor attacks in distributed training for GNNs, e.g., Federated GNNs.*

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. While FL has been widely studied in Euclidean data, e.g., images, texts, and sound, there are increasing studies about FL in graph data. FL on graph data was introduced in [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 overreporting of suspicious activities. Thanks to the reasoning ability of the graph neural network, its advantages can be well-reflected. Considering the need for privacy, [39] proposed the framework for Federated GNNs to optimize the machine learning model. Besides, other research works [21, 44, 57] have been dedicated to enhancing the security of Federated GNNs. By using secure aggregation, [21] proposed a method to predict the trajectories of objects via aggregating both spatial and dynamic information without information leakage. With differential privacy, [57] and [44] put forward a framework to train Federated GNNs for vertical FL and recommendation system, respectively. Moreover, SpreadGNN was proposed in [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 underexplored.*

The Security Assumption of Malicious Majority Clients Cao et al. took into account the situation of backdoor attacks in the malicious majority of clients and proposed a method of defense-*FLTrust* [7]. Before training begins, an honest server collects and trains on a small dataset. The server takes the updates obtained by training on a small dataset as the root of trust in each iteration. It is then compared to the updates uploaded by the clients. If the cosine similarity between them is too small, the updates will be filtered out. With this approach, the accuracy of the global model remains equivalent to that of the baseline. Based on *FLTrust*, Dong et al. considered the setting of two semi-honest servers and malicious majority clients and proposed *FLOD* to ensure that gradients are not leaked on the server side [10].

8 CONCLUSIONS AND FUTURE WORK

This paper explores how Centralized and Distributed Backdoor attacks behave in Federated GNNs. Through extensive experiments on three datasets and three popular GNN models, we showed that generally, DBA achieves a higher attack success rate 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 state-of-the-art defenses for the backdoor attack in FL, necessitating the need for custom defenses. Interestingly, the CBA's ASR is even higher under one defense. The experimental setting in this work verifies the effectiveness of our method in a cross-silo federated learning setting and motivates further research in exploring backdoor attacks in Federated GNNs considering cross-device FL [38]. 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.

REFERENCES

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

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

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

[4] 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).

[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, et al. 2019. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046* (2019).

[6] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, et al. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21 (2005).

[7] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. 2020. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. *arXiv preprint arXiv:2012.13995* (2020).

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

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

[10] Ye Dong, Xiaojun Chen, Kaiyun Li, Dakui Wang, and Shuai Zeng. 2021. FLOD: Oblivious Defender for Private Byzantine-Robust Federated Learning with Dishonest-Majority. In *Computer Security – ESORICS 2021*, Elisa Bertino, Haya Shulman, and Michael Waidner (Eds.).

[11] Shaohua Fan, Junxiong Zhu, Xiaotian Han, et al. 2019. Metapath-guided heterogeneous graph neural network for intent recommendation. In *Proceedings of the 25th ACM SIGKDD*.

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

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

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

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

[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 *NeurIPS*.

[18] Andrew Hard, Kanishka Rao, Rajiv Mathews, et al. 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, Peilin Zhao, Junzhou Huang, Murali Annavaram, and Salman Avestimehr. 2021. FedGraphNN: A Federated Learning System and Benchmark for Graph Neural Networks. arXiv:2104.07145 [cs.LG]

[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]

[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, 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 *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).

[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 *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 *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 *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*.

[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 *AISTATS*. PMLR.

[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, et al. 2022. FLAME: Taming Backdoors in Federated Learning. arXiv:2101.02281 [cs.CR]

[34] Thien Duc Nguyen, Phillip Rieger, Markus Miettinen, and Ahmad-Reza Sadeghi. 2020. Poisoning attacks on federated learning-based IoT intrusion detection system. In *Proc. Workshop Decentralized IoT Syst. Secur.(DISS)*.

[35] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. 2019. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445* (2019).

[36] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. 2018. Generative adversarial perturbations. In *CVPR*.

[37] Pavel Pudlák, Vojtěch Rödl, and Petr Savickỳ. 1988. Graph complexity. *Acta Informatica* 25, 5 (1988), 515–535.

[38] 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 *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1354–1371.

[39] Toyotaro Suzumura, Yi Zhou, Natahalie Baracaldo, et al. 2019. Towards federated graph learning for collaborative financial crimes detection. *arXiv preprint arXiv:1909.12946* (2019).

[40] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *ICLR* (2018). https://openreview.net/forum?id=rJXMpikCZ

[41] 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 *ICDM*. IEEE.

[42] 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.

[43] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world'networks. *nature* 393, 6684 (1998), 440–442.

[44] 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).

[45] 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.

[46] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. 2021. Graph backdoor. In *USENIX Security*.

[47] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. 2019. Dba: Distributed backdoor attacks against federated learning. In *ICLR*.

[48] Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, 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.

[49] Jing Xu, Minhui Xue, and Stjepan Picek. 2021. Explainability-based backdoor attacks against graph neural networks. In *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*.

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

[51] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*. PMLR, 5650–5659.

[52] Ruiping Yin, Kan Li, Guangquan Zhang, and Jie Lu. 2019. A deeper graph neural network for recommender systems. *Knowledge-Based Systems* 185 (2019), 105020.

[53] 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).

[54] 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).

[55] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *AAAI*.

[56] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. 2021. Backdoor attacks to graph neural networks. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*.

[57] 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]

[58] Xinghua Zhu, Jianzong Wang, Zhenhou Hong, and Jing Xiao. 2020. Empirical studies of institutional federated learning for natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*.

A NOTATION

In Table 5, we summarize the notations used throughout the paper.

Table 5: Notations used in this paper.

Notations	Descriptions
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 splitted from dataset D_{train}
D_{test}	testing dataset splitted 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
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

B DATASET STATISTICS

In Table 6, we show various statistics about the datasets used.

C 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 as in Section 5.1. We provide results for TRIANGLES and the GraphSage model as an example as 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 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 9.

Effects of Trigger Size From the ASR results in Figure 9, for both attacks and attack scenarios, with the increase of trigger size, the attack success rate rises significantly, e.g., 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 attack success rate, a higher poisoning intensity gives a higher attack success rate. 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 [46], 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 ASR than a larger poisoning intensity. We consider this an interesting observation and plan to investigate it in future work. 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 Figures 9a and 9b. This can be explained as 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 Figures 9c and 9d. Although more local data is poisoned, the other honest clients (the majority part) still guarantee the performance on the main task.

Effects of Trigger Density From Figure 9b, 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 [37]. 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 considered 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 9, 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 [49], 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.

D ADDITIONAL EXPERIMENTAL RESULTS

D.1 More Clients (Malicious Majority Attack Scenario)

The attack results on TRIANGLES with 10 and 20 clients in the malicious majority attack scenario are shown in Figure 10. In the malicious majority attack scenario, with more clients, the ASR of DBA keeps steady while that of CBA drops dramatically, which is consistent with the observations in the honest majority attack scenario, as shown in Figure 4.

D.2 Less Percentage of Malicious Clients

The attack results with less percentage of malicious clients on TRIANGLES are shown in Figure 11. Similar to the attack results for GraphSage (Figure 5)[7], DBA's ASR is gradually increasing with the rise in the percentage of malicious clients. On the contrary, the attack success rate of CBA keeps steady.

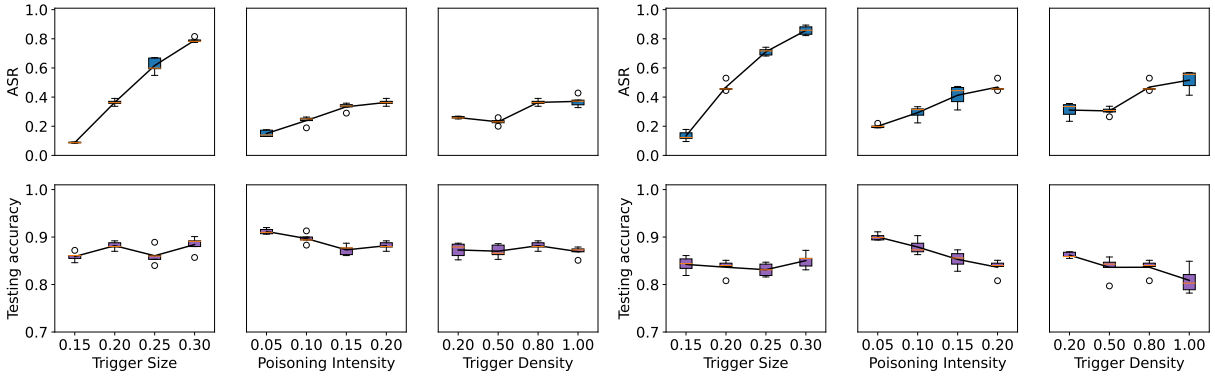
D.3 Additional Defense Results

The attack success rate under defenses on NCI1 and PROTEINS_full datasets (honest majority attack scenario) are shown in Figures 12 and 13, respectively. There is a slight increase in the attack success rate of DBA and CBA under two defenses: FLAME and FoolsGold, which indicates that both defenses fail to identify the malicious

[7]Here, we put the first 5 local triggers in the legend to make the figure more clear. The results for the rest local triggers have the same phenomenon

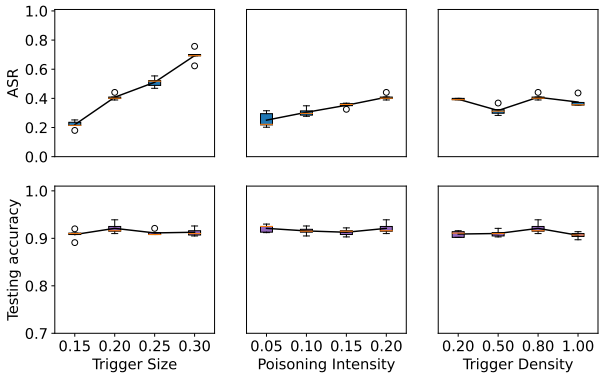
Table 6: Datasets statistics.

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	$4,500[0-9]$

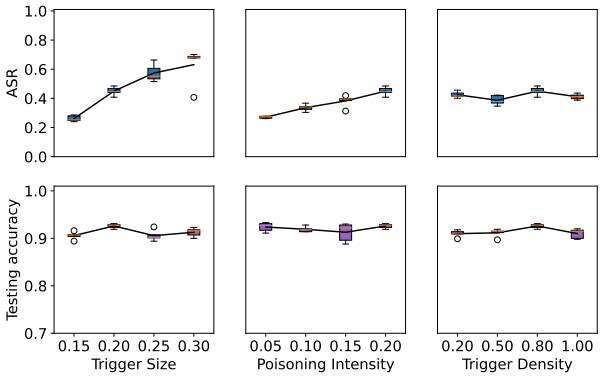


(a) DBA in honest majority attack scenario

(b) DBA in malicious majority attack scenario

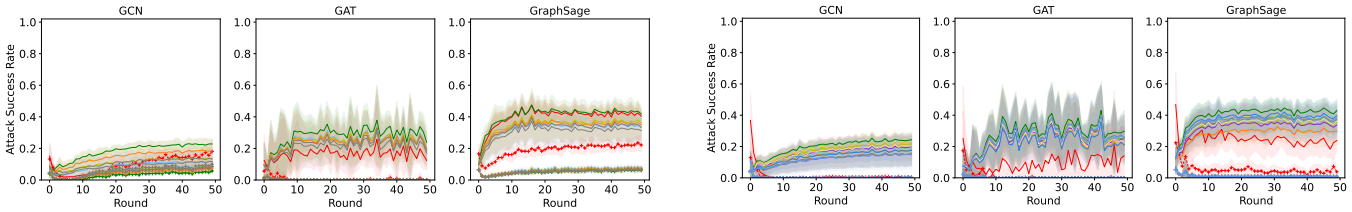


(c) CBA in honest majority attack scenario



(d) CBA in malicious majority attack scenario

Figure 9: Results on TRIANGLES with different trigger parameters.



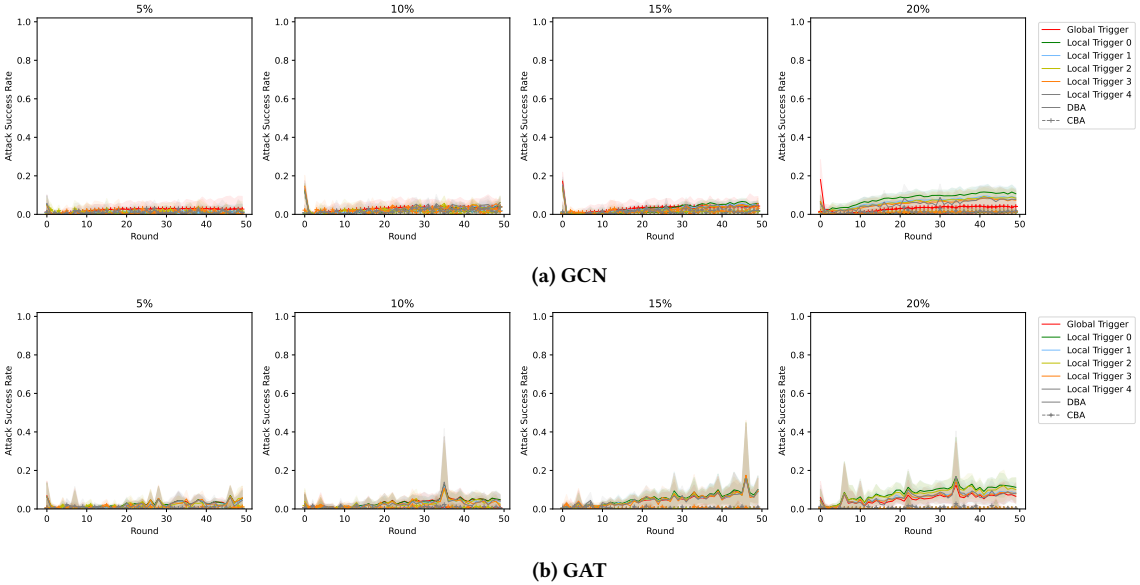
(a) 10 clients

(b) 20 clients

(c) Legend

Figure 10: Backdoor attack results of TRIANGLES with more clients in the malicious majority attack scenario.

updates and misclassify 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, the cosine similarity cannot be used to detect malicious clients based

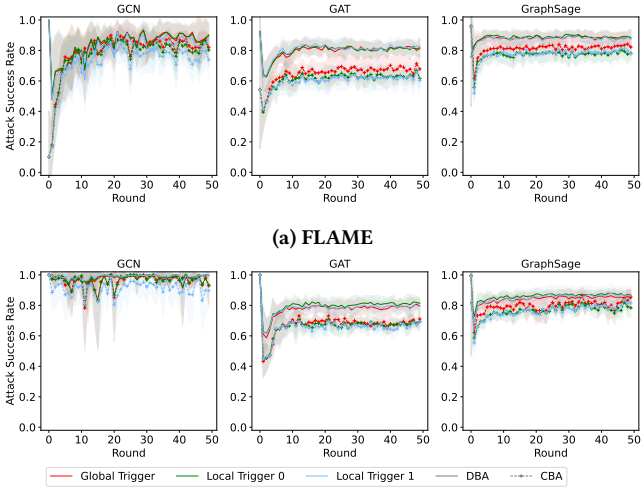


(a) GCN

(b) GAT

Figure 11: Backdoor attack results of TRIANGLES with less percentage of malicious clients ($K = 100$, GCN and GAT).

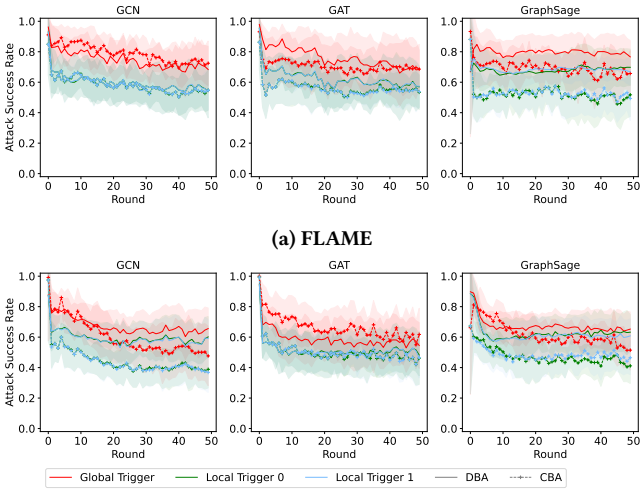
on their updates. Even though there are more malicious clients in the malicious majority scenario and the probability of detecting the malicious updates should be higher, we observe the same behavior. This further verifies our hypothesis that the defenses based on cosine similarity between updates are not very effective in the graph domain. The clean accuracy drop under the defenses on these two datasets is similar to that without the defense. Thus, the defenses do not affect the original task in that case.



(a) FLAME

(b) FoolsGold

Figure 12: Attack success rate on NCI1 on two defenses (in the honest majority attack scenario): FLAME and FoolsGold.



(a) FLAME

(b) FoolsGold

Figure 13: Attack success rate on PROTEINS_full on two defenses (in the honest majority attack scenario): FLAME and FoolsGold.

E IMPACT OF PERCENTAGE OF MALICIOUS CLIENTS

In Figure 14, we show the Pearson Correlation Coefficient of the percentage of malicious clients on the attack performance.

Table 7: FoolsGold weight in DBA and CBA on TRIANGLES (honest majority attack scenario).

Attacks	Attacker 1	Attacker 2 (client 2 in CBA)	Client 3	Client 4	Client 5	Attackers (sum)
DBA	0.57 ± 0.23	0.57 ± 0.23	0.86 ± 0.13	0.86 ± 0.13	1.00 ± 0.00	1.14 ± 0.23
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

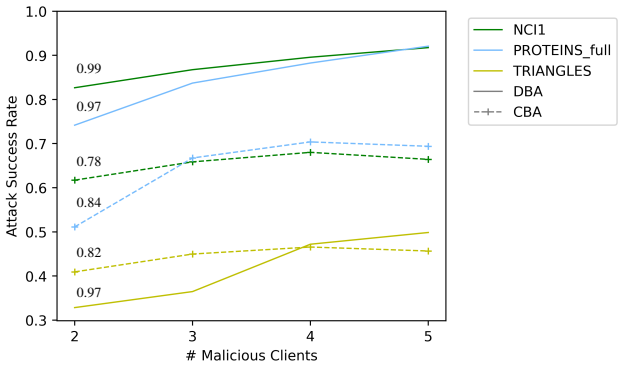


Figure 14: Correlation between ASR and M.

F 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 7. 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 attack success rate of CBA increases significantly under FoolsGold.

On the other hand, in DBA, the weights of the malicious clients are lower than the honest clients, indicating that the honest updates contribute more to the aggregated model. Therefore, there is a decrease of 5% in the DBA's ASR after the defense. The reported weights in Table 7 verify that our hypothesis is valid.