Backdoor Attack on Propagation-based Rumor Detectors

Di Jin[1], Yujun Zhang[1], Bingdao Feng[1,*], Xiaobao Wang[1,2,*], Dongxiao He[1], Zhen Wang[3]

[1]College of Intelligence and Computing, Tianjin University, Tianjin, China
[2]Guangdong Laboratory of Artiffcial Intelligence and Digital Economy (SZ), Shenzhen, China
[3]School of Cybersecurity, Northwestern Polytechnical University, Xi'an, Shaanxi, China
{jindi, zhangyujun, fengbingdao, wangxiaobao, hedongxiao}@tju.edu.cn, w-zhen@nwpu.edu.cn

Abstract

Rumor detection is critical as the spread of misinformation on social media threatens social stability. The propagation structure has garnered attention for its ability to capture discriminative information, such as crowd stance, which has led to the development of enhanced detection methods. However, these detectors are vulnerable to attacks that can manipulate results and evade detection, potentially disrupting public order or influencing public opinion. While adversarial attacks on rumor detectors have been studied, the use of backdoor attacks—an evasive and powerful method—remains unexplored due to the challenges in applying them to propagation trees. In this paper, we introduce the first backdoor attack framework against propagation-based rumor detectors, designed to maintain overall detector performance while enabling targeted attacks on specific rumors. We propose an adaptive discrete trigger generator that injects trigger nodes into critical nodes, creating evasive, transferable attacks. Extensive experiments on three real-world rumor datasets demonstrate that our framework effectively undermines the performance of propagation-based rumor detectors and is transferable across different architectures.

Introduction

With the rapid development of the Internet, social media has become an important channel for communication and information sharing. The widespread and rapid dissemination of inaccurate and unverified content, particularly malicious rumors, is causing significant harm to society. Therefore, the detection of rumors on social platforms has become crucial.

Deep learning plays an important role in rumor detection, as it can auto-efficiently learn feature vectors containing deep semantic information. For example, RNNs-based methods (Ma et al. 2016) can capture temporal relationships between posts. CNN-based methods (Lu and te Li 2020; Yu et al. 2017) can learn local spatial feature representations of rumors. Additionally, some multimodal studies (Dong et al. 2023, 2024; Hua et al. 2023) integrate text and image information for detection. However, these approaches primarily focus on the content of rumors, neglecting the structural information inherent in rumor propagation. This structural information is crucial, as it reflects the interrelationships between posts, reveals the collective wisdom and stance of user comments (Ma, Gao, and Wong 2018; Rosenfeld, Szanto, and Parkes 2020) and serves as a vital feature for distinguishing rumors from non-rumors. Therefore, to bolster detection capabilities, recent studies (Bian et al. 2020; Wei et al. 2021) have begun to incorporate the propagation structure into rumor detection models by leveraging GNN and integrating other advanced techniques. The propagation structure of rumors provides valuable clues on how a claim in the original post spreads and evolves over time.

Despite the excellent performance of propagation-based methods on rumor detection tasks, malicious users and miscreants may exploit the vulnerability of GNNs (Zügner, Akbarnejad, and Günnemann 2018) to evade or interfere with rumor detection results, raising concerns about security issues in rumor detectors. While recent studies have focused on adversarial attacks, such as AdRumor-RL (Lyu et al. 2023) and MARL (Wang et al. 2023), which employ reinforcement learning frameworks to explore the vulnerability of rumor detectors under structural adversarial attacks, a different type of attack, i.e., the backdoor attack (Zhang et al. 2021), has been largely overlooked. In a backdoor attack, the target model is compromised by injecting carefully crafted triggers and modifying labels in a small subset of the training data. Once trained, the model performs normally on clean samples but misclassifies samples containing the triggers as the attacker's desired labels. This allows the attacker to silently alter detection results during system operation. Such attacks can bypass initial security reviews and tests, posing a significant threat to rumor detection models. In this context, we investigate backdoor attacks on propagation-based rumor detectors. Given the critical role of the rumor propagation structure, our backdoor attack specifically targets the rumor propagation tree (RPT) constructed from it, thereby enhancing the attack's generality and effectiveness.

Several studies have explored backdoor attacks against GNNs (Zhang et al. 2021; Xi et al. 2021; Zheng et al. 2024). They typically perturb the graph topology to form fixed-size subgraph triggers, which perform well on graph classification tasks but are unsuitable for rumor detection tasks where propagation trees vary widely in size. More importantly, RPTs differ significantly from graphs, presenting two primary challenges when attempting to attack them: (i) Any

*Corresponding authors.

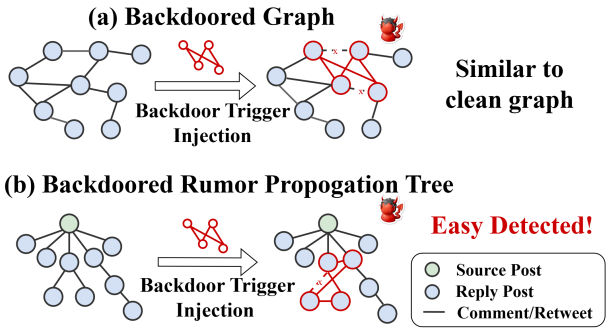


Figure 1: The differences between graphs and propagation trees for backdoor attack. (a) An example of injecting a backdoor trigger on a graph. (b) An example of injecting a backdoor trigger on a RPT. The subgraphs composed of red nodes and edges indicate triggers. Due to the complex graph structure, trigger-embedded graphs remain structurally similar to the original graphs, making the attack evasive. However, on RPTs, it is both easily detected and unrealistic.

perturbation to their structure is highly conspicuous and easily detected. As shown in Figure 1, adding edges may destroy the tree structure, and deleting edges can break the chains. Additionally, on real social platforms, attackers cannot tamper with social records posted by other users, and even if users delete their posts, social platforms may retain historical information. (ii) Choosing appropriate content to be injected as trigger features is also a significant challenge. The content of the injected post must be contextually related to the topic of the post it is replying to while retaining its attack properties to avoid detection and effectively trigger the backdoor. This dual requirement of contextual relevance and efficacy presents significant challenges in implementing backdoor attacks on rumor detectors.

To address these challenges, we propose an Injection-based Backdoor Attack (IBAttack), an adaptive trigger generation framework against propagation-based rumor detectors. To adapt to the rumor propagation structure, we employ a discrete trigger structure and generate adversarial node features that are contextually related to the original posts. These triggers are then attached to key delegate nodes, creating powerful, evasive, and transferable backdoor attacks. Our contributions are summarized as follows:

- To the best of our knowledge, we are the first work to study the vulnerability of propagation-based rumor detectors using backdoor attacks.
- We propose an injection-based backdoor attack framework specifically designed for rumor propagation graphs, which features an adaptive trigger generation module that creates effective and evasive backdoor triggers while dynamically adjusting the injection budget. These discrete triggers are designed to be difficult to detect, offering enhanced flexibility and control.
- Extensive experiments demonstrate that IBAttack significantly reduces rumor detector accuracy and exhibits strong transferability across different architectures.

Related Work

Propagation-based Rumor Detection

Ma et al. (2016) designed a bottom-up and top-down tree-structured recurrent neural network to extract information from RPTs. Similarly, BiGCN (Bian et al. 2020) applied a bidirectional GCN as well as root node feature enhancement techniques for rumor detection. Furthermore, EBGCN (Wei et al. 2021) studied the uncertainty in the propagation structure from a probability perspective. GACL (Sun et al. 2022) combined contrastive loss with adversarial training to learn a representation robust to rumor noise. RAGCL (Cui and Jia 2024) designed an adaptive graph contrastive learning method specifically for rumor detection.

Attacks on Rumor Detectors

Some studies have explored the vulnerability of rumor detectors using adversarial attacks. Some (Zhou et al. 2019) explored the robustness of NLP-based rumor detection by injecting adversarial text. Malcom (Le, Wang, and Lee 2020) modified each news comment to spoof a fake news detector using multi-source data. PEGEN (He, Ahamad, and Kumar 2021) attacked a sequence-based misinformation detector by generating texts that emulate malicious user behavior. AdRumor-RL (Lyu et al. 2023) and MARL (Wang et al. 2023) employed reinforcement learning frameworks to attack social context-based fake news detectors. GAFSI (Zhu et al. 2024) proposed a general black-box adversarial attack framework. Unlike previous work, we are the first to explore the vulnerability of propagation-based rumor detectors using a backdoor attack framework. Existing backdoor attacks on GNNs (Zheng et al. 2024; Yang et al. 2022; Xu, Xue, and Picek 2021) typically perturbed the topology of the graph to form subgraph backdoor triggers, which are not suitable for rumor detectors. In this work, we attempt to explore backdoor attacks against propagation-based rumor detectors.

Preliminaries

In this section, we first define propagation-based rumor detection and then introduce our backdoor attack objective.

Propagation-based Rumor Detection

The rumor detection task can be defined as a graph-level classification task. Specifically, we denote a rumor detection dataset as $\mathcal{C} = \{c_1, c_2, \ldots, c_i, \ldots, c_n\}$, where c_i is i-th event and n is the number of events. Each event $c_i = (G_i, y_i)$ comprises its propagation structure $G_i = (V_i, E_i)$, where V_i and E_i represent the set of graph nodes (a source post and comments of the post) and edges (the relations between pairs of replies or source post and a reply), respectively, and its ground-truth label $y_i \in \{N, R\}$ (i.e., Non-rumor or Rumor) or fine-grained label $y_i \in \{N, F, T, U\}$ (i.e., Non-rumor, False Rumor, True Rumor, Unverified Rumor). Given the dataset, the goal of the rumor detection task is to learn a classifier

$$f : G \to Y, \tag{1}$$

which maps each graph G to one of the labels in $Y = \{y_1, \ldots, y_n\}$. The parameters of f are learned by gradient-descent based optimization with a loss function $\mathcal{L}_{train}$ (e.g., cross-entropy) on the labeled dataset $\mathcal{C}$.

Backdoor Attack on Rumor Detectors

The backdoor attack injects a malicious function as a hidden neuron Trojan into the target model, and when the trigger is present, the hidden Trojan will be activated and mislead the model to get the desired output.

Attacker's Goal: There are two goals for an attacker to utilize a backdoor attack to influence the final trained rumor detectors: (i) the backdoor rumor detection model should predict the label we want on the propagation graph with a trigger. (ii) The backdoor attack should not influence the rumor detection model's accuracy on the benign propagation graph, which makes the attack evasive. Therefore, the attack objective can be defined as:

$$\begin{cases} f_{\theta_t}(m(G; g_t)) = y_t \\ f_{\theta_t}(G) = f_\theta(G) \end{cases}, \tag{2}$$

where G represents the clean graph and y_t is the targeted attack class. $m(\cdot)$ denotes a mixing function that injects trigger g_t into a given graph G to generate a trigger-embedded graph $m(G, g_t)$. f_θ and f_{θ_t} denote the benign rumor detector and the backdoor rumor detector respectively.

Attacker's Knowledge and Capability: In our work, we focus on more realistic attacks and assume the attacker has limited knowledge as shown in the following:

- Considering that platforms may use a wide variety of rumor detectors, we assume that the information of the target rumor detection models including model architecture is unknown to the attacker.
- The attacker could poison a small ratio of the training dataset. However, we restrict the attacker from changing the interaction history of existing posts.

Methodology

In this section, we present IBAttack in detail. The overall framework of the attack is shown in Figure 2. Distinguishing from existing backdoor attack methods for graph classification tasks, the triggers in our attack are more flexible, presented as a number of discrete new nodes. Specifically, we choose the most vulnerable and important nodes as attached nodes and inject new nodes into them to create a trigger-embedded graph. A feature generator is then trained to adaptively produce features for these injected nodes. In this way, we can exploit malicious comments as triggers to simulate controlled users' behavior in rumor propagation, thereby attacking propagation-based rumor detectors. In the following, we elaborate on each key component.

Adaptive Trigger Generation

Due to the previously mentioned restrictions, keeping the original graph intact, we adopt the strategy of the trigger g_t consisting of discrete injection nodes, i.e., connecting a certain budget of new nodes to the propagation graph in order to form a further evolving rumor propagation tree. Based on such a trigger structure, our goal is decomposed into attached node selection, trigger features generation and trigger injection.

Attached Nodes Selection We design an attached nodes selection scheme to decide which nodes the injected nodes should be linked to. Since GNN models aggregate information by learning the features of nodes and neighbors, key nodes are crucial in transferring information in the graph structure. Cui and Jia (2024) analyzed the structural characteristics of RPTs and summarized three principles for assigning importance scores to nodes to improve rumor detection performance. Similarly, we believe that connecting the trigger nodes to important nodes can achieve better attack performance than randomly injecting nodes. Based on the experiments in Table 6, there are two recommended node importance measures used by IBAttack, including degree centrality and node similarity to the root node.

- **Degree centrality** (Linton, C., and Freeman 1978) takes the degree of nodes as the measure of node centrality. Several studies (Wang et al. 2023) have shown that nodes with lower degrees are less robust to attacks than those with higher degrees, and intuitively, posts that are heavily discussed and replied to are harder to influence.
- **Similarity to the root node.** In a rumor propagation tree, the root node, serving as the source post, often contains more important and abundant information. Intuitively, nodes less similar to the root node are more likely to affect the graph.

We use the above two metrics to reflect the importance of node structure and features respectively. In different cases, we can adopt different strategies or combine two metrics to get an importance score:

$$S(v) = \beta sim(x_{root}, x_v) + (1 - \beta)\varphi_c(v), \tag{3}$$

where $sim(\cdot)$ denotes the cosine similarity function, $\varphi_c(\cdot)$ is the node centrality value of node v, and β is the fusion factor used to control the ratio of two indicators. x_{root} and x_v represents the feature of the root node and node v. For a candidate graph, we choose the Top M nodes based on the important score as attached nodes, each of which has a trigger node attached to it. The budget M and the injection process will be described in the trigger injection module.

Trigger Features Generation. Real-world graphs like social networks generally show homophily property, i.e., nodes with similar features are connected by edges. Rumor propagation graphs are no exception, the average similarity distributions between nodes in Twitter15 and Twitter16 are shown in Figure 3. Most graphs exhibit extremely high homogeneity, as all nodes within a propagation graph represent discussions related to the source post. Consequently, it is more likely that the same thematic information is embedded between the content of the reply and the original post. Based on this phenomenon, we add similarity constraint to the features of the nodes so that the trigger nodes can adaptively approach the features of the attached nodes. Let E_g denote the edge set that contains edges that connect trigger nodes

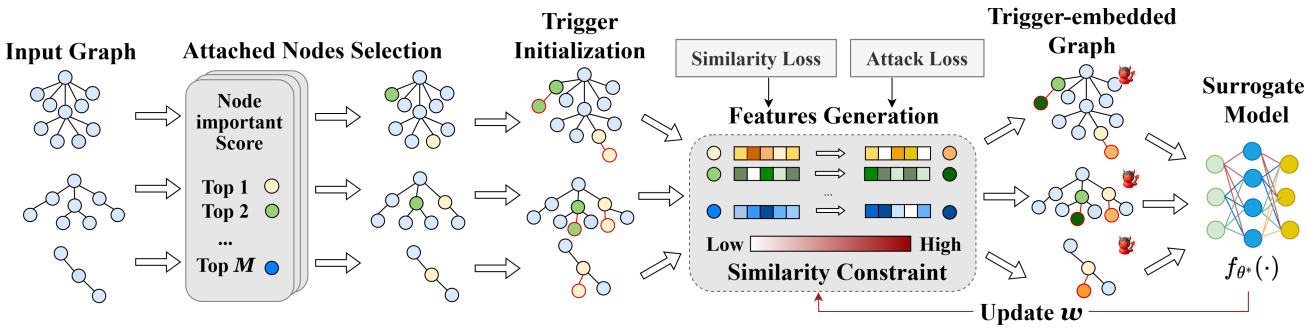
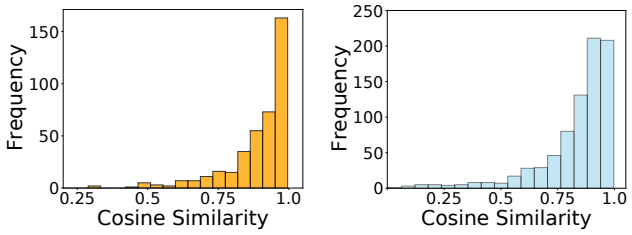


Figure 2: Overall framework of Injection-based Backdoor Attack (IBAttack).



(a) Twitter16 (b) Twitter15

Figure 3: Average similarity distribution of graphs

to attached nodes. The constraint on the generated adaptive triggers can be written as:

$$\min_{(u,v)\in E_g} sim(x_u, x_v) >= T, \tag{4}$$

where x_u, x_v represents the feature of the generated trigger node and attached node, T is a relatively high threshold of the cosine similarity which can be tuned based on datasets.

To generate adaptive triggers that are similar to the attached nodes, the adaptive feature generator takes the node features of the target nodes as input. Specifically, we generate injection node features by:

$$x_u = \sigma(W_1 \cdot x_v + b_1)W_2 + b_2, \tag{5}$$

where W_1, W_2, b_1 and b_2 are learnable parameters for feature generator, collectively referred to as w for simplicity in the subsequent discussion, and $\sigma(\cdot)$ is an activation function. With the feature generator, we can generate adaptive node features for trigger nodes g_t. In contrast to adversarial attacks that generate adversarial perturbations for each graph, we design and train a feature generator in order to mislead the model into capturing hidden associations between triggers and target class.

Trigger Injection For each graph G to be injected into the trigger, we obtain M attached nodes based on attached nodes selection. Subsequently, we adaptively generate the corresponding node feature for each trigger node using the trained feature generator. The mixing function $m(\cdot)$ then inject these M triggers nodes with generated features to their attached nodes as g_t while keeping the original edges and nodes unchanged. Formally, the backdoored graph can be represented as:

$$G_{g_t} = m(G; g_t) \quad \text{s.t.} \quad \|\mathbf{A}_{\mathbf{G}_{\mathbf{g_t}}} - \mathbf{A}_{\mathbf{G}}\|_0 \leq M. \tag{6}$$

To ensure that the backdoor attack remains unnoticeable, a certain number of candidate graphs are randomly sampled to obtain backdoored graphs. Putting these graphs into the training dataset, they participate in the target model training phase to inject hidden Trojan (i.e., the pattern of our triggers). Once the backdoor model is trained, the trigger can be inserted into any graph to activate the Trojan during the testing phase. Notably, due to the large disparities in the sizes of the RPTs, we flexibly control the trigger size (i.e., injection budget M) for different samples, tailoring it to the number of nodes in each graph. The ratio of trigger size to graph size can vary at different phases.

Feature Loss Function. Considering the similarity constraint of Eq. 4, the key idea is to ensure that the attached nodes are connected to a trigger node with high cosine similarity to increase the evasiveness of the triggers and the reliability of the edges. Inspired by UGBA (Dai et al. 2023), we define a differentiable similarity loss to help optimize the feature generator:

$$\mathcal{L}_s = \sum_{(u,v)\in E_g} max(0, T - sim(x_u, x_v)), \tag{7}$$

where T denotes the threshold of the similarity. This loss is applied between all attached nodes and their trigger nodes to ensure the similarity constraint holds.

For a trained model, a candidate graph G prior to trigger injection tends to be classified as its true label y. Within the multiple constraints of structure, content and budget, the attacker needs the model to associate the trigger pattern with the target label as closely as possible. Hence, to guarantee the effectiveness of the generated triggers, we define an attack loss to further optimize the trigger feature generator:

$$\mathcal{L}_{atk} = \log f_{\theta^*}(G_{g_t})_y - \log f_{\theta^*}(G_{g_t})_{y_t}, \tag{8}$$

where $f_{\theta^*}(\cdot)_y$, $f_{\theta^*}(\cdot)_{y_t}$ denotes the logit output for true label y and target label y_t of the surrogate model.

Optimization

We partition dataset $\mathcal{C}$ into two parts, $\mathcal{C}[y_t]$ - the graphs in the target class y_t and $\mathcal{C}[\backslash y_t]$ - the ones in other classes.

Algorithm 1: IBAttack

Input: Graph dataset $\mathcal{C}$, target label y_t, attack budget M
Output: Backdoored dataset $\mathcal{C}'$, ω - parameters of feature generator

train surrogate model f_{θ^*} by descent on $\nabla_\theta \mathcal{L}_{\text{train}}(\mathcal{C})$;
Randomly initialize feature generator w;
while not converged yet **do**
  for all $G \in \mathcal{C}[\backslash y_t]$ **do**
    generate trigger nodes g_t and randomly inject to G;
    update ω by descent on $\nabla_\omega(\mathcal{L}_{atk} + \alpha\mathcal{L}_s)$ based on Eq. (9);
  end for
end while
$G_B \leftarrow$ randomly sample candidate graphs from $\mathcal{C}[\backslash y_t]$;
for $G \in G_B$ **do**
  select M attached nodes based on important score S calculated by Eq. (3);
  generate features of trigger nodes by Eq. (5);
  update G with $m(G; g_t)$;
  Change label of G to y_t;
end for
// The trigger-embedded graphs G_B are put back to form backdoored dataset $\mathcal{C}'$
return $\mathcal{C}', \omega$;

$\mathcal{C}[\backslash y_t]$ are used for training feature generator. $\mathcal{L}_{atk}$ enforces the trigger-embedded graphs have similar embeddings with those in $\mathcal{C}[y_t]$ through surrogate model. Meanwhile, $\mathcal{L}_s$ ensures the similarity constraint. Therefore, our attack goals can be formulated as the following optimization problem:

$$\min_{w} \sum_{G \in \mathcal{C}[\backslash y_t]} \mathcal{L}_{atk}(f_{\theta^*}(G_{g_t}(w))) + \alpha \mathcal{L}_s(f_{\theta^*}(G_{g_t}(w)) \quad \text{s.t.} \quad \theta^* = \arg\min_{\theta} \mathcal{L}_{\text{train}}(f_\theta(\mathcal{C})), \tag{9}$$

where α is a trade-off hyper-parameter to balance two terms of the loss function, $\mathcal{L}_{train}$ is the loss function for training a rumor detector (the surrogate model in our attack). Algorithm 1 sketches our attack flow.

Experiments

In this section, we conduct empirical studies to answer the following research questions:
RQ1: Is IBAttack effective/evasive on RPTs?
RQ2: Is IBAttack effective and transferable across different rumor detectors?
RQ3: What is the impact of node important measures on IBAttack?
RQ4: What is the impact of trigger size on IBAttack?
RQ5: What is the impact of poisoning rate on IBAttack?

Experimental Settings

Datasets. We conduct experiments on three real-world rumor datasets: Twitter16 (Ma, Gao, and Wong 2017), Twitter15 (Ma, Gao, and Wong 2017) and Pheme (Zubiaga, Liakata, and Procter 2017). Table 1 presents detailed statistics. We choose Non-rumor or False Rumor as the target class since attackers generally want rumors to bypass detection.

Surrogate Models and Rumor Detectors. GCN (Kipf and Welling 2017), GAT (Velickovic et al. 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017) are three classical models of GNN that are often used as graph classification tasks, we test the effectiveness of attacks on them and also use them as surrogate models to attack specialized rumor detection models. BiGCN (Bian et al. 2020) is a GCN-based model that captures the global structure of the rumor tree using rumor propagation and dispersion. EBGCN (Wei et al. 2021) adaptively rethinks the reliability of latent relations by adopting a Bayesian approach to capture robust structure features. Their performance on the clean dataset is shown in Table 2.

Matrics. We use two metrics to evaluate the effectiveness and evasiveness of the attacks. The first metric is Attack Success Rate (ASR), which measures the success rate of a backdoor detector in predicting the trigger embedding graph as a specified category:

$$\text{Attack Success Rate (ASR)} = \frac{\textit{success trials}}{\textit{total trials}}. \tag{10}$$

The second metric is Clean Accuracy Drop (CAD), which measures the classification accuracy difference between a clean detector and a trojan detector in prediction on a clean testing dataset.

Baselines. Since there is no attempt to conduct the attack on propagation-based detectors, we choose five existing backdoor attacks on graph classification to testify the performance of IBAttack. Besides, IBAttack is also compared with its heuristic variants IBAttack/F and IBAttack/S as baselines. **ER-B** (Zhang et al. 2021) generates the universal trigger by the Erdős-Rényi (ER) model. **MIA** (Xu, Xue, and Picek 2021) chooses the most important nodes from GNNExplainer and replaces their connections with those of the trigger in the graph. **GTA** (Xi et al. 2021) utilizes a bi-layer optimization algorithm to update the trigger generator, which generates adaptive subgraphs as triggers for graphs. **TRAP** (Yang et al. 2022) generates perturbation triggers for graphs by performing perturbation actions on the graph structure based on a gradient-based score matrix. **Motif** (Zheng et al. 2024) uses the distribution difference of the motifs in the dataset to search the trigger structure. **IBAttack/F** removes the features generation module but randomly selects features from other nodes in the graph as injected trigger nodes features. **IBAttack/S** removes the node selection module to randomly select trigger injection locations.

Implementation Details. The dataset-splitting rules are applied to all methods. Following the settings of BiGCN, we split each dataset into two parts: 80% is used as a training dataset, and 20% is used as a testing dataset. From the non-target classes in the training set, we randomly select 5% to be backdoored. The trigger size is set as 10% and 20% of the number of graph nodes during the training and testing

Datasets	# Graphs	Avg. Max. Min. # of Posts	# Classes	# Graphs in Class	Target Class
Twitter16	818	251, 2765, 81	4	205 [N], 205 [F], 203 [T], 205 [U]	N
Twitter15	1490	223, 1768, 55	4	374 [N], 370 [F], 374 [T], 372 [U]	F
Pheme	6425	31, 345, 1	2	4023 [N], 2402 [R]	N

Table 1: Dataset Statistics

Datasets	GCN	GAT	GSAGE	BiGCN	EBGCN
Twitter16	0.85	0.81	0.87	0.86	0.88
Twitter15	0.81	0.79	0.80	0.81	0.83
Pheme	0.83	0.84	0.84	0.83	0.85

Table 2: Accuracy of models on clean datasets

Baselines	Twitter16	Twitter15	Pheme
ER-B	0.99 | 0.39	0.54 | 1.96	2.09 | 1.27
MIA	7.90 | 6.13	4.32 | 1.53	2.17 | 0.68
GTA	96.30 | 0.77	95.88 | 7.09	96.28 | 5.66
TRAP	0.01 | 1.54	0.31 | 1.41	1.04 | 0.20
Motif	11.57 | 10.0	17.5 | 10.63	68.80 | 5.65
IBAttack/F	0.14 | 0.51	1.62 | **0.39**	0.50 | 0.11
IBAttack/S	98.02 | 0.77	95.38 | 1.57	98.90 | 0.09
IBAttack	**100** | **0.25**	**99.89** | 1.37	**99.33** | **-0.29**

Table 3: Attack results (ASR (%) | CAD (%)) comparing to baselines

phases, respectively. The training epoch and learning rate are set to 100 and 0.001. All baselines use the optimal parameters from the original papers. More details can be found in the appendix.

Experimental Analysis

We present main experimental results. Experiments on the effects of hyperparameters are in the appendix.

Effectiveness and Evasiveness on RPTs. To answer RQ1, we compare IBAttack with baselines on three real-world datasets in terms of attack effectiveness and evasiveness. To be fair, we use GCN as the attacked model. The results are shown in Table 3. We observe that:

- All baseline methods for backdoor attacks, except for GTA, fail on the RPTs even when the tree restriction is removed, which validates our motivation. These methods manipulate the structure as triggers, however, since the nodes of the propagation graph are comment information about the source post, connecting two nodes that are not directly related does not greatly affect the final graph representation. GTA achieves a relatively high performance due to the modification of the trigger's features, but still, its triggers in the form of subgraphs are not only extremely easy to detect but also limit its aggressiveness. In contrast, our approach achieves the best results due to considering both the structure and feature characteristics of the propagation tree.
- Our approach achieves better results compared to IBAttack/F and IBAttack/S, which demonstrates the effectiveness of the attached nodes selection module and features generation module in our framework. When we randomly select node features, the attack becomes extremely ineffective, similar to other baseline methods. This illustrates that the injected properties overwhelmingly influence the effectiveness of the triggers since the propagation tree limits the perturbations of the structure. In addition, choosing the position of the triggers also improves the model's performance to some extent.

Datasets	Metrics	GCN	GAT	GSAGE	BiGCN	EBGCN
Twitter16	ASR (%)	100	68.34	96.97	100	99.63
	CAD (%)	0.25	1.84	0.67	0.77	0.26
Twitter15	ASR (%)	99.89	86.14	95.60	100	99.98
	CAD (%)	1.37	3.54	-0.78	-1.18	0.79
Pheme	ASR (%)	99.33	99.50	100	99.32	99.50
	CAD (%)	-0.29	1.36	0.10	0.11	3.22

Table 4: Attack results on different rumor detection models

Performance across Different Rumor Detectors. To answer RQ2, we test our method on different rumor detection models. The results are shown in Table 4. Our method achieves an extremely high ASR and low CAD on all models. We reach almost 100% on Pheme, likely because it is relatively easy to attack as a binary classification task. We notice that the performance of attacking GAT on Twitter15 and Twitter16 is slightly worse. This occurs because our attack always adds edge nodes to the graph, which tend to be assigned small weights by GAT's attention mechanism, making it less effective compared to the other models. It is worth mentioning that our method also achieves excellent results on the robust model EBGCN. This is due to our similarity constraint, which reduces the suspiciousness of the malicious trigger nodes, thereby increasing the confidence of the edges. This demonstrates the robustness of our attack.

Additionally, we attack rumor detectors of different architectures using simple surrogate models. As shown in Table 5, our attack method performs well in a black-box setup and exhibits transferability across different architectures. Using a different model as a surrogate does not significantly reduce

Model	Surrogate	Twitter16	Twitter15	Pheme
BiGCN	GCN	96.04 | 1.53	99.91 | 1.57	99.83 | 0.29
	GAT	96.11 | 1.54	99.46 | 1.57	99.66 | 0.10
	GSAGE	96.23 | 0.62	96.20 | 1.36	99.83 | 0.10
EBGCN	GCN	87.13 | 0.66	99.90 | 1.19	99.66 | 3.22
	GAT	94.06 | 0.77	99.80 | 0.79	99.32 | 3.07
	GSAGE	82.28 | 0.11	99.80 | 0.40	98.82 | 3.02

Table 5: Attack results (ASR (%) | CAD (%)) using surrogate models with different architectures

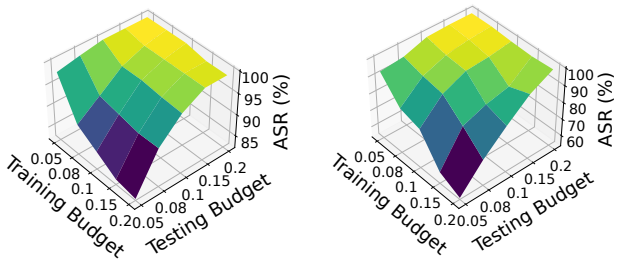
Measures	Twitter16	Twitter15	Pheme
Similarity	99.01 | 0.77	100 | −1.18	99.30 | 2.85
Degree	100 | 1.53	100 | 0.22	99.16 | −1.95
Closeness	98.02 | 2.35	99.15 | 0.11	98.57 | 0.37
PageRank	98.02 | 2.10	99.30 | 0.78	98.01 | −1.30

Table 6: Attack results (ASR (%) | CAD (%)) of different node importance measures

ASR, making our attack applicable to all propagation-based rumor detectors. Although we perform slightly worse when attacking GAT, using GAT as a surrogate model does not compromise the effectiveness of our attack.

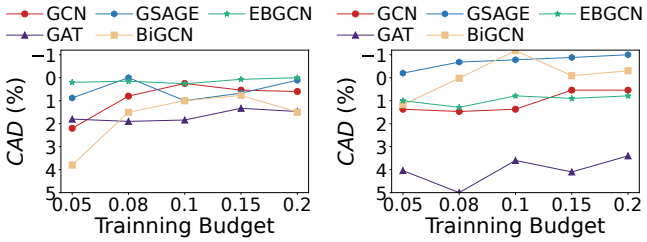
Impacts of the Node Important Measures. To answer RQ3, we conduct experiments in Table 6 to explore the impact of different node importance measures. We report the ASR and CAD when attacking each model using other node important measures on Twitter16. Closeness centrality (Sabidussi 1966) is calculated as the reciprocal of the sum of the length of the shortest paths between the node and all other nodes in the graph. PageRank centrality (Brin and Page 1998), widely used in web page ranking, is based on the principle that a page's importance on the Internet depends on the quantity and quality of its inbound links. Among all measures, similarity to the root node and degree centrality achieve better results across datasets. Thus, we recommend these as indicators for the importance score in IBAttack.

Impacts of the Trigger Size. To answer RQ4, we conduct experiments to explore the attack performance of IBAttack given different budgets in the size of trigger nodes (i.e., trigger size) in the training and testing phase, respectively. In Figure 4, we report the ASR of attacking BiGCN on two datasets. The ASR gradually increases as the ratio of testing trigger size increases, while the trend for the ratio of training trigger size is the opposite. The trend holds for all datasets. Regarding CAD, we only report changes with respect to the ratio of training trigger size as it is only influenced during training phase. As shown in Figure 5, although no clear patterns emerge, it is generally observed that larger values correspond to a decrease in CAD. The overall CAD is less than 2% except for GAT when the trigger size ratio exceeds 0.1. Thus, we choose 0.1 during the training phase and 0.2 during the testing phase to balance effectiveness and evasiveness.



(a) Twitter16 (b) Twitter15

Figure 4: The impact of trigger size on attack effectiveness



(a) Twitter16 (b) Twitter15

Figure 5: The impact of trigger size on attack evasiveness

Impacts of the Poisoning Rate. To answer RQ5, we conduct experiments to explore the effects of the poisoning rate, with results presented in Figure 6. As the poisoning rate increases, the ASR also rises. However, the increment in ASR diminishes with higher poisoning rates, suggesting that our model does not rely on a higher poisoning rate to be successful. Regarding CAD, it shows an overall increase as the poisoning rate rises, but remains below 2% in most cases.

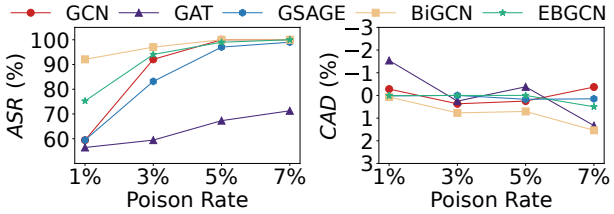


Figure 6: The impact of the poisoning rate on Twitter16

Conclusion and Future Work

In this paper, we propose an injection-based backdoor attack framework for propagation-based rumor detection. Specifically, we employ a nodes selection module to choose critical nodes as attached nodes to fully utilize the attack budget. By employing attack loss and similarity loss, we adaptively generate the features of trigger nodes to ensure both effectiveness and evasiveness. In the future, there are two directions that need further investigation. First, we will explore generating coherent and reasonable semantic information for trigger nodes while maintaining the effectiveness. Second, we will investigate how to defend against IBAttack.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (U22B2036, 92370111, 62422210, 62302333, 62272340, 62276187), the National Science Fund for Distinguished Young Scholarship (62025602), the Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ) (GML-KF-24-16), Hebei Natural Science Foundation (F2024202047) and the XPLORER PRIZE.

References

Bian, T.; Xiao, X.; Xu, T.; Zhao, P.; Huang, W.; Rong, Y.; and Huang, J. 2020. Rumor Detection on Social Media with Bi-directional Graph Convolutional Networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 549–556.

Brin, S.; and Page, L. 1998. The Anatomy of A Large-scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30(1-7): 107–117.

Cui, C.; and Jia, C. 2024. Propagation Tree Is Not Deep: Adaptive Graph Contrastive Learning Approach for Rumor Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 73–81.

Dai, E.; Lin, M.; Zhang, X.; and Wang, S. 2023. Unnoticeable Backdoor Attacks on Graph Neural Networks. In *Proceedings of the ACM Web Conference 2023*, 2263–2273.

Dong, Y.; He, D.; Wang, X.; Jin, Y.; Ge, M.; Yang, C.; and Jin, D. 2024. Unveiling Implicit Deceptive Patterns in Multi-Modal Fake News via Neuro-Symbolic Reasoning. In *AAAI Conference on Artificial Intelligence*.

Dong, Y.; He, D.; Wang, X.; Li, Y.; Su, X.; and Jin, D. 2023. A Generalized Deep Markov Random Fields Framework for Fake News Detection. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23. ISBN 978-1-956792-03-4.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035.

He, B.; Ahamad, M.; and Kumar, S. 2021. PETGEN: Personalized Text Generation Attack on Deep Sequence Embedding-based Classification Models. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 575–584.

Hua, J.; Cui, X.; Li, X.; Tang, K.; and Zhu, P. 2023. Multimodal fake news detection through data augmentation-based contrastive learning. *Appl. Soft Comput.*, 136(C).

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*.

Le, T.; Wang, S.; and Lee, D. 2020. MALCOM: Generating Malicious Comments to Attack Neural Fake News Detection Models. In *2020 IEEE International Conference on Data Mining (ICDM)*, 282–291.

Linton; C.; and Freeman. 1978. Centrality in Social Networks Conceptual Clarification. *Social Networks*.

Lu, Y.-J.; and te Li, C. 2020. GCAN: Graph-aware Co-Attention Networks for Explainable Fake News Detection on Social Media. In *Annual Meeting of the Association for Computational Linguistics*.

Lyu, Y.; Yang, X.; Liu, J.; Xie, S.; Yu, P.; and Zhang, X. 2023. Interpretable and Effective Reinforcement Learning for Attacking against Graph-based Rumor Detection. In *2023 International Joint Conference on Neural Networks (IJCNN)*, 1–9.

Ma, J.; Gao, W.; Mitra, P.; Kwon, S.; Jansen, B. J.; Wong, K.-F.; and Cha, M. 2016. Detecting Rumors from Microblogs with Recurrent Neural Networks. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 3818–3824.

Ma, J.; Gao, W.; and Wong, K.-F. 2017. Detect Rumors in Microblog Posts Using Propagation Structure via Kernel Learning. In *Annual Meeting of the Association for Computational Linguistics*.

Ma, J.; Gao, W.; and Wong, K.-F. 2018. Rumor Detection on Twitter with Tree-structured Recursive Neural Networks. In *Annual Meeting of the Association for Computational Linguistics*.

Rosenfeld, N.; Szanto, A.; and Parkes, D. C. 2020. A Kernel of Truth: Determining Rumor Veracity on Twitter by Diffusion Pattern Alone. In *Proceedings of The Web Conference 2020*, 1018–1028.

Sabidussi, G. 1966. The Centrality Index of A Graph. *Psychometrika*, 31(4): 581–603.

Sun, T.; Qian, Z.; Dong, S.; Li, P.; and Zhu, Q. 2022. Rumor Detection on Social Media with Graph Adversarial Contrastive Learning. In *Proceedings of the ACM Web Conference 2022*, 2789–2797.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y.; et al. 2017. Graph Attention Networks. *stat*, 1050(20): 10–48550.

Wang, H.; Dou, Y.; Chen, C.; Sun, L.; Yu, P. S.; and Shu, K. 2023. Attacking Fake News Detectors via Manipulating News Social Engagement. In *Proceedings of the ACM Web Conference 2023*, 3978–3986.

Wei, L.; Hu, D.; Zhou, W.; Yue, Z.; and Hu, S. 2021. Towards Propagation Uncertainty: Edge-enhanced Bayesian Graph Convolutional Networks for Rumor Detection. In *Annual Meeting of the Association for Computational Linguistics*, 3845–3854.

Xi, Z.; Pang, R.; Ji, S.; and Wang, T. 2021. Graph Backdoor. In *30th USENIX Security Symposium (USENIX Security 21)*, 1523–1540.

Xu, J.; Xue, M. J.; and Picek, S. 2021. Explainability-based Backdoor Attacks Against Graph Neural Networks. In *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*, 31–36.

Yang, S.; Doan, B. G.; Montague, P.; De Vel, O.; Abraham, T.; Camtepe, S.; Ranasinghe, D. C.; and Kanhere, S. S. 2022. Transferable Graph Backdoor Attack. In *Proceedings of the*

25th International Symposium on Research in Attacks, Intrusions and Defenses, 321–332.

Yu, F.; Liu, Q.; Wu, S.; Wang, L.; and Tan, T. 2017. A Convolutional Approach for Misinformation Identification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 3901–3907.

Zhang, Z.; Jia, J.; Wang, B.; and Gong, N. Z. 2021. Backdoor Attacks to Graph Neural Networks. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, 15–26.

Zheng, H.; Xiong, H.; Chen, J.; Ma, H.; and Huang, G. 2024. Motif-Backdoor: Rethinking the Backdoor Attack on Graph Neural Networks via Motifs. *IEEE Transactions on Computational Social Systems*, 11(2): 2479–2493.

Zhou, Z.; Guan, H.; Bhat, M. M.; and Hsu, J. 2019. Fake News Detection via NLP is Vulnerable to Adversarial Attacks. In *International Conference on Agents and Artificial Intelligence*.

Zhu, P.; Pan, Z.; Liu, Y.; Tian, J.; Tang, K.; and Wang, Z. 2024. A General Black-box Adversarial Attack on Graph-based Fake News Detectors. In Larson, K., ed., *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, 568–576. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Zubiaga, A.; Liakata, M.; and Procter, R. 2017. Exploiting Context for Rumour Detection in Social Media. In *Social Informatics*.

Zügner, D.; Akbarnejad, A.; and Günnemann, S. 2018. Adversarial Attacks on Neural Networks for Graph Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2847–2856.