Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.075072

ARTICLE

A Low-Cost Network Topology Obfuscation Method for Critical Node Protection

Yanming Chen[1], **Fuxiang Yuan**[2,*] and **Zekang Wang**[2]

[1]School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou, China
[2]Key Laboratory of Cyberspace Situation Awareness of Henan Province, Zhengzhou, China

*Corresponding Author: Fuxiang Yuan. Email: rookieyfx@163.com

Received: 24 October 2025; Accepted: 18 December 2025; Published: 12 March 2026

ABSTRACT: Network topology obfuscation is a technique aimed at protecting critical nodes and links from disruptions such as Link Flooding Attack (LFA). Currently, there are limited topology obfuscation methods for protecting critical nodes, and the existing approaches mainly achieve obfuscation by extensively modifying network links, resulting in high costs. To address this issue, this paper proposes a low-cost network topology obfuscation method dedicated to critical node protection, with its core innovation lying in a lightweight obfuscation architecture based on Fake Node Clusters (FNCs). Firstly, the protected network is modeled as an undirected graph, and an adjacency matrix is constructed to quantify the network scale and structural characteristics. Then, a fake node cluster generation algorithm is designed to construct an FNC adapted to the target network. Finally, a heuristic obfuscated topology generation algorithm is proposed. By optimizing the deployment positions of Fake Nodes Clusters (FNCs) in the protected network, this algorithm effectively reduces the number of FNCs required to generate the obfuscated topology, further lowering the obfuscation cost. Extensive experiments were conducted on the public Topology Zoo dataset, categorizing network topologies by node count into small-scale ([0, 50)), medium-scale ([50, 100)), and large-scale ([100, 200)) groups. The experimental results demonstrate that the proposed approach achieves excellent obfuscation performance, reducing the critical node recognition rate to 0%. Compared to the typical method, EigenObfu, the proposed approach also reduces obfuscation costs by an average of 97.9%, 99.6%, and 99.3% for small, medium, and large-scale networks, respectively.

KEYWORDS: Topology obfuscation; critical node protection; topology defense; network topology; cyberspace anti-mapping

1 Introduction

The Internet is a complex network system composed of massive nodes interconnected via communication links [1]. As the core carrier for network operations, nodes undertake key tasks such as data reception, processing, storage, and forwarding. According to the differences in the importance of nodes within the network topology, they can be categorized into critical nodes and normal nodes. Among them, the failure or abnormality of critical nodes is highly likely to trigger network cascading failure [2], which in turn leads to the interruption of data transmission links and the deterioration of service quality and may even cause complete network paralysis in severe cases [3–5].

Prior to launching an attack, attackers tend to conduct network reconnaissance. By accurately identifying critical nodes and implementing targeted attacks, they aim to maximize the destructive effect, with typical attack methods including Link Flooding Attack (LFA) [6,7]. Specifically, attackers typically utilize topology probing tools such as Traceroute to scan the target network [8,9], locate critical nodes based on the topology information obtained through probing, and ultimately render them in a state of service unavailability by injecting attack traffic [10,11]. The cascading effects triggered by the failure of such nodes will further expand the scope of the attack's impact, leading to large-scale network communication anomalies. Currently, systematic reconnaissance prior to attacks has become a core means for attackers to enhance their destructive effectiveness.

Network topology obfuscation is currently the main approach to defending against attacks such as LFA [12–15] and protecting critical nodes. As an active defense mechanism [16,17], its fundamental difference from traditional passive defense lies in the fact that this strategy determines critical nodes through node importance evaluation indicators before an attacker launches an attack [18,19], and then actively camouflages [20] the protected network according to the obfuscation strategy, preventing attackers from obtaining the real network topology. Studies have shown that the attack effect of LFA highly depends on the accuracy of the obtained topology information. If the topological information is distorted, attackers need to increase the traffic by at least five times to achieve the effect of blocking network communication [21]. In addition, since the attackers obtain a confused topology through network reconnaissance, they will identify the wrong critical nodes, and the attacks they launch against critical nodes will be redirected to normal nodes, thus effectively protecting the critical nodes [22].

Currently, research on topology obfuscation technology for critical node protection is relatively scarce. Traditional network topology obfuscation methods mainly use local indicators such as degree centrality [23] and betweenness centrality [20] in the stage of evaluating critical nodes. These indicators have played an important role in early network topology research. However, with the continuous evolution of network attack methods and the increasing complexity of network structures, the limitations of traditional indicators have gradually emerged. When determining critical nodes, the above indicators focus on the attributes of the nodes themselves and do not take into account the global characteristics of the network topology enough. However, attackers seem to pay more attention to nodes with high global importance during network attacks because their failures cause more serious damage to the entire network. This leads to the fact that the network topology defense systems built based on traditional indicators are difficult to achieve ideal defense effects when resisting attacks against globally important critical nodes. In addition, the deployment of confused topologies increases the cost for defenders. How to ensure the lowest operation complexity during the conversion from the original topology to the confused topology is the key to improving the practicality of defense.

Zhu et al. were the first to explore the value and potential of eigenvector centrality in evaluating the global importance of nodes. Eigenvector centrality [24] not only considers the importance of the node itself but also incorporates the importance weight of its adjacent nodes [25]. Based on a random link modification strategy, they designed a network topology obfuscation method called EigenObfu, which can defend against attack methods that determine critical nodes based on eigenvector centrality and is currently the only topology obfuscation method targeting attacks based on eigenvector centrality. However, this method has the problem of modifying a large number of original topology links when generating the confused topology, resulting in a high network adjustment cost.

To address the above problems, this paper proposes a low-cost network topology obfuscation method for critical node protection. It generates Fake Node Clusters (FNCs) through a fake node cluster generation algorithm and optimizes the insertion position of the FNCs using a confused topology generation algorithm to construct a low-cost confused topology. The main contributions of this paper are as follows:

- **A low-cost network topology obfuscation framework based on the Fake Node Cluster (FNC) is proposed.** The FNC is designed based on Software-Defined Networking (SDN) technology [26], and the confused topology is generated by optimizing the deployment strategy of the FNC. This framework can effectively resist LFA attacks that use ICMP Traceroute, TCP Traceroute, and UDP Traceroute for network reconnaissance and supports the obfuscation of critical nodes in any proportion.
- **A low-cost fake node cluster generation algorithm is proposed.** The FNC is designed as a chordal graph structure with multiple intermediate nodes in parallel, which can interfere with attackers' identification of critical nodes at a low cost. Meanwhile, the use of fake nodes instead of traditional physical nodes in obfuscation clearly reduces the obfuscation cost.
- **A heuristic obfuscated topology generation algorithm is proposed.** It adopts a greedy strategy to select positions with the maximum hidden gain for critical nodes as FNC insertion positions to generate obfuscated topologies, effectively reducing the required number of FNCs and further lowering the obfuscation cost.

The rest of this paper is organized as follows: Section 2 analyzes the typical method EigenObfu [27]. Section 3 details the principle framework and main stages of the proposed method and provides an in-depth explanation of the core algorithms. Section 4 verifies the effectiveness of the proposed method, analyzes the algorithm complexity, and compares it with the existing typical method, EigenObfu, in terms of obfuscation cost. Section 5 summarizes the proposed method and looks forward to the next steps.

2 Related Work

Zhu et al. proposed a network topology obfuscation method called EigenObfu for critical node protection. This method employs eigenvector centrality as the evaluation metric for critical nodes. Its advantage lies in the fact that eigenvector centrality can take into account the importance of both the node itself and its neighboring nodes, enabling a more comprehensive portrayal of the global importance of nodes [28]. Based on this, EigenObfu generates the obfuscated topology through a candidate topology generation stage and an obfuscated topology selection stage. Its input is the adjacency matrix A of the protected network topology, and the output is the obfuscated topology. The main process of EigenObfu is shown in Fig. 1).

- **Candidate topology generation.** First, the eigenvector centrality (EC) of each node is calculated, and the set of critical nodes K is determined accordingly. Subsequently, the network topology is randomly modified with a fixed probability p. If the nodes associated with the link to be modified are all critical nodes, it is made connected with a probability p and disconnected with a probability of $1 - p$. If both nodes involved are normal nodes, the opposite probabilities are applied. Finally, the topologies that meet the security constraint (the intersection of critical nodes before and after obfuscation is empty) and the connectivity constraint (the second-smallest eigenvalue of the Laplacian matrix is greater than 0) are selected as candidate obfuscated network topologies to ensure the effectiveness of obfuscation and the unchanged connectivity of the obfuscated topology.
- **Topology selection.** First, the similarity of each obfuscated topology is measured based on the adjacency matrix and Laplacian matrix. Then, the one with the maximum similarity is selected as the optimal obfuscated topology and output as the final obfuscated topology.

Although the EigenObfu method uses eigenvector centrality as the critical node evaluation standard, which can more comprehensively reflect the importance of critical nodes and can effectively obfuscate critical nodes, it has certain limitations in the candidate topology generation algorithm. This method disconnects and generates links with a fixed probability, ignoring the differences among different types of topologies. Meanwhile, the randomized link adjustment mechanism results in a high degree of uncertainty in the obfuscation of the generated candidate topologies. It is difficult to precisely guide the topology obfuscation towards the optimal obfuscation direction only through the security constraint and the connectivity constraint, resulting in a lack of targeted modifications to the original topology, excessive modifications to the original topology, and a high cost of network structure adjustment. The team has also recognized this problem. Therefore, in the obfuscated topology selection stage, the similarity is used to screen and reduce the modification range of the topology. However, the obfuscation cost remains at a relatively high level.

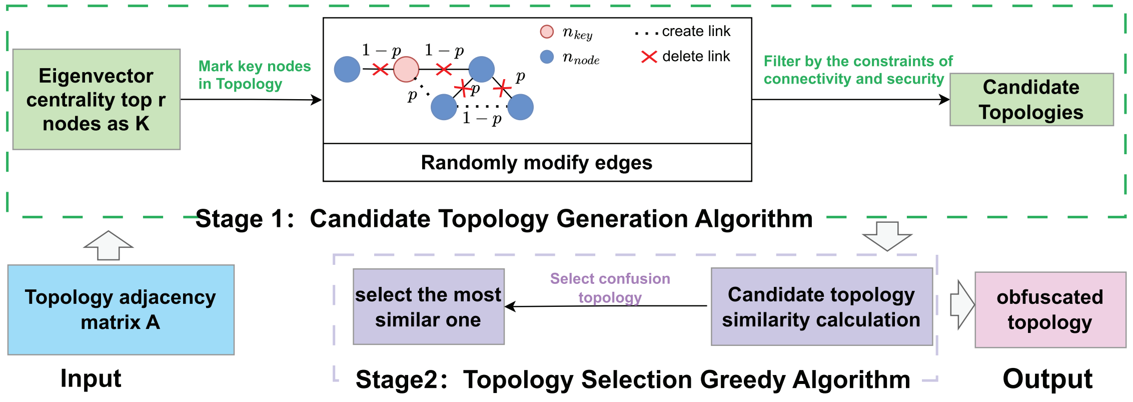

Eigenvector centrality top r nodes as K
Mark key nodes in Topology
$1-p$
$1-p$
p
p
p
$1-p$
n_{key}
n_{node}
create link
delete link
Randomly modify edges
Filter by the constraints of connectivity and security
Candidate Topologies
Stage 1: Candidate Topology Generation Algorithm
Topology adjacency matrix A
Input
select the most similar one
Select confusion topology
Candidate topology similarity calculation
obfuscated topology
Stage2: Topology Selection Greedy Algorithm
Output

Figure 1: Key operations of EigenObfu.

3 Prposed Method

To address the issue of excessive modifications to the original network and high network adjustment costs during the obfuscation process in existing network topology obfuscation methods, this paper proposes a low-cost network topology obfuscation method for critical node protection.

3.1 Framework

Fig. 2 illustrates the proposed method's framework. First, the protected network topology is modeled as an undirected graph G, and its adjacency matrix A is constructed to compute network size $|V|$. Then, by analyzing the impact of local topological structure changes in the network on the ranking of node importance, the FNC structure with multiple parallel intermediary nodes is designed, and based on this, a dynamic fake node cluster generation algorithm is proposed to generate the FNC that adapted to the protected network scale $|V|$. Finally, to protect any proportion of critical nodes while minimizing FNC deployments, a heuristic obfuscated topology generation algorithm is proposed. This algorithm determines the set of protected critical nodes K based on eigenvector centrality, selects and determines an effective sequence of FNC insertion positions $E_{selected}$ by evaluating the critical node hiding gain $\triangle H$, and generates the obfuscated topology G_{obfu}.

The proposed method mainly includes the following three stages. For the convenience of description, the mathematical symbols used in this paper and their meanings are shown in Table 1.

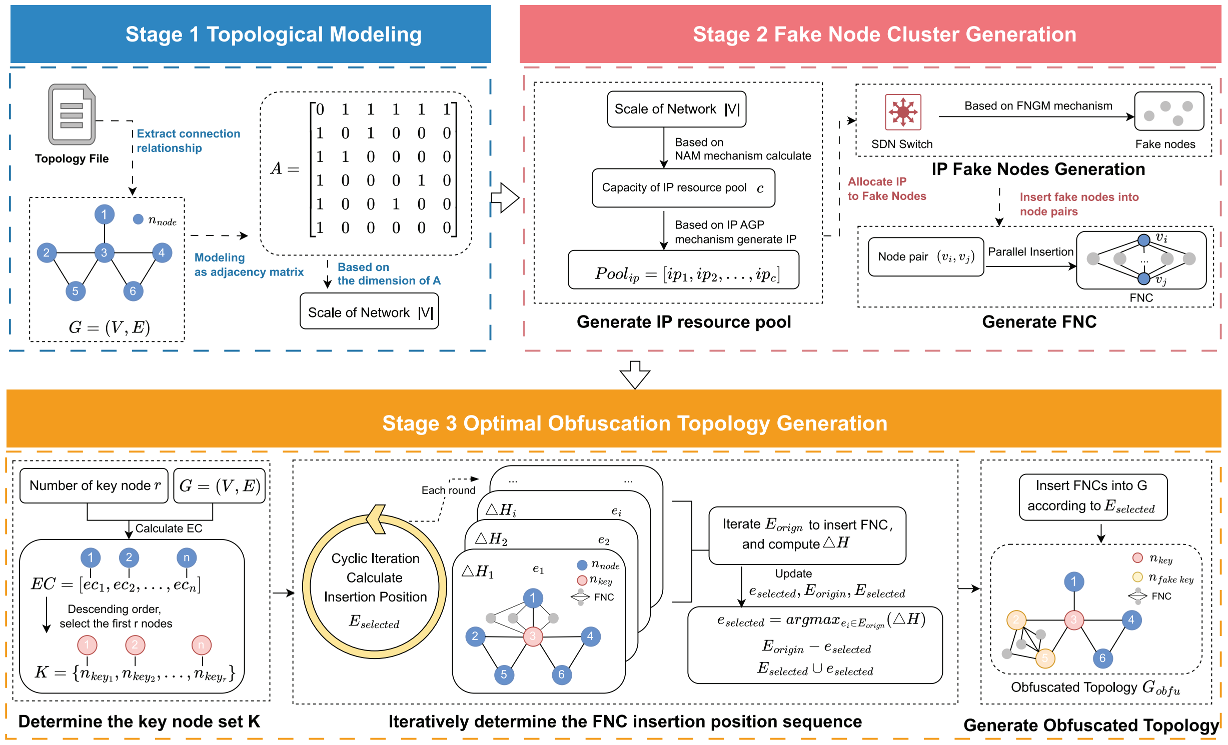

Stage 1 Topological Modeling
Topology File
Extract connection relationship
Modeling as adjacency matrix
Based on the dimension of A
Scale of Network |V|
G = (V, E)
Stage 2 Fake Node Cluster Generation
Scale of Network |V|
Based on NAM mechanism calculate
Capacity of IP resource pool c
Based on IP AGP mechanism generate IP
Generate IP resource pool
SDN Switch
Based on FNGM mechanism
Fake nodes
IP Fake Nodes Generation
Allocate IP to Fake Nodes
Insert fake nodes into node pairs
Node pair
Parallel Insertion
FNC
Generate FNC
Stage 3 Optimal Obfuscation Topology Generation
Number of key node r
Calculate EC
Descending order, select the first r nodes
Determine the key node set K
Each round
Cyclic Iteration Calculate Insertion Position
Iterate E_origin to insert FNC, and compute △H
Update
Iteratively determine the FNC insertion position sequence
Insert FNCs into G according to E_selected
Obfuscated Topology G_obfu
Generate Obfuscated Topology

Figure 2: Method framework.

Table 1: Mathematical symbols and descriptions.

Notation	Description
$G = (V, E)$	Original network topology
$V = \{n_1, n_2, \ldots, n_n\}$	Set of nodes
$E = \{e_1, e_2, \ldots, e_m\}$	Set of edges
$\boldsymbol{A}$	Adjacency matrix of G
r	Number of key nodes
K	Set of key nodes
b	Number of fake nodes in the FNC
$Pool_{ip} = \{ip_1, ip_2, \ldots, ip_n\}$	IP address pool
$\mathrm{G}_{\mathrm{obfu}}$	Obfuscated network topology
$\triangle H$	Hidden effect gain for key nodes
$\mathrm{GED}(g_1, g_2)$	Graph edit distance between g_1 and g_2
S	Safety constraint: key node overlap count (before & after obfuscation)
S_{temp}	Security constraint temporary value
G^*_{obfu}	Temporary obfuscation topology

Stage 1: Topological Modeling. Read the file recording the topology of the protected network N, extract the nodes and connection relationships, and uniformly number them (0 to $n-1$). Based on graph theory, model it as an undirected graph $G = (V, E)$, where V represents the set of all nodes and E represents the set

of edges. Construct the adjacency matrix $A \in \{0,1\}^{n\times n}$, where n is the matrix dimension. A_{ij} equals 1 if and only if nodes i and j $(0 \le i,\ j < n)$ are connected; otherwise, it is 0. The size of N is determined by $|V|$.

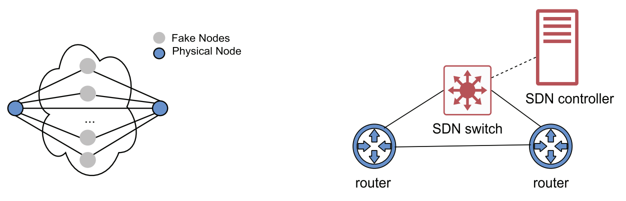
Stage 2: Fake Node Cluster Generation. The FNC is designed as a chordal graph structure with multiple parallel intermediary nodes. By constructing a large number of redundant paths with single parallel intermediary nodes between adjacent nodes, it can effectively disrupt key node identification methods based on EC. Based on this structure, this paper proposes a dynamic fake node cluster generation algorithm. Software-Defined Networking (SDN) switches are used to generate fake nodes instead of physical nodes to reduce hardware resource consumption.

First, according to the scale $|V|$ of the protected network N, a Network Adaptation Model (NAM) is designed to determine the capacity c of the IP address resource pool $Pool_{ip}$, an IP Address Generation Policy (IP AGP) is formulated to build the $Pool_{ip}$. Then, the IP addresses in the $Pool_{ip}$ are allocated to SDN switches, and a Fake Node Generation Mechanism (FNGM) is established to generate multiple fake nodes on the SDN switches. Finally, SDN switches are connected in parallel between adjacent node pairs, enabling the fake nodes and adjacent nodes to form an FNC with a chordal graph structure. For further details, refer to Section 3.2.

Stage 3: Obfuscated Topology Generation. To reduce the number of FNC insertions and achieve the protection of any proportion of critical nodes, a heuristic obfuscated topology generation algorithm is proposed. First, calculate each node's EC in undirected graph G—EC is a widely recognized node importance metric integrating direct connections and neighbor influence. Nodes are sorted descendingly by EC, and the top r nodes (predefined per network security needs) form the critical node set K. Next, the set of edges E in network N is regarded as the candidate set of FNC insertion positions. Through multiple rounds of greedy iteration, the hidden effect gain $\triangle H$ of each candidate edge after FNC insertion is dynamically evaluated. The position with the largest $\triangle H$ is prioritized for insertion. The iteration stops when the intersection of the key node set before and after obfuscation becomes empty or the maximum number of iterations is reached. This process generates the sequence of FNC insertion locations $E_{selected}$. Finally, the obfuscated topology G_{obfu} is generated by inserting FNCs into $E_{selected}$. For further details, refer to Section 3.3.

3.2 Fake Node Cluster Generation

To disrupt attackers' identification of critical nodes, this paper designs a Fake Node Cluster (FNC) with a chordal graph structure featuring multiple parallel intermediary nodes, as shown in Fig. 3. Its core mechanism lies in rapidly constructing multiple redundant paths with single parallel intermediary nodes between two adjacent nodes, which can effectively interfere with key node identification methods centered around EC. The implementation process of the FNC is presented in Algorithm 1, mainly including the following three steps.



(a) FNC Structure (b) FNC Physical Structure

Figure 3: The FNC structure and its physical structure.

Step 1: Generate an IP Address Resource Pool. According to the scale $|V|$ of the protected network N, the number b of fake nodes required for a single unit is dynamically determined, and an IP address resource pool $Pool_{ip}$ with a capacity of c is generated, where $b = c$. An efficient $Pool_{ip}$ is constructed by building a Network Adaptation Model (NAM) and an IP Address Generation Policy (IP AGP).

The NAM describes the relationship between the network scale $|V|$ and the capacity c of the IP address resource pool $Pool_{ip}$, as shown in Eq. (1). It aims to balance the obfuscation cost and the obfuscation effect.

$$c(|V|) = \begin{cases} 3 + \left\lceil \dfrac{|V|}{25} \right\rceil, & \text{if } |V| \le 50 \\ 5 + \left\lceil \dfrac{|V| - 50}{25} \right\rceil, & \text{if } 50 < |V| \le 100 \\ 8 + \left\lceil \dfrac{|V| - 100}{25} \right\rceil, & \text{if } |V| > 100 \end{cases} \tag{1}$$

The reason for limiting the capacity c of $Pool_{ip}$ is that when the network scale $|V|$ is fixed, there is a relationship between the capacity c of $Pool_{ip}$ and the obfuscation cost C as shown in Eq. (2).

$$C(c) = \begin{cases} None, & c \le c_{\min} \\ f(c), & c_{\min} < c \le c^* \\ C_{\min}, & c > c^* \end{cases} \tag{2}$$

Here, c_{min} is the minimum successful obfuscation value, C_{min} represents the lower limit of obfuscation cost, and c^* is the critical saturation value of the $Pool_{ip}$. The function $f(c)$ shows an oscillatory decrease, meaning it generally declines but with fluctuations, and it satisfies:

- When $c \le c_{min}$, obfuscation fails.
- When $c \le c^*$, increasing capacity c reduces obfuscation cost C. However, if c is too large relative to $|V|$, the FNC structure becomes too obvious, increasing obfuscation policy exposure risk.
- When $c > c^*$, obfuscation cost C no longer varies with capacity c, but exposure risk increases.

This indicates that there is a marginal diminishing benefit characteristic between the capacity c of $Pool_{ip}$ and the obfuscation cost C. After exceeding the critical value $c > c^*$, expanding the capacity c of $Pool_{ip}$ cannot improve the obfuscation effect. However, even before reaching the critical value $c > c^*$, a larger capacity c is not always better. The capacity c should be adapted to the network scale $|V|$. If the capacity c is too large, the FNC structure in the obfuscated network will be too obvious, increasing the risk of the obfuscation strategy being exposed. If the capacity c is too small, it may not be possible to fully protect all critical nodes, or the obfuscation cost will be too high. Therefore, dynamically adjusting the capacity c according to the network scale $|V|$ under different network scales can ensure the obfuscation effect while reducing the obfuscation cost and the risk of the obfuscation strategy being exposed.

The IP AGP aims to minimize fake node IP address similarity and consists of the following two policies.

- **Segment Filtering Strategy.** Addresses are preferentially allocated from business network segments with a large host address space (such as /16) and listed in a whitelist, and fake IPs are generated by changing the host number.
- **Maximize IP Differentiation Strategy.** Each fake IP is assigned to a different /30 subnet (host number digits > 2) to reduce IP address similarity against router alias resolution based on IP assignment laws.

Step 2: IP Address Allocation and Fake Intermediary Node Generation. The SDN controller dynamically allocates the IP addresses in the $Pool_{ip}$ to the SDN switches to complete the IP address allocation. Since Traceroute detection infers the routing path and node IP through the TTL (Time to Live) field and the Time Exceeded message, this paper designs a Fake Node Generation Mechanism (FNGM). This mechanism consists of a traceroute detection flow identification policy based on flow tables and a packet forgery policy with dual-strategy dynamic switching. It can generate b fake nodes $(v_1, v_2, \ldots, v_b)$ on SDN switch s, where $b = c$. The specific process is as follows.

First, we design a flow table-based flow identification strategy, which can detect three types of active Traceroute detection traffic: TCP Traceroute, UDP Traceroute, and ICMP Traceroute. During the LFA reconnaissance phase, Traceroute is a common tool for obtaining network topology. When Traceroute detection packets are detected in the network, it can be determined as the preparatory behavior of an attacker before launching an LFA attack. The characteristics of the three types of traceroute detections are shown in Table 2.

Table 2: Three categories of traceroute features.

Traceroute Type	Features
TCP Traceroute	Protocol = 6, Flags = 2
UDP Traceroute	Protocol = 17, Dport = 33,434–33,534
ICMP Traceroute	Protocol = 1, Type = 8

For the above characteristics of Traceroute detection, the algorithm creates a flow table in the SDN controller. Flow table entries for identifying the three types of traceroute detections (TCP Flow Entry, UDP Flow Entry, and ICMP Flow Entry) are designed and configured based on the protocol number and key field values, as shown in Eqs. (3)–(5).

When designing the UDP Flow Entry, while traditional UDP Traceroute uses a destination port (dport) range of 33,434–33,534, attackers can bypass detection by altering this number. Therefore, all UDP messages are recognized when designing this flow table entry.

$$\mathrm{TCPFlowEntry}\ (\mathrm{proto} = 6,\ \mathrm{TCPFlags} = 2) \tag{3}$$

$$\mathrm{UDPFlowEntry}\ \ (\mathrm{proto} = 17) \tag{4}$$

$$\mathrm{ICMPFlowEntry}\,(\mathrm{proto} = 1,\ \mathrm{ICMPType} = 8) \tag{5}$$

The second component is a dual-strategy dynamically switched message forgery strategy, where the selection between the two strategies is determined by a random number, $random\_int$.

- **Direct Forwarding Strategy.** Traceroute probe messages forwarded to the controller are re-routed to switch s for normal forwarding without additional processing.
- **Fake Jamming Strategy.** This strategy employs differentiated processing based on the Time-to-Live (TTL) field value: For packets with a TTL value less than or equal to 1, the original source IP address is set as the destination IP address. Concurrently, an IP address is randomly selected from the IP address pool and assigned as the source IP address. A fake Time Exceeded packet (FP) is then generated using packet forgery techniques and returned to switch s_i for forwarding to the probe source. The original Traceroute probe packet is simultaneously discarded. For packets with a TTL value greater than 1, the original packet is parsed, its TTL value is decremented by 1, and it is re-encapsulated as a fake packet (FP). This packet is then returned to switch s for further forwarding.

Step 3: Fake Node Cluster Construction. To achieve persistent perturbation of the local topology, FNC must be permanently introduced into the protected network topology. For neighboring node pairs (v_i, v_j), FNC construction involves connecting an SDN switch s in parallel between them and simulating the simultaneous connection of multiple fake intermediary nodes with both v_i and v_j.

Algorithm 1: Fake Node Cluster Generation Algorithm

```
Input: ① Network size $|V|$
Output: Fake Node Cluster FNC
Initialization:$c \leftarrow \text{NAM}(|V|)$, $Pool_{ip} \leftarrow \text{IP APG}(IP_{\text{segments}})$;
Flow Table ← TCP, UDP, ICMP Flow Entry;
Establish connection between SDN switch and controller;
if $p_i$ matches Flow Table then
    $random\_int \leftarrow \text{random}(0, 1)$;
    if $random\_int == 1$ then
        packet_out($p_i, s$);
        // Direct forwarding strategy
    end
    else
        if $TTL \leq 1$ then
            reverse(ip_dest, ip_src);
            $ip\_src \leftarrow \text{random}(Pool_{ip})$;
            Generate fake Time Exceeded message FP;
        end
        else
            TTL ← TTL − 1;
            Construct fake message FP by original Traceroute class;
        end
        packet_out(FP, $s$);
        // Send false message
    end
end
Parallelly insert SDN switches between adjacent node pairs $(v_i, v_j)$ to generate FNC;
return FNC
```

3.3 *Obfuscated Topology Generation*

To generate effective obfuscated topologies and minimize FNCs, thus reducing obfuscation cost, a heuristic obfuscated topology generation algorithm is designed, consisting of three main steps.

Step 1: Determine the set of key nodes to be protected.

First, the **EC** value of each node in the undirected graph $G = (V, E)$ is calculated based on the network's adjacency matrix A, following the **EC** calculation Eq. (6):

$$A\mathbf{EC} = \lambda_{\max}\mathbf{EC} \tag{6}$$

where $\lambda_{\max}$ is the largest eigenvalue (dominant eigenvalue) of A, and **EC** is the corresponding eigenvector representing the **EC** scores of all nodes. Then, rank the nodes by their **EC** values in descending order and select the top r nodes (with the highest **EC** values) to form the set of key nodes K. This process is represented in Eq. (7).

$$K = \left\{v \in V \,\middle|\, \text{rank}\left(EC(v)\right) \le r\right\} \tag{7}$$

where rank() denotes the descending order ranking function and r is the specified number of key nodes.

Step 2: Iteratively determine the FNC insertion position sequence.

Firstly, remove e_i from the FNC candidate location set E_{origin} and insert an FNC of scale b to generate the temporary topology G_{temp}. Then, calculate the eigenvector centrality EC_{temp} for each node in G_{temp}, sort them in descending order, and select the top r nodes to form the new set of key nodes K_{temp}. According to Eq. (8), calculate the key node hidden effect gain $\triangle H$:

$$\triangle H(e_i) = |K| - |K \cap K_{temp}| \tag{8}$$

Here, $\triangle H()$ represents the hidden effect function, where K is the original set of key nodes before FNC insertion, and $|K \cap K_{temp}|$ is the number of original key nodes still in the new key node set K_{temp}. A larger $\triangle H$ indicates more original key nodes are hidden. We insert the FNC at the position $e_{selected}$ where $\triangle H$ is largest; if multiple $\triangle H$ values are identical, selection is random. Subsequently, the temporary topology G_{temp} is updated to the obfuscated topology G^*_{obfu}, and $e_{selected}$ is added to the FNC insertion position sequence $E_{selected}$. Simultaneously, $e_{selected}$ is removed from E_{origin}.

Repeat iteration until the maximum number of iterations, max_times, is reached or $OBFU$ becomes $true$. If $OBFU$ remains $false$, the current FNC scale b is incompatible with the network size $|V|$, and obfuscation fails. Otherwise, obfuscation is successful, and the edges in $E_{selected}$ form a set of valid FNC insertion position sequences.

Step 3: Generate the obfuscation topology. If the generated FNC insertion location sequence $E_{selected}$ is valid, the FNCs are inserted at the corresponding locations within the protected network N. This action generates the obfuscated topology G_{obfu}.

4 Experimental Results and Analysis

To validate the effectiveness of the proposed method, we employed the Topology Zoo dataset [29] and built a prototype system of the proposed method on the open-source SDN controller Ryu [30,31] and the network simulator Mininet [32]. The experiments were conducted in two aspects:

- Evaluating the obfuscation effectiveness of the proposed method on network topologies of different scales.
- Comparing the obfuscation costs with the typical method EigenObfu [27].

4.1 Experimental Setup

The experimental setup of this paper is shown in Table 3, which includes experimental categories, data sources, parameter settings, involved methods, and evaluation metrics.

Table 3: Experiment setup.

Experiment Category		Data Source		Parameter Settings		Involved Methods	Evaluation Metric
Experiment	**Description**	**Dataset**	**Description**	**Parameter Name**	**Value**		
Obfuscation effectiveness	To comprehensively evaluate the proposed method's performance across different-scale network topologies.	Topology Dataset	Topology Zoo: Real ISP networks classified by number of nodes: small ([0, 50]), medium ([50, 100]), large ([100, 200]).	FNC Scale b	[2, 10]	Ours	Graph Edit Distance (GED)
Obfuscation cost assessment	Compares the proposed method with EigenObfu under fixed parameters.			FNC Scale b	5, 10	Ours EigenObfu	
				Search Number s	100		
				Connection Probability p	0.32		

4.1.1 Data Sources

The Topology Zoo, a public dataset of real ISP network topologies, is recognized for its representativeness and practical value. To comprehensively evaluate our method's obfuscation effect and cost across network scales, we selected nine representative real ISP topologies from this dataset. These topologies are categorized by node counts: small (0–49), medium (50–99), and large (100–199). We set the critical node proportion to approximately 10%, 20%, 30%, and 40% (as shown in Table 4). Minor deviations from these preset ratios may occur due to the requirement for an integer number of nodes.

Table 4: Critical nodes and their proportions in various network topologies.

Topology Category	Topology Name (Nodes, Edges)	Number of Critical Nodes (r)	Percentage
Small topology	York (23, 24)	r = 2	8.70%
		r = 5	17.39%
		r = 7	26.09%
		r = 9	34.78%
	Sunet (26, 32)	r = 3	11.54%
		r = 5	19.23%
		r = 8	30.77%
		r = 10	38.46%
	Shentel (28, 35)	r = 3	10.71%
		r = 6	21.43%
		r = 8	28.57%
		r = 11	39.29%

(Continued)

Table 4 (continued)

Topology Category	Topology Name (Nodes, Edges)	Number of Critical Nodes (r)	Percentage
Medium topology	Columbus (70, 85)	r = 7	10.00%
		r = 14	20.00%
		r = 21	30.00%
		r = 28	40.00%
	Syringa (74, 74)	r = 7	9.46%
		r = 15	20.27%
		r = 22	29.73%
		r = 30	40.54%
	Sinet (74, 76)	r = 7	9.46%
		r = 15	20.27%
		r = 22	29.73%
		r = 30	40.54%
Large topology	Deltacom (113, 161)	r = 11	9.73%
		r = 23	20.35%
		r = 34	30.09%
		r = 45	39.82%
	Ion (125, 146)	r = 13	10.40%
		r = 25	20.00%
		r = 38	30.40%
		r = 50	40.00%
	TataNld (145, 186)	r = 15	10.34%
		r = 29	20.00%
		r = 44	30.34%
		r = 58	40.00%

4.1.2 Parameter Settings

In the experiment on obfuscation effectiveness, the obfuscation effect of the proposed method mainly depends on the FNC scale b. According to the Network Adaptation Model (NAM), the FNC scale b needs to be matched with the scale $|V|$ of the protected network N: if b is too small relative to $|V|$, the obfuscation effect cannot be achieved; if b is too large, there is a risk of exposing the obfuscation strategy. Based on the scale of the topologies used in the experiment, the interval of the FNC scale b is set to [2,10] to test the effectiveness of the method.

In the experiment on obfuscation cost evaluation, since the obfuscation effect of the proposed method is closely related to the FNC scale, to ensure its adaptability under different network scales, the FNC scale b is set to 5 and 10. For the compared typical method EigenObfu, the number of topology searches and the edge generation probability directly affect the final obfuscation cost. Its parameters are set according to the recommended range in the original literature: the search number $s = 100$ and the connection probability $p = 0.32$.

4.1.3 Evaluation Metric

We quantify network topology obfuscation cost using Graph Edit Distance (GED) [33]. GED represents the minimum edit path cost to transform graph g_1 into graph g_2, computed as shown in Eq. (9):

$$\mathrm{GED}(g_1, g_2) = \min_{\substack{m_1, m_2, \ldots, m_k \\ \in \Gamma(g_1, g_2)}} \sum_{i=1}^{k} c(m_i) \tag{9}$$

Here, m_1, $m_2, \ldots,$ m_k denote individual editing operations within an edit path. $\gamma(g_1, g_2)$ represents the set of all possible edit paths that transform graph g_1 into graph g_2. k is the total number of operations in an edit path, and $c(m_i)$ is the cost of the i-th editing operation. Since GED computation is an NP-hard problem, to simplify the computation, the cost of each operation $c(m_i)$ is set to 1, and the actual number of edit operations during the generation of the obfuscated topology is used as an approximation of the GED.

4.2 Topology Obfuscation Validity

In this subsection, experiments are conducted on network topologies of varying scales retrieved from the Topology Zoo dataset (as shown in Table 4), with an emphasis on evaluating the obfuscation capability of the proposed method under different network scales and critical node proportions and conducting numerical simulations on the key simulation parameter of the proposed method—FNC scale b—to explore its influence on the obfuscation effect of network topologies.

Since the proposed method incorporates security constraints during its design, results are only output when all critical nodes are fully obfuscated. Therefore, the presence of an obfuscation cost (GED) indicates successful obfuscation. The experimental results are shown in Fig. 4.

- **Small topologies (York, Sunet, Shentel).** Across different critical node ratios, setting $b \geq 2$ achieves complete obfuscation of critical nodes. The GED values for York and Sunet remain stable at 3, while Shentel shows GED convergence to 3 when $b \geq 4$). This indicates $b \geq 3$ meets effective requirements in small-scale topologies with relatively stable obfuscation costs.
- **Medium-sized topologies (Columbus, Syringa, Sinet).** Columbus and Syringa exhibit stable GED when $b \geq 6$, whereas Sinet fails to obfuscate critical nodes (for $r = 7$, $r = 15$, $r = 22$) when $b < 5$. This suggests that the FNC scale for medium-scale networks must be set to at least 6 to ensure basic obfuscation effectiveness.
- **Large topologies (Deltacom, Ion, TataNld).** Effective obfuscation requires $b \geq 8$. Specifically, Deltacom achieves stable GED when $b \geq 8$, Ion when $b \geq 9$, and TataNld when $b \geq 9$. Considering the obfuscation demands of complex topologies, the FNC scale for large-scale networks should be set to at least 8 to balance obfuscation efficacy and maintain low GED values.

Based on these experimental conclusions, quantitative analysis of the obfuscation effect and cost equilibrium suggests that for small, medium, and large topologies, FNC scales b of 3 to 5, 6 to 8, and 9 or more, respectively, achieve effective and low-cost obfuscation.

4.3 Comparison of the Cost of Topology Obfuscation

Since network topology obfuscation methods employing different critical node evaluation criteria exhibit fundamental differences in their optimization strategies for generating obfuscated topologies, direct comparison between methods becomes challenging. To ensure experimental rigor and validate the effectiveness of the proposed method, EigenObfu [27]—a typical approach that similarly utilizes EC for critical node evaluation—was selected as the baseline for comparison.

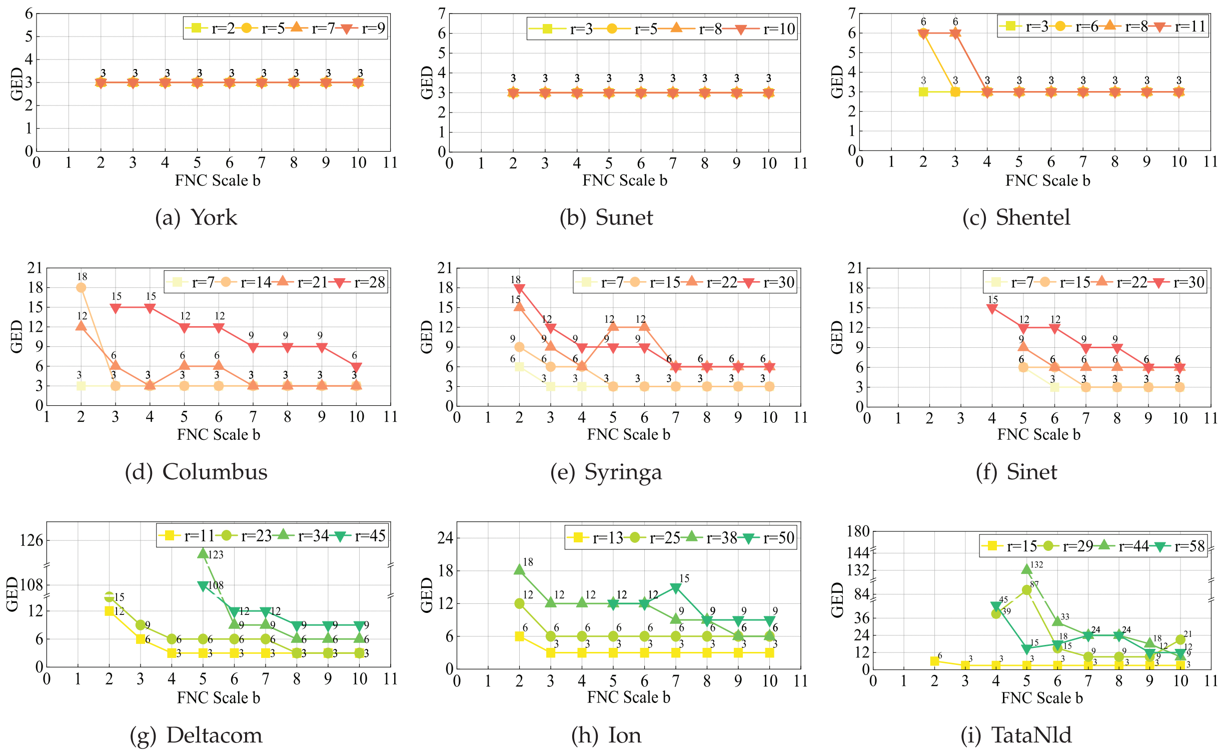


Figure 4: Relationship between different network sizes and FNC: (**a**)–(**c**) small topologies; (**d**)–(**f**) medium topologies; (**g**)–(**i**) large topologies.

The experimental results are shown in Tables 5–7. To present the data intuitively, Fig. 5 compares the obfuscation effects of networks with different topology scales.

Table 5: GED under different methods with varying number of key nodes in small topologies.

Topology Name	Number of Key Nodes	GED		
		EigenObfu [27]	**Ours (#b = 5)**	**Ours (#b = 10)**
York	2	129	3	3
	5	115	3	3
	7	102	3	3
	9	97	3	3
Sunet	3	162	3	3
	5	158	3	3
	8	142	3	3
	10	122	3	3
Shentel	3	199	3	3
	6	184	3	3
	8	172	3	3
	11	145	3	3

Table 6: GED under different methods with varying number of key nodes in medium topologies.

Topology Name	**Number of Key Nodes**	**GED**		
		Enginobfu [27]	**Ours (#b = 5)**	**Ours (#b = 10)**
Columbus	7	1382	3	3
	14	1264	3	3
	21	1109	6	3
	28	1047	12	6
Syringa	7	1519	3	3
	15	1408	3	3
	22	1292	12	6
	30	1157	9	6
Sinet	7	1613	6	3
	15	1433	6	3
	22	1308	9	6
	30	1150	12	6

Table 7: GED under different methods with varying number of key nodes in large topologies.

Topology Name	**Number of Key Nodes**	**GED**		
		Enginobfu [27]	**Ours (#b = 5)**	**Ours (#b = 10)**
Deltacom	11	3755	3	3
	23	3357	6	3
	34	3050	123	6
	45	2756	108	9
Ion	13	4561	3	3
	25	4141	6	6
	38	3696	12	6
	50	3390	12	9
TataNld	15	4738	3	3
	29	4180	87	21
	44	3716	132	9
	58	3332	15	12

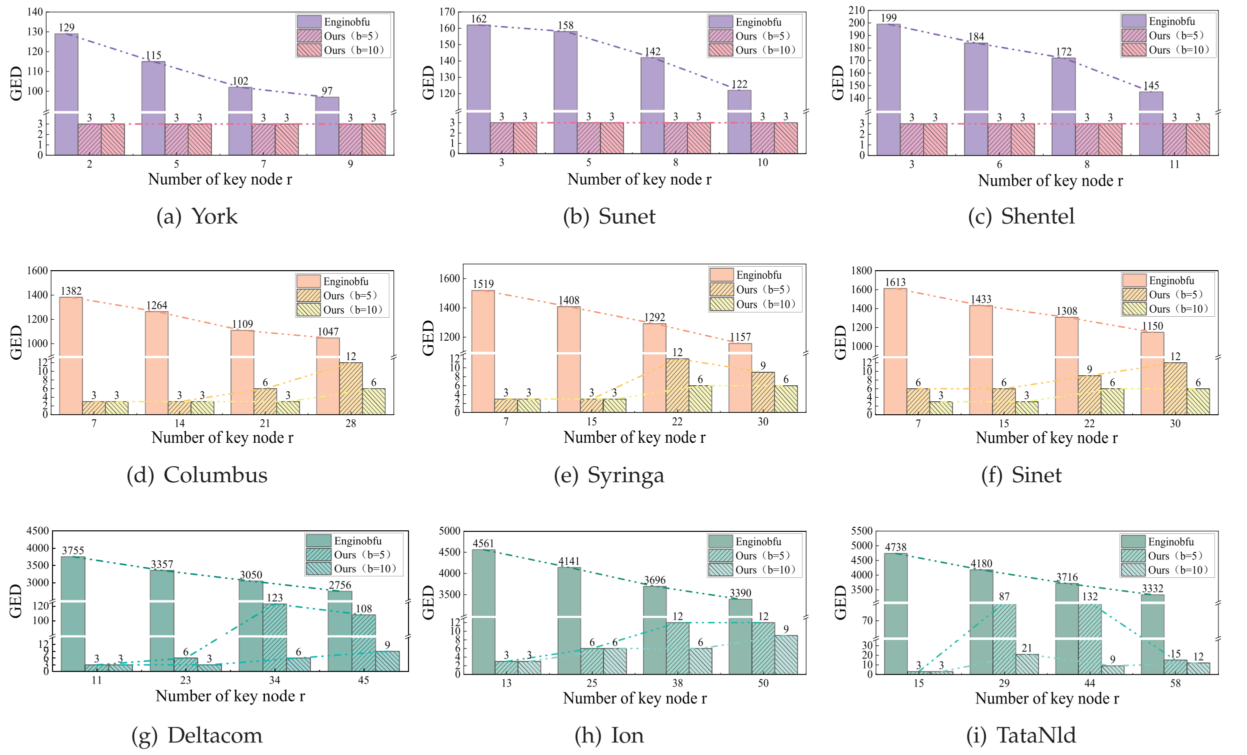


Figure 5: Comparison of EigenObfu's GED across network sizes and key node counts: (**a**)–(**c**) small topologies; (**d**)–(**f**) medium topologies; (**g**)–(**i**) large topologies.

In small-scale topologies, the proposed method with two different parameter settings consistently achieved a GED of 3, while EigenObfu exhibited GED values ranging from 97 to 199. The average GED of the proposed method was merely 2.1% of that of EigenObfu. In medium-scale topologies, the proposed method with both parameter configurations maintained GED values stable between 3 and 12, whereas EigenObfu's GED exceeded 1000. The average GED of the proposed method accounted for only 0.4% of EigenObfu's.

In large-scale topologies, the maximum GED of the proposed method with two parameter settings was 132, compared to EigenObfu's GED exceeding 2700. The average GED of the proposed method was just 0.67% of EigenObfu's.

Experimental results demonstrate that across all topology environments, the proposed method generates obfuscated topologies with significantly lower GED values than EigenObfu when the FNC scale b is set to 5 and 10, indicating substantially reduced obfuscation costs.

4.4 Algorithm Complexity Analysis

This paper involves two primary algorithms. The Fake Node Cluster Generation Algorithm functions as a resident process for continuous monitoring and identification of probe traffic; its computational overhead has a negligible impact on the overall performance of the proposed method and is therefore not detailed here. The system's performance is mainly determined by Algorithm 2 (Heuristic Obfuscated Topology Generation).

Algorithm 2: Heuristic Obfuscated Topology Generation

```
Input: ① Fake node cluster (FNC)
       ② Protected network topology $G$
Output: Obfuscated topology $G_{obfu}$
Initialization: $E_{selected}, E_{origin}, S$, max_times, $G^*_{obfu}$, OBFU $\leftarrow \varnothing, E, +\infty, |E|, G$, false
$EC \leftarrow$ calculate EC($G$) // Compute EC for each node in $G$;
$K \leftarrow \{v \in V \mid \text{rank}(EC(v)) \le r\}$ // Key nodes to protect;
for $time = 1$ to $max\_times$ do
    random_shuffle($E_{origin}$) // Randomize candidate edges;
    $\triangle H_{max} \leftarrow -\infty$;
    for each $e_i \in E_{origin}$ do
        $G_{temp} \leftarrow$ insert FNC($G^*_{obfu}, e_i, b$);
        $EC_{temp} \leftarrow$ calculate EC($G_{temp}$);
        $K_{temp} \leftarrow \{v \in V \mid \text{rank}(EC_{temp}(v)) \le r\}$;
        $\triangle H \leftarrow (|K| - |K \cap K_{temp}|)$ // Evaluate improvement in hiding effect;
        if $\triangle H > \triangle H_{max}$ then
            $\triangle H_{max} \leftarrow \triangle H$;
            $e_{selected} \leftarrow e_i$;
        end
        if $S_{temp} == 0$ then
            OBFU $\leftarrow$ true;
        end
    end
    $E_{selected} \leftarrow E_{selected} \cup \{e_{selected}\}$;
    $G^*_{obfu} \leftarrow$ insert FNC($G^*_{obfu}, e_{selected}, b$);
    $E_{origin} \leftarrow E_{origin} - \{e_{selected}\}$;
end
if $OBFU == true$ then
    Insert FNC into $G$ based on $E_{selected}$ to generate the obfuscated topology $G_{obfu}$;
end
return $G_{obfu}$;
```

Let the target network topology be an undirected graph $G = (V, E)$, with $|V|$ nodes and $|E|$ edges. The objective of Algorithm 2 is to select a set of FNC insertion positions, denoted as E_{selected}, from the E to conceal all critical nodes. The time complexity and space complexity of Algorithm 2 are both polynomial, and the detailed analysis is as follows.

- **Time Complexity:** The initialization phase involves constant-time operations. The identification of critical nodes requires calculating EC and sorting the results. The EC calculation takes $O(|V|^3)$, which dominates the sorting complexity of $O(|V| \log |V|)$. Consequently, the preprocessing complexity is $O(|V|^3)$. The heuristic optimization is constrained to a maximum of 100 iterations. In each iteration, the algorithm traverses candidate edges to generate a temporary topology G_{temp} (complexity related to $|E|$) and recalculates EC values to evaluate the hiding gain ($O(|V|^3)$). Therefore, the total time complexity approximates $O(|E| \cdot |V|^3)$. This polynomial complexity ensures that the computational cost remains controllable as $|V|$ and $|E|$ increase.

- **Space Complexity:** The memory consumption is primarily determined by the storage of the adjacency matrix for the current topology. Thus, the space complexity is $O(|V|^2)$, which is also polynomial.

5 Conclusion

To address the problem of excessively high obfuscation costs in existing network topology obfuscation methods for critical node protection, this paper proposes a novel low-cost network topology obfuscation approach. The method first performs topological modeling on the network to be protected, then constructs target-specific fake node clusters with particular structures, and inserts these clusters into the network based on a greedy strategy to generate the obfuscated topology, thereby disrupting the importance ranking of nodes. Experimental results demonstrate that the proposed method can achieve network topology obfuscation and critical node protection for different-scale networks with low costs.

The limitation of the proposed method is that it is only applicable to the obfuscation of critical nodes evaluated based on eigenvector centrality. Currently, our method mainly resists attack approaches that evaluate critical nodes based on eigenvector centrality. In future work, we will design diverse fake node cluster structures to realize network topology obfuscation in various adversarial scenarios.

Acknowledgement: We thank Dr. Ziliang Zhu from Harbin Institute of Technology for providing valuable assistance in the reproduction of the EigenObfu method.

Funding Statement: This research was funded by the National Key Research and Development Program of China (Grant No. 2022YFB3102900), the Key Science and Technology Project of Henan Province, China (No. 252102211091).

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Yanming Chen and Fuxiang Yuan; Methodology, Yanming Chen; Software, Yanming Chen; Investigation, Yanming Chen; Formal analysis, Yanming Chen; Data curation, Yanming Chen; Writing—original draft preparation, Yanming Chen; Writing—review and editing, Yanming Chen, Fuxiang Yuan and Zekang Wang; Supervision, Fuxiang Yuan and Zekang Wang; Resources, Fuxiang Yuan. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author, upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang Z, Yuan F, Li R, Zhang M, Luo X. Hidden AS link prediction based on random forest feature selection and GWO-XGBoost model. Comput Netw. 2025;262(1):111164. doi:10.1016/j.comnet.2025.111164.
2. Xing L. Cascading failures in Internet of Things: review and perspectives on reliability and resilience. IEEE Internet of Things J. 2020;8(1):44–64. doi:10.1109/jiot.2020.3018687.
3. Pastor-Satorras R, Vespignani A. Epidemic spreading in scale-free networks. Phys Rev Lett. 2023;31(14):97–111. doi:10.1103/physrevlett.86.3200.
4. Fu X, Yang Y. Modeling and analyzing cascading failures for internet of things. Inform Sci. 2021;545:753–70. doi:10.1016/j.ins.2020.09.054.
5. Zhu L, Fu X, Liu X, Du S. Modeling and analysis of cascade failures in Industrial Internet of Things based on task decomposition and service communities. Comput Ind Eng. 2025;74(2):2723–37. doi:10.1016/j.cie.2025.111177.
6. Zhang M, Li G, Wang S, Liu C, Chen A, Hu H, et al. Poseidon: mitigating volumetric ddos attacks with programmable switches. In: Proceedings of the 27th Network and Distributed System Security Symposium (NDSS). Reston, VA, USA: Internet Society; 2020. p. 1–18.

7. Xia Y, Zhang W, Liu Y, Kang J, Zhang H. Learning-based proactive and adaptive link flooding attack mitigation in AIoT. IEEE Internet of Things J. 2025;12(13):23373–88. doi:10.1109/jiot.2025.3553250.
8. Sakuma K, Asahina H, Haruta S, Sasase I. Traceroute-based target link flooding attack detection scheme by analyzing hop count to the destination. In: Proceedings of the 23rd Asia-Pacific Conference on Communications (APCC). Piscataway, NJ, USA: IEEE; 2017. p. 1–6.
9. Li S, Lan T. From network inference errors to utility suboptimality: how much is the impact? IEEE Trans Netw Sci Eng. 2022;9(6):4075–86. doi:10.1109/tnse.2022.3195805.
10. Ravi N, Shalinie SM, Danyson J, Theres D. Balance: link flooding attack detection and mitigation via hybrid-SDN. IEEE Trans Netw Serv Manage. 2020;17(3):1715–29. doi:10.1109/tnsm.2020.2997734.
11. Xing J, Wu W, Chen A. Ripple: a programmable, decentralized link-flooding defense against adaptive adversaries. In: Proceedings of the 30th USENIX Security Symposium. Berkeley, CA, USA: USENIX Association; 2021. p. 3865–81.
12. Meier R, Tsankov P, Lenders V, Vanbever L, Vechev M. NetHide: secure and practical network topology obfuscation. In: Proceedings of the 27th USENIX Security Symposium. Berkeley, CA, USA: USENIX Association; 2018. p. 693–709.
13. Huang X, Xue K, Huang Z, Han J, Chen L, Wei DS, et al. SpiderNet: enabling bot identification in network topology obfuscation against link flooding attacks. IEEE/ACM Trans Netw. 2024;33(1):4075–86. doi:10.1109/tnet.2024.3473742.
14. Ding X, Xiao F, Zhou M, Wang Z. Active link obfuscation to thwart link-flooding attacks for internet of things. In: Proceedings of the 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). Piscataway, NJ, USA: IEEE; 2020. p. 217–24.
15. Liu Y, Zhao J, Zhang G, Xing C. NetObfu: a lightweight and efficient network topology obfuscation defense scheme. Comput Secur. 2021;110(C):102447. doi:10.1016/j.cose.2021.102447.
16. Wang J, Liu Y, Zhang W, Yan X, Zhou N, Jiang Z. ReLFA: resist link flooding attacks via renyi entropy and deep reinforcement learning in SDN-IoT. China Commun. 2022;19(7):157–71. doi:10.23919/jcc.2022.07.013.
17. Kim J, Shin S. Software-defined HoneyNet: towards mitigating link flooding attacks. In: Proceedings of the 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W). Piscataway, NJ, USA: IEEE; 2017. p. 99–100.
18. Hou T, Qu Z, Wang T, Lu Z, Liu Y. Proto: proactive topology obfuscation against adversarial network topology inference. In: Proceedings of the 39th IEEE Conference on Computer Communications. Piscataway, NJ, USA: IEEE; 2020. p. 1598–607.
19. Liu Y, Xing C, Zhang G, Song L, Lin H. AntiTomo: network topology obfuscation against adversarial tomography-based topology inference. Comput Secur. 2022;113:102570. doi:10.1016/j.cose.2021.102570.
20. Trassare ST, Beverly R, Alderson D. A technique for network topology deception. In: Proceedings of the 32th IEEE Military Communications Conference. Piscataway, NJ, USA: IEEE; 2013. p. 1795–800.
21. Kim J, Marin E, Conti M, Shin S. EqualNet: a secure and practical defense for long-term network topology obfuscation. In: Proceedings of the 29th Annual Network and Distributed System Security Symposium. Reston, VA, USA: Internet Society; 2022. p. 1–18.
22. Yan D, Xie W, Zhang Y, He Q, Yang Y. Hypernetwork dismantling via deep reinforcement learning. IEEE Trans Netw Sci Eng. 2022;9(5):3302–15. doi:10.1109/tnse.2022.3174163.
23. Kim J, Nam J, Lee S, Yegneswaran V, Porras P, Shin S. BottleNet: hiding network bottlenecks using SDN-based topology deception. IEEE Trans Inform Forens Secur. 2021;16:3138–53. doi:10.1109/tifs.2021.3075845.
24. Frost HR. A generalized eigenvector centrality for multilayer networks with inter-layer constraints on adjacent node importance. Appl Netw Sci. 2024;9:14. doi:10.1007/s41109-024-00620-8.
25. Elnour M, Atat R, Takiddin A, Ismail M, Serpedin E. Eigenvector centrality-enhanced graph network for attack detection in power distribution systems. Elect Power Syst Res. 2025;241:111339. doi:10.1016/j.epsr.2024.111339.
26. Zhou H, Hong S, Liu Y, Luo X, Li W, Gu G. Mew: enabling large-scale and dynamic link-flooding defenses on programmable switches. In: Proceedings of the 44th IEEE Symposium on Security and Privacy (SP). Piscataway, NJ, USA: IEEE; 2023. p. 3178–92.

27. Zhu Z, Zhu G, Zhang Y, Shi J, Huang X, Fang Y. Eigenobfu: a novel network topology obfuscation defense method. IEEE Trans Netw Sci Eng. 2025;12(1):451–62. doi:10.1109/tnse.2024.3501396.
28. He F, Wang Z, Gu X. Network topology generation based on eigenvector centrality with real-time guarantee. Concurr Comput: Pract Exp. 2023;35(21):1524–40. doi:10.1002/cpe.6955.
29. Knight S, Nguyen HX, Falkner N, Bowden R, Roughan M. The internet topology zoo. IEEE J Selected Areas Commun. 2011;29(9):1765–75. doi:10.1109/jsac.2011.111002.
30. Albu-Salih AT. Performance evaluation of ryu controller in software defined networks. J Al-Qadisiyah Comput Sci Math. 2022;14(1):1–7. doi:10.29304/jqcm.2022.14.1.879.
31. Asadollahi S, Goswami B, Sameer M. Ryu controller's scalability experiment on software defined networks. In: Proceedings of the IEEE International Conference on Current Trends in Advanced Computing (ICCTAC). Piscataway, NJ, USA: IEEE; 2018. p. 1–5.
32. Lantz B, Heller B, McKeown N. A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM Workshop on Hot Topics in Networks. New York, NY, USA: ACM; 2010. p. 1–6.
33. Abu-Aisheh Z, Raveaux R, Ramel JY, Martineau P. An exact graph edit distance algorithm for solving pattern recognition problems. In: Proceedings of the 4th International Conference on Pattern Recognition Applications and Methods. Setubal, Portugal: Scitepress; 2015. p. 271–8.