Computers & Security 144 (2024) 103981

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Combating temporal composition inference by high-order camouflaged network topology obfuscation ☆

Xiaohui Li [a,*], Xiang Yang [a], Yizhao Huang [b], Yue Chen [a]

[a] The School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China
[b] The College of Computer Science, Sichuan University, Chengdu 610065, China

ARTICLE INFO

Keywords:
Topology obfuscation
Proactive defense
Network adversarial
Hypergraph extraction
Discriminator evaluation

ABSTRACT

Topology inference driven by non-collaborative or incomplete prior knowledge is widely used in pivotal target network sieving and completion. However, perceivable topology also allows attackers to identify the fragile bottlenecks and perform efficacious attacks that are difficult to defend against by injecting indistinguishable low-volume attacks. Most existing countermeasures are proposed to obfuscate network data or set up honeypots with adversarial examples. However, there are two challenges when adding perturbations to live network links or nodes. Firstly, the perturbations imposed on the network cannot be conveniently projected to the original network with poor scalability. Secondly, applying significant changes to network information is laborious and impractical. In short, making a good trade-off between concealment and complexity is challenging. To address the above issues, we propose a fraudulent proactive defending tactic, namely *HBB-TSP*, to protect live network privacy by combating attacks of temporal network inference. Specifically, to penetrate the critical network structures, *HBB-TSP* first brings in the Statistical Validation of Hypergraph (SVH) method to identify the pivotal connection information of the network and extract the deep backbone structure. Then, the Temporal Simple Decomposition Weighting (TSDW) strategy is introduced, which can predict the backbone network with evolution rules and add highly obfuscated features at a minimized overhead. Finally, a discriminator with multiple centrality models is used to evaluate the deceptiveness and, in turn, affect the TSDW prediction. The entire process ensures the consistency and robustness of network changes while ensuring effective adversarial resistance. Experimental results on two scale real-world datasets demonstrate the effectiveness and generalization of adversarial perturbations. In particular, it is encouraging that our proposed defending scheme outperforms the advanced countermeasures. It ensures the realization of a deceptive obfuscated network at minimum overhead and is suitable for widespread deployment in scenarios of different scales.

1. Introduction

Network topology is the foundation for devices to connect and communicate, which is crucial for system performance, stability, and security (Zhao et al., 2024a; Liu et al., 2023b; Chee et al., 2024). However, the potential for attackers to exploit accurate network topology information for planning efficient attack routes with the least risk of being detected cannot be ignored. Typically, attackers follow multiple processes in attacking a network (Zhang and Chen, 2024). For example, (*i*) detection attacks (Liu et al., 2024; Doriguzzi-Corin and Siracusa, 2024) are executed to gather information about the target system and network security, including scanning technology, architecture probing, system information service collection, and traceless information detection. (*ii*) Blocking attacks (Yoo et al., 2024; Li et al., 2023) are launched to forcefully consume channel resources, network connections, and storage space, causing the server to crash or become exhausted and unable to provide external services. Denial of Service (DoS) is a typical blocking attack that attacks the target through protocols or seizes system vulnerabilities until the other party's network is paralyzed. Common methods include email bombs, Link-Flooding Attacks (LFA), TCP SYN flood attacks, Land attacks, etc. Taking mail servers as an example, attackers can trace network communication paths through technologies such as Traceroute to identify and exploit highly connected mail servers or key transmission nodes for building

☆ This document is the results of the research project funded by the National Natural Science Foundation of China under Grant 62101368 and Grant U2133208; and the Sichuan Youth Science and Technology Innovation Team, China under Grant 2022JDTD0014.

* Corresponding author.
E-mail addresses: lixiaohui@scu.edu.cn (X. Li), 2021141530106@stu.scu.edu.cn (X. Yang), huangyizhao@stu.scu.edu.cn (Y. Huang), yuechen@scu.edu.cn (Y. Chen).

https://doi.org/10.1016/j.cose.2024.103981
Received 2 March 2024; Received in revised form 21 May 2024; Accepted 29 June 2024
Available online 5 July 2024

advanced DoS attacks to maximize the attack's impact (Coscia et al., 2024; Zhao et al., 2024b). Hence, it is imperative to effectively camouflage the network topology and other relevant information to induce attackers or obfuscate the adversary's understanding, called beggarthy-neighbor. Topology obfuscation is necessary to improve network topology security and enhance the concealment of critical details.

The accurate identification of bottleneck links and nodes by attackers depends on their ability to access the network topology. However, the strategy of obfuscated network generation can be adopted to construct a network topology that exhibits a similar structure while concealing critical information. This makes it challenging for attackers to use network path tracing tools, such as Traceroute, to obtain the actual and detailed network topology information. As a result, the cost of Distributed DoS (DDoS) and other attacks increases, and obfuscated network generation emerges as an effective means to enhance network security and privacy. Therefore, obfuscated network generation holds practical significance and should be considered for improving network security measures.

In the current landscape of active defense, attackers are increasingly leveraging intelligent analyzers to exploit observable properties in the obfuscated network generation. This poses a significant threat to user privacy, and dynamic transfer methods based on the network attack level are being used to counter it. Active defense strategies based on deception, such as Moving Target Defense (MTD) (Celdrán et al., 2024) and honeypots (Javadpour et al., 2024), are gaining popularity among these methods. The MTD approach employs dynamic or static permutation, deformation, transformation, or obfuscation to divert the attacker's attack, with MTDCAP (Sun et al., 2024), MMTD (Wang et al., 2023a), and LOKI (Schloegel et al., 2022) being some of its well-known examples. Honeypot, on the other hand, lures attackers and helps grasp attack or isolate attack traffic (Ye et al., 2020; Wang et al., 2023b; Maesschalck et al., 2024). However, the selection of dynamic transfer decisions is often blind, and random transfer methods significantly reduce the effectiveness of the defense. Therefore, obfuscation defense technology based on complex networks can analyze the network topology from a theoretical level to generate multi-dimensional elements that resist network adversary analysis (Ye et al., 2020). This technology includes obfuscated network generation, intelligent algorithm obfuscation, and traffic distribution obfuscation and can serve as an effective defense against network attacks.

Multiple types of solutions have been derived on this subject. For mitigating LFAs, ONSET (Nance-Hall et al., 2024) was proposed, which operates by leveraging optical topology programming to dynamically scale network capacity and create new paths during LFA incidents. Accordingly, in response to cyber threats from network reconnaissance, the Intelligence-Driven Host Address Mutation (ID-HAM) (Zhang et al., 2023) scheme significantly reduces the scanning hit rate through a Markov decision process and dominant actor criticism algorithms while maintaining minimal communication impact. Another approach is to add virtual nodes or links to balance the tracking flow of nodes and links. For example, EqualNet (Kim et al., 2022) adds virtual nodes or links to balance the tracking flow, effectively countering fingerprinting and topology inference attacks. NetHide (Meier et al., 2018) generates a virtual topology using physical network topology graphs and forwarding behavioral specifications, which can flexibly trade off security and availability. This approach prevents attackers such as LFAs from using intelligent obfuscation strategies, such as generating random IP addresses or plausible pseudo-network topologies, making it difficult for attackers to recognize the network topology. Trassare et al. (2013) proposed the strategies of random obfuscation by returning randomly generated IP addresses to create noise and intelligent obfuscation by generating plausible pseudo-network topologies, which make it difficult for attackers to recognize the actual network topology. ProTo (Hou et al., 2020) generates an obfuscated topology to successfully counteract attacks based on external end-to-end topology probing, such as DDoS attacks. Regarding behavior distribution confusion, Netobfu (Liu et al., 2021) iteratively adjusts the traffic density by selecting optimal virtual paths and honeypot links through a greedy algorithm. This prevents attackers from using topological information to implement harmful attacks, such as LFA. AntiTomo model (Liu et al., 2022) selects the optimal correlation-delay obfuscated network topology against topology-detection-based network attacks, such as DDoS attacks and LFAs, using the Candidate Forest Generation algorithm that randomly generates a set of candidate trees. Finally, Linkbait (Wang et al., 2017) proposes a selective traffic rerouting strategy that effectively blocks LFAs by rerouting traffic on decoy links and achieves a high detection rate of compromised devices.

The use of adversarial perturbations on network topology has emerged as a promising research area for protecting network privacy. However, several challenges remain unsolved.

- **Obscure concealment:** Current research only focuses on safeguarding network topology information without considering network authenticity constraints. Conducting a comprehensive and in-depth analysis based on network dynamics and sequential characteristics is imperative to create a concealed and undetectable obfuscated network construction based on partial conditional networks (King and Huang, 2023; Qin and Yeung, 2023).
- **Confrontational robustness:** Current methods for generating obfuscated networks are limited in their applicability. For instance, most of these methods are designed to secure the connection links between routers and prevent LFA (Meier et al., 2018; Trassare et al., 2013; Wang et al., 2017). Therefore, we must develop more organized-based solutions, improving the adaptability, operability, and effectiveness of obfuscation methods.
- **Finite overhead:** Most of the existing obfuscation methods are associated with high overhead (Kim et al., 2022). Protecting subnet information can result in significant resource overhead. Generating virtual nodes is often required, and deployment may be challenging.

In this paper, a framework called *HBB-TSP* generates obfuscated networks to protect complex network structures from cyberattacks by maintaining anonymity. *HBB-TSP* operates as a network topology space framework, safeguarding network topology privacy through adversarial interference on real-time network topology. To ensure the authenticity of obfuscated network topology generation, *HBB-TSP* utilizes Hypergraph BackBone extraction (HBB) to extract bottleneck links of the network, thus extracting the most important links and ensuring that any added links have a significant impact on the network and aligns with changes to the topology. Then, a Temporal Simplicial closure event Prediction (TSP) is employed to reinforce the generated obfuscation part, all while adhering to overhead constraints. Finally, an evaluation scheme is utilized to assess the developed obfuscation network and provide feedback for refining the generator. We posit that *HBB-TSP* represents a significant advancement in network security. As cyberattacks become increasingly sophisticated, preserving the anonymity and privacy of complex network structures is critical.

The rest of the paper is organized as follows. Section 2 provides detailed information on the algorithms used in the framework. In Section 3, we compare *HBB-TSP* with the baselines and Section 4 describes the experimental results, analysis, and discussion, and finally, Section 5 conclude the proposed work.

2. Defeating temporal analyzers with *HBB-TSP*

2.1. Overview

This section presents the formulations and algorithms for generating obfuscated networks by complying with timing and cost constraints with *HBB-TSP* being illustrated in Fig. 1. The notations utilized in the *HBB-TSP* model are summarized in Table 1 to facilitate a better understanding of the model. The following subsection gives a detailed discussion of the proposed *HBB-TSP*.

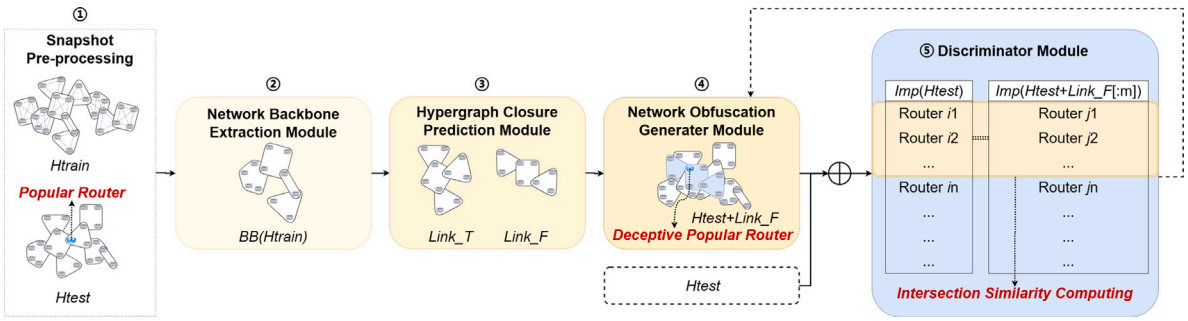


Fig. 1. *HBB-TSP* algorithm framework.

Table 1
HBB-TSP symbol descriptions.

Symbol	Description
H	Hypergraph snapshots with historical H_{train}, H_{test}
$Link\_T/F$	Prediction of hyperedges in $T_{(i+1)}$
$BB(\cdot)$	Extracted backbone network
$TSP(\cdot)$	Prediction results for simplicial closure events
$Imp(\cdot)$	Ranking of important nodes
$IS(\cdot,\cdot,n)$	Intersection similarity of the top n nodes
$Pr\_Auc(H_1, H_2)$	AUC of the predicted hypergraph H_1 and actual hypergraph H_2

- **Snapshot Pre-processing** By representing the network topology as a series of dynamic hypergraph snapshots, we can comprehensively understand its evolution over time. These snapshots are further partitioned into historical snapshots H_{train} with the generated obfuscated network H_{test} at time T, allowing us to delve deeper into the network's past and analyze its growth patterns. This approach can provide valuable insights for optimizing network performance and improving overall efficiency.
- **Network Backbone Extraction** In this module, to discover insights and visualize complex relationships, the SVH algorithm is introduced. The network backbone from a historical hypergraph snapshot H_{train} is extracted, and the benefits of a more diverse edge relationship representation are utilized. Hypergraphs offer a more natural and effective way of depicting multiple relationships, especially in scenarios where multiple connections must be expressed. Especially when dealing with complex relationships and dependency structures, hypergraphs can represent relationships more intuitively.
- **Hypergraph Closure Prediction & Obfuscation Generator** By implementing hypergraph closure prediction on the network backbone, the precise prediction of hyperedges in the next phase is made possible. The TSDW algorithm ensures that the most current and relevant information is available, thereby facilitating informed and strategic decision-making and ultimately enabling organizations to maintain a competitive edge. Moreover, this approach effectively combines the in-depth analysis of hypergraphs with the equilibrium stability of closure prediction, resulting in a robust and effective modeling strategy. Including the predicted hyperedge $Link\_F$ is crucial in finalizing the obfuscation process. It is paramount to incorporate the expected hyperedge into the network H_{test} to guarantee the production of a thoroughly obfuscated network.
- **Discriminator** By utilizing spectral centrality analysis, we can accurately detect significant nodes and hyperedges in the discriminator. In addition, the essential nodes and links are identified, and then, the fundamental information of the generated obfuscated network is compared with that of the authentic network. This comprehensive approach ensures that the generator is influenced by the superimposition of supersedes, resulting in a more robust and reliable network.

2.2. The definition of dynamic heterogeneous hypergraph

Formally, *HBB-TSP* aims to minimize the $IS(\cdot)$ between authentic and obfuscated networks, with the constraints of hyperedge prediction accuracy requirements and hyperedge number limitations. Our objective function represents the ultimate goal of the obfuscated network generated by *HBB-TSP* in Eq. (1), which is to minimize the similarity of the intersection with the original sequence of the top n important nodes in the network. Two constraints are essential for *HBB-TSP*, which is formulated as follows:

$$\begin{aligned} \underset{m}{\text{minimize}} \quad & IS\left(Imp\left(H_{\text{test}}\right), Imp\left(H_{\text{test}} + Link\_F[:m]\right), n\right) \\ \text{subject to} \quad & \begin{cases} pr\_Auc\left(TSP\left(BB\left(H_{\text{train}}\right)\right), H_{\text{test}}\right) > \lambda & (i) \\ 0 < m < \mu & (ii) \end{cases} \end{aligned} \tag{1}$$

where m represents the number of obfuscated links to be modified; note that constraint (i) represents that the accuracy of simplicial closure event prediction should exceed the parameter λ, and constraint (ii) indicates that the number of added obfuscated links should not surpass the parameter μ, namely, the dual constraints on obfuscation (constraint (i)) and cost (constraint (ii)). Specifically, the accuracy used to compare with λ in (i) refers to the similarity between the predicted link and the real link obtained by extracting the network backbone of H_{train} for temporal link prediction. A higher value indicates better prediction accuracy. An accuracy greater than λ means that the predicted link aligns with the pattern of change in the real network in this experiment, i.e., more obfuscation. The metric m used to compare with μ in (ii) represents the number of joined links. We believe that this value is too large to increase the cost and be easier to detect by attackers. Therefore, the number of joined links being less than μ indicates that the cost is feasible.

In mathematics, a hypergraph is a generalization of a traditional graph, where edges can connect any number of vertices. Hypergraphs extend graph theory to represent relationships among node sets abstractly, enabling hyperedge to connect multiple nodes, thus enhancing structural diversity. Dynamic heterogeneous hypergraphs depict the hypergraph as a time-evolving structure, where nodes and hyperedges can simultaneously belong to different types. Typically, a dynamic heterogeneous hypergraph $H = \left(\{H^t\}_{t=1}^{T}\right)$ is structured across temporal slices, where T represents the number of time slices. At each time point t, $H^t = \{V^t, E^t\}$, where $V^t = \bigcup_{i=1}^{vn} v_i$ denotes the set of nodes, with vn representing the total number of nodes. Simultaneously, $E^t = \bigcup_{i=1}^{en} e_i$ denotes the set of hyperedges, with en indicating the count of hyperedges. A hyperedge e_i can be succinctly expressed as a set of nodes, exemplified by $\{v_1, v_2, \ldots, v_j, v_k, \ldots\}$. To illustrate, consider a topology graph in router packet transmission, where a third-order hyperedge could signify a scenario in which three routers collectively transmit a critical message.

A simplex provides an alternative representation of a hyperedge, where a d-dimensional simplex (or d-simplex) signifies a set of $d+1$ fully interacting nodes. Precisely, a hyperedge involving three nodes corresponds to a 2-simplex, and a hyperedge involving four nodes is

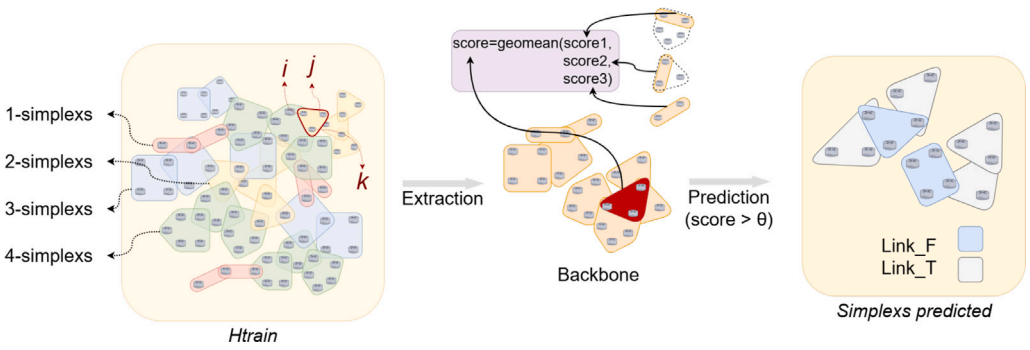


Fig. 2. The detailed diagram of the *HBB-TSP* algorithm. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

denoted as a 3-simplex, symbolically expressed as (a, b, c, d), the simplex form of each dimension is clearly labeled in the dark yellow region of Fig. 2. A 3-simplex encapsulates not only the interactions among the four nodes but also delineates interactions within its subsets, as exemplified by $\{(a, b, c, d), (a, b, c), (b, c, d), \ldots, (c), (d)\}$.

2.3. Network backbone extraction

To predict obfuscated links concealing crucial network information, the primary objective is to extract essential links from H_{train}. However, considering our aim is to utilize these extracted significant links for link prediction, it is crucial to ascertain that the extracted sub-networks form a set of critical links aligning with the structural characteristics of the original network. Employing important links directly may generate sub-networks that deviate from the structural characteristics of the original network, thus reducing the effectiveness of link prediction.

Inspired by the previous work (Musciotto et al., 2021), we use the SVH algorithm to extract the network backbone. we assess the significance of a hyperedge by comparing the frequency of node interactions within the cluster to its probability of interaction in a random group. For each hyperedge e_i with n nodes $\{v_1, v_2, \ldots, v_n\}$, SVH collects all the hyperedges of size n and filters out the hyperedges with overlapping nodes with e_i, and finally computes the hypergeometric distribution to determine whether e_i belongs to the backbone network.

Illustratively, considering the hyperedge composed of nodes i, j, and k within the yellow region in Fig. 2, we first extract the other hyperedges that have the same number of nodes, such as all the 2-simplex portion of the yellow region of Fig. 2. The network is then divided into different regions, which are the set of all 2-simplices associated with node i (e.g., N_i^3), the set of all 2-simplices associated with node j (e.g., N_j^3) and the set of all 2-simplices associated with node k (e.g., N_k^3). Formally, the probability $p(N_{ijk}^3)$ quantifies the likelihood of three nodes interacting N_{ijk}^3 times under a random null model, expressed as:

$$p(N_{ijk}^3) = \sum_X H(X|N^3, N_i^3, N_j^3) \times H(N_{ijk}^3|N^3, X, N_k^3) = \frac{\sum_X \binom{N_i^3}{X}\binom{N^3-N_i^3}{N_j^3-X}\binom{X}{N_{ijk}^3}\binom{N^3-X}{N_k^3-N_{ijk}^3}}{\binom{N^3}{N_j^3}\binom{N^3}{N_k^3}} \tag{2}$$

where $p(\cdot)$ represents the probability of the count of hyperedges connecting nodes i, j, and k, and $H(N_{AB}|N, N_A, N_B)$ denotes the hypergeometric distribution, determining the probability of an intersection sized N_{AB} between sets A and B, each possessing sizes N_A and N_B respectively, within a total set of N elements.

Based on $p(N_{ijk}^3)$, the survival function is computed by assessing the probability that the count of hyperedges formed by nodes i, j, and k is more or equal to N_{ijk}^3. This calculation allows the evaluation of the significance of the observed hyperedges, indicating the extent of deviation from the anticipated conditions prescribed by the null model. A lower observed probability suggests an overexpression of the hyperedge.

$$p(x \geq N_{ijk}^3) = 1 - \sum_{x=0}^{N_{ijk}^3-1} p(x) \tag{3}$$

For hyperedges with a generic size n, a parallel methodology is employed to initially compute the probability $p(N_{x_1\ldots x_n}^n)$ representing the interaction among n nodes within the hyperedge. Where $N_{x_1\ldots x_n}^n$ denotes the count of hyperedges comprising the specified n nodes, analogous to the previous N_{ijk}^3. Formally, the probability $p(N_{x_1\ldots x_n}^n)$ can be formulated as follows:

$$\begin{aligned} p(N_{x_1\ldots x_n}^n) = \sum_{X_{x_1x_2}} \Bigg(& H(X_{x_1x_2}|N^n, N_{x_1}^n, N_{x_2}^n) \\ & \times \sum_{X_{x_1x_2x_3}} \Bigg(H(X_{x_1x_2x_3}|N^n, X_{x_1x_2}, N_{x_3}^n) \\ & \times \cdots \\ & \times \sum_{X_{x_1x_2\ldots x_{n-1}}} \Bigg(H(X_{x_1x_2\ldots x_{n-1}}|N^n, X_{x_1x_2\ldots x_{n-2}}, N_{x_{n-1}}^n) \\ & \times H(N_{x_1x_2\ldots x_n}^n|N^n, X_{x_1x_2\ldots x_{n-1}}, N_{x_n}^n) \Bigg) \cdots \Bigg) \Bigg) \end{aligned} \tag{4}$$

Similarly, the survival function is also modified as follows:

$$p\left(x \geq N_{x_1\ldots x_n}^n\right) = 1 - \sum_{x=0}^{N_{x_1\ldots x_n}^n-1} p(x) \tag{5}$$

To rigorously filter out statistically insignificant hyperedges and determine the backbone, the SVH algorithm employs a stringent set of criteria including a significance threshold of $\alpha = 0.01$, a Bonferroni correction, and a False Discovery Rate (FDR) method. The formula for determining the backbone is expressed as:

$$SVH(N_{x1\ldots xn}^n) = \begin{cases} 1, & \text{if } p(x \geq N_{x1\ldots xn}^3) < FDR \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where FDR is computed by sorting p-values in ascending order and comparing them to a critical value derived from the desired FDR level (e.g., 0.01) and the total number of tests. The critical value k is calculated as $i * \alpha/m$, where i is the rank of the p-value, α is the desired FDR level, and m is the total number of tests. The FDR threshold is the largest p-value less than its corresponding critical value. p-values equal to or below the FDR threshold are considered significant, while those exceeding the threshold are deemed insignificant. By applying the SVH algorithm, the original networks, which typically contain much irrelevant information, can be compressed into a backbone most representative of the underlying structure and significantly impact the network. Using a null hypothesis model (Musciotto et al., 2021), SVH effectively filters out links lacking statistical significance.

2.4. Hypergraph closure prediction & obfuscation generator

Given the extracted network backbone, we aim to predict obfuscated links that can alter the sequence of critical nodes in the network (H_{test}) at the next time. To address this, we employ the SDW algorithm (Liu et al., 2023a), a hyperedge prediction method grounded in localized information. Unlike conventional model training that abstracts network changes as sequences, SDW assesses the likelihood of simplicial closure by calculating weights between node pairs. Here, simplicial closure denotes the transition of a simplex from exclusively possessing its subset simplices to its initial occurrence at a new time point.

Taking the prediction of the existence of a hyperedge $(v_1, \ldots, v_k)$ at the next time point as an example, we conduct the prediction by computing the scores of all 1-simplex it encompasses, precisely calculating $sdw(v_i, v_j)$ where $i, j \in \{v_1, \ldots, v_k\}$. The computation of $sdw(v_i, v_j)$ first involves identifying simplices containing these two nodes and their interaction data. Subsequently, we obtain the k-faces of the simplex set, designating these k-faces as the initial candidate sequence. Following this, we add q-faces of the initial candidate sequence ($q = k-1, \ldots, 2$) and update the candidate sequence. Each updated element involves nodes x_i and x_j for prediction, and the weight calculation is performed by averaging the weights for each pair of nodes within a predicted k-simplex. This process yields the score for that simplex, where a higher score signifies a higher likelihood of occurrence the next time.

As introduced above, the computation of the fraction of 1-simplex depends on the order of the simplex to be predicted. In addition, since the historical data of network topology taken in this experiment may contain a lot of redundant information, in order to avoid SDW learning information from too long ago, we propose Temporal Simple Decomposition Weighting (TSDW) here, which means that in combination with the time of appearance of simplices, the earlier the appearance time of the simplices has a lower weight, and the later the simplices have a higher weight, i.e., the one that we consider to be the one that is more closely related to the new network topology. Aiming to reduce the computational complexity and minimize the probability of detecting obfuscated links, we mainly utilize TSDW for 2-simplex and 3-simplex. Since the proportion of 2-simplex and 3-simplex is higher than that of higher-order simplices in authentic network scenarios. Formally, the computation of the fraction of 1-simplex in 2-simplex and 3-simplex is summarized as follows:

$$
\begin{aligned}
&TSDWFor2(n_1, n_2) = \\
&\sum_{i=1}^{n} \begin{cases} (\text{len}(simplex_i) - 2) \times 2 \times T_i, & \text{if len}(simplex_i) \geq 3, \\ T_i, & \text{otherwise} \end{cases} \\
&TSDWFor3(n_1, n_2) = \\
&\sum_{i=1}^{n} \begin{cases} C^2_{\text{len}(simplex_i)-2} \times 6 \times T_i, & \text{if len}(simplex_i) \geq 4, \\ 2 \times T_i, & \text{if len}(simplex_i) = 3, \\ T_i, & \text{otherwise} \end{cases}
\end{aligned} \tag{7}
$$

where len(·) denotes the number of nodes in the simplex to be predicted. Here, $simplex_i$ belongs to a subset of that simplex, and T_i is a weight based on the time of occurrence of this simplex, where n represents the length of its subset. The weights are then computed using the geometric mean algorithm with the following formula:

$$
s^{TSDWG}_{(x_1,\ldots,x_k)} = \frac{\left(\prod_{(x_i,x_j)\in k\text{-simplex}} \text{tsdw}(x_i, x_j)\right)^{\frac{2}{k(k-1)}}}{\frac{k(k-1)}{2}} \tag{8}
$$

By applying the TSDW algorithm, we can predict crucial links at the next point by leveraging the learned information from the network backbone. This ensures that the generated obfuscated links align with the temporal dynamics of the network. Furthermore, the TSDW algorithm reduces the overhead of learning by employing local information for link prediction.

2.5. Discriminator

To ascertain the deceptive characteristics of the generated obfuscated network, a spectral centrality measure is employed to identify significant nodes in the higher-order network (Tudisco and Higham, 2021). The evaluation of deception involves calculating the Intersection Similarity (IS) of influential nodes between the authentic and obfuscated networks.

A framework was introduced for computing nonlinear feature vector centrality measures tailored for hypergraphs, which is based on a set of nonlinear functions f, g, ϕ, and ψ, mapping nonnegative real numbers x to nonnegative real numbers y. These functions model the impact of hyperedges on nodes and nodes on hyperedges and capture self-influence. The chosen functions are nonlinear to reflect the complex interactions within the hypergraph. Formally, the framework is expressed through the following equations:

$$
\begin{cases} \lambda \boldsymbol{x} = g\left(BWf(\boldsymbol{y})\right) \\ \mu \boldsymbol{y} = \psi\left(B^{\top} N \varphi(\boldsymbol{x})\right) \end{cases} \quad \boldsymbol{x}, \boldsymbol{y} > 0, \quad \lambda, \mu > 0 \tag{9}
$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are nonnegative vectors of node and hyperedge weights, B is the incidence matrix of the hypergraph, and W, N is the adjacency tensor, and diagonal matrices of edge and node weights, respectively.

Three node-edge eigenvector centrality models, distinguished by the functions f, g, ϕ, and ψ, are employed to validate the effectiveness of *HBB-TSP* from multiple perspectives.

- **LINEAR centrality model**
  1. $f = g = \varphi = \psi = \text{id}$
  2. Essentially equivalent to standard eigenvector centrality, providing a score proportional to the sum of neighboring edge weights.
- **LOG-EXP centrality model**
  1. $f = \text{id}$, $\phi(x) = \ln(x)$, $\psi(x) = \exp(x)$, $g(x) = x^{-1/2}$
  2. Utilizes logarithmic and exponential functions to capture complex interactions, assigning scores based on the weighted sum of neighboring edges.
- **MAX centrality model**
  1. $f = g = \text{id}$, $\phi(x) = x^{\alpha}$, $\psi(x) = \phi^{-1}(x) = x^{1/\alpha}$
  2. Assigns scores based on the maximum weights of neighboring edges focusing on the maximum node centrality.

The "Linear" model favors nodes with many involved hyperedge nodes, while the "Log-Exp" model emphasizes nodes with fewer active hyperedge nodes. The "Max" model, on the other hand, assigns scores based on the maximum weights of neighboring edges.

3. Experimental setup

The *HBB-TSP* framework has been developed to mitigate the effects of active spoofing attacks and to generate obfuscated hyperedges for network topology efficiently and cost-effectively in real-time. This section aims to provide an overview of the framework's effectiveness by presenting the experimental dataset used, outlining the evaluation criteria employed to assess its utility, and identifying the benchmark algorithm implemented to compare link prediction.

3.1. Datasets

To evaluate the efficacy of *HBB-TSP*, we employed two real-world IP traffic datasets in our experiments, each corresponding to a different network topology. One dataset involves a large-scale network, while the other involves a small-scale network. For further details regarding these network typologies, please refer to Table 2.

Table 2
Datasets.

Dataset	Nodes	Edges	Simplices	Unique simplices
D_{ss} (Rossi and Ahmed, 2015)	97	88 934	75 584	15 342
D_{ls} (Rossi and Ahmed, 2015)	34 762	171 403	161 602	31 616

Table 3
Evaluation metrics overview.

Category	Metrics	Description
Overhead	(*i*) Time cost	Time consumption
	(*ii*) Resource cost	Num of obfuscated links
Obfuscation	(*i*) Temporality	PR-AUC
	(*ii*) Obfuscation	IS
Defense	Defensive	FR
	Capability	AL

3.1.1. TECH-IP-TRAFFIC-FLOW(D_{ss})

The dataset sourced from TECH-IP-TRAFFIC-FLOW pertains to a temporal IP network, capturing the flow attributes of IP traffic within a technological context. This dataset serves as a valuable resource for analyzing the behavior of IP traffic flows, discerning trends, anomalies, or patterns, and gaining insights into the operational dynamics of networks within the technological domain. Additionally, it affords opportunities for comparative analysis with other network datasets, facilitating broader insights into network dynamics across diverse domains and categories.

3.1.2. TECH-AS-TOPOLOGY(D_{ls})

The datasets provided within the TECH-AS-TOPOLOGY repository belong to the category of Dynamic Networks, elucidating the topology within the technology domain. In this context, each node typically represents a distinct entity, such as a device, server, or network component within the technological infrastructure. The edges between nodes denote connections, relationships, or interactions among these entities, thereby depicting communication links, data transfers, or dependencies among different components within a technological network.

3.2. Evaluation metrics

To evaluate the performance of *HBB-TSP*, we considered three metrics: *Overhead*, *Obfuscation* and *Defense*. The evaluation includes verifying whether the generated obfuscation links conform to network dynamics, determining if adding obfuscation links alters the order of fundamental network nodes, and assessing the ability to counteract attacks on critical nodes. The evaluation methodology is overviewed by Table 3.

3.2.1. Overhead

The evaluation of overhead encompasses both (*i*) time and (*ii*) resource dimensions: (*i*) Time overhead is assessed through comparative analysis with benchmark algorithms to gauge methodological efficiency. (*ii*) Resource overhead, on the other hand, is regulated by constraints on the number of joined obfuscated links, controlled by distinct μ values corresponding to various network sizes. It is hypothesized that increasing the number of joined obfuscated links will lead to higher resource overhead.

3.2.2. Obfuscation

(*i*) Temporality and (*ii*) obfuscation are two metrics used to evaluate the effectiveness of a discriminator: (*i*) To assess the impact of temporality on the network topology, a link prediction technique employing simplex scoring of the network backbone is utilized. Precision–Recall Area Under the Curve (PR-AUC) measures the accuracy of link predictions made on the network backbone. During this process, the parameter λ in Eq. (1) is constrained to 0.8 to ensure adherence of predicted links to the network's dynamic pattern. (*ii*) Obfuscation assessment involves the addition of obfuscated links indicated by *HBB-TSP* and the baseline algorithm to the network, respectively. Subsequently, the Intersection Similarity(IS) with the original important node sequence of the network is examined. Lower IS values indicate a higher level of obfuscation.

3.2.3. Defense

Resistance to real-world attacks serves as a crucial evaluation criterion for the *HBB-TSP*. We assume that attackers prioritize nodes based on their importance, under the assumption that higher importance correlates with a higher likelihood of attack. Specifically, attackers target 10% of nodes in D_{ss} and 1% of nodes in D_{ls}, with a presumed attack success rate of 50%. The framework's resilience to real network attacks is evaluated by measuring the failed rate of traffic transmission(FR) and the additional latency(AL) required after nodes are attacked. Lower FR and AL indicate better resilience to attacks.

Table 4
Comparison baseline for overhead and obfuscation.

Methods	Temporality	Data models
GraphGAN (Wang et al., 2018)	Static	Embedding matrix
TLP (Li et al., 2018; Goyal et al., 2020; Lei et al., 2019; Chen et al., 2019)	Dynamic	Evenly-sampled snapshot sequence

3.3. Comparison baseline

3.3.1. Overhead and obfuscation

To the best of our knowledge, limited proposed techniques exist for generating obfuscated networks adapted to temporal evolution, particularly when considering overhead constraints. Prior work that aims to learn network temporal evolution for link prediction shares similarities with *HBB-TSP*. In light of this, we compare our generation framework with such predictive approaches. Owing to the specificity of the algorithm, we transform the dataset into a paired dataset. For instance, in the case of an undirected graph, the hyperedge $\{(1,2,3)\}$ is converted into a paired dataset, i.e., $\{(1,2),(1,3),(2,3)\}$. Table 4 presents the essential information regarding benchmark algorithms.

- **GraphGAN**: Initially, node embedding vectors for the original dataset are generated using the node2vec approach. Subsequently, these vectors undergo training using the GraphGAN (Wang et al., 2018) graph representation learning method to minimize the cross-entropy loss between the generator and the discriminator through alternating training. The optimization objective is expressed as:

$$\begin{aligned}&\min_{\theta_G}\max_{\theta_D} V(G,D)\\&=\sum_{c=1}^{V}\Big(\mathbb{E}_{v\sim p_{\text{true}}(\cdot|v_C)}\left[\log D\left(v,v_c;\theta_D\right)\right]\\&\quad+\mathbb{E}_{v\sim G(\cdot|v_c;\theta_G)}\left[\log\left(1-D(v,v_c;\theta_D)\right)\right]\Big)\end{aligned}\tag{10}$$

  where positive samples ("v") and negative samples ("v_c") are used. Using the discriminator D and its parameter θ_D, we compute $D(v,vc;\theta_D)$ and $1-D(v,vc;\theta_D)$ and add them. The generator G minimizes this function while the discriminator D maximizes it. The discriminator focuses on learning important edge features using two disconnected nodes generated in a spanning tree as negative samples. After training, we use the embedding vectors for subsequent node link prediction.
- **TLP**: The present study proposes a neural network-based approach for link prediction that treats the dynamics of the network as a sequence and learns higher-order features of the network for accurate prediction. To facilitate the direct application of the

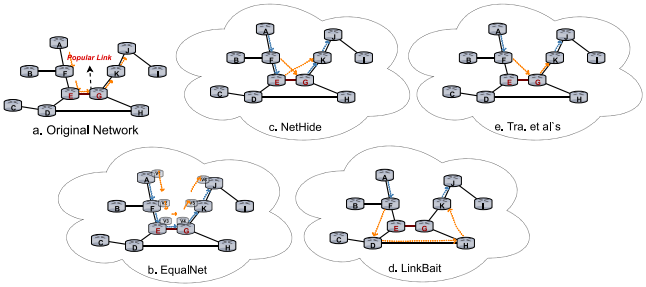


Fig. 3. The state-of-art comparison method for topological confusion link diagram.

algorithm, the network's history is divided into a sequence of uniformly sampled snapshots based on the actual size of the network. The proposed models can identify patterns in the evolution of network dynamics, enabling the prediction of links for subsequent phases.

1. **DDNE** (Li et al., 2018) incorporates a nonlinear transformation by training a twin neural network that projects pairs of nodes into the same low-dimensional space.
2. **Dyngraph2vec** (Goyal et al., 2020) is a technique for learning graph representations that map different properties of graphs to vector spaces.
3. **GCN_GAN** (Lei et al., 2019) is a method that combines the strengths of Graph Convolutional Networks (GCN), Long Short-Term Memory (LSTM), and Generative Adversarial Networks (GAN) to improve the accurate prediction of link evolution in weighted dynamic networks.
4. **E_LSTM_D** (Chen et al., 2019) utilizes the Encoder-Long Short-Term Memory-Decoder (E_LSTM_D) architecture to address potential relationships between nodes in dynamic networks in an end-to-end manner.

3.3.2. Defense

To demonstrate *HBB-TSP*' defensive capability, we introduce several obfuscation network generation methods for comparative analysis under the "defense" metric: EqualNet (Kim et al., 2022), NetHide (Meier et al., 2018), LinkBait (Wang et al., 2017), and Trassare et al.'s solution (Trassare et al., 2013). Fig. 3 provides illustrative examples of these methods. In these examples, the original path is depicted as $A \rightarrow F \rightarrow E \rightarrow G \rightarrow K \rightarrow J$, while the obscured paths represent the obfuscation methods applied to the traffic path. Subsequently, we will detail each of these methods with the aid of examples.

- **EqualNet** (Kim et al., 2022) achieves a balanced distribution of traffic in the network by introducing virtual nodes and virtual links. Specifically, in Fig. 3, EqualNet confuses traffic paths by introducing virtual nodes V_1, V_2, V_3, V_4, V_5, and V_6 to balance the flow between real nodes A, F, E, G, K, and J (especially E and G).
- **NetHide** (Meier et al., 2018) constructs a virtual topology based on physical network topology graphs and forwarding behavioral specifications, allowing for a flexible balance between security and availability. The fundamental principle underlying NetHide's approach is to permit the appearance of nodes E and G in traceroute responses In Fig. 3, provided they do not occur consecutively. To prevent consecutive occurrences, NetHide adjusts the Time-To-Live (TTL) field of all traceroute packets passing through node E to expire at node G (and vice versa), effectively skipping one of these nodes. Consequently, the adversary may observe either the path $A \rightarrow F \rightarrow E \rightarrow K \rightarrow J$ or $A \rightarrow F \rightarrow G \rightarrow K \rightarrow J$ in their traceroute responses.
- **Linkbait** (Wang et al., 2017) introduces a strategy for selective traffic rerouting to effectively obstruct LFAs by diverting traffic onto decoy links, achieving a high detection rate of compromised devices. Fig. 3 illustrates a potential virtual topology that Linkbait could reveal to conceal link E–G. Assuming the bait link consists of two links: C–D and D–H, chosen based on their proximity to link E–G and adequate available bandwidth, Linkbait selects the most suitable link within the bait link (e.g., link D–H) based on its current latency. Consequently, if Linkbait is utilized, the adversary would receive a traceroute response containing the sequence $A \rightarrow F \rightarrow D \rightarrow H \rightarrow K \rightarrow J$.
- Trassare et al. (2013) proposed strategies involving random obfuscation through the introduction of randomly generated IP addresses to introduce noise, and intelligent obfuscation by creating plausible pseudo-network topologies. In contrast to NetHide, where all routers (except for nodes E or G) respond to expired tracing packets, Trassare et al.'s solution ensures that tracing packets are consistently answered by the ingress router (i.e., node A). Fig. 3 illustrates a potential virtual topology that Trassare et al.'s solution could generate to conceal link E–G, achieved by employing a single virtual link. If this defense mechanism is implemented, the adversary would receive the traceroute response $A \rightarrow F \rightarrow G \rightarrow K \rightarrow J$.

4. Experimental results

We use *HBB-TSP* to generate the obfuscated networks for both D_{ss} and D_{ls} network topologies mentioned in Section 3.1. In addition, we compare our generation with five existing methods introduced in Section 3.3. The experiments are performed on the following hardware platforms: Intel(R) Xeon(R) Platinum 8255C CPU @ 2.50 GHz, and RTX 3080 (10 GB). Configurations for the deep learning model include PyTorch 1.10.0, Python 3.8 (Ubuntu 20.04), and CUDA 11.3.

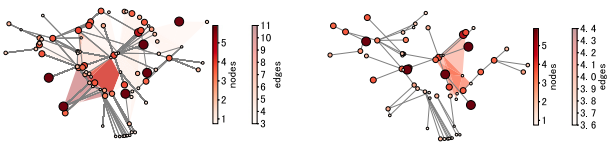
In this section, we aim to answer the following questions:

- What are the network structure characteristics used for obfuscated network generation (4.1)?
- How does *HBB-TSP* perform in accommodating temporal variations in the network (4.2)?
- To what extent does the *HBB-TSP* algorithm minimize resource consumption (4.3)?
- How does *HBB-TSP* perform in generating obfuscated links with maximum deception (4.3)?
- Apart from link generation, what other factors affect the quality of the obfuscated network (4.3)?
- How effective is *HBB-TSP* in defending against real attacks (4.4)?

Note that we will adhere to the experimental procedure outlined in Section 2.1 for generating the obfuscated network topology. However, the snapshot pre-processing module is not delineated in the subsequent experimental elaboration. Instances where multiple nodes interact extensively within a limited timeframe will be treated as simplices.

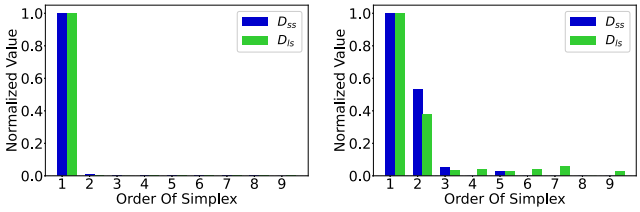
4.1. The evaluation for network backbone extraction module

In this section, we discuss the impact of extracting backbones from the original network on obfuscated network generation. which is conducted to assess the effect of predicted obfuscated links in *HBB-TSP*, Specifically, we compare the network topology structure of D_{ss} in Fig. 4, where each node represents a device. The utilization of D_{ss} aims to enhance visualization clarity, and the findings are similar to those of D_{ls} presented in this study. As expected, despite filtering out 96.3% of the links, the overall structure of the network remained intact. Pre-filtered networks tend to exhibit a larger number of 1-simplices, and an excess of high-order or low-order simplices may hinder the discernment



(a) Before filtering (b) After filtering

Fig. 4. Visualization of network structure with backbone network extraction.



(a) Before filtering (b) After filtering

Fig. 5. Comparison of simplicial structure distribution in backbone network extraction.

of critical information, as discussed in Section 2.5. Consequently, the results demonstrate that *HBB-TSP* effectively removes redundant connections through backbone extraction, thereby enhancing the distribution of nodes within each hyperedge.

For further knowledge, the detailed distribution of simplices of various orders before and after filtering is presented in Fig. 5. To simplify the structure analysis and prediction work and consider the dominant number of 1-simplex, we used the min–max method for data normalization. It is evident that backbone extraction notably amplifies the weights of simplices of all orders except for 1-simplices in both D_{ss} and D_{ls} from Fig. 5. Moreover, the distribution within the original network remains consistent in both scales. Take D_{ss} as an example, the weight variation interval for the 2-simplex (order 2 bar) was [0.05, 0.52], showing an increasing trend. Similarly, for the 3-simplex (order 3 bar), the interval was [0(approximation), 0.06], with weights of larger-order simplex also exhibiting an increase after filtering. In D_{ls}, the interval for weight variation of the 2-simplex was [0.05, 0.38], again indicating an increasing trend. For the 3-simplex, it was [0(approximation), 0.4], indicating a similar upward trend. Similarly, the larger-order simplex quantities showed an increasing trend as well.

Both aforementioned analyses are aimed at illustrating the efficacy of the network backbone extraction strategy employed by *HBB-TSP* in equilibrating the number of nodes within hyperedges. Subsequently, we will empirically demonstrate its tangible impact on network dynamics.

Fig. 6 compares the IS of different lengths in the sequences of significant nodes in the network when links are removed. These links were filtered by SVH or selected randomly. Three node sequence identification methods introduced in Section 2.5 were used to identify significant node sequences in the network after removing the links and computing their IS with the original network. The results show that removing SVH-filtered backbone edges leads to a significantly lower IS between the obfuscated and original networks than random link removal. This was found in both networks. Overall, apart from the linear algorithm, the sequences of significant nodes computed by the other two algorithms exhibit changes after removing the backbone links. We attribute this phenomenon to the bias of the linear algorithm towards higher-order simplices, whereas the backbone links we removed primarily consisted of 1-simplices. Hence, the linear algorithm struggles to yield significant results especially in D_{ss}, whereas the impact is pronounced for the log-max algorithm. Additionally, random edge removal is likely

Table 5
Runtime comparison (in **ms**).

Dataset	TSDW	SVH&TSDW
D_{ss}	456.39	25.18
D_{ls}	4787.53	3372.57

to eliminate edges associated with higher-order links that are crucial for the linear algorithm, thereby resulting in better performance compared to SVH. Specifically, in D_{ss}, the IS for removing SVH-filtered links is 7% higher than removing random links, according to the linear algorithm when comparing the lowest IS achievable, but the IS value for the max algorithm is 15% lower, and the IS value for the log-exp algorithm is 48% lower. Considering D_{ls}, removing SVH-filtered links is 10% lower than removing random links for the linear algorithm. For the max algorithm, the IS for deleting SVH-filtered links is 15% lower, while for the log-exp algorithm, the IS for deleting SVH-filtered links is 39% lower than deleting random links. The results demonstrate that removing SVH-filtered links is more effective in reducing the IS of necessary node sequences, and the IS decreases with the number of edges removed, as expected.

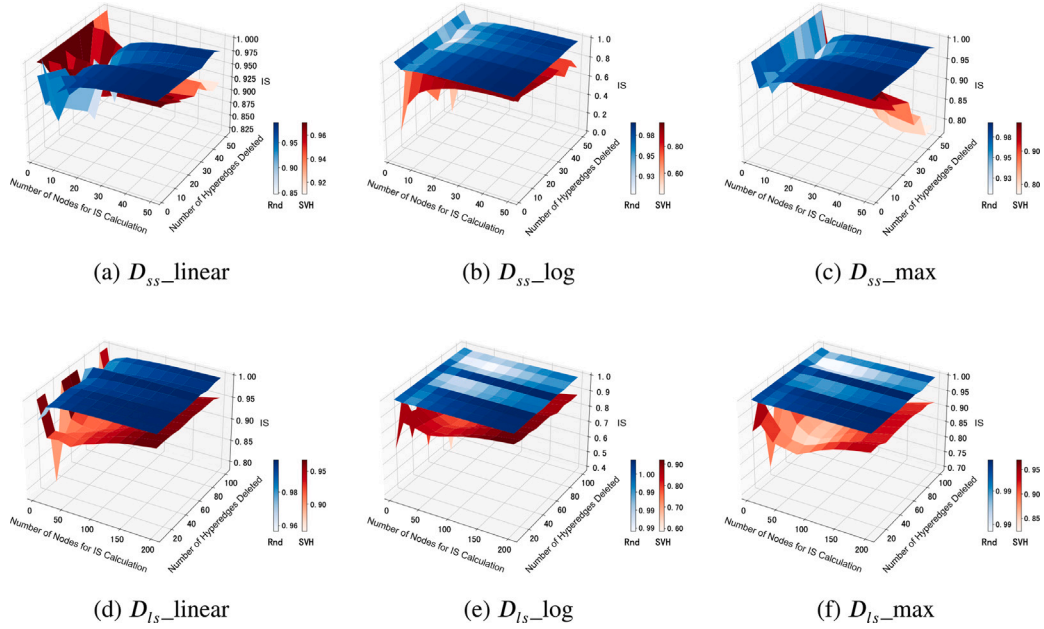
4.2. The evaluation for simplicial closure prediction module

This section aims to provide empirical evidence that the obfuscated links generated by link prediction through the backbone network are consistent with temporal. Specifically, we demonstrate the accuracy of the simplicial closure prediction through the backbone, which is shown in Fig. 7. The figure displays the prediction results using backbone and original networks with varying proportions of training and test sets, where the training set represents historical data and the test sets embody novel network data. Evidently, the precision–recall area under the curve (PR-AUC) of closure prediction with backbone extraction surpasses that without backbone extraction in both D_{ss} and D_{ls}. For instance, considering D_{ss}, the PR-AUC of closure prediction after extracting the network backbone by SVH at training set share 0.8 reaches 0.97, which is consistent for 2-simplex and 3-simplex, while the accuracy of closure prediction without extracting the network backbone is only 0.4 for 2-simplex and 0.2 for 3-simplex. Similar results in D_{ls}, with the 2-simplex achieving an accuracy of 0.97 and the 3-simplex of 0.78 at a training set share of 0.8 after extracting the network backbone, whereas without extracting the network backbone the 2-simplex is only 0.62 and the 3-simplex is 0.24. As expected, the accuracy of the closed prediction increases as the proportion of the training set continues to increase.

Based on the results shown in Fig. 7, it can be illustrated that we satisfy the temporal requirement in Section 3.2, which is that the PR-AUC should be greater than 0.8 (λ). Therefore, we consider the simplices generated by *HBB-TSP* to be plausible, which is crucial for the quality of the obfuscated links generated.

Next is the evaluation of time cost, which is presented in Table 5, we compare the chronological efficiency of various prediction strategies that use backbone networks, which can be demonstrated by the comparison of running times. Specifically, we compare the execution time for link prediction before and after extracting the network backbone and find that extracting the network backbone can reduce the overall execution time. Such as the runtime decreased from 456.39 ms to 25.18 ms for D_{ss}, the runtime of D_{ls} is reduced from 4787.53 ms to 3372.57 ms. Note that deep learning-based link prediction techniques are not included in the comparison, as they depend on comprehending the entire network topology, which leads to variable execution times influenced by factors such as GPU configuration. Therefore, algorithms based on deep learning may incur higher time costs.

Note that although the above only compares the time cost of a single obfuscated network generation experiment, it also demonstrates that *HBB-TSP* does not require significant computational resources in



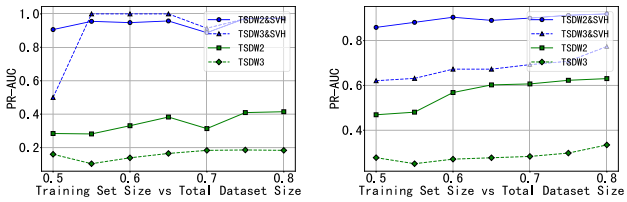
(a) D_{ss}_linear (b) D_{ss}_log (c) D_{ss}_max

(d) D_{ls}_linear (e) D_{ls}_log (f) D_{ls}_max

Fig. 6. Comparative analysis of IS across significant nodes: Random edge deletion vs. SVH backbone edge deletion.

Table 6
PR-AUC comparison with baseline.

Algorithms	Dataset		Algorithms		Dataset	
	D_{ss}	D_{ls}			D_{ss}	D_{ls}
E_LSTM_D (Chen et al., 2019)	0.70	0.79	Dyngraph2vec (Goyal et al., 2020)		0.45	0.53
DDNE (Li et al., 2018)	0.58	0.69	GraphGAN (Wang et al., 2018)		0.60	0.72
GCN_GAN (Lei et al., 2019)	0.39	0.45	TSDW	For2	**0.97**	**0.90**
				For3	**0.97**	0.78



(a) D_{ss} (b) D_{ls}

Fig. 7. PR-AUC comparison: Link prediction via network backbone vs. original network.

a continuous obfuscation process for the following reasons: (*i*) *HBB-TSP* generates obfuscated networks using historical traffic paths of the network, which are easier to collect and continuously update compared to real-time traffic paths. (*ii*) While an excessive amount of historical traffic data can increase the computational resource consumption of *HBB-TSP*, experiments have shown that an appropriate amount of historical traffic data is sufficient to generate effective obfuscated links. Older data tends to have lower weight and may not provide substantial support for *HBB-TSP*. Therefore, by controlling the amount of historical traffic data, the resource cost of generating obfuscated networks with *HBB-TSP* is manageable. (*iii*) The obfuscated networks generated by *HBB-TSP* are effective over a period rather than at a specific moment, thus spreading the resource cost over a longer duration. In conclusion, although *HBB-TSP* requires continuous updates to maintain effective obfuscated networks, controlling the amount of historical data can effectively manage its computational resource consumption.

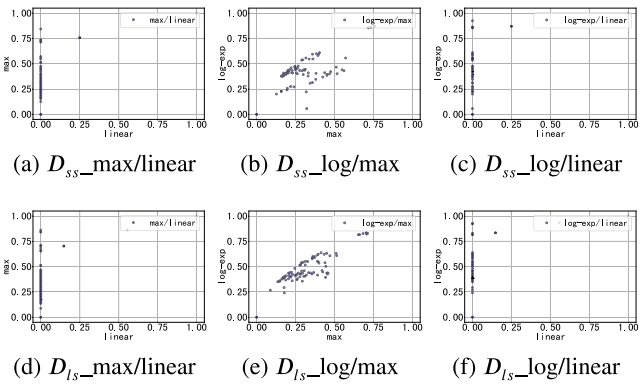
Finally, the accuracy of link prediction using the *HBB-TSP* algorithm was compared with other benchmark algorithms shown in Table 6. Note that the rate of the train set is 0.8 there, and it can be seen that overall the TSDW algorithm obtains higher accuracy than all other prediction algorithms, such as D_{ss}, the results showed that the accuracy of *HBB-TSP* in predicting 2-simplex was 0.97, and the accuracy in predicting 3-simplex was 0.90, surpassing different prediction algorithms. Note the accuracy here is independent of the algorithms' performance. Since only links associated with important nodes are extracted as points in the network topology to ensure that the algorithm is learning the important structures of the network. Moreover, the network topology is delineated for only five-point snapshots based on the specific amount of data in the dataset, which may affect the prediction results, especially for GCN-GAN. Similarly, considering the D_{ls}, where the accuracy of *HBB-TSP* in predicting 2-simplex was 0.97, and the accuracy in predicting 3-simplex was 0.78.

4.3. The evaluation for generating obfuscated networks and validation model

In this section, we aim to establish the efficacy of the *HBB-TSP* algorithm. To achieve this objective, we will compare this algorithm with select conventional link prediction algorithms introduced in Section 3.3. Specifically, we will perform a quality assessment of obfuscated network generation under two strategies, Additionally, a supplementary experiment is introduced to evaluate the positioning of the added obfuscated links.

4.3.1. Defense against analyzers with sequence generator

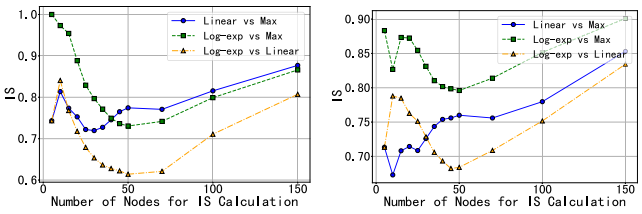
we briefly demonstrate the validity of our evaluation method before evaluating the obfuscation network. Three algorithms are used to achieve this: linear, max, and log-exp. Fig. 8 illustrates the interrelation among these algorithms, providing a normalized comparison of scores



(a) D_{ss}_max/linear (b) D_{ss}_log/max (c) D_{ss}_log/linear

(d) D_{ls}_max/linear (e) D_{ls}_log/max (f) D_{ls}_log/linear

Fig. 8. Visualization of node centrality scores across various centrality algorithms.



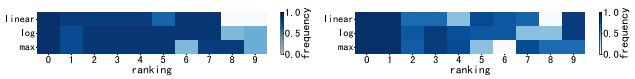
(a) D_{ss} (b) D_{ls}

Fig. 9. The IS of different algorithms for calculating important node sequences.

computed for the same node. It is observed that a discernible linear relationship exists between the log-exp and max algorithms, whereas linear and these two algorithms do not exhibit linearity, but rather display a positive correlation. Specifically, after normalizing the scores computed by the nodes in both networks by the linear algorithm, only 2.6% and 0.29% of the nodes have scores of 0.5 and above, whereas the scores computed by the max algorithm are normalized to 24.1% and 44.3% of the nodes. log-exp algorithm has similar results, with 16.7% and 42.9% of the nodes.

The linear algorithm prioritizes hyperedges with a greater node count, which may lead to overlooking essential nodes. On the other hand, the max algorithm exhibits a more uniform distribution of node identification across the network. Comparing the linear and log-exp algorithms, the scores identified by the log-exp algorithm tend to be in the mid-to-upper range, suggesting that only a few nodes are closely connected in most authentic networks. However, cases where multiple nodes are tightly connected should not be ignored. Moreover, we tested the network's suitability for our analyses by conducting a comparative study between two datasets with different numbers of nodes in the expanded network. The comparative results between the two datasets remained similar, confirming the network's suitability for our analyses.

The analysis depicted in Fig. 8 explores the bias inherent in the computation of node scores across three algorithms. Furthermore, a comparison of the resultant node rankings is illustrated in Fig. 9. We conducted experiments to assess the IS of significant node sequences derived from the linear, max, and log-exp algorithms. The findings reveal a notable degree of similarity among the node rankings computed by these algorithms. In both D_{ss} and D_{ls}, it is evident that the IS of log-max surpasses that of linear-max, as well as linear-log. For instance, in D_{ss}, as the number of nodes (n) used for IS calculation increases from 5 to 10, the IS of significant node sequences computed by the linear and max algorithms gradually increases from 0.74 to 0.82 before rising to 0.88, indicating high relevance. Similarly, the IS of sequences computed by the log-exp and max algorithms also exhibit a high degree of relevance, achieving 1.0 when n is 5, decreasing to 0.72 at n of 50, but gradually rising to 0.87. Moreover, the IS of sequences computed



(a) D_{ss} (b) D_{ls}

Fig. 10. Comparative analysis of top ten important nodes of different algorithms. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

by the log-exp and linear algorithms increases from 0.74 to 0.84 as n rises from 5 to 10 but subsequently decreases to 0.61 at n of 50, followed by a gradual increase, ultimately reaching 0.80.

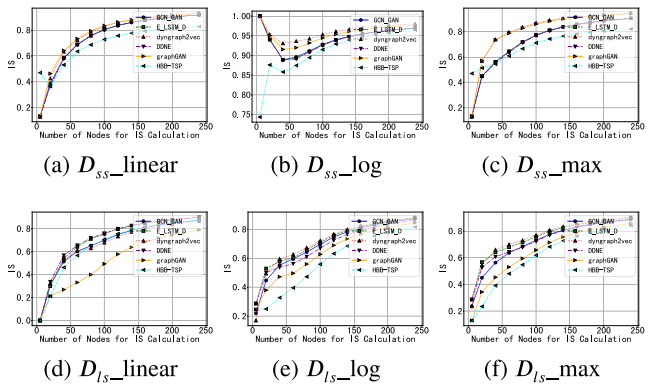
In D_{ls}, the IS of significant node sequences computed by the linear and max algorithms exhibit an upward trend, increasing from 0.72 to 0.85. Conversely, the IS of sequences computed by the log-exp and max algorithms display a strong correlation, consistently surpassing the other two algorithms, reaching 0.88 at n of 5, and remaining above 0.8, particularly reaching 0.90 at n of 150. Furthermore, the IS of sequences computed by the log-exp and linear algorithms increases from 0.72 to 0.78 as n increases from 5 to 10, but then decreases to 0.67 at n of 50, similar to the trend observed in D_{ss}, and reaches 0.83 when n increases to 150.

Notably, the high IS of log-exp and max in both networks indicates that these algorithms exhibit similarity in identifying crucial nodes, possibly attributed to the two network topologies we used are both heavily weighted towards low-order simplices, and log-exp itself is more weighted towards low-order simplices, so it does not show a particularly pronounced bias, and the result is very similar to that of the max algorithm, which is also reflective of our network topology. However, the IS of linear-max is relatively higher than log-linear. This result is also in line with expectations, as the log-exp and linear algorithm is biased in the opposite direction, which makes the difference in the final node rankings even greater.

The preceding discussion provides an overview of the overall similarity in node rankings. Now, we proceed to visualize the repetition frequency for the top important nodes. In Fig. 10, we conduct a comparative analysis of the top ten significant nodes identified by three algorithms to assess the consistency of significant node analysis. The results are depicted in a heat map, where color intensity signifies the degree of repetition, and each column represents the outcomes obtained by different algorithms for computing important node sequences. Specifically, in D_{ss}, only the ninth and tenth significant nodes identified by the linear algorithm do not appear in the results of the other algorithms. Conversely, the significant nodes computed by the log-exp and max algorithms overlap with those of all other algorithms, with only two nodes exhibiting lower repetition rates. This finding aligns with our preceding analysis. Similarly, D_{ls} exhibits analogous results, wherein despite its larger size, the duplication rate of calculated important nodes remains exceedingly high, with nearly all algorithms yielding identical important node selections. From these observations, we infer the credibility of the algorithms employed for important node evaluation. Despite inherent biases among the three algorithms, they consistently identify particularly significant nodes, a critical aspect for our subsequent evaluation of obfuscated network quality.

4.3.2. Analysis of obfuscation generator and discriminator

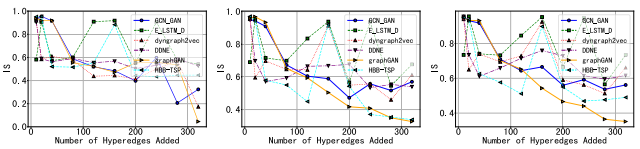
In prior experiments, an exhaustive analysis of network topology revealed a prevalence of hyperedges with a small node count compared to those with a larger node count. Additionally, it was established that the backbone network encapsulates pivotal information about the entire network. The evaluation of a link prediction model based on local information proved highly reliable, ensuring the generated obfuscated links are more adept at misleading potential attackers. Utilizing diverse algorithms to identify sets of important node sequences showcased the efficacy of the obfuscated network in concealing critical network information.



(a) D_{ss}_linear (b) D_{ss}_log (c) D_{ss}_max

(d) D_{ls}_linear (e) D_{ls}_log (f) D_{ls}_max

Fig. 11. Comparison of IS for different lengths of significant node sequences with a fixed number of obfuscated links.



(a) D_{ls}_linear (b) D_{ls}_log (c) D_{ls}_max

Fig. 12. Comparison of IS of fixed-length significant node sequences with different numbers of obfuscated links.

The current investigation centers on enhancing obfuscated network generation by incorporating links that emulate real network structures. To achieve this objective, partially interchangeable algorithms are amalgamated in the generation process. This research endeavors to assess the generation of obfuscated networks under various prediction algorithms to validate the rationale behind the algorithm employed in *HBB-TSP*. Two types of experiments were conducted to address this aim. The first involved fixing the number of added obfuscated links (m) and comparing the IS between the generated obfuscated network and the top n important node sequences of the actual network. The second experiment aimed to fix the number of important node sequences that required comparison (n) and to compare the IS when adding different numbers of obfuscated links (m). The subsequent sections delve into the discussion of these experiment results.

As shown in Fig. 11, which reports a technical analysis of the IS of significant nodes with variable sequence length n in the generated obfuscated network, while keeping the number of added obfuscated links (m) fixed.

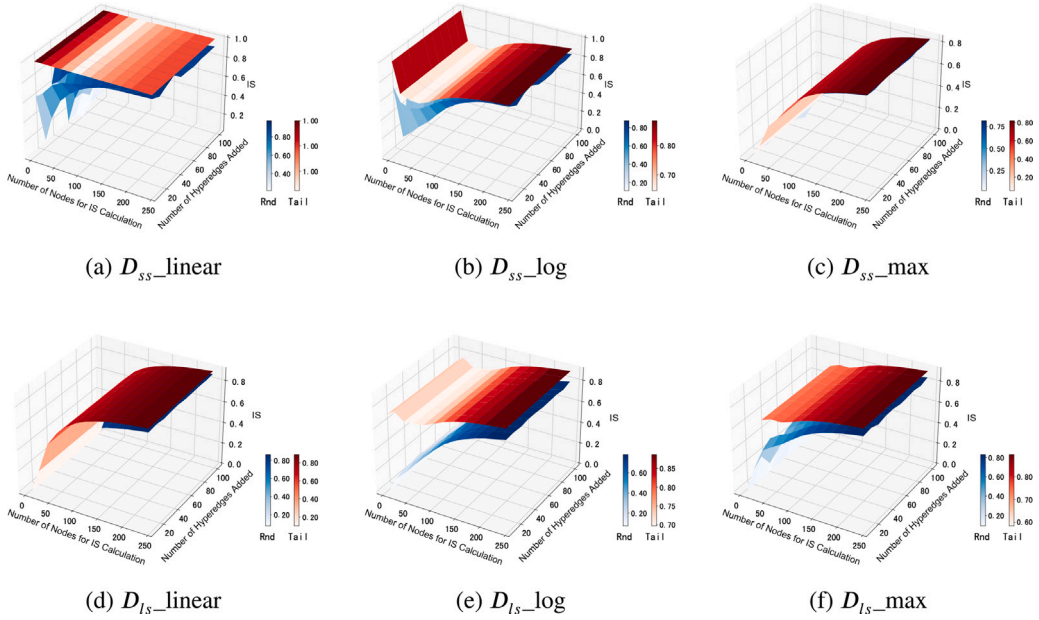
Initially, we discuss the choice of the number of obfuscated links (m) in this experiment, which is set to 3 in D_{ss} and 100 in D_{ls}. The value of m in D_{ss} is so low mainly because due to the limitation of the network size and the requirement of temporal, we predicted the number of obfuscated links to be only 3, Despite this constraint, the results obtained by *HBB-TSP* with a limited number of added obfuscated links are still considerable, also due to the network size. Specifically, for the linear algorithm, Specifically, regarding the linear algorithm, *HBB-TSP* does not consistently exhibit a clear advantage. Especially before n reaches 25, the obfuscated network generated by *HBB-TSP* performs the worst compared to other algorithms. However, with increasing n, *HBB-TSP* gradually demonstrates some advantages, consistently yielding optimal results when n reaches 150. This observation can be attributed to the bias of the linear algorithm and the relatively low order of the added obfuscated links, limiting the reflection of *HBB-TSP*'s strong advantage. This phenomenon can be explained in the results of the log-exp, which can be seen in the algorithm's results that the *HBB-TSP* shows a powerful advantage, regardless of n values, the similarity computed by *HBB-TSP* between the obfuscated network and the original network is the lowest. For instance, at $n = 5$, the similarity between the obfuscated network and the original network is 0.74, albeit higher than the linear algorithm. This disparity may stem from the linear algorithm's identification of only a few highly important nodes, causing a significant change in node scores and final ranking results upon adding a small number of obfuscated links. Considering the results of the max algorithm, *HBB-TSP* exhibits a similar trend to the linear algorithm, showcasing an advantage primarily at later stages. This suggests that for small networks, where the number of obfuscated links that can be added is limited, *HBB-TSP* may not consistently outperform other algorithms in protecting particularly important nodes. However, it remains feasible to safeguard important nodes.

Moving to the results in D_{ls} shown in Fig. 11, *HBB-TSP* does not demonstrate a significant advantage over the linear algorithm. Conversely, the outcomes of the GraphGAN algorithm are more noteworthy, possibly due to its static prediction nature, which learns from the real network, thereby yielding high prediction accuracy and prioritizing predicted links. Nonetheless, *HBB-TSP* still induces considerable changes in the significant nodes calculated by the linear algorithm. But besides the linear algorithm, the log-exp and max algorithms get the best results for *HBB-TSP*, no matter how much n is taken. Specifically, at $n = 50$, the IS of *HBB-TSP* is 0.37, increasing to 0.58 at $n = 100$, further rising to 0.68 at $n = 150$, and finally reaching 0.8 at $n = 250$. On the other hand, for the max algorithm, *HBB-TSP* achieves optimal results with an IS of 0.12 at $n = 5$, 0.42 at $n = 50$, 0.6 at $n = 100$, and maintaining this optimal result until reaching 0.8 at $n = 250$.

From the above experiments conducted on networks of varying sizes, we infer that the obfuscated links generated by *HBB-TSP* effectively alter the critical structure of the network, as evidenced by comparisons with other algorithms. Our algorithms exhibit inconsistent results in calculating the critical structure with varying favoritism, particularly evident in the case of the linear algorithm. However, despite suboptimal performance, *HBB-TSP* demonstrates overall effectiveness. Notably, the most pronounced degree of change in the sequence of significant nodes is observed in the results produced by the linear algorithm. While *HBB-TSP* performs comparatively better in other algorithms, the degree of change is less pronounced.

Next up is the second experiment. As observed in Fig. 12, we systematically explore the impact of different numbers of obfuscated links m under a fixed number of significant nodes ($n = 40$). Note that we did not conduct experiments with D_{ss} in this scenario. This decision was made due to the previous demonstration of the effectiveness of *HBB-TSP* in small-scale network obfuscation and only predicted single-digit obfuscated links, considering the limitations imposed by network size and temporal requirements. In this experiment, the first 40 nodes are designated as vulnerable nodes, and the performance of the *HBB-TSP* algorithm is evaluated by introducing different numbers of obfuscated links.

Firstly we observe the experimental results as a whole, we find that *HBB-TSP* is not considered to be good in the linear algorithm. Among all the values of m that were experimented with, *HBB-TSP* demonstrates the most advantage only in the case of $m = 40$, 60, and 240. This outcome aligns with the findings from the previous two experiments. Additionally, we observe a correlation between the quality of the obfuscated network and the number of obfuscated links added. However, it is important to note that adding more obfuscated links does not necessarily translate to a better-quality obfuscated network. For instance, in the results obtained from the log-exp and max algorithms, *HBB-TSP* generally maintains a superior ranking. However, at an m value of 160, the results of *HBB-TSP* take a turn, with the IS increasing significantly in both the log-exp and max scenarios, from 0.42 to 0.86 and from 0.45 to 0.90 respectively. Except for this point, the IS remains relatively low for all other m values compared to the other algorithms, particularly in the range of $40 - 120$, where *HBB-TSP*



(a) D_{ss}_linear (b) D_{ss}_log (c) D_{ss}_max

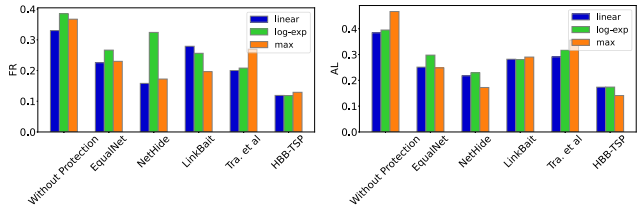
(d) D_{ls}_linear (e) D_{ls}_log (f) D_{ls}_max

Fig. 13. IS comparison of important node sequences when network obfuscated links are added to the network at different times. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

exhibits optimal results compared to the other algorithms. Furthermore, we note that the results obtained from GraphGAN are notably impressive, maintaining a consistent decreasing trend with increasing m. Although the results may not surpass those of *HBB-TSP* in certain cases, they exhibit greater stability. We attribute this phenomenon to the characteristics of the *HBB-TSP* hypergraph algorithm. The simplex generation in this network topology may not accurately reflect the underlying network structure, potentially leading to a network backbone extraction that is insufficiently comprehensive.

Based on the above experimental results, it is evident that *HBB-TSP* can achieve optimal performance when the number of obfuscated links is carefully controlled. However, this performance appears to be somewhat unstable. Therefore, the process of obfuscated network generation should not be approached as a rigid, one-time endeavor. In practical applications, after generating the obfuscated network topology, it is imperative to compare it with the original network and continuously adjust the number of obfuscated links or how they are added. This iterative approach is essential to ensure the attainment of optimal obfuscated network quality.

The above two experiments evaluated the obfuscation effectiveness of the generated obfuscated links from two distinct perspectives. One experiment aimed to explore the impact of introducing obfuscated links into authentic networks at different time points on the final obfuscated network's quality, as illustrated in Fig. 13. Specifically, the study sought to compare the effectiveness of randomly adding obfuscated links versus adding them at the end of the network formation process. The results suggest that random addition of obfuscated links is more effective, regardless of the network size. Notably, the red region (Tail) consistently exhibited higher IS values compared to the blue region (Rnd), observed in both D_{ss} and D_{ls}. For instance, in D_{ss}, the linear, max, and log-exp algorithms achieved the lowest IS values of 0.23, 0.1, and 0.15, respectively, with randomly added obfuscated links, whereas these values were 1.00, 0.12, and 0.65, respectively, when links were added at the tail. Similarly, in D_{ls}, the lowest IS values for the linear, max and log-exp algorithms were 0.09, 0.10, and 0.08 with random placement, compared to 0.12, 0.59, and 0.69 with tail placement. The study concludes that the random placement strategy is more effective than the tail placement strategy for generating obfuscated links. This strategy reduces the likelihood of attackers discovering obfuscated links and more effectively conceals important information within obfuscated networks. These findings carry implications for designing secure and resilient networks.



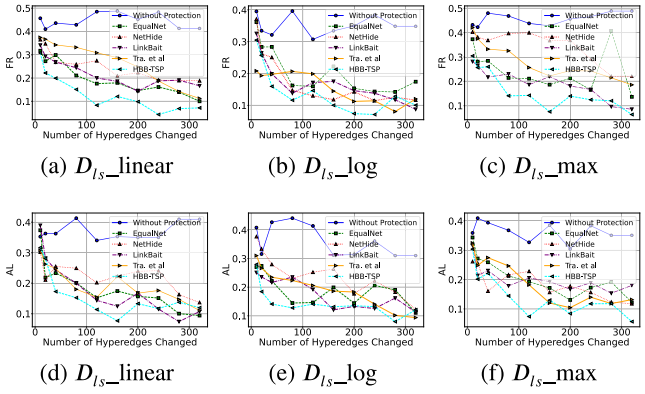
(a) D_{ss}_FR (b) D_{ss}_AL

Fig. 14. FR and AL across different algorithms in D_{ss}.

4.4. The evaluation of simulated node attacks in network systems

Obfuscated networks once generated should be applied in real-world scenarios to protect our network. Therefore, we evaluated the effectiveness of our generated obfuscated networks against real attacks. Specifically, we compared the impact of obfuscated networks generated by the *HBB-TSP* and various obfuscation network generation strategies on FR and AL after experiencing node attacks. Note that while other methods predominantly alter existing links (i.e., redirection) rather than adding new links, the number of altered or added links in the comparative experiments remains consistent for fair experimentation, rather than modifying the entire network's links.

First, we conducted an evaluation on D_{ss}. However, due to the limited number of obfuscation links generated by *HBB-TSP*, we were unable to observe the impact of increasing links on the reduction of FR and AL. Fig. 14 shows the FR and AL under different algorithms when three links are changed. Without any topology obfuscation mechanisms, the FR and the AL are relatively high when attackers target key nodes. Nevertheless, these values remain between 0.3 and 0.5. With the application of methods such as EqualNet, NetHide, LinkBait, and Tra. et al. the FR and AL decrease to approximately 0.2, with NetHide performing the best, reducing the values to below 0.2. The application of *HBB-TSP* results in both the FR and AL being around 0.1, indicating



(a) D_{ls}_linear (b) D_{ls}_log (c) D_{ls}_max

(d) D_{ls}_linear (e) D_{ls}_log (f) D_{ls}_max

Fig. 15. FR and AL across different algorithms in D_{ls}.

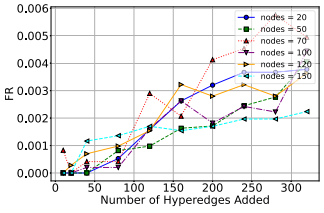


Fig. 16. FR for normal users in D_{ls}.

that *HBB-TSP* provides the best improvement in reducing FR and AL in small-scale networks.

Fig. 15 compares the effects of FR and AL on the D_{ls}. The results show that the attacker's targeting of inferred key nodes leads to relatively high FR and AL when no topology obfuscation mechanism is used. However, due to the limited number of attacked nodes, FR and AL remain slightly below 0.5 but above 0.3, demonstrating relative stability. The methods of EqualNet, NetHide, LinkBait, and Tra. et al. reduce FR and AL to slightly above 0.1 when applied to the network. The application of *HBB-TSP* achieves a faster reduction in FR and AL as the number of altered super edges increases, ultimately reducing them to below 0.1. This demonstrates that *HBB-TSP* effectively counters attacks on critical nodes by adding influential obfuscated links generated from the network backbone. It outperforms the aforementioned methods that directly modify real links within limited cost constraints. We attribute this to *HBB-TSP*'s consideration of generating the most obfuscated links, whereas other methods tend to apply continuous changes. Consequently, when a large number of links are altered, *HBB-TSP* does not show significant superiority over other algorithms. However, It becomes more pronounced when the number of altered links is limited.

Due to our current algorithm's inability to precisely target attackers while exposing the obfuscation network, and considering that legitimate users typically aim to send traffic via the shortest path in network topologies, we opted for an evaluation method where any two nodes (simulating normal users and their respective services) attempt to find the shortest path through the obfuscation network. The metric used for this evaluation is FR when specifying the sending path. Considering the number of obfuscated links and nodes, we conducted the comparative experiment on D_{ls}, as shown in Fig. 16. In this figure, $nodes$ represent the number of nodes simulating mutual traffic transmission, and FR indicates the failed rate with varying numbers of added obfuscation links. As expected, FR shows an upward trend with the addition of obfuscation links. However, within controlled limits, we observed that FR does not exceed 1% for any number of nodes. We attribute this to the fact that the obfuscation links we added are not traditional pairwise links but hyperedges consisting of multiple edges. Therefore, the added obfuscation links, when converted to pairwise links, include redundant links. Consequently, even with a large number of added obfuscation links, the actual number of unique pairwise links is limited. This demonstrates that the impact of the *HBB-TSP*-generated obfuscation network on the benign functions used by normal users is negligible.

5. Conclusion

This paper proposes a network defense strategy, *HBB-TSP*, to safeguard network connectivity priority from potential threats posed by network topology analyzers. *HBB-TSP* employs a heuristic active defense methodology, which incorporates overhead constraints and network dynamic changes, to effectively interfere with real-time traffic and offer untargeted and targeted defense. The proposed approach involves identifying the deep hypergraph-based backbone network in the network topology. The complex network structure prediction based on timing is then utilized to add obfuscated nodes/links in real-time, making it challenging for network topology analyzers to comprehend the network structure. The efficacy of *HBB-TSP* was evaluated using an advanced network topology discriminant analyzer, and the results demonstrate that our defense scheme outperformed traditional methods in terms of active defense. Additionally, we demonstrate that *HBB-TSP* is stealthy and deceptive against existing countermeasures. The current study believes that *HBB-TSP* is appropriate for more general scenarios and can perform targeted active defense against general network structure analyzer-based ones. Furthermore, the proposed defense scheme is cost-effective and accounts for network dynamic changes. In conclusion, our findings suggest that *HBB-TSP* is a promising network defense strategy that can enhance network security in the face of network topology analyzers. In future research, we propose that *HBB-TSP* can be more extensively employed to combat a broader spectrum of network topology mapping attacks, particularly those leveraging latency inference methods. This is underpinned by the efficacy of *HBB-TSP* in strategically introducing influential obfuscated links to mitigate attacks. Furthermore, future enhancements to *HBB-TSP* could involve implementing mechanisms that selectively expose the obfuscated network only to attackers, while ensuring that normal users' access and usage remain unaffected.

CRediT authorship contribution statement

Xiaohui Li: Writing – review & editing, Software, Methodology, Investigation. **Xiang Yang:** Writing – original draft, Validation, Methodology, Investigation. **Yizhao Huang:** Visualization, Resources, Methodology. **Yue Chen:** Writing – review & editing, Validation, Methodology.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Xiaohui Li reports financial support was provided by National Natural Science Foundation of China. Xiaohui Li reports financial support was provided by Sichuan Youth Science and Technology Innovation Team. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Celdrán, A.H., Sánchez, P.M.S., Von Der Assen, J., Schenk, T., Bovet, G., Pérez, G.M., Stiller, B., 2024. Rl and fingerprinting to select moving target defense mechanisms for zero-day attacks in iot. IEEE Trans. Inf. Forensics Secur..

Chee, K.O., Ge, M., Bai, G., Kim, D.D., 2024. IoTSecSim: A framework for modelling and simulation of security in Internet of things. Comput. Secur. 136, 103534.

Chen, J., Zhang, J., Xu, X., Fu, C., Zhang, D., Zhang, Q., Xuan, Q., 2019. E-LSTM-D: A deep learning framework for dynamic network link prediction. IEEE Trans. Syst. Man Cybern.: Syst. 51 (6), 3699–3712.

Coscia, A., Dentamaro, V., Galantucci, S., Maci, A., Pirlo, G., 2024. Automatic decision tree-based NIDPS ruleset generation for DoS/DDoS attacks. 82, Elsevier, 103736,

Doriguzzi-Corin, R., Siracusa, D., 2024. FLAD: adaptive federated learning for DDoS attack detection. Comput. Secur. 137, 103597.

Goyal, P., Chhetri, S.R., Canedo, A., 2020. Dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. Knowl.-Based Syst. 187, 104816.

Hou, T., Qu, Z., Wang, T., Lu, Z., Liu, Y., 2020. ProTO: Proactive topology obfuscation against adversarial network topology inference. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications. IEEE, pp. 1598–1607.

Javadpour, A., Ja'fari, F., Taleb, T., Shojafar, M., Benzaïd, C., 2024. A comprehensive survey on cyber deception techniques to improve honeypot performance. Elsevier, 103792,

Kim, J., Marin, E., Conti, M., Shin, S., 2022. EqualNet: a secure and practical defense for long-term network topology obfuscation. In: Proceedings of the USENIX NDSS.

King, I.J., Huang, H.H., 2023. Euler: Detecting network lateral movement via scalable temporal link prediction. ACM Trans. Priv. Secur..

Lei, K., Qin, M., Bai, B., Zhang, G., Yang, M., 2019. GCN-GAN: A non-linear temporal link prediction model for weighted dynamic networks. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, pp. 388–396.

Li, D., Hu, Y., Xiao, G., Duan, M., Li, K., 2023. An active defense model based on situational awareness and firewalls. Concurr. Comput.: Pract. Exper. 35 (6), 1.

Li, T., Zhang, J., Philip, S.Y., Zhang, Y., Yan, Y., 2018. Deep dynamic network embedding for link prediction. IEEE Access 6, 29219–29230.

Liu, J., Deng, W., Yang, C., Qin, A., Huang, K., 2023b. SI-LSGAN: Complex network structure inference based on least square generative adversarial network. Chaos Solitons Fractals 173, 113739.

Liu, L., Kuang, X., Liu, L., Zhang, L., 2024. Defend against adversarial attacks in malware detection through attack space management. Comput. Secur. 141, 103841.

Liu, Y., Xing, C., Zhang, G., Song, L., Lin, H., 2022. AntiTomo: Network topology obfuscation against adversarial tomography-based topology inference. Comput. Secur. 113, 102570.

Liu, B., Yang, R., Lü, L., 2023a. Higher-order link prediction via local information. Chaos 33 (8).

Liu, Y., Zhao, J., Zhang, G., Xing, C., 2021. NetObfu: A lightweight and efficient network topology obfuscation defense scheme. Comput. Secur. 110, 102447.

Maesschalck, S., Fantom, W., Giotsas, V., Race, N., 2024. These aren't the PLCs you're looking for: Obfuscating PLCs to mimic honeypots. IEEE,

Meier, R., Tsankov, P., Lenders, V., Vanbever, L., Vechev, M., 2018. NetHide: Secure and practical network topology obfuscation. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 693–709.

Musciotto, F., Battiston, F., Mantegna, R.N., 2021. Detecting informative higher-order interactions in statistically validated hypergraphs. Commun. Phys. 4 (1), 218.

Nance-Hall, M., Liu, Z., Sekar, V., Durairajan, R., 2024. Analyzing the benefits of optical topology programming for mitigating link-flood DDoS attacks. IEEE Trans. Dependable Secure Comput. 1–18. http://dx.doi.org/10.1109/TDSC.2024.3391188.

Qin, M., Yeung, D.-Y., 2023. Temporal link prediction: A unified framework, taxonomy, and review. ACM Comput. Surv. 56 (4), 1–40.

Rossi, R.A., Ahmed, N.K., 2015. The network data repository with interactive graph analytics and visualization. In: AAAI. URL https://networkrepository.com.

Schloegel, M., Blazytko, T., Contag, M., Aschermann, C., Basler, J., Holz, T., Abbasi, A., 2022. Loki: Hardening code obfuscation against automated attacks. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 3055–3073.

Sun, H., Su, H., Weng, J., Liu, Z., Li, M., Liu, Y., Zhong, Y., Sun, W., 2024. MTDCAP: Moving target defense-based CAN authentication protocol. IEEE,

Trassare, S.T., Beverly, R., Alderson, D., 2013. A technique for network topology deception. In: MILCOM 2013-2013 IEEE Military Communications Conference. IEEE, pp. 1795–1800.

Tudisco, F., Higham, D.J., 2021. Node and edge nonlinear eigenvector centrality for hypergraphs. Commun. Phys. 4 (1), 201.

Wang, Y., Su, Z., Benslimane, A., Xu, Q., Dai, M., Li, R., 2023b. Collaborative honeypot defense in uav networks: A learning-based game approach. IEEE Trans. Inf. Forensics Secur. 1963–1978.

Wang, J., Tian, J., Liu, Y., Yang, D., Liu, T., 2023a. MMTD: Multi-stage moving target defense for security-enhanced D-FACTS operation. IEEE,

Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., Xie, X., Guo, M., 2018. Graphgan: Graph representation learning with generative adversarial nets. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 32.

Wang, Q., Xiao, F., Zhou, M., Wang, Z., Li, Q., Li, Z., 2017. Linkbait: active link obfuscation to thwart link-flooding attacks. arXiv preprint arXiv:1703.09521.

Ye, D., Zhu, T., Shen, S., Zhou, W., 2020. A differentially private game theoretic approach for deceiving cyber adversaries. IEEE Trans. Inf. Forensics Secur. 16, 569–584.

Yoo, S., Chen, X., Rexford, J., 2024. SmartCookie: Blocking large-scale SYN floods with a split-proxy defense on programmable data planes. In: USENIX Security.

Zhang, L., Chen, Y., 2024. Distributed finite-time ADP-based optimal secure control for complex interconnected systems under topology attacks. IEEE Trans. Syst. Man Cybern.: Syst..

Zhang, T., Xu, C., Shen, J., Kuang, X., Grieco, L.A., 2023. How to disturb network reconnaissance: A moving target defense approach based on deep reinforcement learning. IEEE Trans. Inf. Forensics Secur. 18, 5735–5748. http://dx.doi.org/10.1109/TIFS.2023.3314219.

Zhao, Z., Li, Z., Zhou, Z., Yu, J., Song, Z., Xie, X., Zhang, F., Zhang, R., 2024b. DDoS family: A novel perspective for massive types of DDoS attacks. 138, Elsevier, 103663,

Zhao, C., Zhou, Y., Zhang, S., Wang, T., Sheng, Q.Z., Guo, S., 2024a. DEthna: Accurate ethereum network topology discovery with marked transactions. arXiv preprint arXiv:2402.03881.

Xiaohui Li received the B.S. and Ph.D. degrees in computer science from the College of Computer Science, Sichuan University, Chengdu, China, in 2012 and 2017, respectively. She is currently an Associate Researcher with the School of Cyber Science and Engineering, Sichuan University, Chengdu, China. Her research interests cover several areas in network and information security (e.g., wireless networks, satellite networks, internet of thing, cyber threat intelligence, etc.), with emphasis on design of network forensic, complex network analysis, efficient transport and routing protocols.

Xiang Yang is a college student pursuing her B.S. degree at Sichuan University, Chengdu, China. Her research interests include network security, code auditing, software security, complex network analysis and machine learning.

Yizhao Huang received the M.S. degree s in Computer Science and Technology from the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China. He is currently working on his Ph.D. in Sichuan University. His recent research interests include program analysis, cyber security and machine learning.

Yue Chen is a graduate student pursuing her M.S. degree at Sichuan University, Chengdu, China. Her research interests include cyber threat analysis. machine learning in cyber security, and intelligent decision making.