Toward Dynamic Topology Obfuscation for IoT Networks With Evolutionary Defense

Xiang Yang, Zhentian Zhong, Yue Chen, Xiaohui Li, *Member, IEEE*, Junfeng Wang, and Zhiping Cai

***Abstract*—Traditional defenses against Distributed Denial of Service and link flooding attack in Internet of Things (IoT) networks, such as static network topology obfuscation and traffic engineering, suffer from critical limitations. These include high computational overhead that is incompatible with low-power devices, rigid policies that are vulnerable to evolving attacks, and isolated modules that create exploitable gaps. Accordingly, a new problem setting for IoT network defense is introduced, i.e., dynamic open-world network defense that poses two critical tasks: 1) open defense, aiming not only to mitigate attacks from known patterns but also to detect and counteract unknown adversarial strategies through real-time traffic analysis and adaptive obfuscation and 2) dynamic adaptation, aiming to continuously learn new attack types (e.g., novel probing techniques or topology inference methods) without retraining the entire model while preserving robustness against previously encountered threats and avoiding catastrophic forgetting of learned defense policies. To address these problems, we propose OptiPathNet, a unified framework integrating multiobjective evolutionary optimization with dynamic topology obfuscation and real-time traffic analysis. At its core, OptiPathNet leverages two innovations. First is a GAN-based constrained multiobjective evolutionary algorithm-based method that dynamically balances security, latency and energy efficiency through Pareto-optimal tradeoffs, reducing attackers' topology inference accuracy while ensuring control packet delays and consumption. Moreover, a synergistic detection–obfuscation framework combining lightweight traffic classification with adaptive obfuscation policies, enabling protocol-compliant noise injection to disrupt both reconnaissance for traceroute-based probing and forensic analysis for timing-based attack traces. Evaluations across IoT scenarios, i.e., smart cities and industrial systems, demonstrate OptiPathNet's superiority over the state-of-the-art methods. Reduces the structural similarity between the attacker-inferred and real topologies by 43%, while the delay overhead was reduced by 17%, and achieves optimal defense in 95% of the scenarios tested. By transforming IoT's inherent constraints, including resource limitations and dynamic topologies, into defensive assets, OptiPathNet establishes a scalable, adaptive security paradigm for next-generation networks, bridging the gap between theoretical innovation and practical deployment in resource-constrained environments.**

***Index Terms*—Dynamic adaptation, Internet of Things (IoT) security, link flooding attacks (LFAs), multiobjective evolutionary optimization, topology obfuscation.**

I. Introduction

THE Internet of Things (IoT) has gained attention for its ability to enhance user services. However, IoT systems face various security threats, one of the most serious being Distributed Denial-of-Service (DDoS) attacks driven by botnets [1], [2], [3], [4]. In recent years, a new type of DDoS attack known as link flooding attack (LFA) [5], [6], [7], [8], [9] has emerged, posing a serious challenge to Internet availability. Research exemplified by the Coremelt attack [10] shows that using target-agnostic traffic can create wide-area link congestion, causing large-scale disruptions without directly targeting specific hosts. This highlights the potential of indirect attack methods to undermine network integrity and performance. LFA disrupts many communication networks by inundating critical infrastructure, such as routers and bottleneck links. In 2016, the Mirai botnet executed a large-scale LFA using infected IoT devices, including smart cameras and home routers [11]. This attack paralyzed multiple Internet service providers and IoT service platforms worldwide, affecting millions of users. In 2020, an LFA attack targeted smart city infrastructure. It disrupted traffic monitoring systems and smart grid management platforms in a European country, causing significant economic losses and service interruptions [12]. According to CloudFlare statistics, DDoS attacks on the IoT infrastructure have been increasing rapidly in recent years, underscoring the urgent need to address these threats [1], [13].

To launch an LFA, attackers target critical network links that handle aggregated traffic. For example, the Crossfire attack [5] enhanced its effectiveness by utilizing path discovery tools, such as traceroute, to pinpoint and precisely inundate bottleneck links in specific regions, facilitating coordinated and stealthy disruptions. Although encrypted communication mechanisms, such as VPN tunnels and onion routing (e.g., Tor), offer some degree of traffic anonymity, their primary design goals focus on protecting message content and source–destination associations, rather than defending against structural link-level attacks. This vulnerability is especially pronounced in resource-constrained environments like IoT networks, where the overhead and complexity hinder scalability. Moreover, LFA attackers typically do not rely on

Received 14 June 2025; revised 4 August 2025; accepted 19 August 2025. Date of publication 21 August 2025; date of current version 24 October 2025. This work was supported in part by the National Natural Science Foundation of China under Grant U24B20147 and Grant U2133208, and in part by the Major Science and Technology Special Project of Sichuan Province under Grant 2024ZDZX0044, Grant 2024ZHCG0195, and Grant 2024ZYD0269. *(Corresponding author: Xiaohui Li.)*

Xiang Yang, Yue Chen, and Xiaohui Li are with the School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China (e-mail: lixiaohui@scu.edu.cn).

Zhentian Zhong is with the Pittsburgh Institute, Sichuan University, Chengdu 610065, China.

Junfeng Wang is with the College of Computer Science, Sichuan University, Chengdu 610065, China.

Zhiping Cai is with the Department of Network Engineering, School of Computer Science, National University of Defense Technology, Changsha 410073, China.

Digital Object Identifier 10.1109/JIOT.2025.3601528

precise end-to-end path knowledge. Instead, they infer partial or full topologies using repeated measurements, replays, and spoofing to disrupt targeted links. Thus, defending against such attacks calls for dedicated topology obfuscation methods that counter structure-aware threats.

Defense strategies against LFA can be categorized into passive and active defenses. Passive defense uses traffic engineering to mitigate attack impacts and restore network functionality [9], [14], [15], [16], [17], while active seeks to obfuscate network nodes and links to deter attackers from identifying targets [18], [19], [20], [21], [22], [23]. Various topology obfuscation techniques have been developed to enhance security in IoT networks, with dynamic network topology obfuscation (NTO) emerging as a leading solution [18], [19], [20], [24], [25], [26], [27], [28], [29]. NTO creates a secure virtual topology to prevent attackers from identifying critical nodes. Technologies like NetObfu [30] generate fake identities, Linkbait [20] redirects traffic to less critical paths, EqualNet [26] balances path tracking to weaken topology reasoning, and NetHide [27] aims for a balance between security and availability. Additionally, machine learning approaches, such as Proto [28], implement advanced obfuscation methods. However, building a robust virtual topology faces three challenges: 1) limited adaptability to evolving attacker strategies; 2) high computing and network overhead; and 3) absence of joint optimization between topology obfuscation and attack detection.

To address the above challenges, we propose OptiPathNet, which advances dynamic NTO technologies through the integration of a GAN-based constrained multiobjective evolutionary algorithm (GCMOEA). It establishes a collaborative framework that merges dynamic obfuscation with traffic detection, thereby enhancing the efficiency of the defense mechanism. The proposed method encompasses two primary components.

1) *Obfuscation-Detection Collaboration Module:* This module integrates knowledge distillation (KD) to reduce detection complexity, while enabling feedback-driven adaptive obfuscation.
2) *GAN-Based Constrained Multiobjective Evolutionary Optimization Module:* This module dynamically optimizes obfuscation parameters by simultaneously balancing obfuscation strength, Quality of Service (QoS), and computational overhead.

Experimental results demonstrate that OptiPathNet reduces topology reconstruction similarity by 43% with only 17% delay overhead, achieving optimal defense in 95% of test scenarios, confirming its practical robustness. Our main contributions are summarized as follows.

1) *Unified Defense Architecture:* OptiPathNet establishes an adaptive defense architecture that tightly couples dynamic obfuscation with traffic detection in a feedback-driven loop, enabling intelligent, low-overhead, and real-time strategy refinement against evolving threats.
2) *GAN-Augmented Multiobjective Topology Obfuscation:* The design integrates GAN-driven multiobjective optimization to co-adjust obfuscation strength and service quality in real time, achieving superior resilience and adaptability compared to conventional approaches.
3) *IoT-Compatible Design:* The design pioneers a lightweight, self-adaptive architecture that safeguards against topology inference attacks while flexibly accommodating heterogeneous low-power IoT protocols, ensuring broad applicability and deployment feasibility.

The remainder of this article is organized as follows. Section II outlines the scenarios and the topology inference model. Section III details OptiPathNet, and Section IV compares it with state-of-the-art baselines, presenting experimental results. Finally, Section VII concludes the study.

II. Topology Inference Model

This section discusses two representative topology inference strategies employed by adversaries during the reconnaissance phase of lateral movement: 1) internal cooperative topology inference and 2) external end-to-end topology inference. These models reflect distinct but complementary approaches to adversarial path discovery and serve as critical precursors to targeted attack planning. While the core objective of this work is to defend against the later, we also propose a lightweight countermeasure to address the former, thereby enhancing the robustness of the defense under diverse reconnaissance scenarios.

1) *Internal Cooperative Topology Inference:* This model operates under the premise that the attacker has control over multiple internal hosts within the target network. By coordinating their probing efforts, often employing low time-to-live (TTL) packets, such as ICMP or UDP-based probes, the attacker gathers responses, including time-exceeded messages, to reconstruct internal routing paths. Furthermore, variations in hop-level round-trip time (RTT) can be leveraged to identify high-load or bottleneck links with greater precision. This model achieves high inference accuracy with minimal probing overhead, owing to the reliability and stability of its internal paths. However, its practical application is constrained by the strong assumption of internal compromise, which may not hold in many real world.
2) *External End-to-End Topology Inference.* This model assumes that the attacker only has control of the communication endpoints, typically involving multiple receivers and a single source, without access to internal network components. The attacker manipulates traffic characteristics, such as packet size, interarrival time, or protocol type, to induce and observe variations in performance metrics (e.g., delay or loss rate) at the receivers. These observations are then used to infer the underlying network structure and the distribution of load. In this study, we assume the attacker employs a recursive neighbor joining (RNJ)-based inference algorithm, which constructs a tree-like similarity graph of end-to-end paths based on observed link-level attributes. This approach enables the attacker to probabilistically identify critical forwarding links without requiring internal access.

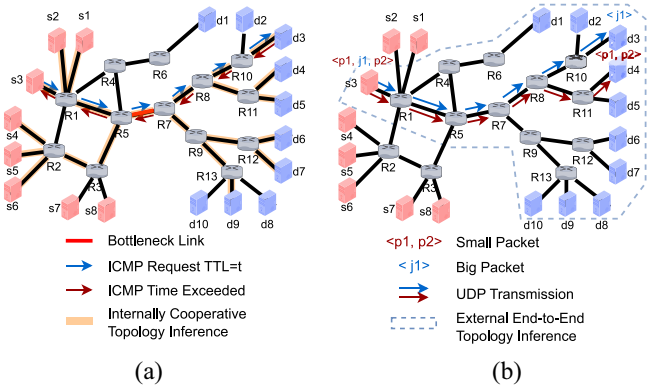

Fig. 1. Topology inference model. (a) Internal inference. (b) External inference.

A. Internal Cooperative Topology Inference

1) Attack Model: In the internal inference model, attackers utilize tools, such as traceroute to manipulate TTL values and determine network paths during the reconnaissance phase. As shown in Fig. 1, a representative example involves sending a packet with a TTL value of 6, which results in an ICMP time-exceeded response from router d_3. The response reveals the router's IP address R_t, enabling the attacker to map the sequence of IP addresses along the path between s_3 and d_3 as follows:

$$\text{Path}(s_3, d_3) = \{s_3, R_1^{ttl=1}, R_5^{ttl=2}, R_7^{ttl=3}, R_8^{ttl=4}, R_{10}^{ttl=5}, d_3\}. \quad (1)$$

The delay for a specific link from R_i to R_j, such as (R_1, R_5), can be calculated based on the difference in delays measured at subsequent hops

$$T_{R_1 \to R_5} = T_{s_3 \to d_3}^{ttl=2} - T_{s_3 \to d_3}^{ttl=1}. \quad (2)$$

Then, bottleneck links that are potential targets for concentrated attacks, such as DDoS or LFAs, are identified

$$\left(R_i^*, R_j^*\right) = \arg \max_{(R_i, R_j)} \left(T_{R_i \to R_j}, \text{Freq}(R_i, R_j)\right) \quad (3)$$

where $T_{R_i \to R_j}$ represents the maximum delay for a specific pair of (s_3, d_3), and $Freq(R_{k-1}, R_k)$ represents the frequency of occurrences of the link in multiple Path (s_3, d_3).

2) Defense Strategy: This defense scenario assumes that the attacker has already gained control of several internal nodes, typically located at the edges of the topology. These nodes are capable of launching low-TTL probe packets (e.g., ICMP or UDP) that reach only a few hops and then trigger ICMP Time Exceeded replies. By aggregating such responses across multiple nodes, the attacker attempts to infer internal routing structure and identify key links.

To counter such behavior, a common defense strategy is to proactively modify the TTL values of certain low-TTL probe packets, particularly those targeting key links. This modification makes it appear that the probing path traverses more hops than it actually does, effectively flattening the topology and obfuscating path depth. The defense mechanism is designed with the following objectives.

1) *Security Objective:* Achieve a more balanced traffic distribution across paths, preventing attackers from identifying highly concentrated flows that reveal bottlenecks.
2) *Usability Constraint:* Ensure that the modified responses do not disrupt normal business routing behavior. In practice, this often involves blending real and spoofed topology responses in a way that minimizes structural distortion.

B. External End-to-End Topology Inference

1) Attack Model: External end-to-end topology inference is predicated on the measurement of end-to-end delays, as exemplified in Fig. 1, which portrays the utilization of the back-to-back detection approach. In this instance, we define $T_{s \to d_i}^{(t)}$ as the measured one-way delay of the tth probe from the source node s to the target node d_i. A value of $T_{s \to d_i}^{(t)} = \infty$ signifies that the destination d_i did not receive the tth probe. We utilize $T_{s \to d_i}^{\min} = \min_t T_{s \to d_i}^{(t)}$ to estimate the propagation delay from source s to destination d_i. Moreover, we introduce the symbol $\hat{\rho}(s, d_i)$ to denote the total length of the path from source s to destination d_i. Similarly, $\hat{\rho}(s, [d_i, d_j])$ represents the total length within the shared portion of the paths from source s to destinations d_i and d_j. Assuming that an attacker consecutively transmits small, large, and small packets, denoted as p_1, j_1, and p_2, respectively. Here, p_1 and p_2 are destined for the same endpoint d_4, while j_1 is addressed to destination d_3. By measuring the discrepancy in arrival times between j_1 and p_2, we can ascertain the end-to-end length $[\hat{\rho}(d_3), \hat{\rho}(s, d_4)]$ and the shared path length $\hat{\rho}(s, [d_3, d_4])$ by employing explicit estimators

$$\begin{aligned} \hat{\rho}(s, d_i) &= \hat{\text{var}}\left(\text{T}_{\text{s}\to\text{d}_\text{i}}\right) \\ \hat{\rho}\left(s, \left[d_i, d_j\right]\right) &= \hat{\text{cov}}\left(T_{s \to d_i}, T_{s \to d_j}\right) \end{aligned} \quad (4)$$

where

$$\begin{aligned} \hat{var}\left(T_{s \to d_i}\right) &= \frac{1}{n-1} \sum_{t=1}^{n} \left(T_{s \to d_i}^{(t)} - \bar{T}_{s \to d_i}\right)^2 \\ \hat{cov}\left(T_{s \to d_i}, T_{s \to d_j}\right) &= \frac{1}{n-1} \sum_{t=1}^{n} \left(T_{s \to d_i}^{(t)} - \bar{T}_{s \to d_i}\right) \\ &\quad \left(T_{s \to d_j}^{(t)} - \bar{T}_{s \to d_j}\right) \\ \bar{T}_{s \to d_i} &= \frac{1}{n} \sum_{t=1}^{n} T_{s \to d_i}^{(t)}. \end{aligned} \quad (5)$$

The source node denoted as s, a set of target nodes denoted as D, the end-to-end length denoted as $\hat{\rho}(s, D)$, the shared path length denoted as $\hat{\rho}(s, D^2)$, and a length threshold denoted as $\Delta > 0$, which are derived from RNJ algorithm [31]. Then, the node and edge set are defined as follows:

$$\begin{aligned} V &= \{s\} \cup D \\ E &= \varnothing. \end{aligned} \quad (6)$$

Afterward, determine i^* and j^* to maximize $\hat{\rho}(s, [i^*, j^*])$, and select arbitrarily from multiple candidate solutions

$$\hat{\rho}\left(s, \left[i^*, j^*\right]\right) - \hat{\rho}\left(s, \left[i^*, k\right]\right) \leq \frac{\Delta}{2}. \quad (7)$$

A node f is created to serve as the parent of i^* and j^* and then proceeds to update the set

$$\begin{aligned} D &= D\backslash\{i^*, j^*\} \\ V &= V \cup \{f\} \\ E &= E \cup \{(f, i^*), (f, j^*)\}. \end{aligned} \tag{8}$$

The following path length $\hat{\rho}(f, i^*)$, $\hat{\rho}(f, j^*)$ are calculated:

$$\begin{aligned} \hat{\rho}(f, i^*) &= \hat{\rho}(s, i^*) - \hat{\rho}(s, [i^*, j^*]) \\ \hat{\rho}(f, j^*) &= \hat{\rho}(s, j^*) - \hat{\rho}(s, [i^*, j^*]). \end{aligned} \tag{9}$$

Then, the connections can be updated. For each node $k \in D$, it should be connected to f, and the shared path length and set should be updated accordingly

$$\begin{aligned} \hat{\rho}(f, k) &= \hat{\rho}(s, k) - \hat{\rho}(s, [i^*, j^*]) \\ \hat{\rho}(s, [k, f]) &= \frac{1}{2}\left[\hat{\rho}(s, [k, i^*]) + \hat{\rho}(s, [k, j^*])\right] \\ D &= D \cup \{f\} \\ \hat{\rho}(s, f) &= \hat{\rho}(s, [i^*, j^*]) \end{aligned} \tag{10}$$

where, when there is only one node k remaining in set D, it is necessary to connect the source node s to node k and update the edge set. If multiple nodes remain in set D, the above steps must be repeated

$$E = E \cup (s, k). \tag{11}$$

We denote the final tree $\hat{T} = (V, E)$, where V corresponds to the set of nodes and E corresponds to the set of edges. The length of each edge e is denoted by $\hat{\rho}(e)$ for all $e \in E$.

The proposed attack model is conceptually aligned with CrossPoint [7], which highlights iterative, fine-grained topology inference through end-to-end probing and historical path analysis. On this basis, we consider adversarial sensitivity to both latency and TTL variations, and introduce a structure-aware delay adjustment and TTL emulation mechanism for obfuscation resilience.

2) Defense Strategy: This scenario assumes the attacker has no control over any internal node and can only manipulate communication between end-hosts. The attacker sends customized probe traffic between sending and receiving endpoints, then infers intermediate link properties based on round-trip delay patterns.

In this case, a commonly adopted defense strategy is delay-oriented topology camouflage. Specifically, slight delays are added to selected packets traversing key links, thereby distorting RTT-based inferences and introducing structural deviation between the perceived and actual topologies. The design objectives here are as follows.

1) *Security Objective:* Minimize the structural similarity between inferred and true topologies to confuse the attacker's path reconstruction.
2) *Usability Constraint:* Maintain acceptable total end-to-end latency to avoid significant impact on user-perceived service quality. This requires precise control over added delays so as not to degrade network performance.

III. OptiPathNet Framework

To address the critical challenges in topological obfuscation, including limited adaptability, fragmented strategies, and computational inefficiency, we propose OptiPathNet, a unified framework that dynamically optimizes multidimensional defense objectives while seamlessly integrating traffic detection mechanisms. The proposed methodology consists of two tightly coupled components.

1) OptiPathNet presents a unified defense architecture that couples dynamic topology obfuscation with lightweight traffic detection in an adaptive feedback loop. Detection outcomes continuously inform obfuscation strategies, enabling attacker-aware, resource-efficient, and real-time defense adaptation without unnecessary network overhead.
2) We propose a GAN-enhanced multiobjective optimization module designed to refine obfuscation strategies by simultaneously optimizing two critical objectives.
   a) *Path Delay Adjustment:* The expected communication delay between nodes R_i and R_j, denoted as $T_{R_i \to R_j}$, serves as a key indicator of service quality. This objective ensures that obfuscation does not excessively degrade network performance.
   b) *Path Similarity Constraint:* The normalized shared ratio, $\hat{\rho}_v(s, [d_i, d_j])$, measures the structural similarity between the true path and its obfuscated counterpart. This metric helps design deceptive paths that closely resemble the true route, thereby increasing the probability of attacker misjudgment while preserving functional connectivity.

A. Overall Architecture

To effectively mitigate topology-aware attacks LFA and DDoS, our primary objective is to disrupt precise topology inference by adversaries. However, real-world network environments often exhibit a significant presence of benign probing traffic alongside malicious flows. To address this, we incorporate a lightweight, model-driven traffic classification mechanism that enables targeted, adaptive defense without compromising system usability.

As illustrated in Fig. 2, incoming traffic at critical network nodes (e.g., core routers and edge firewalls) is first processed through a compact classifier distilled from an offline-trained deep model. This module categorizes flows into three types: 1) *malicious probing traffic* (*red*), which exhibits stealthy path exploration characteristics and is prioritized for suppression to safeguard structural privacy; 2) *benign communication traffic* (*blue*), which is nonprobing and must be preserved with minimal disruption to uphold service quality; and 3) *suspicious probing traffic* (*gray*), which displays RTT-based or path-scanning features suggestive of structural inference attempts. The third category is the principal target of our topology obfuscation strategy.

Upon detection of suspicious flows, the system activates a delay-aware camouflage module that dynamically generates delay injection rules. These rules are computed in real-time,

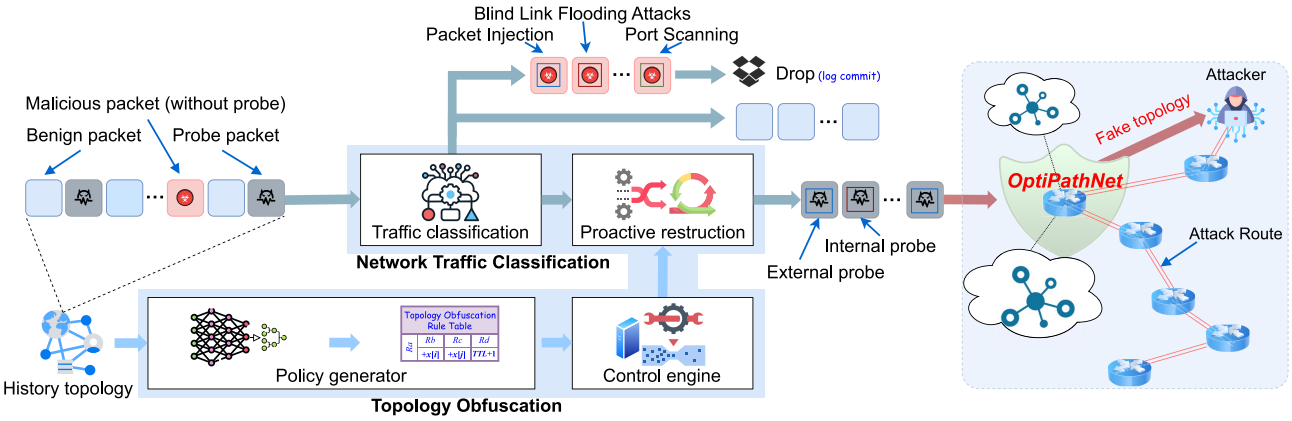

Fig. 2. Framework of OptiPathNet.

conditioned on the current load status and routing policy constraints, and are designed to alter the perceived link characteristics without degrading the QoS for legitimate flows. This mechanism reduces the topological fidelity of attacker observations while ensuring compliance with performance constraints. Fig. 2 further distinguishes between internal cooperative and external end-to-end probing behaviors to inform differential defense strategies. While our design predominantly targets the latter, this classification framework supports fine-grained response mechanisms tailored to diverse adversarial reconnaissance modalities.

Specifically, OptiPathNet comprises two primary modules.

1) Topology Obfuscation Mechanism Grounded in Multiobjective Evolutionary Algorithm: The topology obfuscation algorithm, leveraging a multiobjective evolutionary approach, allows OptiPathNet to implement targeted obfuscation strategies for varying types of probe traffic. In internal collaborative detection traffic, the system mitigates the adversary's capability for topological reasoning by modifying TTL values and creating virtual nodes, thereby offloading traffic from critical bottleneck links to diminish their relevance to potential attackers. The system dynamically identifies critical links regarding external end-to-end detection traffic, discerning which paths necessitate increased delays. These delay values are optimized through a multiobjective evolutionary algorithm to achieve delay homogenization while effectively balancing the effects of topological confusion and QoS.

2) Collaborative Optimization Framework for Dynamic Obfuscation and Traffic Detection: Under the collaborative design of dynamic topology obfuscation and traffic identification, OptiPathNet integrates a distillation-optimized lightweight malicious traffic detection model. The core objective is to maintain high detection accuracy while substantially reducing resource consumption, addressing the stringent latency, computation, and memory constraints typical of IoT environments. Rather than designing a lightweight architecture from scratch, the approach leverages KD from an existing high-capacity classifier. A teacher–student framework is employed, wherein the student model learns to approximate the output distribution of the teacher while significantly compressing parameter size and inference overhead, thus achieving a balance between efficiency and detection fidelity. The model filters out identified nonprobing malicious traffic, retaining its features for further reference, while allowing benign communication to proceed uninterrupted. For probing traffic, subsequent analysis is performed to distinguish between internal cooperative and external end-to-end reconnaissance behaviors.

Moreover, OptiPathNet integrates a dynamic policy update mechanism that empowers the network to adapt its obfuscation strategies at varying levels of granularity within the IoT environment, responsive to temporal changes and evolving conditions. OptiPathNet aims to optimize network security while ensuring stability and communication efficiency by consistently applying and recalibrating the established obfuscation strategies.

B. Traffic-Aware Dynamic Obfuscation—Synergistic Defense Framework

The synergistic framework for dynamic obfuscation and traffic analysis involves fine-tuning a pretrained patch generation model to categorize various traffic types, including probe traffic. This process begins with extensive pretraining on distinct sets of malicious and benign traffic, followed by fine-tuning with a smaller dataset tailored to the current task. Model distillation is incorporated to maintain high detection rates and low false positives while minimizing network load.

For this, the Flow-MAE model uses masked autoencoders (MAEs) [32], with its core design centered on self-supervised learning to extract discriminative representations directly from raw network traffic. Unlike traditional approaches that rely on handcrafted statistical features, the Flow-MAE model operates on byte-level flow sequences. After preprocessing steps, such as session segmentation and length normalization, the input traffic is partitioned into fixed-length patches, a subset of which is randomly masked. The encoder learns contextual representations from the unmasked patches, aiming to reconstruct the masked portions during training. This process encourages the model to capture temporal dependencies and contextual structures inherent in network flows. Following pretraining, the original decoder is replaced with a lightweight fully connected classifier, and the model undergoes supervised fine-tuning for multiclass traffic classification.

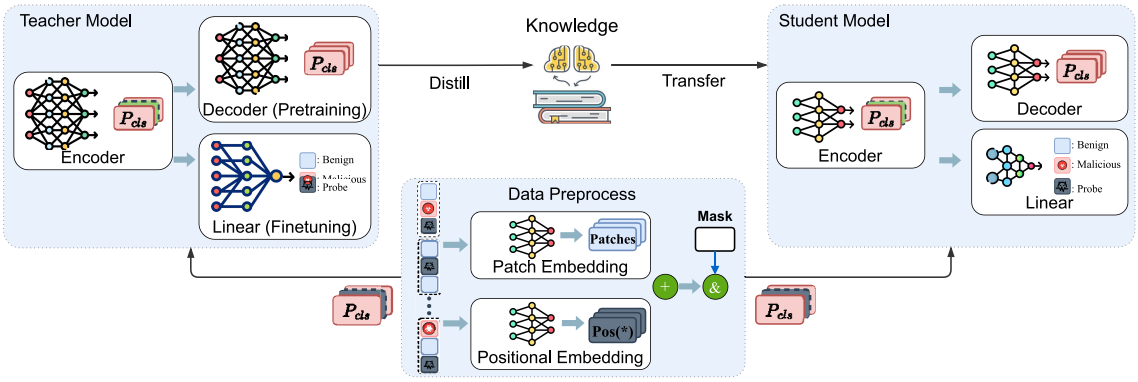

Fig. 3. Design of traffic classification.

Despite its strong performance in modeling sequential structure and learning generalizable representations, the deep Transformer-based encoder incurs considerable computational and memory overhead, limiting its applicability on resource-constrained devices. To address this, we introduce a KD framework wherein the pretrained Flow-MAE serves as the teacher model. A smaller student model is trained to approximate both the soft output distribution and intermediate feature representations of the teacher. The resulting Distilled-Flow-MAE achieves substantial reductions in model size and inference cost while retaining classification accuracy, thus enhancing its practical deployability and responsiveness in real-world network security scenarios. The algorithm design for traffic classification is illustrated in Fig. 3.

1) Preprocess and Pretraining: Building on Flow-MAE research, packets ($P = \{p_i | i = 1, 2, 3 \cdots\}$) are segmented into sessions ($S = p_j \mid p_j \in P$) by quintuple, then further split into bursts ($B = p_k \mid p_k \in S$) to capture network traffic patterns. Explicit identifiers like MAC, IP, and port fields are removed to reduce bias. Bursts are then standardized to a uniform length using padding and cropping, followed by generating attention masks using masked sequences

$$B = \begin{cases} B \| 0^{\ell_{\text{MAX}} - \ell_B}, & \text{if} \quad \ell_B < \ell_{\text{MAX}} \\ B[0{:}\ell_{\text{MAX}}], & \text{otherwise} \end{cases}, \tag{12}$$

$$M = \begin{cases} 1^{\ell_B} \| 0^{\ell_{\text{MAX}} - \ell_B}, & \text{if} \quad \ell_B < \ell_{\text{MAX}} \\ 1^{\ell_{\text{MAX}}}, & \text{otherwise} \end{cases} \tag{13}$$

where ℓ_{MAX} represents the maximum limit input length of MAE[32], while ℓ_B signifies the actual length of the burst flow$B$. The input sequence B is initially encoded using a 1-D convolutional layer by the MAE, generating nonoverlapping contiguous segments referred to as patches ($\mathcal{P}$)

$$\underset{d,k}{\text{Conv1D}}(B) \to \mathcal{P} = \left\{ p_i \mid p_i \in \mathbb{R}^d, i = 0, 1, \ldots \lceil \frac{\ell_B}{k} \rceil - 1 \right\} \tag{14}$$

where d represents the number of output channels, and k denotes the convolution kernel size. The attention mask $\mathcal{M}$ determines whether p_i is generated from bytes padded or B, based on the mask sequence M

$$\mathcal{M} = \{ m_i \mid m_i \in \{0, 1\}, i = 0, 1, \ldots, |\mathcal{P}| - 1 \} \tag{15}$$

$$\text{s.t. } m_i = \begin{cases} 0, & \text{if} \quad M[i \cdot k, (i+1) \cdot k] = 0^k \\ 1, & \text{otherwise.} \end{cases} \tag{16}$$

Finally, positional embedding encodes the positional information of each patch into the model, denoted as follows:

$$\text{Pos}(i) \in \mathbb{R}^h. \tag{17}$$

In pretraining, random masking and shuffling are applied to the generated patches. Each patch p_i has a masking probability r (where $0 < r < 1$), making it invisible during training. The patches are then divided into visible (P_v) and invisible (P_m) subsets, with a category patch ($p_{cls} = 0^d$) added to form the input $\mathcal{P}_{cv} = \{p_{cls}\} || \mathcal{P}_v$ for the encoder.

The Masked Encoder, based on the transformer architecture, uses self-attention to capture relationships between P_v at various hierarchical levels, resulting in latent representations $\mathcal{P}_{i+1} = \text{Encoder}_i(\mathcal{P}_i, \mathcal{M}_v)$. The Decoder reconstructs blocked patches using self-attentive blocks. Decoding begins with an "Unshuffle" operation, where placeholders are inserted at masked positions, and visible patches are returned to their original spots

$$\mathcal{P}'_0 \leftarrow \text{Unshuffle}\left(\mathcal{P}_{|E|}, 0^d, 0^d, \ldots, 0^d\right) \tag{18}$$

where $|\mathcal{P}_{|E|}| = |\mathcal{P}_v|$ and $|0^d, \ldots, 0^d| = |\mathcal{P}m|$, the decoder replaces placeholders with recovered patches, denoted as $\mathcal{P}i + 1' = \text{Decoder}_i(\mathcal{P}'_i)$. The pretraining loss is evaluated using the mean-square error (MSE) between the recovered patches ($\mathcal{P}'_r$) and the original masked patches ($\mathcal{P}_m$)

$$L_{\text{pre-train}} = \text{MSE}\left(\mathcal{P}'_r, \mathcal{P}_m\right) \tag{19}$$

$$= \frac{1}{|\mathcal{P}_m|} \sum_{i=0}^{|\mathcal{P}_m|-1} \| p'_{r_i} - p_{m_i} \|_2. \tag{20}$$

2) Finetuning: Using large-scale unlabeled traffic for pretraining allows the encoder to generate deep latent representations for any given burst, which can be leveraged for various tasks, thus addressing data scarcity for task-specific traffic classification. During the fine-tuning process, the pretrained encoder and associated parameters are retained, while the decoder is replaced with a fully connected linear classifier for classification. The loss function is the cross-entropy loss,

quantifying the disparity between the predicted probability vector and the true label probability

$$L_{\text{CrossEntropy}} = R \cdot \text{CrossEntropy}\left(R \cdot \text{Softmax}\left(\hat{Y}\right), Y\right) = -\sum_{i=1}^{c} \log \frac{\exp(\hat{y}_i)}{\sum_{j=1}^{c} \exp(\hat{y}_j)} y_i \quad (21)$$

where, for a classification task with c categories and input burst B, the true label probabilities are $Y = \{y_1, y_2, \ldots, y_c\}$, and the predicted probabilities are $\hat{Y} = \{\hat{y_1}, \hat{y_2}, \ldots, \hat{y_c}\}$.

3) Knowledge Distillation: Larger models often result in increased inference times to detect malicious traffic in IoT environments, making real-time responses more challenging. KD [33] can help compress models and enhance the understanding of sample similarity. Our study proposes a framework utilizing a heterogeneous teacher model and a relation miner to extract specific traffic knowledge while minimizing knowledge contrast loss. This approach includes the following strategies.

1) *Response-based KD* aims for the student model to replicate the teacher's predictions by minimizing output differences.
2) *Feature-based KD* aligns the teacher and student models' internal feature representations.
3) *Relation-based KD* focuses on capturing the structural relationships within the feature representations of both models.

C. Multiobjective Evolutionary Topology Obfuscation

Multiobjective evolutionary topology obfuscation provides a unified control interface for OptiPathNet, generating a topology obfuscation rule table (TORT) for each network interface as shown in Figs. 4 and 5, which illustrate the link target determination for delay obfuscation and the multiobjective optimization process for delay computation, respectively. This approach deliberately delays identified probe packets, preventing attackers from accurately inferring the network topology, resulting in an obfuscated structure with misleading bottleneck links. The obfuscation module must meet the following criteria.

1) *Security:* The Attacker's inferred topology should significantly differ from the actual network.
2) *Deception:* Delay increments should remain within manageable limits to ensure operational integrity.
3) *Efficiency:* The policy generation must be rapid to maintain performance.
4) *Concealment:* Bottleneck identification should significantly diverge from the true layout.

The process consists of two stages: identifying links for delay increases or adding virtual nodes and optimizing delay increments. The precise TORT is ultimately produced through iterative processes, relying on network stability for effective obfuscation.

1) Max Delay Table Determination: In the initial stage, the maximum delay value table is computed to identify links needing delay increases. This is illustrated on the right side of Fig. 1, simulating an attacker probing traffic along s_3–d_3 and s_3–d_1. If the delay of s_3–d_3 significantly exceeds that of s_3–d_1, it indicates potential topology inference issues or critical links in the path. As delay adjustments are unidirectional, the strategy involves increasing the delay of R_1–R_4, which then becomes part of the delay-increase link set. Maximum delay comparisons between links determine significant disparities, guiding whether R_1–R_5 should also be included. If R_1–R_4 has the highest delay, it is identified as a bottleneck, necessitating traffic management through TTL adjustments. Thus, the maximum delay for each node in various directions is established. The breadth-first search (BFS) algorithm, described in [34], calculates the maximum delay for each node along a specific interface and updates the MDT accordingly.

Algorithm 1: Maximum Delay Table Determination

Input: G: Graph with nodes and edges; D: Link delay dictionary
Output: G: Acyclic graph; MDT: Max delay table

```
/* Remove the shortest link in the shortest cycle */
while G has cycles do
    RemoveCycle(G)
/* Initialize the Max Delay Table */
MDT ← Dict
foreach n ∈ G do
    vis ← Set
    foreach nbr ∈ n.neighbors() do
        MDT[n][nbr] ← (n, nbr, D[n, nbr])
        vis ← {nbr}, q ← [nbr]
        /* Find the maximum delay path */
        while q do
            cur ← dequeue(q)
            foreach next ∈ cur.neighbors() do
                if next ∉ vis then
                    /* Check if the current path delay is greater than the stored max delay */
                    if D[cur, next] > MDT[n][nbr][2] then
                        MDT[n][nbr] ←
                        (cur, next, D[cur, next]);
                    q.enqueue(next);
                    vis.add(next);
return G, MDT
```

For example, to compute the maximum delay of $R1$ along $R5$, the $R1$–$R5$ link is added to the MDT with its delay set as the maximum. As $R5$ is processed, the delays of its connected links ($R13$ and $R12$) are evaluated, updating the MDT. This iterative process continues until all leaf nodes are processed. The algorithm, mentioned in Algorithm 1, includes two steps: 1) removing rings to prevent loops and 2) using BFS to identify the bottleneck delay link for MDT construction.

Extracting the network backbone is crucial for computing the MDT, focusing on maximum delays in different directions.

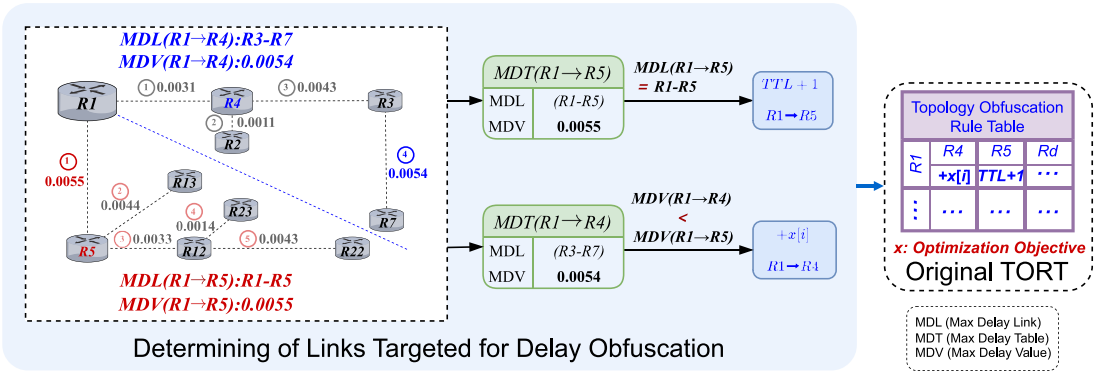

Fig. 4. Determining links targeted for delay obfuscation.

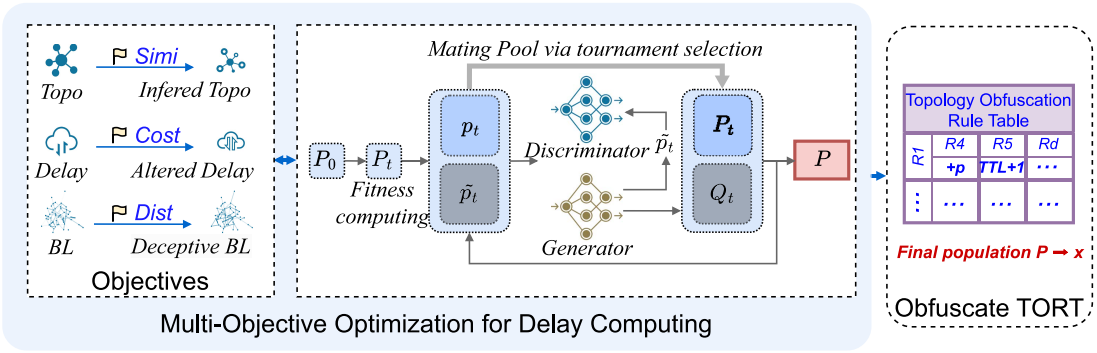

Fig. 5. Multiobjective optimization for delay computing.

Algorithm 2: Determine Latency and TTL Strategy

Input: G: Graph with nodes and edges; *MDT*: Max delay table; D: Link delay dictionary
Output: *delay_add*: Links needing delay added; *ttl_add*: Links needing TTL added

```
/* Initialize delay_add and TTL_add */
delay_add ← Dict
ttl_add ← Dict
foreach n ∈ G do
    /* Find the maximum delay for the current node */
    max_delay ← max(MDT[n])
    foreach each nbr ∈ n.neighbors() do
        delay_add[n][nbr], ttl_add[n][nbr] ← False
        /* Check if the current link delay is less than the maximum delay */
        if MDT[n][nbr] < max_delay then
            delay_add[n][nbr] ← True
        else
            ttl_add[n][nbr] ← True
return delay_add, ttl_add
```

Due to ring structures in real-world topologies, identical delays can occur in multiple paths, necessitating the removal of links with negligible delays to achieve a loop-free network.

2) Delay Value Calculation and Virtual Node Assignment: In the second stage, the delay increment value, represented as the variable x, is optimized by integrating security, deception, and bottleneck concealment. A multiobjective evolutionary approach combining GAN with a population-based algorithm, GMOEA [35], is used, with additional constraints introduced to expedite evolution and enhance outcomes. This is referred to as the GCMOEA.

Determining which links require added delay or TTL based on the MDT is essential. If the MDT of the current interface is lower than others, the delay is increased; if it equals the maximum MDT, TTL is added. The algorithm for this process is detailed in Algorithm 2.

The added delay values can be optimized with specified objective functions and constraints as follows:

$$\begin{aligned}
\min f_1(\mathbf{x}) &= \mathrm{Simi}(T, T') \\
&= 1 - \frac{\mathrm{TED}(T, T')}{\mathrm{TED}(T, Z) + \mathrm{TED}(T', Z)} \\
f_2(\mathbf{x}) &= \mathrm{Cost}(T, T') \\
&= \sum_{i,j \in D} \left(\frac{d'_{i,j}}{d_{i,j}} - 1 \right) \\
f_3(\mathbf{x}) &= -\mathrm{Dist}(l_B, l'_B) \\
&= -\frac{H\left(n_{l_{Bs}}, n_{l'_{Bs}}\right) + H\left(n_{l_{Bd}}, n_{l'_{Bd}}\right)}{2d_T} \\
\text{s.t.} 0 \le x_i &\le \max(\mathrm{MDT}), \quad i = 1, \ldots, N \\
g_1(\mathbf{x}) &= f_1(\mathbf{x}) - F_1 \le 0 \\
g_2(\mathbf{x}) &= f_2(\mathbf{x}) - F_2 \le 0
\end{aligned} \tag{22}$$

where Simi employs normalized tree edit distance (*TED*) to assess the similarity between the obfuscated topology T' and the original topology T, with Z representing the zero-node tree. The variable Cost indicates the growth delay ratio, taking into account the delay correlations $d_{i,j}$ and $d'_{i,j}$ between nodes in both topologies. The notations $n_{l_{Bs}}$, $n_{l_{Bd}}$, $n_{l'_{Bs}}$, and $n_{l'_{Bd}}$ refer to the starting and terminating nodes of the original and fabricated bottleneck links, respectively, while d_T indicates the longest network path.

The variable x_i is bounded between 0 and the maximum network delay, with constraints $g_1(\mathbf{x})$ ensuring that similarity does not exceed 0.8 and $g_2(\mathbf{x})$ capping growth delay at 0.8. The GCMOEA approach optimizes x for Pareto solutions using an elite reservation strategy for multiobjective optimization via GAN. The algorithm initializes a population P of size N, classifying solutions as "real" or "fake" through the GAN discriminator. Half of the population is labeled as fake for further refinement, and the GAN minimizes the discriminator loss during training.

To enhance robustness in constrained multiobjective optimization, GCMOEA integrates an adaptive constraint-aware ranking mechanism that jointly considers feasibility, constraint violation degree, dominance relations, and objective-space density. This approach enables the algorithm to effectively maintain evolutionary pressure in both feasible and infeasible regions, facilitating convergence while preserving diversity.

Given the solution $\mathbf{x}_i$, the total constraint violation is defined as

$$G_i = \sum_{k=1}^{K} \max(0, g_k(\mathbf{x}_i)) \tag{23}$$

where $g_k(\mathbf{x}_i)$ denotes the kth inequality constraint, and K is the number of constraints, which is set to 2. To incorporate constraint information into selection, we redefine the dominance relation $\mathrm{Dom}(i, j)$ as

$$\mathrm{Dom}(i,j) = \begin{cases} \text{true}, & \text{if } G_i = 0,\ G_j > 0 \\ \text{true}, & \text{if } G_i = G_j = 0 \text{ and } F_i < F_j \\ \text{true}, & \text{if } G_i > 0,\ G_j > 0 \text{ and } G_i < G_j \\ \text{false}, & \text{otherwise} \end{cases} \tag{24}$$

where F_i denotes the objective vector of x_i. This refined dominance structure ensures that feasible solutions are always preferred over infeasible ones and promotes progression toward feasibility when all candidates are infeasible.

To further differentiate among solutions, we define a raw fitness score for x_i based on the number and quality of other solutions it dominates

$$R(i) = \sum_{j:\mathrm{Dom}(j,i)=\text{true}} |S(j)| \tag{25}$$

where $S(j)$ is the support set of $\mathbf{x}_j$ defined as $S(j) = \{k \mid \mathrm{Dom}(j, k) = \text{true}\}$. This formulation prioritizes solutions that dominate many strong candidates. In parallel, a density penalty term $D(i)$ is introduced to avoid overcrowding in the objective space

$$D(i) = \frac{1}{d_i^{(k)} + \epsilon} \tag{26}$$

where $d_i^{(k)}$ is the Euclidean distance from $\mathbf{x}_i$ to its kth nearest neighbor in the objective space, and $\epsilon > 0$ is a small constant to avoid division by zero. The final fitness value is computed as

$$\mathrm{Fitness}(i) = R(i) + D(i) \tag{27}$$

enabling a balance between convergence pressure and solution diversity.

Building on this fitness-guided ranking mechanism, GCMOEA selects high-potential solutions—particularly those exhibiting distinctiveness from the current Pareto front—as pseudo-labeled elite samples for generative modeling. These samples serve as reference points for the training of the GAN-based generator, which aims to explore the objective space more effectively by modeling the distribution of promising solutions.

The latent code $\mathbf{z}$ for the GAN generator is sampled from a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, estimated from the top-$N/2$ elite solutions r_i in the objective space

$$\mu = \frac{1}{N/2} \sum_{i=1}^{N/2} \mathbf{r}_i \tag{28}$$

$$\Sigma = \frac{1}{N/2 - 1} \sum_{i=1}^{N/2} (\mathbf{r}_i - \mu)(\mathbf{r}_i - \mu)^T. \tag{29}$$

This estimation reflects the evolving geometry of the objective space and allows the generator to sample from promising regions adaptively. The GAN is trained with a standard adversarial loss composed of three components

$$\max_D V(D) = \mathbb{E}_{x_r \sim P_r}\big[\log D(r)\big] + \mathbb{E}_{x_f \sim P_f}\big[\log(1 - D(f))\big] + \mathbb{E}_z\big[\log(1 - D(G(z)))\big] \tag{30}$$

where P_r and P_f denote the distributions of real and generated samples, respectively, and $G(\cdot)$ is the generator function. The discriminator D is trained to distinguish between real elite solutions and generated candidates, thereby guiding G to match the target distribution.

During the offspring generation phase, GCMOEA integrates generative modeling with traditional evolutionary operators to achieve a balance between global exploration and local exploitation. Specifically, a GAN-based generator samples candidate solutions from the latent distribution learned from high-quality individuals, facilitating exploration of underrepresented but promising regions in the search space. Currently, the algorithm employs simulated binary crossover (SBX) and polynomial mutation (PM) to enhance solution diversity and improve convergence precision.

Given two parent individuals x_1 and x_2, SBX generates two offspring as follows:

$$y_1 = 0.5 \cdot [(x_1 + x_2) - \beta \cdot |x_1 - x_2|] \tag{31}$$

$$y_2 = 0.5 \cdot [(x_1 + x_2) + \beta \cdot |x_1 - x_2|] \tag{32}$$

where β is a spread factor used to control the exploration range, which is drawn from a distribution parameterized by crossover distribution index η_c and the solution dimensionality N. A higher η_c results in more similarity to the parents. We

set it to 20 following the default configuration in GMOEA, which provides a balance between exploration depth and convergence rate. Each offspring gene is further subject to PM with probability $1/N$. The mutated gene value y_i' is computed as

$$y_i' = y_i + \Delta \cdot \left(x^{\text{upper}} - x^{\text{lower}}\right) \tag{33}$$

where the mutation strength Δ follows a polynomial distribution

$$\Delta = \begin{cases} \left(2\mu + (1-2\mu)(1-y_{\text{norm}})^{d_m+1}\right)^{\frac{1}{d_m+1}} - 1, & \mu \le 0.5 \\ 1 - \left(2(1-\mu) + 2(\mu-0.5)(1-y_{\text{norm}})^{d_m+1}\right)^{\frac{1}{d_m+1}}, & \text{otherwise} \end{cases} \tag{34}$$

where $\mu \sim U(0,1)$ and d_m is the distribution index that controls the mutation granularity, a larger d_m results in smaller perturbations. We also set d_m to 20, consistent with the original GMOEA design, which aims to evaluate the effectiveness of the baseline under standard parameter settings rather than to fine-tune it through hyperparameter optimization.

Here, y_{norm} denotes the normalized position of x_i within its predefined bounds

$$y_{\text{norm}} = \frac{x_i - x_i^{\text{lower}}}{x_i^{\text{upper}} - x_i^{\text{lower}}}. \tag{35}$$

To mitigate potential instability introduced by the GAN component and avoid premature overfitting, a decay-based scheduling mechanism is introduced to adaptively control the involvement of GAN in offspring generation. The probability of utilizing GAN at generation i is defined as

$$P_{\text{GAN}}(i) = \left(1 - \frac{i}{W_{\max}}\right)^2 \tag{36}$$

where $W_{\max}$ is the total number of generations. This scheduling strategy ensures that the GAN contributes more during early stages for facilitating global exploration, and gradually recedes in later stages, allowing traditional operators to dominate for convergence refinement. The overall design maintains a controlled balance between generative diversity and evolutionary stability throughout the optimization process.

3) Theoretical Guarantee and Pareto Approximation Analysis: Despite the heuristic nature of GCMOEA, this section provides a theoretical perspective to demonstrate its capability to approximate the Pareto front asymptotically through iterative evolution. Specifically, we analyze the convergence properties of the population dynamics and the preservation of Pareto-optimal solutions under the elitist evolutionary framework.

Let $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$ be two candidate solutions. According to the standard Pareto dominance relation, $\mathbf{x}_1$ is said to dominate $\mathbf{x}_2$ (denoted $\mathbf{x}_1 \prec \mathbf{x}_2$) if

$$\mathbf{x}_1 \prec \mathbf{x}_2 \Longleftrightarrow \begin{cases} f_i(\mathbf{x}_1) \le f_i(\mathbf{x}_2) & \forall i \in \{1,\ldots,m\} \\ f_j(\mathbf{x}_1) < f_j(\mathbf{x}_2), & \exists j \in \{1,\ldots,m\} \end{cases} \tag{37}$$

where f_i denotes the ith objective function in an m-objective optimization task.

To ensure convergence toward the Pareto front, GCMOEA adopts a generational elitism mechanism. At generation t, the current population $\mathcal{P}_t$ produces an offspring population $\mathcal{Q}_t$ through variation operators. The population for the next generation $\mathcal{P}_{t+1}$ is then selected from the union $\mathcal{P}_t \cup \mathcal{Q}_t$ using nondominated sorting and diversity preservation, such that

$$\mathcal{P}_{t+1} \subseteq \mathcal{ND}(\mathcal{P}_t \cup \mathcal{Q}_t) \tag{38}$$

where $\mathcal{ND}(\cdot)$ denotes the operator that returns the nondominated set. This ensures that no individual in $\mathcal{P}_{t+1}$ is dominated by any solution in the previous generation or among its peers, thereby maintaining Pareto dominance.

To quantify the population's convergence toward the Pareto front, we employ a dominance coverage metric $C(\mathcal{P}_{t+1}, \mathcal{P}_t)$ defined as

$$C(\mathcal{P}_{t+1}, \mathcal{P}_t) = \frac{|\{\mathbf{x} \in \mathcal{P}_t \mid \exists \mathbf{y} \in \mathcal{P}_{t+1},\ \mathbf{y} \preceq \mathbf{x}\}|}{|\mathcal{P}_t|} \tag{39}$$

where a higher value of C ($\to 1$) implies stronger selection pressure and faster convergence toward the Pareto-optimal set, since a larger proportion of the parent population is weakly dominated or improved upon by offspring.

To mitigate premature convergence and maintain solution diversity, a crowding distance mechanism is applied during the selection phase, ensuring a well-distributed approximation across the objective space. The interplay between convergence pressure and diversity control underpins the theoretical soundness of GCMOEA.

Furthermore, the incorporation of GAN-based offspring generation extends the exploratory capacity by introducing diversity beyond the convex hull of existing solutions. However, this mechanism inherently carries the risk of generating infeasible or suboptimal candidates, especially in early generations. To address this, we design a linearly decaying probability function $P_{\text{GAN}}(i)$, which ensures the GAN plays a more exploratory role in the early stage but gradually gives way to SBX and PM in later generations for stable convergence and refined Pareto compliance.

Compared with traditional evolutionary algorithms, such as NSGA-II and SPEA2, which rely solely on variation operators and population heuristics, GCMOEA exhibits a data-driven adaptation capability. The integration of GAN-based modeling enables fine-grained exploration in high-dimensional search spaces, where conventional dominance and crowding-based strategies often exhibit premature convergence or insufficient coverage of the Pareto front.

More critically, in many-objective scenarios where the Pareto set is complex and sparsely distributed, the adversarial dynamics allow the generator to internalize latent structural patterns, facilitating targeted sampling and diversity enhancement. Despite the minimal overhead introduced by GAN training, the framework adopts a lightweight network architecture and semi-supervised updates, ensuring computational feasibility.

Ultimately, this synergy between constraint-aware ranking and GAN-guided sampling significantly improves the evolutionary efficiency of GCMOEA, yielding superior performance in terms of both convergence speed and Pareto front diversity—advantages that are substantiated by our experimental results across multiple benchmark settings.

Under the assumptions of finite population size and fixed genetic operator parameters (crossover and mutation rates) satisfying the conditions of weak ergodicity, the evolutionary process of GCMOEA can be modeled as a finite-state Markov chain. Within this formalism, the algorithm's dynamics induce a stochastic transition system where, over an infinite time horizon, the distribution over populations asymptotically approaches a stationary distribution. This stationary distribution is biased toward regions in the decision space that correspond to high-quality, nondominated solutions, thereby enabling weak convergence to the Pareto-optimal set.

Moreover, the effective search domain is projected onto the feasible subspace by introducing a constraint-aware ranking mechanism and feasibility-biased selection. This introduces an implicit regularization on the Markov process, steering the evolutionary trajectory toward the constrained Pareto front. Although a strict proof of global convergence in the sense of attaining the actual Pareto front is generally intractable due to the combinatorial and stochastic nature of the problem, convergence in expectation, that is, the population converging in distribution toward a set of high-quality feasible solutions, can be theoretically ensured. This level of convergence is consistent with the analytical frameworks adopted by mainstream evolutionary multiobjective optimization algorithms, and aligns with the concept of asymptotic stability under constraint-preserving dynamics.

IV. Experiments

We evaluate the effectiveness of OptiPathNet in real networks, detailing the experiment design, evaluation setting, and core model performance. To assess OptiPathNet's performance, we created three real-world network topologies using Mininet, based on small, medium, and large networks from the Internet Topology Zoo shown in Fig. 6. We simulated probe and benign traffic, along with malicious activities like blind LFAs (BLFAs), port scanning (PS), and packet injection (PI).

A. Experiment Setup

1) Cost Objectives: To ensure deployability in resource-constrained environments, such as IoT networks, we impose strict control over system overhead across three key dimensions.

1) *Traffic Identification Overhead:* To reduce computational cost without compromising detection accuracy, we employ KD to compress deep detection models into lightweight variants. This significantly lowers memory and computation demands, enabling practical deployment on edge devices while retaining robust identification performance.
2) *Camouflage-Induced Latency Cost:* The performance impact of delay injection is assessed via the average ratio of added latency to normal transmission time. This metric reflects the tradeoff between obfuscation strength and user-perceived service quality, and is critical for evaluating system acceptability in latency-sensitive scenarios.

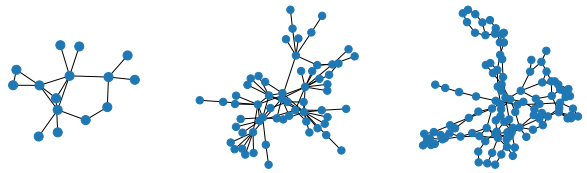

Fig. 6. Multiscale topological architectures in IoT networks. (a) Claranet (Node:15, Link:18). (b) Garr201112 (Node:61, Link:75). (c) Deltacom (Node:113, Link:161).

3) *Decision-Making Overhead:* We measure the runtime cost of generating delay injection strategies and executing control logic to assess real-time feasibility. This is complemented by the probe traffic detection accuracy, which jointly reflects the system's responsiveness and its ability to maintain effective defense under constrained execution budgets.

2) Baseline: For comparison, we selected specific schemes to evaluate traffic detection performance and NTO. It is worth noting that OptiPathNet focuses on balancing link traffic by increasing latency and adding TTL to bottleneck links rather than modifying the network topology, distinguishing it from existing obfuscation schemes (Table I).

3) Implementation Details: OptiPathNet is implemented within a software-defined networking (SDN) framework that programs packet forwarding rules in SDN controllers. It utilizes real-world SDN topology data to simulate dynamic networking environments in a virtual setting using Mininet. The RYU controller handles probe packet inspection and delay manipulation, allowing for customizable SDN control logic and OpenFlow communication with switches for dynamic traffic control. Written in Python, OptiPathNet enables seamless integration and simulation of real-time network topological changes.

4) Dataset: The Internet Topology Zoo[1] contains 261 real-world network topologies from around the globe, designed for network-related research and experimentation. The topologies in Topology Zoo are provided in GML or GraphML formats, enabling flexibility in network simulations and analysis for IoT and other applications.

1) *Claranet:* This topology encapsulates the intricate network configuration of Claranet, a managed service provider offering cloud, security, and connectivity services across Europe. It is a smart home IoT network, consisting of a central gateway and multiple interconnected room devices, reflecting the growing complexity of modern residential IoT systems.
2) *GARR201112:* This topology models a multilayer IoT infrastructure, such as the Industrial IoT. The structure includes edge servers, field controllers, and sensors, making it ideal for simulating IIoT scenarios like smart manufacturing or energy management.
3) *Deltacom:* This structure depicts the network topology of a telecommunications provider in the Southeastern

[1] https://topology-zoo.org/explore.html

TABLE I
BASELINE METHODS

Model	Description	Q1|Q2
Flow-MAE [36]	The original model without simplification.	◐
Mini-Flow-MAE	Simplify the model by reducing parameters without employing KD.	◐
NoProt	Doesn't run any topology obfuscation mechanism on the target network topology.	◑
Antitomo [25]	Candidate forests are randomly generated, and the optimal candidate is selected based on multiple objectives. A delay increase is then applied to the probe flows.	◑
RndPathNet	The links requiring delay growth are identified through BFS, and delays are randomly added to the probe flow.	◑
NSGAPathNet	The BFS algorithm is used to identify the links necessitating delay swiftly, and the suitable delay value for probe flows is reckoned by $NSGA\text{-}II$.	◑
ProTo [28]	The semi-supervised KNN algorithm with incremental updating is used to detect probe flow. Then, a routing matrix is randomly generated as an obfuscation topology to compute the delay.	●
OptiPathNet	Distilled-Flow-MAE identifies probe flow, while BFS quickly finds links necessitating delay/TTL increase without pre-constructing the obfuscation matrix or topology. $GCMOEA$ then determines the appropriate delay for probe flows.	●

United States. It focuses on backbone connectivity and regional coverage and is well-suited for simulating IoT use cases involving wide-area IoT deployments, such as regional smart city applications or large-scale sensor networks.

5) Attack Traffics for Evaluating: We evaluate the effectiveness of OptiPathNet in both reconnaissance traffic detection and link-level mitigation. To do this, we introduce a diverse set of representative traffic types in our experimental setup, covering the full attack lifecycle of LFAs, from initial probing to execution.

1) *Probe traffic* models typical end-to-end inference behaviors, characterized by high-frequency packet transmissions targeting a narrow set of hosts, with either pronounced or minimal payload variability and size fluctuations designed to perturb RTT and reveal path structure.
2) *Traceroute* simulates internal cooperative probing, where attackers incrementally increase TTL values to elicit ICMP responses and infer hop-level topology; such traffic typically features small, tightly ranged TTLs and regular interpacket timing.
3) *BLFA* reflects the execution phase, wherein attackers induce congestion by sending high volumes of legitimate yet redundant packets to targeted links; this traffic is highly bursty, short-lived, and concentrated on specific paths with medium to large payload sizes.
4) *PS* emulates service fingerprinting via diverse TCP/UDP port access attempts, with minimal variation in destination IPs but clear signatures in transport-layer flags.
5) *PI* involves the insertion of malformed or unauthorized packets into ongoing sessions to achieve command injection or protocol spoofing, exhibiting structural anomalies and protocol violations.
6) *Challenge collapsar (CC)* represents an application-layer DoS attack using massive volumes of syntactically valid requests (e.g., HTTP GET) to deplete server resources; such traffic shows high request frequency, concentrated target domains, and superficially normal protocol behavior.

Among these, Probe and Traceroute represent the reconnaissance stage, enabling assessment of OptiPathNet's early-stage detection and reaction capabilities. The remaining types correspond to flooding or DoS variants, reflecting the system's ability to generalize across attack execution scenarios. By incorporating this heterogeneous traffic spectrum, we aim to rigorously validate OptiPathNet's detection adaptability and defense robustness in realistic network environments.

6) Evaluation Metrics: In evaluating the traffic-aware dynamic obfuscation method, we assess OptiPathNet's recognition performance through various metrics shown in Table II.

1) *Accuracy* measures the ratio of correctly identified packets of a specific type to the total identified in Mininet. Let the set of traffic be $T = \{(t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n)\}$, where t represents network traffic and y denotes its category, which can be "normal," "probe," "traceroute," "BLFA," "PS," "PI," or "C&C." Here, δ equals 1 if the condition holds and 0 otherwise. We denote y_i as the true category and y_i' as the detected category.
2) *False positive rate* is the number of incorrect identifications divided by the total for that type.
3) *Recall rate* indicates how many true packets are accurately identified.
4) *Detection cost* is the time incurred divided by delay, gauged by sending the same packet density and number twice using Mininet. The first test monitors traffic without the SDN switch configuration; the second uses RYU controller for monitoring. Latency before deployment is represented as $RL = \{(l_1, d_1), (l_2, d_2), \ldots, (l_m, d_m)\}$, with the post-deployment delay as $RL' = \{(l_1, d_1'), (l_2, d_2'), \ldots, (l_m, d_m')\}$.

For evaluating multiobjective evolutionary topology obfuscation in OptiPathNet, four metrics are used.

1) *Security:* Measures the similarity between the attacker's inferred network topology based on end-to-end delay and the actual topology.
2) *Obfuscation Cost:* Indicates the growth rate of associated delay.
3) *Obfuscation Efficiency:* Time required to compute the added delay.
4) *Concealment:* Ratio of the distance between the deviation of the bottleneck link (DBL) and the longest path.

All reported results are computed as the average of 100 independent trials, each with randomized seeds and reinitialized experimental conditions. Mean performance metrics are reported along with standard deviation intervals ($\pm 1\sigma$) to quantify statistical significance and demonstrate result stability.

TABLE II
EVALUATION METRICS

Metrics	Formula
Accuracy	$\mathrm{ACC}(y) = \frac{\sum_{i=1}^{n} \delta(y_i'=y \wedge y_i=y)}{\sum_{i=1}^{n} \delta(y_i'=y)}$
False Positive Rate	$\mathrm{FPR}(y) = \frac{\sum_{i=1}^{n} \delta(y_i'=Ry \mid y_i \neq Ry)}{\sum_{i=1}^{n} \delta(y_i \neq Ry)}$
Recall Rate	$\mathrm{RR}(y) = \frac{\sum_{i=1}^{n} \delta(y_i=y \wedge y_i'=y)}{\sum_{i=1}^{n} \delta(y_i=y)}$
Detection Cost	$\mathrm{DC} = \frac{\sum_{i=1}^{m}(d_i'-d_i)}{\sum_{i=1}^{m} d_i}$
Security	$\mathrm{Simi}(T,T') = 1 - \frac{\mathrm{TED}(T,T')}{\mathrm{TED}(T,Z)+\mathrm{TED}(T',Z)}$
Obfuscation Cost	$\mathrm{Cost}(T,T') = \sum_{i,j \in D} \left(\frac{d_{i,j}'}{d_{i,j}} - 1 \right)$
Obfuscation Efficiency	$T = T_{\mathrm{obfus}} - T_{\mathrm{orig}}$
Concealment	$\mathrm{Dist}(l_B, l_B') = \frac{H(n_{l_{Bs}}, n_{l_{Bs}'}) + H(n_{l_{Bd}}, n_{l_{Bd}'})}{2d_T}$

B. Evaluation of Traffic-Aware Dynamic Obfuscation

1) Accuracy: To evaluate the effect of traffic-aware dynamic obfuscation, we conducted experiments using the distillation method for the traffic identification model. We compared three distillation algorithms, response-based, feature-based, and relationship-based, with a nondistilled direct parameter reduction model in Fig. 7.

Fig. 7(a) and (b) compares loss values, showing that distillation algorithms effectively reduced training and evaluation losses while preserving the teacher model's performance. The nondistilled model had high loss values, indicating a loss of essential features during parameter reduction. Response-based distillation improved losses but focused mainly on final outputs. In contrast, feature-based (training: 0.137, evaluation: 0.063) and relationship-based distillation (training: 0.098, evaluation: 0.032) performed best, capturing fine-grained knowledge from the teacher model.

Fig. 7(c) and (d) shows that distilled models surpassed the nondistilled model in accuracy and recall. The nondistilled model had an accuracy and recall of 0.976, with instability during training. Feature-based distillation led with 0.990, relationship-based achieved 0.987, and response-based reached 0.986, with the latter excelling in capturing feature interdependencies.

In summary, distillation's effectiveness hinges on utilizing intermediate features rather than just the final output. Feature-based and relationship-based distillation methods excel in managing complex IoT traffic features, enhancing the student model's generalization and robustness. These insights aid in designing efficient traffic identification models and optimizing traffic-aware dynamic obfuscation. Experimental results show that the method effectively balances model compression with performance preservation, making it ideal for IoT traffic identification tasks.

2) Efficiency–Accuracy Tradeoff: Fig. 8 evaluates the algorithm's effectiveness in balancing detection performance and computational efficiency, which is critical for IoT scenarios with resource constraints. Through KD, Distilled-Flow-MAE achieved significant model simplification while maintaining high accuracy, making it particularly suitable for IoT applications like edge computing and lightweight device deployments.

Distilled-Flow-MAE preserved 99% of the accuracy of the original Flow-MAE, i.e., ACC: 98.5%, RR: 99%, while reducing model size by 11.93 times and computational costs from 1.13% to 0.71%. In contrast, though efficient, Mini-Flow-MAE showed degraded accuracy; CC dropped by 2.46%, and FPR increased by 59.15%, highlighting the risk of direct parameter reduction in losing critical feature representations. Distilled-Flow-MAE processed 114.98 samples/s, making it ideal for real-time IoT traffic detection. Its reduced size 28.53 MB and low computational requirements enable deployment on resource-constrained IoT devices without compromising detection accuracy.

By leveraging KD, Distilled-Flow-MAE strikes the optimal balance between efficiency and accuracy. It addresses IoT-specific challenges, such as limited memory, computation, and real-time traffic monitoring, making it well-suited for applications like anomaly detection in smart homes, industrial IoT systems, and edge AI deployments.

3) Robustness: The experiments demonstrated in Fig. 9 depicted that OptiPathNet excels in detection robustness, maintaining high accuracy and low FPR across diverse and complex traffic scenarios, making it ideal for IoT applications with dynamic and heterogeneous traffic. OptiPathNet achieves over 99% detection rates across all network sizes, stabilizing faster than ProTo. This early saturation highlights its ability to adapt to increased classification complexity without performance degradation. With an FPR consistently below 1%, OptiPathNet reliably distinguishes detection flows from regular traffic, outperforming ProTo, which struggles in more complex scenarios. Unlike ProTo, OptiPathNet links fine-grained classification with detection, ensuring efficiency and accuracy even under high traffic diversity, a crucial feature for IoT networks.

OptiPathNet's robustness makes it particularly valuable for IoT environments, such as anomaly detection in smart homes, industrial IoT, and real-time edge monitoring. Its ability to handle diverse traffic patterns and scale efficiently ensures reliable performance in resource-constrained and dynamic IoT scenarios.

C. Evaluation of Multiobjective Evolutionary Topology Obfuscation

1) Obfuscation Effectiveness: Fig. 10(a) evaluates the obfuscation effectiveness of topology manipulation mechanisms, emphasizing their ability to reduce network similarity and confuse attackers. It can be seen that OptiPathNet achieves a similarity score below 0.4 for networks like Garr201112, outperforming NSGAPathNet, while a more simple method like ProTo exceeds 0.6. This demonstrates OptiPathNet's ability to obscure topology effectively by altering delay values, making it more resilient to attacker inference in complex networks. As network size grows, OptiPathNet and NSGAPathNet leverage more conditions to confound attackers, while ProTo struggles. Larger networks amplify obfuscation effectiveness, highlighting OptiPathNet's scalability and adaptability to diverse traffic patterns. To empirically substantiate the convergence behavior and algorithmic stability of GCMOEA, we analyze the evolution of objective values over increasing generation counts and subpopulation sizes. As shown in Table III, results are

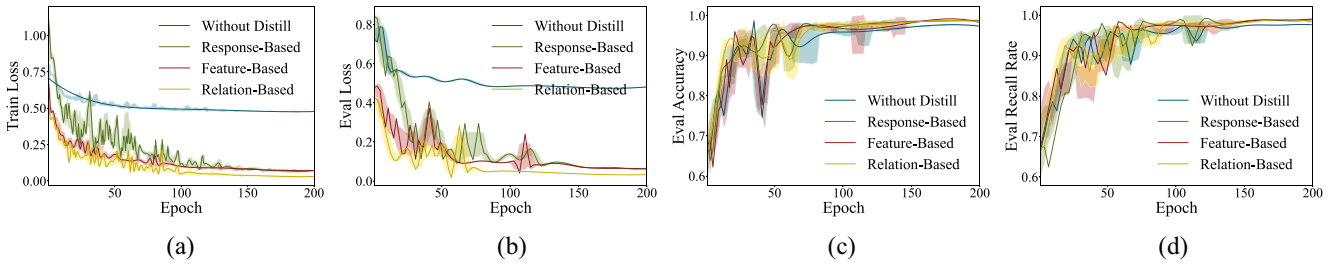

Fig. 7. Evaluation of traffic-aware dynamic obfuscation. (a) Train Loss. (b) Eval. Loss. (c) Eval. Accuracy. (d) Eval. Recall.

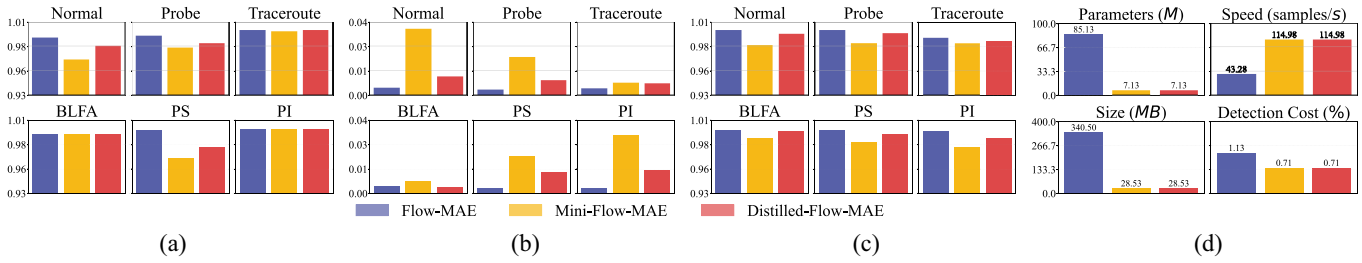

Fig. 8. Evaluation of efficiency and accuracy tradeoff. (a) ACC. (b) FPR. (c) RR. (d) DC.

reported for early (iter 6), intermediate (iter 50), and later (iter 100) stages of evolution. Across all reported objectives, a clear trend of monotonic improvement is observed, particularly in high-dimensional objectives where the gain in later generations becomes more prominent. This reflects the progressive refinement of the solution set and indicates that the approximated Pareto front becomes increasingly dense and representative over time. Such empirical trends are in line with the theoretical convergence expectations discussed above. The algorithm demonstrates not only convergence in terms of objective value improvement but also increased solution stability across generations. Note that the convergence rate is coupled with computational cost. As the number of generations increases, the per-run computational burden also rises, which can present practical constraints in resource-limited deployment scenarios. To this end, the experimental configuration adopts a pragmatic tradeoff: selecting an upper bound on the number of generations that balances convergence quality with computational efficiency. This design choice ensures the feasibility of the algorithm for real-world applications, while still preserving its convergence advantages.

Accordingly, the robustness and scalability of OptiPathNet render it suitable for IoT networks, where topology obfuscation is imperative for defending against targeted attacks. Its capacity to adapt dynamically to diverse traffic patterns ensures effective obfuscation in smart homes, industrial IoT, and edge computing environments, providing a robust defence mechanism with minimal computational overhead.

2) Obfuscation Cost: The experiment in Fig. 10(b) evaluates the obfuscation cost by analyzing delay growth rates, highlighting how different methods balance obfuscation effectiveness with network performance. OptiPathNet maintains delay growth rates below 0.6 in networks like Garr201112 and Deltacom, outperforming NSGAPathNet (close to 0.7) and significantly surpassing ProTo (above 1). Targeting probe traffic reduces costs while avoiding unnecessary delays, unlike

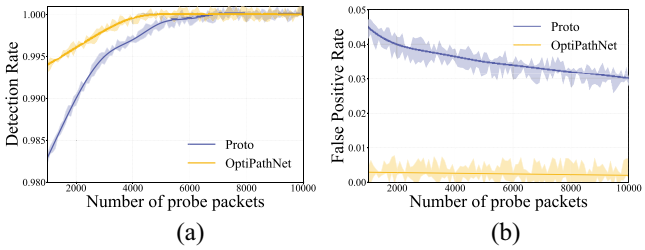

Fig. 9. Performance cross diverse and complex traffic scenario. (a) Detection rate. (b) FPR.

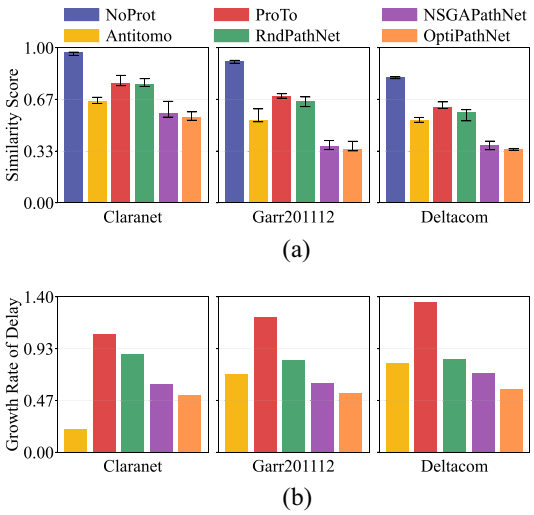

Fig. 10. Evaluation of obfuscation effectiveness with multiscale topologies. (a) Similarity with the original topology. (b) Obfuscation cost.

Antitomo, which delays all traffic indiscriminately, leading to inefficiency. For Scalability and Cost Efficiency, OptiPathNet excels in larger networks, achieving lower obfuscation costs with fewer iterations compared to NSGAPathNet in Table III.

TABLE III
EVALUATION OF DIFFERENT POPULATION SIZE OVER ITERATION

Dataset	Iteration	Method	Population Size (Simi.: %, Cost: %, Time: s)																							
			10			20			25			30			35			40									
			Simi	Cost	Time	Simi	Cost	Time	Simi	Cost	Time	Simi	Cost	Time	Simi	Cost	Time	Simi	Cost	Time	Simi	Cost	Time	Simi	Cost	Time
Claranet	6	$Nsga$	65.1	68.9	0.3	64.1	68.4	0.4	64.2	67.4	0.5	64.1	66.4	0.6	63.2	66.1	0.8	63.1	66.0	0.9	62.9	65.9	1.2	62.4	65.4	2.5
		$Opti$	62.1	64.1	11.4	62.1	63.4	12.8	61.0	62.4	14.9	60.1	61.1	19.9	60.1	61.8	24.9	59.1	60.4	31.1	59.0	59.4	35.0	59.0	58.4	38.0
	50	$Nsga$	62.9	65.4	2.5	61.8	64.1	2.6	60.8	64.0	2.5	59.8	63.4	2.5	59.0	63.2	3.2	58.8	63.1	4.0	57.8	62.6	5.1	57.0	62.4	5.7
		$Opti$	60.2	58.4	23.6	59.1	57.4	38.0	50.0	56.5	49.8	57.9	55.1	63.5	57.1	54.4	74.5	56.4	54.4	88.7	56.1	54.4	118.1	56.0	51.1	125.6
	100	$Nsga$	60.8	62.1	3.1	59.8	61.4	3.9	57.8	61.3	4.3	57.2	61.0	4.8	56.1	61.6	6.1	56.2	61.4	7.6	56.1	60.9	10.0	55.8	60.4	11.2
		$Opti$	57.4	57.4	29.6	55.1	56.4	45.6	55.0	55.4	57.8	55.1	54.1	69.5	55.1	54.3	80.5	55.1	53.4	98.7	54.9	53.3	128.1	54.9	52.4	135.6
Garr201112	6	$Nsga$	50.0	70.1	9.3	50.0	69.8	18.2	49.5	69.4	23.4	49.4	68.8	35.9	49.0	68.0	63.7	48.9	68.0	66.9	48.9	67.8	78.1	48.5	68.0	92.0
		$Opti$	47.2	65.1	18.9	46.5	64.8	29.1	46.9	64.8	37.2	46.5	64.6	57.1	46.5	64.4	69.6	46.1	64.4	91.6	46.0	64.4	115.1	45.9	64.3	137.7
	50	$Nsga$	43.9	66.8	53.3	42.4	66.8	91.9	41.6	66.0	110.9	39.0	65.8	183.0	38.2	64.7	236.5	36.9	63.7	310.1	36.3	62.8	409.5	35.6	62.8	516.0
		$Opti$	44.6	62.5	82.8	43.6	61.5	137.0	41.0	60.8	188.1	39.9	59.8	263.9	39.4	57.9	342.0	39.0	57.7	432.0	38.4	57.4	503.4	38.0	56.8	600.1
	100	$Nsga$	40.3	63.8	96.1	38.5	62.8	179.1	37.4	62.8	260.0	35.4	62.4	350.2	35.1	62.6	460.1	35.1	62.0	613.2	57.4	57.4	852.6	55.1	56.4	977.6
		$Opti$	38.4	59.8	168.6	37.9	58.0	275.9	36.8	57.8	342.5	35.7	56.7	484.3	35.1	55.6	580.5	34.6	54.8	683.1	34.5	54.3	812.4	34.0	54.4	935.6
Deltacom	6	$Nsga$	48.5	80.2	23.6	47.5	79.5	24.4	46.5	79.5	31.9	45.4	78.5	48.2	44.5	78.5	62.8	44.9	78.4	82.2	43.5	77.5	102.0	42.4	77.5	123.4
		$Opti$	47.5	64.6	29.7	46.9	64.1	48.0	46.2	63.5	53.0	45.9	63.7	79.2	45.6	63.7	79.2	45.1	63.5	130.4	45.0	62.8	163.2	44.9	62.8	193.3
	50	$Nsga$	45.9	76.3	119.4	44.7	75.4	122.0	43.5	74.5	155.3	42.4	74.3	269.0	41.4	74.4	340.7	40.5	74.2	459.3	39.4	74.1	567.2	39.1	74.0	666.6
		$Opti$	42.5	62.4	100.2	41.4	61.4	208.2	39.4	59.5	328.8	38.3	59.1	421.5	37.1	59.4	530.5	37.4	59.4	648.7	36.1	58.9	758.1	35.6	58.5	875.6
	100	$Nsga$	42.1	74.2	241.7	42.4	73.5	246.8	40.4	72.5	352.5	39.4	71.5	446.7	39.1	71.5	627.8	38.7	71.4	789.3	38.6	71.9	1158.1	38.5	71.4	1429.6
		$Opti$	39.5	58.9	300.6	38.4	58.3	465.6	38.0	58.2	587.8	37.1	58.1	709.5	36.5	58.0	890.5	35.1	57.9	1058.7	35.9	57.6	1128.1	35.4	57.4	1205.6

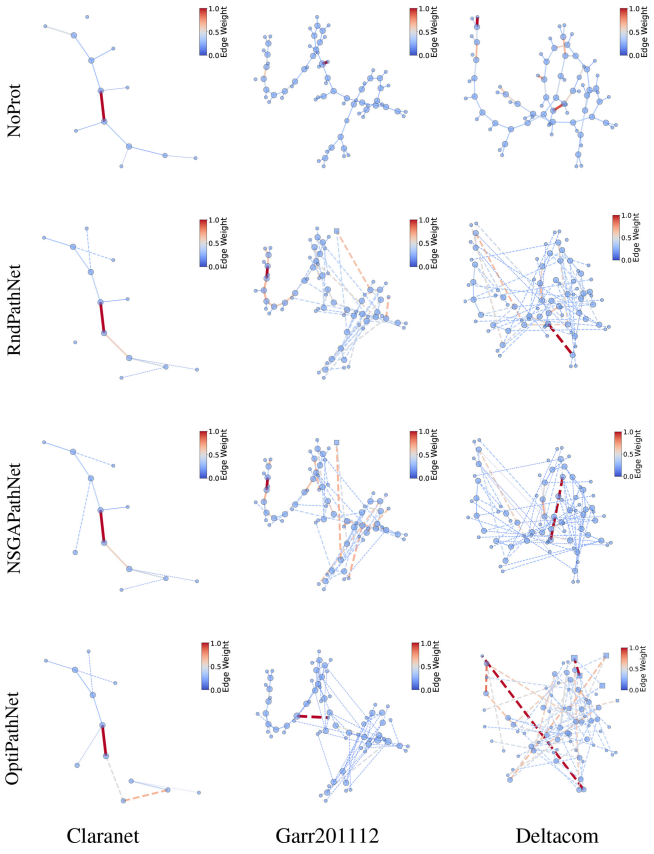

Fig. 11. Visual comparison of obfuscated network structures under different topologies.

It balances scalability and performance, well-suited for complex, resource-constrained environments.

In IoT networks, where low latency and efficiency are critical, OptiPathNet's ability to minimize delay growth while maintaining robust obfuscation ensures seamless operation for latency-sensitive applications, such as smart homes and industrial IoT. Its selective targeting approach avoids performance degradation, offering a scalable, cost-effective solution for securing IoT networks against attackers.

3) Computational Efficiency: Table III illustrates the variation in runtime across small, medium, and large-scale network topologies as the offspring population size and optimization iterations increase. Experimental results indicate that larger populations significantly increase computational latency—exponentially so in large topologies—yet yield only marginal improvements in security (as measured by structural similarity) and limited gains in cost detection metrics, resulting in suboptimal efficiency. From a practical standpoint, we recommend adopting moderate population sizes and iteration counts that satisfy minimum security requirements while ensuring a favorable balance between runtime and defense effectiveness. Under default settings (50 iterations and population size of 10), the average runtime per optimization round is approximately 23.6 s in small topologies and increases to 100.2 s in large ones (evaluated on an Intel Xeon 64-core CPU). Runtime is closely tied to both topological complexity and evolutionary parameters, but remains controllable and adjustable.

It is worth emphasizing that the OptiPathNet framework is particularly well-suited for periodic defense scenarios. Although network traffic is dynamic, the macro-level structure of routing paths and connectivity tends to remain stable over short time windows. As such, real-time updates at the packet level are unnecessary; instead, topology obfuscation can be periodically optimized using snapshot-based strategies. This approach effectively balances defense robustness and computational overhead, making it especially suitable for environments with high demands on stability and security, such as critical infrastructure, enterprise networks, or data centers.

4) Obfuscation and Concealment: The experiment shown in Fig. 11 fundamentally examines how strategic distortion versus naive noise injection impacts adversarial perception of network infrastructure. The visualization in Fig. 3 (solid lines: actual Noprot links; dashed lines: attacker-inferred phantom links; red lines: core bottleneck links) reveals critical distinctions in obfuscation quality. While all three methods attempt to mislead topology inference through delay manipulation, their concealment efficacy diverges in how they exploit the attacker's cognitive blind spots.

It is demonstrated that how network obfuscation strategies manipulate adversarial perception through cognitive topology poisoning. While baseline methods

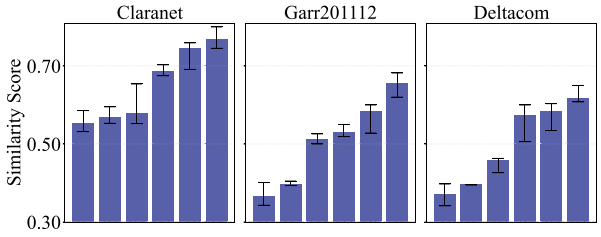

Fig. 12. Obfuscation effectiveness variation over time.

(RndPathNet/NSGAPathNet) generate isolated decoys (dashed lines), OptiPathNet's multiobjective optimization creates self-reinforcing phantom bottlenecks (red/dashed overlaps) that mimic organic congestion patterns across scales. In IoT ecosystems, this approach delivers unique advantages: 1) resource-aware deception leverages micro-delays indistinguishable from inherent device heterogeneity; 2) dynamic camouflage exploits IoT networks' natural topology fluctuations to mask artificial distortions; and 3) decentralized persistence sustains obfuscation through coordinated edge-device interactions without centralized control. The critical innovation lies not in hiding infrastructure but in weaponizing attackers' inference models to accept fabricated bottlenecks (red lines) as legitimate network artifacts—particularly vital for IoT deployments where physical concealment is impossible and attack surfaces evolve unpredictably.

Fig. 12 unveils time-aware concealment engineering where OptiPathNet's decaying obfuscation (confounding effect < 0.67 in large networks) strategically exploits IoT's intrinsic dynamics. Unlike rigid methods whose static deception patterns conflict with real-world network evolution, OptiPathNet synchronizes phantom bottleneck erosion with IoT's organic churn. Its adaptive delay distortions mimic natural device intermittency while maintaining critical red-line bottlenecks as persistent perceptual anchors. The diminishing confounding effect becomes a weaponized feature; attackers interpret controlled deception decay as evidence of "authentic" network behavior. At the same time, IoT's inherent instability perpetually renews the attack surface for sustained misdirection.

V. Related Work

Topology protection remains a central topic in network security [19], [20], [25], [26], [27], [28], [30], particularly under the persistent threat of active probing attacks. Extensive research has explored various forms of topology obfuscation and camouflage strategies. These methods are generally categorized based on adversarial capabilities into two major classes: 1) defenses against internal cooperative probing and 2) defenses against external end-to-end inference. To provide a structured overview, we summarize representative approaches in each category and discuss their applicability.

A. Topology Obfuscation for Internal Cooperative Inference

These techniques typically manipulate TTL values of probe packets, fabricate return paths, or spoof response behaviors to mislead attackers' inferences regarding the underlying topology.

Representative methods include Netobufu [30] and Linkbait [20]. Netobufu generates fake routing paths and node identifiers to mislead attackers and reduce their ability to identify critical nodes. Linkbait redirects attacker attention to decoy links near bottlenecks, the so-called "trap links" by manipulating traffic redirection and flow distribution. EqualNet [26] adjusts the routing of probe traffic across paths to balance the load and obscure bottleneck routes, based on the idea that flattening the flow density reduces the accuracy of the inference.

A more recent line of work, exemplified by SpiderNet [19], combines traffic classification with structure-aware topology mutation. These methods leverage accurate probe detection and AI-guided decision mechanisms to determine when and how to apply latency distortion or randomized topological responses, thereby enhancing robustness and adaptability in obfuscation.

However, methods like EqualNet [26] show limited effectiveness in highly dynamic environments involving IP mobility or virtualization. To improve adaptability, NetHide [27] constructs virtual topologies by controlling node observability, enabling tunable tradeoffs between usability and obfuscation strength. Its approach emphasizes structural camouflage rather than merely obfuscating traffic characteristics, although its performance remains sensitive to the attack model and probing patterns.

B. Topology Obfuscation for External End-to-End Inference

In this setting, adversaries launch probes from edge nodes and infer the topology based on the latency profiles of returned paths. The primary defense objective is to distort the inference process by injecting controlled delay or randomness into the latency matrix perceived by the attacker.

Representative examples include ProTo [28] and Antitomo [25]. ProTo identifies abnormal probing behavior through large-scale PIs and KNN-based classification, selectively delaying suspicious probe flows to disrupt inference accuracy. Antitomo builds upon this idea by generating multiple synthetic delay matrices, dynamically selecting the most obfuscating topology under utility constraints. Similarly, ChameleonNet [37] adopts a two-stage pipeline that first constructs decoy topologies and then applies distortion to the selected structure.

Recent approaches have also emerged, such as HBB-TSP [18], which abstracts the network as a higher-order temporal graph to generate time-evolving deceptive topologies. EigenObfu [24] adopts eigenvector centrality as an alternative to degree centrality for critical node detection and protection.

VI. Discussion

Based on the preceding analysis, we provide new insights into key limitations and unresolved issues in existing topology obfuscation approaches.

1) *Lack of Comprehensive Attacker Modeling:* Most existing approaches adopt a two-stage defense paradigm, namely, topology obfuscation followed by latency perturbation, but often fail to incorporate the attacker's adaptive and iterative inference capabilities. Specifically,

adversaries can incrementally refine their attack strategies through repeated measurements and feedback loops, progressively converging on the true underlying path structure [6], [38]. Representative methods include ProTo, Antitomo, ChameleonNet, ARIEL, and HBB-TSP. Although some internal inference defenses attempt to mislead attackers through static camouflage, they tend to incur considerable computational and deployment overhead.

2) *Over-Simplified Security Metrics:* Several studies (e.g., Antitomo and NetHide) optimize for single objectives, such as structural similarity or flow uniformity. While these metrics serve as reasonable proxies for attacker observability, they overlook the attacker's epistemic uncertainty, particularly in scenarios where bottleneck locations are unknown a priori. This may result in blind spots under stealthy or probabilistic attack models.
3) *Inadequate Probing Identification Mechanisms:* Although some works propose classification-based detection mechanisms (e.g., ProTo uses lightweight k-NN classifiers, and SpiderNet filters only traffic requiring obfuscation), these methods often assume a clear separation between probing and benign flows. In practice, background traffic exhibits high variability, and probe-only detection is insufficient for robust camouflage.
4) *Rigid Multiobjective Optimization Frameworks:* Some approaches, such as NetHide and Antitomo, require balancing multiple objectives with equal priority. However, real-world deployment typically requires dynamic prioritization based on runtime constraints.
5) *Limited Generalizability:* Many methods cannot ensure that attacker behavior conforms to a single model. For example, EqualNet and Linkbait rely on IP anchoring or static decoy nodes, which limits their adaptability in dynamic environments. Meanwhile, methods like ProTo and Antitomo rely heavily on latency injection, which may cause backfire when used against internal cooperative inference.

Our approach distinguishes itself by integrating fine-grained structural similarity metrics with lightweight, real-time latency modulation, thereby enabling a scalable and stealth-resilient defense against both internal and external inference threats.

Furthermore, to facilitate a fair and relevant evaluation, we select ProTo and Antitomo as our primary comparison baselines for the following reasons.

1) Both methods are designed to defend against external end-to-end inference attacks, which aligns with the primary threat model addressed by our framework.
2) They exemplify delay-based obfuscation mechanisms triggered by probing traffic, offering representative baselines for comparison with our real-time camouflage strategy.
3) Compared to structural rewriting methods, such as NetHide and EqualNet, ProTo and Antitomo demonstrate greater practicality and deployability under the evaluation constraints defined in our study.

Note that this article discusses the feasibility of applying TTL-based camouflage to defend against internal cooperative probing, but this is not the core focus of our work. Rather than directly addressing internal scenarios, we focus on experimentally validating the effectiveness of our end-to-end defense strategy, ensuring consistency between theoretical design and empirical results.

VII. Conclusion

This article introduces OptiPathNet, an innovative framework for proactive defense in IoT networks that combines dynamic topology obfuscation with real-time traffic detection. Unlike traditional methods that isolate these tasks, OptiPathNet enables a closed-loop interaction, allowing traffic analysis to inform adaptive obfuscation policies and disrupt attackers' learning of network patterns. Our key innovations include traffic-aware dynamic obfuscation, utilizing lightweight traffic classification with multiobjective evolutionary optimization for real-time adaptation to network changes and attack strategies, and multiobjective evolutionary topology obfuscation. The GCMOEA algorithm applies multiobjective principles to IoT security, balancing security, operational efficiency, and energy constraints, representing a shift from single-objective tradeoffs to comprehensive optimization in resource-constrained environments.

Future work will explore federated learning for distributed policy updates and blockchain-integrated trust mechanisms. By bridging the gap between adaptive defense and IoT's unique constraints, OptiPathNet lays the foundation for a new era of resilient, self-healing networks.

References

[1] J. K. Chahal, A. Bhandari, and S. Behal, "DDoS attacks & defense mechanisms in SDN-enabled cloud: Taxonomy, review and research challenges," *Comput. Sci. Rev.*, vol. 53, Aug. 2024, Art. no. 100644.
[2] L. Huang, P. Liu, X. Chen, C. Jiang, L. Kuang, and J. Lu, "A consolidated game framework for cooperative defense against cross-domain cyber attacks in satellite-enabled Internet of Things," *IEEE Internet Things J.*, vol. 12, no. 9, pp. 12853–12868, May 2025.
[3] P. Kumari and A. K. Jain, "A comprehensive study of DDoS attacks over IoT network and their countermeasures," *Comput. Security*, vol. 127, Apr. 2023, Art. no. 103096.
[4] D. Kong et al., "Combination attacks and defenses on SDN topology discovery," *IEEE/ACM Trans. Netw.*, vol. 31, no. 2, pp. 904–919, Apr. 2023.
[5] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *Proc. IEEE SP*, 2013, pp. 127–141.
[6] M. Tran, M. S. Kang, H.-C. Hsiao, W.-H. Chiang, S.-P. Tung, and Y.-S. Wang, "On the feasibility of rerouting-based DDoS defenses," in *Proc. IEEE SP*, 2019, pp. 1169–1184.
[7] X. Huang et al., "You can obfuscate, but you cannot hide: CrossPoint attacks against network topology obfuscation," in *Proc. USENIX Security*, 2024, pp. 5735–5750.
[8] M. Nance-Hall, Z. Liu, V. Sekar, and R. Durairajan, "Analyzing the benefits of optical topology programming for mitigating link-flood DDoS attacks," *IEEE Trans. Depend. Secure Comput.*, vol. 22, no. 1, pp. 146–163, Jan./Feb. 2025.
[9] S. Javanmardi, M. Ghahramani, M. Shojafar, M. Alazab, and A. M. Caruso, "M-RL: A mobility and impersonation-aware IDS for DDoS UDP flooding attacks in IoT-fog networks," *Comput. Security*, vol. 140, May 2024, Art. no. 103778.
[10] A. Studer and A. Perrig, "The coremelt attack," in *Proc. Eur. Symp. Res. Comput. Security*, 2009, pp. 37–52.

[11] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng, "FlowGuard: An intelligent edge defense mechanism against IoT DDoS attacks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9552–9562, Oct. 2020.
[12] V. Sivaraman and B. Sikdar, "A game-theoretic approach for enhancing data privacy in SDN-based smart grids," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10583–10595, Jul. 2021.
[13] V. Hnamte, A. A. Najar, H. Nhung-Nguyen, J. Hussain, and M. N. Sugali, "DDoS attack detection and mitigation using deep neural network in SDN environment," *Comput. Security*, vol. 138, Mar. 2024, Art. no. 103661.
[14] Z. Han, W. Wang, C. Wen, and L. Wang, "Distributed adaptive consensus control for nonlinear systems with active-defense mechanism against denial-of-service attacks," *IEEE Trans. Ind. Informat.*, vol. 20, no. 8, pp. 10440–10451, Aug. 2024.
[15] L. Zhang, Y. Du, J. Xu, and X. Wang, "UAV-enabled IoT: Cascading failure model and topology-control-based recovery scheme," *IEEE Internet Things J.*, vol. 11, no. 12, pp. 22562–22577, Jun. 2024.
[16] X. Jiang et al., "Credible link flooding attack detection and mitigation: A blockchain-based approach," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 3, pp. 3537–3554, Jun. 2024.
[17] X. Xu, Y. Wang, Y. Zhang, and D. Li, "A fault diagnosis method to defend Scapegoating attack in network tomography," *Theor. Comput. Sci.*, vol. 939, pp. 237–249, Jan. 2023.
[18] X. Li, X. Yang, Y. Huang, and Y. Chen, "Combating temporal composition inference by high-order camouflaged network topology obfuscation," *Comput. Security*, vol. 144, Sep. 2024, Art. no. 103981.
[19] X. Huang et al., "SpiderNet: Enabling bot identification in network topology obfuscation against link flooding attacks," *IEEE/ACM Trans. Netw.*, vol. 33, no. 1, pp. 99–113, Feb. 2025.
[20] X. Ding, F. Xiao, M. Zhou, and Z. Wang, "Active link obfuscation to thwart link-flooding attacks for Internet of Things," in *Proc. IEEE 19th TrustCom*, 2020, pp. 217–224.
[21] S. Yoo, X. Chen, and J. Rexford, "SmartCookie: Blocking large-scale SYN floods with a split-proxy defense on programmable data planes," in *Proc. USENIX Security*, 2024, pp. 217–234.
[22] X.-Y. Zhao and J.-L. Wang, "Analysis and control for synchronization of coupled reaction–diffusion neural networks with multiple couplings subject to topology attacks," *Neurocomputing*, vol. 555, Oct. 2023, Art. no. 126653.
[23] Y. Liu, Y. Zhao, X. Guo, and L. Liu, "Inferring router ownership based on the classification of intra-and inter-domain links," *Sci. Rep.*, vol. 13, no. 1, p. 5090, 2023.
[24] Z. Zhu, G. Zhu, Y. Zhang, J. Shi, X. Huang, and Y. Fang, "EigenObfu: A novel network topology obfuscation defense method," *IEEE Trans. Netw. Sci. Eng.*, vol. 12, no. 1, pp. 451–462, Jan./Feb. 2025.
[25] Y. Liu, C. Xing, G. Zhang, L. Song, and H. Lin, "AntiTomo: Network topology obfuscation against adversarial tomography-based topology inference," *Comput. Security*, vol. 113, Feb. 2022, Art. no. 102570s.
[26] J. Kim, E. Marin, M. Conti, and S. Shin, "EqualNet: A secure and practical defense for long-term network topology obfuscation," in *Proc. NDSS*, 2022, pp. 1–18.
[27] R. Meier, P. Tsankov, V. Lenders, L. Vanbever, and M. Vechev, "NetHide: Secure and practical network topology obfuscation," in *Proc. USENIX Security*, 2018, pp. 693–709.
[28] T. Hou, Z. Qu, T. Wang, Z. Lu, and Y. Liu, "ProTO: Proactive topology obfuscation against adversarial network topology inference," in *Proc. IEEE INFOCOM*, 2020, pp. 1598–1607.
[29] D. Halder and S. Ray, "PhD project: Reconfigurable network on chip architecture through topology obfuscation for protecting SoC against reverse engineering," in *Proc. IEEE FCCM*, 2024, pp. 227–228.
[30] Y. Liu, J. Zhao, G. Zhang, and C. Xing, "Netobfu: A lightweight and efficient network topology obfuscation defense scheme," *Comput. Security*, vol. 110, Nov. 2021, Art. no. 102447.
[31] J. Ni, H. Xie, S. Tatikonda, and Y. R. Yang, "Efficient and dynamic routing topology inference from end-to-end measurements," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 123–135, Feb. 2010.
[32] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. CVPR*, 2022, pp. 16000–16009.
[33] L. Beyer, X. Zhai, A. Royer, L. Markeeva, R. Anil, and A. Kolesnikov, "Knowledge distillation: A good teacher is patient and consistent," in *Proc. CVPR*, 2022, pp. 10925–10934.
[34] S. Beamer, K. Asanović, and D. Patterson, "Direction-optimizing breadth-first search," *Sci. Program.*, vol. 21, nos. 3–4, pp. 137–148, 2013.
[35] C. He, S. Huang, R. Cheng, K. C. Tan, and Y. Jin, "Evolutionary multiobjective optimization driven by generative adversarial networks (GANs)," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3129–3142, Jun. 2021.
[36] Z. Hang, Y. Lu, Y. Wang, and Y. Xie, "Flow-MAE: Leveraging masked autoencoder for accurate, efficient and robust malicious traffic classification," in *Proc. 26th RAID*, 2023, pp. 297–314.
[37] C. Qiu, B. Ren, G. Tang, L. Luo, and D. Guo, "ChameleonNet: Topology obfuscation against tomography with critical information hiding," *IEEE Trans. Netw.*, early access, May 16, 2025, doi: 10.1109/TON.2025.3567771.
[38] C. Liaskos, V. Kotronis, and X. Dimitropoulos, "A novel framework for modeling and mitigating distributed link flooding attacks," in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.

Xiang Yang is currently pursuing the B.S. degree with the School of Cyber Science and Engineering, Sichuan University, Chengdu, China.

Her research interests span a wide range of fields, including network security, complex networks, machine learning, and large model safety.

Zhentian Zhong is currently pursuing the B.S. degree with the Pittsburgh Institute, Sichuan University, Chengdu, China.

His research focuses on AI-driven security solutions and network security protocols, with a particular emphasis on time-series analysis for anomaly detection and the development of intelligent systems to enhance cyber threat detection.

Yue Chen is currently pursuing the M.S. degree with the School of Cyber Science and Engineering, Sichuan University, Chengdu, China.

Her research focuses on network forensics, traffic analysis and attribution, and machine learning for cybersecurity, with particular emphasis on cyber threat intelligence and intelligent security decision-making.

Xiaohui Li (Member, IEEE) 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 Professor with the School of Cyber Science and Engineering, Sichuan University. Her research interests cover several areas with emphasis on spatial information networks and network and information security.

Junfeng Wang received the M.S. degree in computer application technology from Chongqing University of Posts and Telecommunications, Chongqing, China, in 2001, and the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2004.

He is currently a Professor with the College of Computer Science, Sichuan University, Chengdu. His recent research interests include spatial information networks and network and information security.

Zhiping Cai received the B.Eng., M.A.Sc., and Ph.D. degrees in computer science and technology from the National University of Defense Technology (NUDT), Changsha, China, in 1996, 2002, and 2005, respectively.

He is a Full Professor with the College of Computer, NUDT. His current research interests include artificial intelligence, network security, and big data.