Hierarchical Micro-Segmentations for Zero-Trust Services via Large Language Model-Enhanced Graph Diffusion

Yinqiu Liu, *Member, IEEE*, Guangyuan Liu, *Graduate Student Member, IEEE*, Hongyang Du, *Member, IEEE*, Dusit Niyato, *Fellow, IEEE*, Jiawen Kang, *Senior Member, IEEE*, Zehui Xiong, *Senior Member, IEEE*, Dong In Kim, *Life Fellow, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—In the rapidly evolving Next-Generation Networking (NGN) era, the adoption of zero-trust architectures has become increasingly crucial to protect security. However, provisioning zero-trust services in NGNs poses significant challenges, primarily due to the environmental complexity and dynamics. Motivated by these challenges, this paper explores efficient zero-trust service provisioning using hierarchical micro-segmentations. Specifically, we model zero-trust networks via hierarchical graphs, thereby jointly considering the resource- and trust-level features to optimize service efficiency. We organize such zero-trust networks through micro-segmentations, which support granular zero-trust policies efficiently. To generate the optimal micro-segmentation, we present the Large Language Model-Enhanced Graph Diffusion (LEGD) algorithm, which leverages the diffusion process to realize a high-quality generation paradigm. Additionally, we utilize gradient ascent and Large Language Models (LLM) to enable LEGD to optimize the generation policy and understand complicated graphical features. Moreover, realizing the unique trustworthiness updates and service upgrades in zero-trust NGN, we further present LEGD-Adaptive Maintenance (LEGD-AM), providing an adaptive way to perform task-oriented fine-tuning on LEGD. Extensive experiments demonstrate that the proposed LEGD achieves 90% higher efficiency in provisioning services compared with other baselines. Moreover, the LEGD-AM can reduce the service outage time by over 50%.**

***Index Terms*—Zero trust, micro-segmentation, diffusion, service function chain (SFC), next-generation networking.**

Received 9 October 2024; revised 8 May 2025; accepted 8 January 2026; approved by IEEE TRANSACTIONS ON NETWORKING Editor L. Huang. Date of publication 14 January 2026; date of current version 2 February 2026. This work was supported in part by the Seatrium New Energy Laboratory, Singapore Ministry of Education (MOE) Tier 1 under Grant RT5/23 and Grant RG24/24; in part by Nanyang Technological University-Centre for Computational Technologies in Finance (NTU-CCTF); in part by the Research Innovation and Enterprise (RIE) 2025 Industry Alignment Fund-Industry Collaboration Projects (IAF-ICP) administered by the Agency for Science, Technology and Research (A*STAR) under Award I2301E0026; in part by the Ministry of Science and Information and Communications Technology (MSIT), South Korea, under the ICT Creative Consilience Program supervised by the Institute for ICT Planning and Evaluation (IITP) under Grant IITP-2020-0-01821. *(Yinqiu Liu and Guangyuan Liu contributed equally to this work.) (Corresponding author: Hongyang Du.)*

Yinqiu Liu, Guangyuan Liu, and Dusit Niyato are with the College of Computing and Data Science, Nanyang Technological University, Jurong West, Singapore 639798 (e-mail: yinqiu001@e.ntu.edu.sg; liug0022@e.ntu.edu.sg; dniyato@ntu.edu.sg).

Hongyang Du is with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong, SAR, China (e-mail: duhy@eee.hku.hk).

Jiawen Kang is with the School of Automation, Guangdong University of Technology, Guangzhou, Guangdong 510006, China (e-mail: kavinkang@gdut.edu.cn).

Zehui Xiong is with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, BT7 1NN Belfast, U.K. (e-mail: z.xiong@qub.ac.uk).

Dong In Kim is with the Department of Electrical and Computer Engineering, Sungkyunkwan University, Seou l16419, South Korea (e-mail: dongin@skku.edu).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/TON.2026.3654170

I. INTRODUCTION

IN CYBERSECURITY, conventional trust models operate on the principle that once one entity is authenticated, it can permanently access network resources [1]. This implicit trust creates significant vulnerabilities, as it allows attackers who breach the perimeter from inside to move laterally, thus damaging the entire network [1]. Zero-trust, however, represents a fundamental shift from this paradigm. Specifically, it follows the *never trust, always verify* principle, ensuring that every access request is authenticated, authorized, and continuously validated. By strictly enforcing least-privilege access, both the risk of insider threats and outsider attacks can be alleviated [1]. For Next-Generation Networking (NGN) [2], the demand for zero-trust is projected to be higher than ever, driven by the increasing volume of data, devices, and applications.

Recent research has made substantial progress in zero-trust networks, primarily focusing on advancing per-session authentication and defending against security threats. Three main technical directions have emerged. First, reputation-based mechanisms [3], [4], [5] have been widely adopted to quantify mutual trustworthiness between nodes using historical behavior and contextual features. These schemes facilitate continuous authentication by dynamically updating trust levels. Second, multi-factor authentication incorporates both software-based credentials and hardware-level identifiers such as radio-frequency fingerprints or device-specific signatures, thereby increasing the difficulty of launching attacks [6], [7]. Third, blockchain is increasingly integrated into zero-trust networks to eliminate reliance on trusted authorities [3], [8]. Through immutable logging and decentralized consensus, blockchain supports distributed verification and tamper-proof auditing. Beyond academia, various industrial zero-trust solutions have been presented, such as Microsoft Entra [9] and IBM MaaS360 [10]. However, we observe that while the fundamental zero-trust authentication has been addressed, a subsequent important problem remains underexplored, i.e., the *service provisioning* in next-generation zero-trust networks. Unlike traditional network architectures, the pursuit of zero-trust in NGNs introduces unique challenges

- **Challenge 1**: Zero-trust networks are typically partitioned into micro-segmentations, each tailored to a specific application and governed by an independent access policy

[11]. Although enabling fine-grained zero-trust policy deployments and minimizing the cross-segmentation privacy leakage, this structural fragmentation complicates the overall network topology, making it difficult to coordinate service providers across segments.

- **Challenge 2**: With increasingly complex user demands in NGNs, services are commonly realized via Service Function Chains (SFCs) spanning multiple nodes with different computation capabilities and data [12]. In the zero-trust setting, each hop in the SFC should undergo real-time trust verification, which necessitates modeling the end-to-end efficiency of service execution under dynamic physical and trust conditions [12], [13].
- **Challenge 3**: Zero-trust networks exhibit great dynamics due to the necessity for continuous trustworthiness maintenance [1]. For instance, a widely trusted entity with a high reputation may be removed in the next period due to a single malicious behavior [4]. Consequently, the service provisioning strategies defined at the initial stage may quickly become outdated, leading to significant performance degradation.

In this paper, we present efficient service provisioning for zero-trust NGNs and solve the above challenges. Specifically, to align with the structural characteristics and security principles of zero-trust NGNs, we consider a network composed of application-specific micro-segmentations [14], where services are provisioned through SFCs. We model such zero-trust networks using a hierarchical graph that integrates physical resources and trustworthiness relationships. Then, we present the Large Language Model-Enhanced Graph Diffusion (LEGD) algorithm, leveraging state-of-the-art diffusion-based generative architecture [15] for controllable micro-segmentation generation. Particularly, inspired by human feedback mechanisms [16], we incorporate an LLM-empowered agent, which generates and activates heuristic filters to improve LEGD's efficiency. Finally, we propose an adaptive micro-segmentation maintenance mechanism that promptly updates micro-segmentations to adapt to continuous trustworthiness updates and service upgrades in zero-trust NGNs. Our main contributions are summarized as follows.

- **Zero-trust Service Provisioning Framework**: *To the best of our knowledge, this is the first work that systematically studies efficient service provisioning for zero-trust NGN.* Using graph theory, we model zero-trust networks with micro-segmentations and SFCs through a hierarchical graph, which jointly considers physical and trust-level features. We then formulate the service provisioning as an optimization problem of controllable micro-segmentation generation. Additionally, we derive key factors that determine the efficiency of micro-segmentation.
- **LEGD for Micro-segmentation Generation**: We present the LEGD algorithm to solve the formulated problem and maximize the micro-segmentation's efficiency. Leveraging diffusion architecture, LEGD exhibits excellent exploration capability via a denoising process. We adopt gradient ascent to optimize the generation policy, allowing LEGD to reinforce itself through interacting with zero-trust networks. We also propose an LLM-empowered agent to provide human-like perceptions of the graphical network environment, thus activating heuristic filters to improve LEGD's efficiency.
- **Adaptive Micro-segmentation Maintenance**: To adapt to the dynamic zero-trust NGN environments, we propose an adaptive micro-segmentation maintenance algorithm named LEGD-Adaptive Maintenance (LEGD-AM). By fine-tuning well-trained LEGD models, LEGD-AM can respond to trustworthiness updates and service upgrades promptly. Moreover, adaptive masks and reward engineering are developed to ensure LEGD-AM aligns with updated environments and tasks.

The remainder of this paper is organized as follows. Section II reviews related works. The system models are described in Section III. Then, in Section IV, we elaborate on the design of the LEGD algorithm. LEGD-AM is described in Section V. The numerical results are discussed in Section VI. Finally, Section VII concludes the paper.

II. Related Work and Motivation

A. Zero-Trust Network Architecture

Recent research on zero-trust networks primarily focused on advancing per-session authentications that eliminate assumed trust. Mainstream authentication strategies fall into three categories: reputation mechanism [3], [4], [5], multi-factor authentication [6], [7], and blockchain [3], [8]. Reputation mechanisms are particularly effective in behavior-driven environments such as social Internet-of-Things and collaborative edge computing, where trust levels can be dynamically inferred from historical interactions [3]. For instance, Tian et al. [4] developed a reputation evaluation system for vehicular networks using evolutionary game theory to dynamically model attacker behaviors. Multi-factor authentication enhances security by integrating software-based credentials or physical-layer identifiers, making it well-suited for applications involving biometric or device-specific information [6]. For instance, Jing et al. [6] replaced traditional key-based authentication with a physical-layer identifier named radio frequency fingerprint. Finally, blockchain technologies offer immutable logging and decentralized trust verification, effectively eliminating reliance on centralized authorities [8]. However, existing works only established zero-trust networks. Ensuring efficient and reliable *service provisioning* in zero-trust networks remains an open and underexplored problem. The authors in [13] employed SFCs for organizing services but relied on heuristic device selection without considering the physical and trust-level network environments. Hence, in this paper, we propose a systematic, learning-based approach to realize efficient service provisioning in zero-trust NGN.

B. Micro-Segmentation

Micro-segmentation is regarded as the fundamental manner for organizing zero-trust NGNs, enabling granular network isolation to limit the lateral movement of threats [11], [17]. Early micro-segmentations are generated in static ways [1], which are often rigid and lack flexibility in dynamic environments. Contemporary approaches, such as illumio [14] and [18], leveraged Software-Defined Networking (SDN) to achieve programmable micro-segmentation, enabling real-time policy enforcement to defend against intelligent attacks. Nonetheless, challenges still remain in ensuring the generated micro-segmentation can maximize user experience and service efficiency, especially in large heterogeneous NGNs. To this end, Yousefi-Azar et al. [19] explored this issue by fine-tuning segmentation strategies via unsupervised learning. In this paper, we define multiple performance indicators for evaluating zero-trust services. Moreover, we abstract micro-segmenting as a generation problem and present the

LEGD algorithm based on the state-of-the-art generative model named diffusion [15] to realize controllable micro-segmentation generation.

C. Graph Neural Networks for Networking

Graphs can model various networking and communication problems like resource allocation, routing optimization, and traffic prediction [20]. This is because network topologies naturally form graph structures, enabling efficient representation of complex relationships. Graph Neural Networks (GNNs) have been extensively studied for these problems. For example, Liu et al. [21] used graph attention networks to predict connection failures. Peng et al. [22] introduced Vertex-GNN and Edge-GNN for resource allocation in wireless networks. Hou et al. [23] developed a GNN-based caching scheme for SDN-based networks. Micro-segmentation generation, however, differs from conventional graphical optimizations. Instead of learning mappings from graphical features to optimal variables, generation requires learning to produce graph outputs that match a desired distribution under specific constraints. To this end, we present the LEGD with the state-of-the-art generative architecture, i.e., diffusion.

III. System Model: Hierarchical Micro-Segmentations for Zero-Trust NGN

In this section, we present the system model. As mentioned in Section I, we consider a representative zero-trust NGN that is divided into application-specific micro-segmentations and organizes services by SFCs.

A. Network Modeling via Hierarchical Graph

1) Stakeholder: We consider two types of stakeholders, namely users and service providers. Users request services from the SFC containing multiple service providers. For instance, Fig. 1-A shows a three-step SFC for video generation, in which the user leverages textual prompts to draw images and then selects the most preferred one to make videos. Note that such AI-generated video services have reached a market size of more than USD 554.9 million by 2023.[1] Accordingly, the SFC should consist of text generation, text-to-image, and image-to-video service providers in sequence.

2) Network: We model the network topology using a hierarchical graph with physical and trust layers. As shown in Fig. 1-B, the physical layer is represented by a directed graph $\boldsymbol{G}^P(\boldsymbol{V}^P, \boldsymbol{E}^P, \boldsymbol{F}_V^P, \boldsymbol{F}_E^P)$, where all the devices construct node set $\boldsymbol{V}^P$, and all the communication links between each pair of devices form edge set $\boldsymbol{E}^P$. Accordingly, $\boldsymbol{F}_V^P$ and $\boldsymbol{F}_E^P$ preserve the node- and edge-level features, respectively, i.e.,

$$\boldsymbol{F}_V^P = \left\{ f_{v_i}^P = ((x_i, y_i), c(v_i), \varpi(v_i)) \mid \forall v_i \in \boldsymbol{V}^P \right\}, \quad (1)$$

$$\boldsymbol{F}_E^P = \left\{ f_{e_{i\to j}}^P = (d(v_i, v_j)) \mid \forall e_{i\to j} \in \boldsymbol{E}^P \right\}, \quad (2)$$

where $(x_i,\ y_i)$, $c(v_i)$, and $\varpi(v_i)$ indicate the 2D-location, computing, and transmission power of node v_i, respectively. $d(v_i, v_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ denotes length of edge $e_{i\to j}$, i.e., the physical distance between nodes v_i and v_j.

Unlike conventional one-time identifications, zero-trust requires dynamic trustworthiness management and per-session authentications. To this end, a trust layer is built atop the physical layer, preserving the network-wide trustworthiness relationship. Specifically, we leverage an undirected graph $\boldsymbol{G}^R(\boldsymbol{V}^R, \boldsymbol{E}^R, \boldsymbol{F}_V^R, \boldsymbol{F}_E^R)$ to model the trust layer, where $\boldsymbol{V}^R \in \boldsymbol{V}^P$ and $\boldsymbol{E}^R \in \boldsymbol{E}^P$, denoting the participants of zero-trust service provisioning. Accordingly, the node and edge features $\boldsymbol{F}_V^R$ and $\boldsymbol{F}_E^R$ are defined as

$$\boldsymbol{F}_V^R = \left\{ f_{v_i}^R = (s_i) \mid \forall v_i \in \boldsymbol{V}^R \right\}, \quad (3)$$

$$\boldsymbol{F}_E^R = \left\{ f_{e_{i\leftrightarrow j}}^R = (\omega_{i\leftrightarrow j}) \mid \forall e_{i\leftrightarrow j} \in \boldsymbol{E}^R \right\}, \quad (4)$$

where s_i represents the roles that the node plays in SFC, i.e., type-$\{1,\ 2,\ \ldots,\ K\}$ service provider. $\omega_{i\leftrightarrow j}$ denotes the mutual trustworthiness between nodes v_i and $v_j \in \boldsymbol{V}^R$, which is defined below. Through this hierarchical graph, we can perform joint optimization regarding resources and trustworthiness.

3) Zero-Trust Policy and Micro-Segmentation: As shown in Fig. 2-A, the entire trust layer is partitioned into multiple isolated micro-segmentations, each of which accommodates multiple SFCs catering to a specific application. Within each micro-segmentation, we suppose that the NIST standards [11] are adopted to implement zero-trust policies (see Fig. 2-B). Specifically, each pair of nodes is monitored by one Policy Enforcement Point (PEP). All PEPs are coordinated and managed by a centralized Policy Engine (PE), which grants or denies the request for node access. To do so, a trustworthiness scheme $\mathcal{M}()$ is established, measuring the mutual trust level of each pair of nodes by a value in [0, 1], i.e.,

$$\omega_{i\leftrightarrow j} = \omega_{j\leftrightarrow i} = \mathcal{M}(v_i, v_j) \in [0, 1], \forall v_i, v_j \in \boldsymbol{V}^R. \quad (5)$$

The higher the $\omega_{i\leftrightarrow j}$ value, the higher the level of trustworthiness that two nodes maintain. Following zero-trust principles, mutual trustworthiness is updated dynamically, and PE performs continuous authentication, i.e., the access request will only be granted if the real-time mutual trustworthiness is greater than the user-required threshold. Finally, PEP executes the trust policies by opening/blocking the communication links according to the PE commands.

Remark 1: Since this paper focuses on service provisioning in the zero-trust context, we adopt the general zero-trust policy,

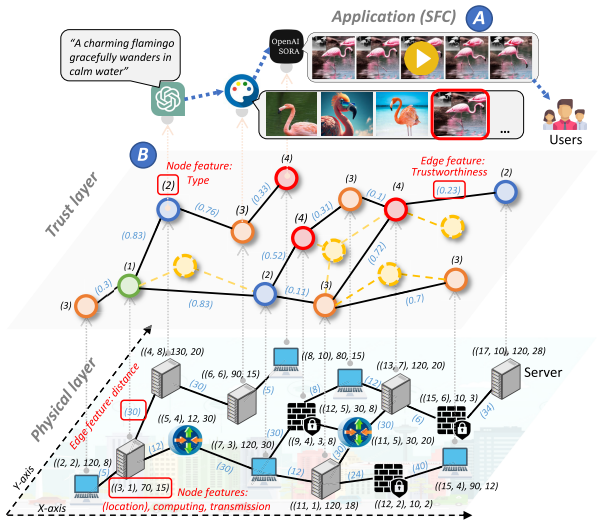


Fig. 1. System model. **A**: The illustration of a three-step SFC for text-guided video generation. **B**: Zero-trust network modeling using a hierarchical graph.

[1]Data available at: https://www.grandviewresearch.com/industry-analysis/ai-video-generator-market-report

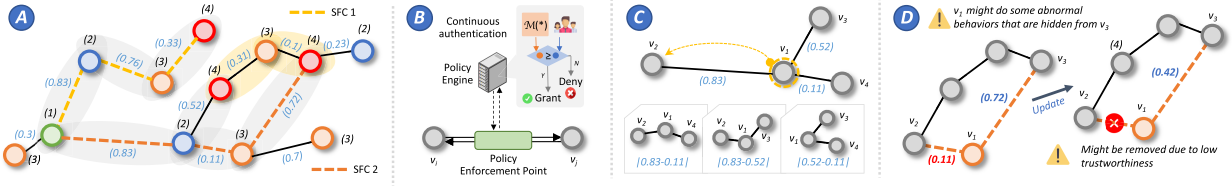


Fig. 2. **A**: The illustration of two micro-segmentations (marked by gray and yellow, respectively) over the trust layer of Fig. 1. **B**: The zero-trust policy. **C**: The calculation of E_s on node v_1. **D**: The example to explain trustworthiness equilibrium.

i.e., authenticating access according to mutual trustworthiness thresholds. To ensure the adaptability, $\mathcal{M}(.,.)$ is regarded as a pluggable module, supporting various well-proven proposals, e.g., reputation [4] and blockchain-based voting [3].

Next, we define several critical performance indicators for both the physical and trust layers, thus evaluating the efficiency of service provisioning.

B. Physical Layer Performance Indicators

1) Service Latency: First, we model the latency required by one micro-segmentation to accomplish the given task. Denote an n-step service as $\mathbf{r}^n = \{r_1, r_2, \ldots, r_n\}$. r_j ($j \in \{1, 2, \ldots, n\}$) represents the j-th step, which calls a certain type of service provider. As aforementioned, $\mathbf{r}^n$ is handled by an n-step SFC in the generated micro-segmentation. The mapping between the zero-trust network and generated micro-segmentation can be expressed as

$$\mathcal{F}(\boldsymbol{G}^R|\boldsymbol{G}^P) \overset{\pi_\theta}{\longmapsto} \boldsymbol{G}^S = (\boldsymbol{S}_1^n, \boldsymbol{S}_2^n, \ldots, \boldsymbol{S}_Q^n), \tag{6}$$

where $\mathcal{F}(*)$ represents the mechanism to fuse two hierarchical layers. $\boldsymbol{G}^S$ denotes the generated micro-segmentation that contains Qn-step SFCs. Such a mapping is performed by LEGD, which relies on a learning-based network π_θ parameterized by θ. The LEGD design will be described in Section IV. When each service request arrives at micro-segmentation $\boldsymbol{G}^S$, it will be randomly assigned to an SFC.

The first component of service latency is performing computation to obtain the required results. Denote the computation complexity of $\mathbf{r}^n$ as $\mathbf{c}^n = \{c(r_1), c(r_2), \ldots, c(r_n)\}$, where $c(r_j) \sim \mathcal{N}\left(\mu_j^c, (\sigma_j^c)^2\right), j \in \{1, 2, \ldots, n\}$. Hence, the long-term computation latency can be derived as

$$L_c = \lim_{T \mapsto \infty} \sum_{\mathbf{r}_n \in \mathcal{R}} \sum_{q=1}^{Q} \sum_{j=1}^{n} \frac{\tau\left(\pi_{\theta,q}^{-1}(r_j)\right)^{-1} c(r_j)}{c\left(\pi_{\theta,q}^{-1}(r_j)\right)} \cdot (TQ)^{-1}, \tag{7}$$

where $\mathcal{R}$ denotes tasks, arriving following a Poisson distribution. $\pi_{\theta,q}^{-1}(r_j)$ is the inverse process of Eq. (6), which maps to the device serving r_j in the q-th SFC. We suppose that the computing power of each device is allocated equally among all SFCs. $c(*)$ and $\tau(*)$ denote the computing power of device $*$ and the number of SFCs that $*$ belongs to, respectively.

Apart from computation, each service provider in an SFC receives outputs generated by its predecessor and sends its computation results to the successor, causing transmission latency. Based on [24], the one-hop transmission bandwidth (bit/s) between devices v_i and $v_j \in \boldsymbol{V}^P$ can be defined as

$$b(e_{i \to j}) = W \log_2\left(1 + \frac{\varpi(v_i)\, d\,(v_i, v_j)^{-\gamma}\, |h_0|^2}{N_0}\right), \tag{8}$$

where W and γ are uplink channel bandwidth between devices and path-loss exponent, respectively. h_0 represents the complex Gaussian channel coefficient following complex normal distribution $\mathcal{CN}(0, 1)$. N_0 denotes the additive white noise. Suppose that the bandwidth consumption of intermediate results transferred along the SFC is $\{b(r_1), b(r_2), \ldots, b(r_n)\}$. The long-term transmission latency can be derived as

$$L_t = \lim_{T \mapsto \infty} \sum_{\mathbf{r}_n \in \mathcal{R}} \sum_{q=1}^{Q} \sum_{j=1}^{n-1} \Phi\left(\pi_{\theta,q}^{-1}(r_{j \leftrightarrow j+1})\right) \cdot \frac{\tau\left(\pi_{\theta,q}^{-1}(r_j)\right)^{-1} b(r_j)}{b\left(\pi_{\theta,q}^{-1}(r_{j \leftrightarrow j+1})\right)} \cdot (TQ)^{-1}. \tag{9}$$

Similar to Eq. (7), $\pi_{\theta,q}^{-1}(r_{j \leftrightarrow j+1})$ denotes the edge that connects the j-th and $j+1$-th service providers in the q-th SFC. Accordingly, $\Phi\left(\pi_{\theta,q}^{-1}(r_{j \leftrightarrow j+1})\right)$ represents the number of hops existing in this edge. In this paper, we suppose each pair of service providers directly exchange information without relay, i.e., $\Phi\left(\pi_{\theta,q}^{-1}(r_{j \leftrightarrow j+1})\right) = 1$ in all the cases.

Combining computation and transmission latency, the total service latency can be expressed as $L_s = L_c + L_t$.

2) Service Throughput: Besides service latency, another critical consideration is the throughput, which refers to the number of service requests that the micro-segmentation can simultaneously process. Given that the micro-segmentation accommodates Q SFCs, the service throughput is $T_s = Q$.

C. Trust Layer Performance Indicators

1) Trustworthiness-Aware Service Accomplishment: During the execution of an n-step SFC, each service provider v_{i+1} requests v_i ($i \in \{1, 2, \ldots, n-1\}$) for the intermediate results. Nonetheless, if mutual trustworthiness $\omega_{i \leftrightarrow i+1}$ is less than the user threshold, PE will deny the access request and terminate the entire SFC execution. Denote trustworthiness threshold of service $\mathbf{r}^n$ as $\{t(r_1), t(r_2), \ldots, t(r_n)\}$, where $t(r_j) \sim N\left(\mu_j^t, (\sigma_j^t)^2\right)$. In this case, the probability of successful service execution can be derived as

$$P_s = \sum_{q=1}^{Q}\left(\text{Prob}(\boldsymbol{S}_q^n|\boldsymbol{G}^S) \prod_{i=1}^{n-1} \text{Prob}\left(t(r_i) \le \omega_{i \leftrightarrow i+1}^q\right)\right) \tag{10a}$$

$$= \sum_{q=1}^{Q}\left(\prod_{i=1}^{n-1} \frac{Q^{-1}}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\omega_{i \leftrightarrow i+1}^q} \exp\left(-\frac{(\omega_{i \leftrightarrow i+1}^q - \mu_i^t)^2}{2\,(\sigma_i^t)^2}\right) dt(r_i)\right), \tag{10b}$$

where $\omega_{i,q}$ represents the mutual trustworthiness between the i-th and $i+1$-th service provider in the q-th SFC. By Eq. (10), we can observe that efficient micro-segmentation generation policies should try to configure highly trusted nodes for each step of the SFCs, thus improving the probability of successful service execution.

2) Trustworthiness Equilibrium: For each single node, its mutual trustworthiness with a neighbor determines the probability that data access between them can be established. Furthermore, from the viewpoint of the entire zero-trust network, these mutual trustworthiness scores convey more than just pairwise access control, embedding critical stability information.

Example 1: We use Fig. 2-D as an example to illustrate the relationship between mutual trustworthiness and network stability. As shown, node v_1 maintains high trust with v_3 but is distrusted by v_2. This inconsistency reflects a potential security risk: abnormal behavior from v_1 may go undetected or be tolerated by v_3, thus undermining zero-trust policy enforcement [25]. In contrast, if both v_2 and v_3 assign high trust to v_1, the consistency of evaluations reinforces v_1's credibility and reduces the likelihood of sudden node removal. Interestingly, even if both neighbors assign *low* trust to v_1, the network remains more stable than in the inconsistent case, because the distrust is unanimous, and appropriate mitigation (e.g., isolation) can be reliably applied.

To qualitatively capture the impact of trustworthiness consistency on network stability, we introduce the concept of trustworthiness equilibrium. This concept is inspired by signed network theory [25], in which edges are labeled as positive or negative to represent cooperative or antagonistic relationships. These signed relationships have been widely used to guide stability analysis, community detection, and recommendation strategies in various social and organizational networks [25]. We observe that the mutual trustworthiness between nodes in a zero-trust network can be naturally analogized to the signed edges. In signed networks, the minimal analytical unit consists of one target node and two evaluators (as shown in Fig. 2-D). Hence, we define trustworthiness equilibrium as the average inconsistency in mutual trustworthiness among all units in the generated micro-segmentation, i.e.,

$$E_s = \sum_{q=1}^{Q} \sum_{i=1}^{n} \sum_{j,z} \frac{|\omega_{i\leftrightarrow j} - \omega_{i\leftrightarrow z}|}{n\binom{|\boldsymbol{P}_{v_i}|}{2} Q},\ v_j, v_z \in \boldsymbol{P}_{v_i},\ j < z, \tag{11}$$

$$\boldsymbol{P}_{v_i} = \{v_k \mid \exists\, e_{i\leftrightarrow k} \in \boldsymbol{E}^R\}, \tag{12}$$

where $\boldsymbol{P}_{v_i}$ denotes the set of neighbors directly connected to v_i in $\boldsymbol{G}^R$. As shown in Fig. 2-C, Eq. (11) iterates through all nodes in the micro-segmentation and identifies every unit that involves the node and two of its neighbors. For each unit, we compute the absolute difference in mutual trustworthiness values that the two neighbors assign to the target node. The final value E_s is the normalized average across all such differences, reflecting the global consistency of trust assessments.

Remark 2: A high E_s indicates substantial inconsistency in trust evaluations, i.e., the same node receives divergent levels of trustworthiness from its neighbors. This asymmetry may expose hidden vulnerabilities, where malicious behavior could be overlooked. Conversely, $E_s = 0$ implies perfect consensus: all neighbors assign the same trustworthiness value to each node. Such consistency contributes to structural stability, as the network exhibits a coherent perception of node reliability. Nonetheless, a low E_s does not necessarily mean that the network is trustworthy. For instance, if all nodes are unanimously distrusted, it still results in a low E_s due to the uniformity of assessments. Moreover, E_s is not necessarily required to converge to zero. The overall efficiency of micro-segmentations also depends on several additional factors, including service latency, throughput, and service accomplishment rate. We intend to acquire the optimal micro-segmentations that strike the great balances among these factors.

IV. Micro-Segmentation Generation via LLM-Enhanced Graph Diffusion

In this section, we formulate the zero-trust service provisioning problem as optimal micro-segmentation generation. We then introduce LEGD, detailing the diffusion paradigm for graph generation, the policy optimization framework, and LLM enhancements.

A. Problem Formulation

Based on the models in Section III, we formulate the problem for LEGD. As aforementioned in Eq. (6), we aim to explore an optimal policy for generating micro-segmentations in a controllable way. Specifically, given the format of service and the zero-trust network modeled by hierarchical graphs $\boldsymbol{G}^P$ and $\boldsymbol{G}^R$, a micro-segmentation $\boldsymbol{G}^S$ should be generated, which accommodates at least one qualified SFC for service provisioning. Moreover, the generated micro-segmentations should be optimized, i.e., maximizing the overall utility that jointly considers physical- and trust-level performance. Such a problem can be formulated as

$$\max_{\boldsymbol{G}^S}\ U_{\mathrm{U}}\left(\{\boldsymbol{G}^P, \boldsymbol{G}^R, L_s, T_s, P_s, E_s\}\right), \tag{13a}$$

$$s.t.\quad T_s \geq 1, \tag{13b}$$

$$L_s \leq L_{\max}, \tag{13c}$$

where Eq. (13b) requires that $\boldsymbol{G}^S$ must contain at least one SFC that caters to user services. Eq. (13c) means the service latency cannot exceed the threshold. Otherwise, the service execution will be terminated and regarded as failed. Jointly considering L_s, T_s, P_s, and E_s, the user utility U_{U} can be defined as

$$U_{\mathrm{U}} = P_s\left[\alpha_1 \log_{P_s}\left(\frac{L_s}{L_{\max}}\right) + \alpha_2 T_s + \alpha_3(1 - E_s)\right], \tag{14}$$

where α_1, α_2, and α_3 are the weighting factors. Note that we apply the Weber-Fechner Law [26] to model the sensitivity of users regarding service latency, which states that the increased latency has a logarithmic relationship with the degradation of user service experience. Moreover, we adopt P_s as the base of the $\log(\cdot)$ function because the higher the trustworthiness level, the higher the user's tolerance for latency.

Theorem 1: The micro-segmentation generation problem defined in Eq. (13) is NP-hard.

Proof: We prove that the micro-segmentation generation problem is NP-hard by reducing it from the graph partition problem, which is a known NP-hard problem [27]. First, we introduce the graph partition problem.

Definition 1 ***(Graph Partition):*** Given an undirected graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, partition the vertex set $\boldsymbol{V}$ into k disjoint subsets $\{\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_k\}$ such that the number of edges crossing between different subsets is minimized [27].

Then, we construct a new instance I_P of the graph partition problem, in which $k = 2$. Next, we prove that this problem is a simplified case of our micro-segmentation generation problem.

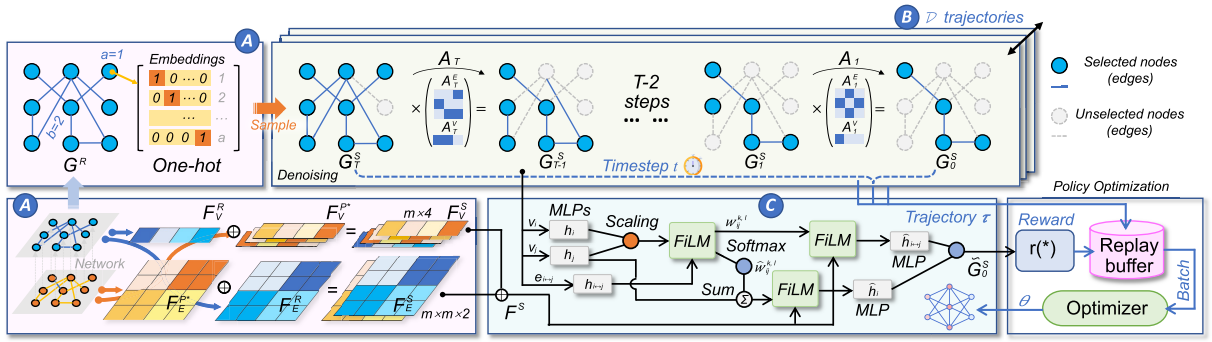


Fig. 3. The illustration of the LEGD algorithm. **A**: The layer fusion and one-hot embeddings of the graph structure. **B**: The illustration of the denoising process and the trajectory collection module. **C**: The architecture design of the denoising network.

Specifically, problem I_P can be regarded as a simplified case of our optimization problem defined in Eq. (13).

To prove this, we consider that there is only one type of service provider in the network and they fully trust each others, i.e., $\omega_{i\leftrightarrow j} = \omega_{j\leftrightarrow i} = 1$, $\forall V_i \,\&\, V_j \in \boldsymbol{V}^R$. Hence, we can acquire $P_s = 1$, since the users' requirements for mutual trustworthiness can always be satisfied. Then, we can let $\alpha_1 = 0$, $\alpha_2 = 0$, and $\alpha_3 = 1$. We further let $E_s = |E_c|$, which represents the number of edges crossing between the generated micro-segmentation and the remaining part of the original graph. It is obvious that such mappings can be accomplished in polynomial time. Moreover, we ignore constraints (13b). We consider physical resources (including both computing and communication power) to be infinite. Hence, the constraint (13c) can always be satisfied, since $L_s \to 0$. In this way, our problem becomes $\max_{\boldsymbol{G}^S}(1 - |E_c|)$, which can be further derived to $\min_{\boldsymbol{G}^S}(|E_c|)$, i.e., problem P_I. Since constraints (13b) and (13c) are ignored, P_I can be regarded as a simplified case of our problem. Finally, given the NP-hardness of the graph partitioning problem P_I [27], we can conclude that our optimization problem is NP-hard.□

B. Diffusion on Graph for Micro-Segmentation Generation

We present LEGD, which leverages the state-of-the-art generative model named diffusion [15] and employs an LLM-empowered agent to enhance the capability of topology understanding. Next, we demonstrate the core of LEGD, i.e., diffusion on graph-structured data.

1) Forward Diffusion: Inspired by non-equilibrium thermodynamics, diffusion consists of two Markov decision processes, namely forward diffusion and denoising. Denote the optimal micro-segmentation as $\boldsymbol{G}_0^S = (\boldsymbol{V}^S, \boldsymbol{E}^S)$. From Eq. (6), we know that $\boldsymbol{G}^S$ is generated by selecting a subset of $\boldsymbol{G}^R$, given the environmental features saved in $\boldsymbol{G}^R$ and $\boldsymbol{G}^P$. Such a generative problem is discrete, i.e., the nodes and edges should be embedded into categories. As shown in Fig. 3, we suppose the categories of each node and edge in $\boldsymbol{G}^R$ have a and b possibilities, respectively, and utilize the one-hot embedding to encode such categorical features.

The forward diffusion process gradually disturbs $\boldsymbol{G}_0^S$ into a random graph $\boldsymbol{G}_T^S$ by adding noise incrementally for T steps. Inspired by Digress [28], we define noise addition as applying a transition matrix to the current micro-segmentation state, which is suitable for discrete and graph-structured data. Denote the t-th ($t \in \{0, 1, \ldots, T\}$) transition matrices for nodes and edges as $\boldsymbol{A}_t^V$ and $\boldsymbol{A}_t^E$, respectively. The transition probabilities can follow uniform or marginal distributions [28]. Therefore, for each node v_i and edge $e_{i\leftrightarrow j} \in \boldsymbol{G}^R$, the category transition at the t-th step is defined as follows:

$$[\boldsymbol{A}_t^V]_{fg} = q\left(v_{i|t+1} = g | v_{i|t} = f\right), \tag{15a}$$

$$[\boldsymbol{A}_t^E]_{hk} = q\left(e_{i\leftrightarrow j|t+1} = k | e_{i\leftrightarrow j|t} = h\right), \tag{15b}$$

where $f, g \in \{1, \ldots, a\}$ and $h, k \in \{1, \ldots, b\}$, representing the category of the given node and edge, respectively. Hence, the forward diffusion process can be expressed as

$$q(\boldsymbol{G}_T^S | \boldsymbol{G}_0^S) = \prod_{t=1}^{T} q(\boldsymbol{G}_t^S | \boldsymbol{G}_{t-1}^S) = \prod_{t=1}^{T} (\boldsymbol{V}^S \boldsymbol{A}_t^V, \boldsymbol{E}^S \boldsymbol{A}_t^E) \tag{16}$$

Denoting $\bar{\boldsymbol{A}}^V = \boldsymbol{A}_1^V \cdots \boldsymbol{A}_T^V$ and $\bar{\boldsymbol{A}}^E = \boldsymbol{A}_1^E \cdots \boldsymbol{A}_T^E$, Eq. (16) can be rewritten as $(\boldsymbol{V}^S \bar{\boldsymbol{A}}^V, \boldsymbol{E}^S \bar{\boldsymbol{A}}^E)$.

2) Denoising: Denoising is the inverse process of forward diffusion, aiming to generate micro-segmentations that can maximize the utility defined in Eq. (14) from a random graph. Consequently, each denoising step can be expressed as [28]

$$p_\theta(\boldsymbol{G}_{t-1}^S | \boldsymbol{G}_t^S) = \prod_{1 \le i \le m} p_\theta(v_i^{t-1} | \boldsymbol{G}_t^S) \prod_{1 \le i,j \le m} p_\theta(e_{i\leftrightarrow j}^{t-1} | \boldsymbol{G}_t^S) \tag{17a}$$

$$= \int_{\boldsymbol{G}_0^S} p_\theta(\boldsymbol{G}_{t-1}^S | \boldsymbol{G}_t^S, \boldsymbol{G}_0^S) dp_\theta(\boldsymbol{G}_0^S | \boldsymbol{G}_t^S) \tag{17b}$$

$$= \sum_{\widetilde{\boldsymbol{G}}_0^S \in \mathcal{G}} \underbrace{p_\theta(\boldsymbol{G}_{t-1}^S | \boldsymbol{G}_t^S, \boldsymbol{G}_0^S)}_{\to q(\boldsymbol{G}_{t-1}^S | \boldsymbol{G}_t^S, \boldsymbol{G}_0^S)} p_\theta(\widetilde{\boldsymbol{G}}_0^S | \boldsymbol{G}_t^S), \tag{17c}$$

where m denotes the number of nodes, $p_\theta(\widetilde{\boldsymbol{G}}_0^S | \boldsymbol{G}_t^S)$ means a prediction on $\boldsymbol{G}_0^S$ given $\boldsymbol{G}_t^S$. To efficiently represent graph-structured data, we adopt a graph transformer as the denoising network to predict $\boldsymbol{G}_0^S$, which is demonstrated in Section IV-C. Since $q(\cdot)$ is pre-defined, $p_\theta(\boldsymbol{G}_{t-1}^S | \boldsymbol{G}_t^S)$ can be calculated. Hence, we can model the denoising process as a Markov chain, following [29]

$$\begin{aligned} \boldsymbol{s}_t &\triangleq \left(\boldsymbol{G}_{T-t}^S, T - t\right), \ \boldsymbol{a}_t \triangleq \boldsymbol{G}_{T-t-1}^S, \\ \pi_\theta\left(\boldsymbol{a}_t \mid \boldsymbol{s}_t\right) &\triangleq p_\theta\left(\boldsymbol{G}_{T-t-1}^S \mid \boldsymbol{G}_{T-t}^S\right), \\ r\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) &\triangleq \begin{cases} r\left(\boldsymbol{G}_0^S\right), & \text{if } t = T, \\ 0, & \text{if } t < T, \end{cases} \end{aligned} \tag{18}$$

where $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$ represent the state and action at t-th step, respectively. π_θ denotes the policy for micro-segmentation

and $r(\boldsymbol{s}_t, \boldsymbol{a}_t)$ denotes the reward. With the Markov decision processes being fixed, we can acquire a series of micro-segmentation generation trajectories, denoted as $\boldsymbol{\tau} = \{\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_T, \boldsymbol{a}_T\}$. The accumulative reward of each trajectory is $\sum_{t=0}^{T} r(\boldsymbol{s}_t, \boldsymbol{a}_t) = r(\boldsymbol{G}_0^S)$. Note that $r(\cdot)$ is defined based on Eq. (14) to measure the performance of the generated micro-segmentation in addressing the optimization problem, which is discussed in Section IV-E. Then, the expected accumulative reward of the agent can be derived as $\mathcal{J}(\theta) = \mathbb{E}_{p_\theta(\widetilde{\boldsymbol{G}_0^S}|\boldsymbol{G}_T^S)}\left[r(\boldsymbol{G}_0^S)\right]$.

Recall that our goal is to refine denoising network parameters θ, facilitating LEGD to generate the optimal micro-segmentations with the maximum utility, i.e., $\max_\theta \mathcal{J}(\theta)$. Following the policy-based learning principle, we use gradient ascent [30] to optimize the policy gradient $\nabla_\theta \mathcal{J}(\theta)$, thereby training LEGD to learn the optimal policy for micro-segmentation generation. Given $\mathcal{J}(\theta) = \mathbb{E}_{p_\theta(\widetilde{\boldsymbol{G}_0^S}|\boldsymbol{G}_T^S)}\left[r(\boldsymbol{G}_0^S)\right]$, the policy gradient can be derived as

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}\left[r(\boldsymbol{s}, \boldsymbol{a}) \nabla_\theta \log \pi_\theta(\boldsymbol{a}|\boldsymbol{s})\right] \tag{19a}$$

$$= \mathbb{E}_{\boldsymbol{\tau}}\left[r\left(\boldsymbol{G}_0^S\right) \sum_{t=1}^{T} \nabla_\theta \log p_\theta\left(\boldsymbol{G}_{t-1}^S \mid \boldsymbol{G}_t^S\right)\right], \tag{19b}$$

where Eq. (19b) is acquired by substituting $\boldsymbol{s}$, $\boldsymbol{a}$, and π_θ in Eq. (18) to Eq. (19a). Then, we perform Monte Carlo estimation on Eq. (19b), acquiring

$$\nabla_\theta \mathcal{J}(\theta) \approx \frac{T}{|\mathcal{D}| \cdot |\mathcal{T}|} \sum_{\boldsymbol{\tau} \in \mathcal{D}} \sum_{t \in \mathcal{T}} r(\boldsymbol{\tau}.\boldsymbol{s}_0) \nabla_\theta \log p_\theta(\boldsymbol{\tau}.\boldsymbol{s}_{t-1} \mid \boldsymbol{\tau}.\boldsymbol{s}_t), \tag{20}$$

where $\mathcal{D}$ and $\mathcal{T}$ represent the sets of sampled trajectories and timesteps, respectively. $\boldsymbol{\tau}.\boldsymbol{s}_0$ refers to the last generated micro-segmentation in trajectory $\boldsymbol{\tau}$, i.e., $\boldsymbol{G}_0^S$. In this way, we can efficiently estimate policy gradient and update parameters by interacting with the zero-trust network and generating a batch of trajectories that record the denoising process. Particularly, to facilitate training convergence and alleviate the effect of limited and unreliable Monte Carlo samples, we revise Eq. (20) to *eager policy gradient*, following [31]

$$g(\theta) = \frac{T}{|\mathcal{D}| \cdot |\mathcal{T}|} \sum_{\boldsymbol{\tau} \in \mathcal{D}} \sum_{t \in \mathcal{T}} r(\boldsymbol{\tau}.\boldsymbol{s}_0) \nabla_\theta \log p_\theta(\boldsymbol{\tau}.\boldsymbol{s}_0 \mid \boldsymbol{\tau}.\boldsymbol{s}_t). \tag{21}$$

Compared with Eq. (20), eager policy gradient focuses on the prediction of $\boldsymbol{G}_0^S$, thereby reducing the variance of gradient estimates. The rationality proof of this strategy can be found in [31]. Finally, the denoising network parameter θ can be updated by gradient ascent, i.e., $\theta' = \theta + \eta \cdot g(\theta)$, where η represents the learning rate.

C. LEGD Framework

We present our LEGD framework, consisting of layer fusion, trajectory collection, and the denoising network. Furthermore, we show the data and control flows along these components, which realize the LEGD training.

1) Layer Fusion: First, we design the input format for LEGD. As shown in Fig. 3-A, the inputs of the denoising process, i.e., $\boldsymbol{G}_T^S$, are randomly sampled micro-segmentations. Recall that micro-segmentation is generated by involving nodes and edges from $\boldsymbol{G}^R$. We let $\boldsymbol{G}^S$ share the same size with $\boldsymbol{G}^R$, i.e., m nodes and $m \times m$ edges. Then, since each edge may be in the Selected or Unselected state while node states are associated with connected edges, we use one-hot encoding with cardinalities *1* and *2* to encode nodes and edges, respectively. Consequently, we have $v_i \in \mathbb{R}, e_{i \leftrightarrow j} \in \mathbb{R}^2$, $\forall v_i, e_{i \leftrightarrow j} \in \boldsymbol{G}^S$, and $\boldsymbol{G}_T^S$ is constructed by randomly initializing these embeddings, as shown in Fig. 3-A.

Apart from $\boldsymbol{G}_T^S$, another input for LEGD is the environmental feature $\boldsymbol{F}^S$, which serves as the additional condition of p_θ, i.e., $p_\theta(\boldsymbol{G}_{t-1}^S | \boldsymbol{G}_t^S, \boldsymbol{F}^S)$. We can observe that $\boldsymbol{F}^S$ is the prerequisite to ensure that the generated micro-segmentations can maximize utility in the given network states. Particularly, both the physical and trust features from the hierarchical graph should be fused and incorporated in $\boldsymbol{F}^S$. Hence, we filter $\boldsymbol{G}^P$, deleting all the nodes and corresponding edges that do not exist in $\boldsymbol{G}^R$. Accordingly, the corresponding node- and edge-level features are also removed from $\boldsymbol{F}_V^P$ and $\boldsymbol{F}_E^P$. Denoting the filtered $\boldsymbol{G}^P$ as $\boldsymbol{G}^{P*}$, $\boldsymbol{F}^S$ is defined as

$$\boldsymbol{F}^S = \begin{cases} \boldsymbol{F}_V^S \leftarrow \text{Concat}(\boldsymbol{F}_V^{P*}, \boldsymbol{F}_V^R), \\ \boldsymbol{F}_E^S \leftarrow \text{Concat}(\boldsymbol{F}_E^{P*}, \boldsymbol{F}_E^R). \end{cases} \tag{22}$$

As illustrated by Fig. 3-A, $\boldsymbol{F}_V^{P*} \in \mathbb{R}^{m \times 3}$, $\boldsymbol{F}_V^R \in \mathbb{R}^{m \times 1}$, $\boldsymbol{F}_E^{P*}$ and $\boldsymbol{F}_E^R \in \mathbb{R}^{m \times m \times 1}$. Consequently, we can conclude that $\boldsymbol{F}_V^S \in \mathbb{R}^{m \times 4}$ and $\boldsymbol{F}_E^S \in \mathbb{R}^{m \times m \times 2}$. Different from $\boldsymbol{G}_T^S$ that initializes the denoising process, $\boldsymbol{F}^S$ is fed into the denoising network for learning $p_\theta(\widetilde{\boldsymbol{G}}_0^S | \boldsymbol{G}_t^S)$, thus facilitating the calculation of $p_\theta(\boldsymbol{G}_{t-1}^S | \boldsymbol{G}_t^S, \boldsymbol{F}^S)$. The denoising network design is demonstrated below.

2) Trajectory Collection: With $\boldsymbol{G}_T^S$ sampled, the current denoising network performs the denoising process. As illustrated in Fig. 3-B, once $\boldsymbol{G}_0^S$ is generated, the entire trajectory is saved in a replay buffer, where numerous trajectories are split into multiple training batches. The reward for each trajectory is $r(\boldsymbol{G}_0^S)$. As shown in Eq. (17), to calculate the *eager policy gradient*, we need the prediction of $\boldsymbol{G}_0^S$ given $\boldsymbol{G}_t^S$ and the additional condition $\boldsymbol{F}^S$, i.e., $p_\theta(\boldsymbol{G}_{t-1}^S | \boldsymbol{G}_t^S, \boldsymbol{F}^S)$. To achieve this, we randomly sample timesteps from $1, 2, \ldots, T$ for predictions, reducing computation overhead and avoiding model overfitting [15]. The LEGD algorithm continuously interacts with the zero-trust network, denoising different $\boldsymbol{G}_T^S$ and $\boldsymbol{F}^S$, thus enriching the replay buffer. A batch of trajectories is fetched at each epoch, and the *eager policy gradient* is calculated to update model parameters.

3) Denoising Network: The denoising network is the learnable component of LEGD, which predicts the clean micro-segmentation $\boldsymbol{G}_0^S$ based on input $\boldsymbol{G}_t^S$ and condition $\boldsymbol{F}^S$. As shown in Eq. (17c), two requirements should be satisfied since the prediction of $\boldsymbol{G}_0^S$ directly determines the performance of micro-segmentation generation. First, the denoising network should effectively represent complicated topologies and graphical features, learning the graph generation policy. Moreover, it should support additional conditions (i.e., $\boldsymbol{F}^S$), thus enabling the controllable generation for maximizing utilities. To this end, we build the denoising network for LEGD based on graph transformer architecture [31], [32].

As shown in Fig. 3-C, nodes $\boldsymbol{V}_t^S$ and edges $\boldsymbol{E}_t^S$ first undergo the corresponding Multi-layer Perception (MLP) modules [28] to acquire the embeddings. Afterward, a graph transformer layer is deployed, incorporating the graph attention mechanism. Through graph attention scores, graphical features, especially the message transfers among nodes along edges,

can be reflected. Similar to the conventional attention [33], the score of the ℓ-th attention layer is defined as

$$\hat{\boldsymbol{w}}_{ij}^{k,\ell} = \text{FiLM}\left(\left(\frac{\boldsymbol{Q}^{k,\ell}\boldsymbol{h}_i^{\ell} \cdot \boldsymbol{K}^{k,\ell}\boldsymbol{h}_j^{\ell}}{\sqrt{d_k}}\right), \boldsymbol{E}^{k,\ell}\boldsymbol{h}_{i\leftrightarrow j}^{\ell}\right), \quad (23a)$$

$$\boldsymbol{w}_{ij}^{k,\ell} = \text{softmax}(\hat{\boldsymbol{w}}_{ij}^{k,\ell}), \quad (23b)$$

where $k \in \{1, 2, \ldots, H\}$ denotes the index of attention head. $\boldsymbol{h}_i$ and $\boldsymbol{h}_{i\leftrightarrow j}$ represent the embeddings of v_i and $e_{i\leftrightarrow j} \in \boldsymbol{G}^S$, respectively. Eq. (23a) shows that for each node, its attention to all neighbors and the corresponding edges will be incorporated, thus informing the denoising network with graph-perspective knowledge. Note that FiLM refers to the *Feature-wise Linear Modulation* layer [34], which can modify the output of the transformer layer by applying a scaling and shifting operation that is conditioned on $\boldsymbol{F}^S$. Hence, the processed node and edge features can be defined as

$$\hat{\boldsymbol{h}}_i = \text{FiLM}\left(\|_{k=1}^{H}\left(\sum_j \boldsymbol{w}_{ij}^{k,\ell}\boldsymbol{V}^{k,\ell}\boldsymbol{h}_j^{\ell}\right), \boldsymbol{F}^S\right), \quad (24a)$$

$$\hat{\boldsymbol{h}}_{i\leftrightarrow j} = \text{FiLM}\left(\|_{k=1}^{H}\left(\hat{\boldsymbol{w}}_{ij}^{k,\ell}\right), \boldsymbol{F}^S\right), \quad (24b)$$

where $||$ represents the Concat operation that combines the outputs of H attention heads. Afterward, the processed node and edge features go through residual connection and layer normalization [33], which are utilized to combat the vanishing gradient problem and stabilize the training process, respectively. Finally, two MLPs are deployed to decode node and edge, respectively. In this way, the embeddings can be transferred to node and edge matrices that correspond to the predicted micro-segmentation $\boldsymbol{G}_0^S$.

D. Reward Engineering

To provide efficient feedback for every generated $\boldsymbol{G}_0^S$ and guide the LEGD training, the reward function $r(\cdot)$ should be crafted, so-called reward engineering. Considering both the objective function and constraints, we present the reward function that considers both the direct reward and the penalty terms. The former reflects the unconstrained user utility, and the latter ensures compliance with the constraints. To harmonize the relationship between these two aspects, we define a penalty-constrained reward expression, i.e.,

$$r(\boldsymbol{G}_0^S) = U_{\text{U}}\left(\{\boldsymbol{G}^P, \boldsymbol{G}^R, L_s, T_s, P_s, E_s\}\right) \times \underbrace{(\Omega_{\text{T}} \times \Omega_{\text{L}})}_{\text{penalty terms}}, \quad (25)$$

where $\Omega_{\text{T}} = \mathcal{I}(T_s, 1)$ and $\Omega_{\text{L}} = \mathcal{I}(L_{\max}, L_s)$. Note that $\mathcal{I}(*, \circ)$ denotes the indicator function, which outputs *1* if $* \geq \circ$ and *0* otherwise. Substituting Eq. (25) into Eq. (21) and establishing the framework illustrated in Section IV-C, the basic LEGD algorithm is completely demonstrated and can address the controllable micro-segmentation generation problem defined by Eq. (13).

E. LLM Enhancement

Similar to GNNs, LEGD may take a long time to converge as the denoising network struggles to capture the graphical features and generate optimal micro-segmentations in varying environments. With the increasing network scale and complexity in the NGN era [2], we aim to improve LEGD's efficiency. Inspired by Reinforcement Learning with Human Feedback (RLHF) [16], we exploit human perception and expertise. While humans cannot directly generate optimal micro-segmentations due to numerous possibilities and intensive calculations, they excel at recognizing patterns and identifying outliers [35]. This allows us to remove certain graph parts and simplify the generation problem. For example, in Fig. 1-A, humans can easily filter out text-to-3D service providers since they are not required by users. In contrast, neural networks require numerous samples and training iterations to learn such patterns. Conventional RLHF relies heavily on real-time human feedback, which is labor-intensive and unstable. Fortunately, LLMs, trained on massive datasets, possess human-like expertise and multimodal understanding. To this end, we leverage an LLM-empowered agent to simulate human perception and compress LEGD's action space. Specifically, the agent is informed with the following information

$$\mathcal{A}_{\text{LEGD}} = \text{LLM}(\boldsymbol{f} \,|\, \boldsymbol{G}_T^S, \boldsymbol{G}^{P*}, \boldsymbol{G}^R, U_{\text{U}}), \quad (26)$$

where $\boldsymbol{G}_T^S$ informs LLM with the original state space structure. Graphs $\boldsymbol{G}^{P*}$ and $\boldsymbol{G}^R$ provide the environmental context, i.e., the features of each node and edge. U_{U} indicates the requirements for the generated micro-segmentations. Then, $\mathcal{A}_{\text{LEGD}}$ can analyze the aforementioned information and generate a series of heuristic features $\boldsymbol{f} = \{f_1, f_2, \ldots, f_n\}$. Each filter corresponds to one rule that removes certain nodes' candidacy for constructing micro-segmentation. Thus, the state space of LEGD can be effectively shrunk by applying these filters, i.e.,

$$\boldsymbol{G}_T^{S*} = \Psi_{\varphi\sim[0,1]}\left(\left(\prod_{i=1}^{n} f_i\right)\boldsymbol{G}_T^S\right), \quad (27)$$

where temperature mechanism Ψ controls the strictness of compliance with $\mathcal{A}_{\text{LEGD}}$'s rules. When $\varphi = 1$, all nodes filtered by $\mathcal{A}_{\text{LEGD}}$ are removed, minimizing the state space. Lower φ values allow better exploration, as the filters adopted by $\mathcal{A}_{\text{LEGD}}$ may not be perfect. To ensure adaptability, $\mathcal{A}_{\text{LEGD}}$ is designed as a pluggable module with the trajectory collector. Any LLM with graph understanding capability, such as ChatGPT-4 [36] or LLaVA [37], can be applied. Moreover, $\mathcal{A}_{\text{LEGD}}$ works in an online manner during the LEGD training, thereby dynamically configuring and activating filters. The entire workflow of LEGD is detailed in **Algorithm 1**.

F. Complexity Analysis

1) Computational Complexity: We denote L the number of graph transformer layers in LEGD's denoising network. At the beginning of each generation, LEGD initializes the graph structure $\boldsymbol{G}_T^S$ (including node and edge states), causing $\mathcal{O}(m^2)$ complexity. Then, $\mathcal{A}_{\text{LEGD}}$ is activated, whose computation complexity is denoted as $\mathcal{O}(C_{\text{LLM}})$. The filter is applied to deactivate certain action space elements. Such operation requires traversing the action space $|\mathcal{A}|$, causing $\mathcal{O}(m^2)$ overhead. Hence, overall initialization complexity is $\mathcal{O}(C_{\text{LLM}} + m^2)$.

In each training round, the denoising network generates micro-segmentation $\boldsymbol{G}_0^S$ from $\boldsymbol{G}_T^S$ over T successive steps. Given that LEGD's denoising network is Transformer-based, the computational complexity for attention score calculation at each layer is $\mathcal{O}(m^2)$. Hence, the overall computational

Algorithm 1 The Proposed LEGD algorithm

Input: LLM-empowered agent $\mathcal{A}_{\text{LEGD}}$, initial denoising network π_θ, network structure and features $\boldsymbol{G}^R$ and $\boldsymbol{G}^{P*}$, temperature value φ, # of diffusion steps T, # of timestep samples $|\mathcal{T}|$, # of trajectory sample $|\mathcal{D}|$, learning rate η, # of training epochs E;
Procedure 1: LLM-based Enhancement;
  Inform LLM with $\boldsymbol{G}^R$ and $\boldsymbol{G}^{P*}$
  Acquire filters $\boldsymbol{f} = \{f_1, f_2, \ldots, f_n\}$
  $\boldsymbol{G}_T^{S*} = \Psi_\varphi\left(\left(\prod_{i=1}^n f_i\right)\boldsymbol{G}_T^S\right)$
Procedure 2: LEGD Training;
  for $i = 1, 2, \ldots, E$ **do**
    for $d = 1, 2, \ldots, |\mathcal{D}|$ **do**
      Sample $\boldsymbol{G}_T^{S*}$ based on *Procedure 1*
      for $t = 1, 2, \ldots, T$ **do**
        Perform denoising using π_θ based on Eq. (17)
      Acquire trajectory $\boldsymbol{\tau}_d$
      Sample timesteps $\mathcal{T}_d \sim \text{Uniform}([1, T])$
      Calculate reward r_d using the $\boldsymbol{G}_0^S$ of all the timesteps $\in \mathcal{T}_d$ based on Eq. (25)
    Calculate reward mean $\bar{r} \leftarrow \frac{1}{|\mathcal{D}|}\sum_{d=1}^{|\mathcal{D}|} r_d$
    Calculate reward variance std$[r] \leftarrow \sqrt{\frac{1}{|\mathcal{D}|-1}\sum_{d=1}^{|\mathcal{D}|}(r_d - \bar{r})^2}$
    Calculate eager policy gradient $g(\theta)$ based on $\bar{r}$, std$[r]$, and Eq. (21)
    Update π_θ parameters
Procedure 3: LEGD Inference;
  Sample $\boldsymbol{G}_T^{S*}$ based on *Procedure 1*
  for $t = 1, 2, \ldots, T$ **do**
    Perform denoising based on Eq. (17)
Output: Generated micro-segmentation $\boldsymbol{G}_0^S$

complexity of L-layer graph transformer for one diffusion step is $\mathcal{O}(L\,m^2)$. Repeating for T denoising steps yields:

$$\text{C}_{\text{denoising}} = \mathcal{O}\Big(T\,L\,m^2\Big). \tag{28}$$

During training, LEGD collects $|\mathcal{D}|$ trajectories per episode. Each trajectory is a whole denoising process with T denoising steps, calling the complexity in Eq. (28). Thus, the computational complexity of trajectory collection in a single episode is $\mathcal{O}\big(|\mathcal{D}|\,T\,L\,m^2\big)$. Additionally, the policy-gradient backpropagation has a complexity of $\mathcal{O}\big(b\,|\theta|\big)$, where b and $|\theta|$ are the batch size and parameter size of the denoising network, respectively. Repeating over E episodes, the total training complexity can be expressed as

$$\text{C}_{\text{total}} = \mathcal{O}\Big(E \times \big(|\mathcal{D}|\left(C_{\text{LLM}} + (T\,L+1)\,m^2\right) + b|\theta|\big)\Big). \tag{29}$$

Inference corresponds to generating one finalized micro-segmentation once the LEGD model is well-trained. In this case, no parameter update is required, and the computational complexity per inference call can be expressed as:

$$\text{C}_{\text{inference}} = \mathcal{O}\Big((T\,L+1)\,m^2\Big). \tag{30}$$

2) Storage Complexity: During the running of LEGD, two parts of data should be preserved, i.e., the denoising network and the trajectories. First, the storage complexity for saving the denoising network equals its parameter size, i.e., $|\theta|$. Additionally, the sampled trajectories should be preserved, whose format is $\{\boldsymbol{G}_T^S, \boldsymbol{G}_{T-1}^S, \ldots, \boldsymbol{G}_0^S\}$. Considering that each graph state consumes $\mathcal{O}(m^2 + m)$ storage and $|\mathcal{D}|$ trajectories are collected, the corresponding storage complexity is $\mathcal{O}\left(T\left(m^2 + m\right)\right)$. Note that the $\mathcal{O}(m)$ part can be omitted. Therefore, the overall storage complexity of LEGD is

$$\text{Storage Complexity} = \mathcal{O}\left(|\theta| + |\mathcal{D}|\,T\,m^2\right). \tag{31}$$

3) Sample Complexity: Sample complexity refers to the number of environment interactions needed to learn an ε-optimal policy, where ε represents the precision or optimality gap [38]. LEGD models discrete graph generation as an MDP and utilizes policy gradient for optimization. In addition, as shown in Eq. (21), the policy gradient is estimated using the Monte Carlo method. Each Monte Carlo sample can be denoted as

$$X_j = r(\boldsymbol{\tau_j}.\boldsymbol{s}_0)\nabla_\theta \log p_\theta(\boldsymbol{\tau_j}.\boldsymbol{s}_0 \mid \boldsymbol{\tau_j}.\boldsymbol{s}_t), \forall j \in \{1, 2, \ldots, N\}. \tag{32}$$

We denote the mean and variance of X_j as g_i and σ^2, respectively. By the Central Limit Theorem, as N increases, the distribution of the estimator $\hat{g}_i$ approaches a normal distribution, i.e.,

$$\hat{g}_i \sim \mathcal{N}\left(g_i, \frac{\sigma^2}{N}\right). \tag{33}$$

Hence, the mean squared error (MSE) of the estimator is:

$$\text{MSE}(\hat{g}_i) = \mathbb{E}\left[(\hat{g}_i - g_i)^2\right] = \frac{\sigma^2}{N}. \tag{34}$$

To achieve ε precision, we require $\text{MSE}(\hat{g}_i) \le \varepsilon^2$. Substituting the MSE expression in Eq. (34), we can acquire $\frac{\sigma^2}{N} \le \varepsilon^2$. Hence, $N \ge \frac{\sigma^2}{\varepsilon^2}$. Since σ^2 is a constant that depends on the environment and policy, we can acquire

$$N = \mathcal{O}\left(\frac{1}{\varepsilon^2}\right). \tag{35}$$

Eq. (35) demonstrates that for accurate gradient estimation to ε precision, we need $\mathcal{O}(1/\varepsilon^2)$ samples per policy parameter. In our implementation, we use a diffusion-based policy, whose trainable module is a graph transformer-based denoising network with multiple attention heads for $\boldsymbol{G}_0^S$ prediction. In this case, the number of policy parameters $|\theta|$ is determined by the dimensions of the input state space and output action space. This is because, as shown in Fig. 3, the MLP layers map from state dimensions to hidden dimensions and from hidden dimensions to action dimensions. The dimensions of the MLP layers are directly influenced by the sizes of the state and action spaces since each additional dimension in the state or action representation generally requires additional neurons to accurately capture the underlying functional relationships and ensure sufficient representational capacity [39]. In LEGD, we scale MLP dimensions with $|\mathcal{S}|$ and $|\mathcal{A}|$. Hence, we can acquire that

$$|\theta| \propto |\mathcal{S}| \;\&\; |\theta| \propto |\mathcal{A}|. \tag{36}$$

Since $|\theta|$ independently scales with both $|\mathcal{S}|$ and $|\mathcal{A}|$ (because the MLPs for input encoding and output decoding are designed independently), we can derive that $|\theta| \propto |\mathcal{S}||\mathcal{A}|$. Then, based on such observation and Eq. (35), we can derive that

$$\text{Sample Complexity} \propto \frac{|\mathcal{S}||\mathcal{A}|}{\varepsilon^2}. \tag{37}$$

Additionally, the distribution mismatch coefficient D_∞ influences the sample complexity by quantifying how different the

policy-induced distribution is from the state distribution used for training. A larger mismatch typically requires more samples to accurately estimate policy performance [38]. Similarly, the discount factor $(1-\gamma)$ controls the effective horizon length, significantly affecting the sample complexity by necessitating more interactions to reliably estimate the value of long-term rewards [38]. According to [38] and [40], the sample complexity of the proposed LEGD algorithm can be expressed as:

$$\text{Sample Complexity} = \mathcal{O}\left(\frac{D_\infty^2|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^{w_1}\varepsilon^2}\right), \tag{38}$$

where $D_\infty := \max_s\left(\frac{d_{s_0}^{\pi^*}(s)}{\mu_s}\right)$. $d_{s_0}^{\pi^*}(s)$ is the fraction of time spent in state s when executing an optimal policy π, starting from the state s_0 [38]. μ is the starting state distribution for LEGD. w_1 is determined by the specific training settings (e.g., the usage of softmax parameterization) [38]. From the above analysis, we can theoretically demonstrate the rationale of applying $\mathcal{A}_{\text{LEGD}}$ to enhance graph diffusion: LLM effectively compresses the actual action space $\mathcal{A}$ by filtering out irrelevant or low-utility actions based on domain knowledge and heuristic rules, allowing us to improve the sample efficiency. In addition, such strategic reduction transforms the original expansive action space of $\mathcal{O}(m^2)$ into a more manageable subspace. Consequently, the efficiency of action space exploration can be significantly improved [41].

V. Adaptive Micro-Segmentation Maintenance

LEGD realizes the optimal micro-segmentation generations for the specific network states. Nonetheless, in zero-trust NGN, two situations may occur.

- **Trustworthiness Update**: Zero-trust paradigms involve continuous trustworthiness updates [3]. As presented in Section III, an increase in trustworthiness benefits the service experience. In contrast, if one entity loses the trust of neighbors due to suspicious behaviors, the execution success rate of its SFC will be sharply reduced. Moreover, if its mutual trustworthiness drops below the threshold, it might be removed by PE, causing the failure of the entire micro-segmentation.
- **Service Type Upgrade**: The services in the NGN era are rapidly evolving. Taking Fig. 1-A as an example, the emergence of GPT-4, which supports conversational graph generations, greatly reduces the demand for text-to-image services, such as Stable Diffusion. Hence, a topology of micro-segmentation should evolve according to the changing service requirements.

Both situations require dynamic micro-segmentation maintenance, i.e., keeping the overall micro-segmentation topology while involving/removing certain nodes/edges to adapt to the new environment/services. To this end, we further present LEGD-AM, which performs task-oriented fine-tuning atop LEGD to fit varying zero-trust networks.

A. Fine-Tuning of LEGD

First, we utilize the well-trained LEGD model (parameterized by θ) as the checkpoint. Suppose that after the LEGD training converges, the model can efficiently generate micro-segmentation for {type-*1*, type-*2*, type-*3*} services. During the operation, one type-*2* device is removed, or the micro-segmentation needs to upgrade the service type to {type-*1*, type-*2*, type-*3*, type-*4*}, we should involve/remove certain nodes (and the corresponding edges) while retaining the overall topology. Hence, we leverage the fine-tuning technique [42] to align the pre-trained model with new objectives. The fine-tuning process of LEGD can be expressed as

$$\theta^* = \theta + \eta_f g_\theta\left(\theta; r_f(\boldsymbol{G_0^{S*}} \mid \boldsymbol{G_0^S})\right), \tag{39}$$

where η_f represents the fine-tuning learning rate. $\boldsymbol{G}_0^{S*}$ means the refined micro-segmentation based on the new objective, which is expressed by the fine-tuning reward function $r_f(*)$. $g_\theta\left(\theta; r_f(\boldsymbol{G_0^{S*}} \mid \boldsymbol{G_0^S})\right)$ denotes the eager policy gradient for fine-tuning. Compared with training a model from scratch, fine-tuning requires much fewer epochs to converge since the pre-trained knowledge can be utilized.

B. LEGD-AM Design

1) Adaptive Mask: To perform fine-tuning, first, we present adaptive masks to constrain the structure of $\boldsymbol{G}_0^{S*}$, thereby aligning it to the fine-tuning objective. Masks act similarly to the filters mentioned in Section IV, which disable a part of nodes' candidacy for participating in $\boldsymbol{G}_0^{S*}$ during the entire denoising process. Such an operation can improve training efficiency by allowing the denoising network π to selectively focus on the most relevant graph parts. Moreover, it can minimize the number of nodes and edges that can be adjusted, thus reducing micro-segmentation costs. This is because the involvement/removal of each node requires PE to redeploy the network virtualization, re-evaluate the trustworthiness, etc. [14]. To develop adaptive masks for LEGD-AM, we first define the concept of an ϵ-*Interest Zone*.

Definition 2 (ϵ-Interest Zone): For given graph $\boldsymbol{G}^R$, an ϵ-interest zone is defined as the subset of $\boldsymbol{G}^R$, which includes all nodes (and the corresponding edges) that provide the required type of services and their ϵ-degree neighbors.

Fig. 4 demonstrates the meaning of ϵ-interest zone. Suppose that one type-*2* node is removed due to over-low trustworthiness. In this case, to fix the micro-segmentation, the required service type is *2*. The corresponding *1*- and *2*-interest zones, denoted by $\mathbb{Z}^1$ and $\mathbb{Z}^2$, respectively, are illustrated in Fig. 4-B and Fig. 4-C. Accordingly, masks filter the graph parts outside the interest zone and can be designed as

$$\boldsymbol{M} \in \mathbb{R}^{m\times m},\ [\boldsymbol{M}]_{ij} = [\boldsymbol{M}]_{ji} = \begin{cases} 1, & \text{if } f_{v_i}^R \notin \mathbb{Z}^\epsilon \\ 0, & \text{otherwise,} \end{cases} \tag{40}$$

where $i,\ j \in \{1, 2, \ldots, m\}$. Similar to Eq (27), the adaptive masks are applied to $\boldsymbol{G}_T^S$. Moreover, a temperature mechanism like Ψ can also be applied to balance training complexity and exploration capability.

2) Adaptive Reward Engineering: The reward $r_f(*)$ should be crafted to guide fine-tuning, effectively indicating the desirability of each action in refining the micro-segmentation. Considering that fine-tuning aims to balance the utility of the update micro-segmentations and the re-configuration costs, we define the following two-term reward for LEGD-AM:

$$r_f(\boldsymbol{G}_0^{S*} \mid \boldsymbol{G}_0^S) = U_{\text{U}} - \underbrace{\alpha_4 \Delta(\boldsymbol{G}_0^S, \boldsymbol{G}_0^{S*})}_{\text{re-configuration costs}}, \tag{41}$$

where U_{U} is defined in Eq. (14) and represents the utility of the refined micro-segmentation $\boldsymbol{G}_0^{S*}$. α_4 is a weighting factor. $\Delta(\boldsymbol{G}_0^S, \boldsymbol{G}_0^{S*})$ measures the topological difference between the original micro-segmentation $\boldsymbol{G}_0^S$ and the refined one $\boldsymbol{G}_0^{S*}$,

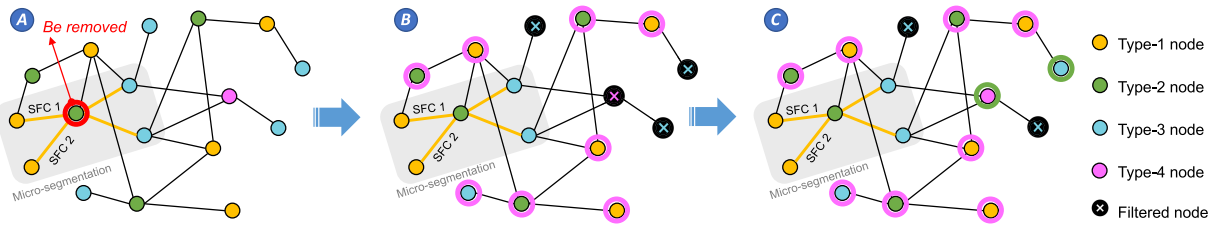


Fig. 4. The illustration of an interest zone. **A**: The zero-trust network and the original micro-segmentation. Note that a type-*2* node is removed due to over-low trustworthiness. **B**: 1-degree interest zone, which is highlighted by pink circles, including all remaining type-*2* nodes and their 1-degree neighbors. **C**: 2-degree interest zone, adding the 2-degree neighbors, which are highlighted in green. The nodes marked by black crosses will be marked and filtered by the adaptive mask.

which can be calculated using well-established metrics such as *graph edit distance* [43]. From Eqs. (40) and (41), we can observe that LEGD-AM ensures the adaptability of our service provisioning framework in zero-trust NGN since both the mask and reward function can be flexibly designed according to the latest environment/requirements.

VI. Numerical Results

In this section, we implement the proposed zero-trust service provisioning framework and the micro-segmentation generation and update algorithms. Then, we conduct extensive experiments that aim to answer: 1) whether the LEGD algorithm can generate the micro-segmentations that maximize utility U_{U} and 2) whether the LEGD-AM algorithm can efficiently maintain micro-segmentation topologies according to the varying network environments.

Experimental Settings. The experiments are conducted on a server with an NVIDIA RTX A5000 GPU with 24 GB of memory and an AMD Ryzen Threadripper PRO 3975WX 32-Core CPU with 263 GB of RAM in Ubuntu 20.04 LTS. We utilize this server to simulate a nine-node zero-trust AIGC network. We suppose that the network contains four types of service providers, namely text generation, text-to-image synthesis, image-to-video creation, and interactive video enhancement. These types are indexed by *1*, *2*, *3*, and *4*, respectively. Users request text-controlled video generation services, i.e., {type-*1*, type-*2*, type-*3*}. The zero-trust architecture is built following the NIST standard [11]. Additionally, we adopt reputation [4] and ChatGPT-4 [36] as the trustworthiness scheme and LLM-empowered agent, respectively.

A. Filter Generation and Dynamic Activation

The LLM-empowered agent improves LEGD by generating and dynamically activating multiple filters. Based on the graphic network modeling (in Section III) and the utility function for micro-segmentation generation (in Section IV), the heuristic filters provided by the agent include:

- **Functional Filter**: Targets and removes nodes that do not meet current service requirements, streamlining the network structure to better fit operational needs.
- **Threshold Filter**: Evaluate nodes and edges based on real-time assessments of resource capacity and trustworthiness, selectively removing those with low resources or trustworthiness during the denoising process.
- **Chain Filter**: Analyzes and excludes unnecessary service chains, optimizing the path of service function chains to enhance overall network efficiency.

As shown in Fig. 5, during LEGD training, the agent keeps monitoring the dynamic user requirements, network states, and optimization objective U_U, and manages the filter activation accordingly. For example, agent's decision

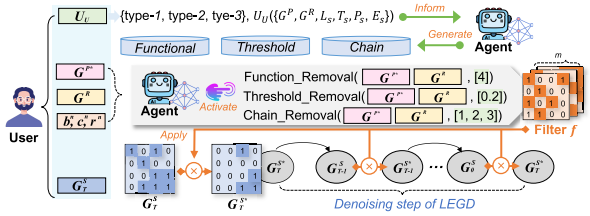


Fig. 5. The generation and activation of heuristic filters by the LLM-empowered agent.

{"Function": [4], "Trust": [0.2], "Chain":[1, 2, 3]} means activating the following mechanisms to form a combined filter that: 1) Remove any connection with type-*4* nodes; 2) Remove the edges where the trustworthiness is lower than 0.2; and 3) Remove connections outside service chains {type-*1*, type-*2*, type-*3*}. This setup enables LEGD to combine the neural network's learning capability and heuristic expertise knowledge, thus further improving efficiency. Note that more application-specific filters can be generated to accommodate more specific application requests and user preferences. Since our experiments mainly intend to demonstrate the effectiveness of LLM enhancement, we adopt the filters mentioned above.

B. Efficiency of LEGD

1) Baseline Settings: We adopt the following baselines: i) **No-Segmentation**, where the entire zero-trust network forms a single micro-segmentation and service requests are randomly assigned to one SFC; ii) **Random**, where the zero-trust network is randomly divided into multiple micro-segmentations and service requests are randomly assigned to one micro-segmentation; iii) **Greedy-Trust**, which uses Metis [44], a widely adopted graph partitioning algorithm with minimal costs. We set mutual trustworthiness as the cost to reflect the greedy strategy for maximizing mutual trustworthiness of generated micro-segmentations; iv) **Greedy-Resource**, which also uses Metis but sets the total volume of computing and transmission power as the costs, reflecting the greedy strategy for maximizing the total resources of generated micro-segmentations; v) **GNN-enhanced DRL**, which follows the conventional policy-based DRL paradigm [30] rather than generative architectures. Note that to ensure fairness, the GNN adopted by DRL for graph representation and understanding is the same as that of LEGD.

2) Service Provisioning Optimization: Fig. 6 illustrates the average reward (i.e., *AvgReward*) obtained by LEGD and the aforementioned baselines. We can observe that no-segmentation acquires an AvgReward of 9.79 since the entire network is unorganized. Random segmentation performs better than no-segmentation, providing the basic service load balancing. By greedily segmenting networks via resources and trustworthiness, greedy algorithms can further improve the

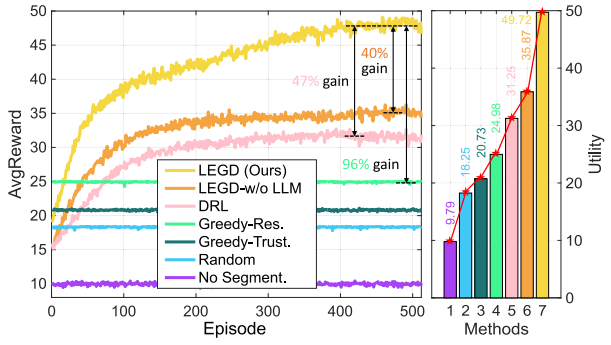


Fig. 6. The AvgReward of LEGD and other baselines over training episodes.

utility of generated micro-segmentations. With GNNs for representing graphical features and learning policies, DRL greatly outperforms non-learning methods. The proposed LEGD, however, achieves a higher AvgReward due to the outstanding exploration capability empowered by the generative diffusion architecture. Moreover, we incorporate LLM, which activates the filter that removes all edges connected to type-*4* nodes. We can observe that initially, LEGD and LEGD (w/o LLM) start from similar values. However, as the number of episodes increases, the superiority of LEGD gradually increases and finally achieves 40% higher AvgReward, which demonstrates that the LLM filters facilitate the training of LEGD to achieve higher efficiency and avoid sub-optimal. High AvgReward means that the generated micro-segmentation can provision services with high throughput, low latency, and high execution success rate. Hence, the effectiveness of the proposed LEGD in optimizing service provisioning can be demonstrated.

3) Impact of Hyper-Parameters: In this part, we examine LEGD's performance under different configurations and explore the impact of hyper-parameter settings on it. First, Fig. 7(a) illustrates the training curves of LEGD using different learning rates. We can observe that setting the learning rate to 10^{-4} can lead to an AvgReward of 68.5. In comparison, training LEGD by a learning rate of 10^{-5} achieves faster convergence, while the model fails into sub-optimal. Learning rate 5×10^{-5} requires more epochs to converge, and the utility of the generated micro-segmentation is lower than that of 10^{-4}. Hence, we fix the learning rate for all the remaining experiments as 10^{-4}.

Then, Fig. 7(b) compares the AvgReward of LEGD with varying numbers of diffusion steps. Performing 45 diffusion steps for each denoising process can lead to the best performance. However, the time consumption of the denoising process increases linearly with the increasing diffusion step numbers [15]. As shown in Fig. 7(b), generating a 2048-graph batch takes 18.9s, 37.8s, and 75.6s, for 15, 30, and 45 diffusion steps, respectively. Consequently, even if fewer approaches are required to converge, the blue curve's training time is long. Therefore, we set 30 as the number of diffusion steps, thus striking a balance between performance and training costs.

Finally, we explore the effectiveness of heuristic filter activation by the LLM agent. To do so, we adjust the agent configuration and temperature values to activate different types of filters. As shown in Fig. 7(c), trust-oriented filters can effectively improve the performance of LEGD since they mask the edges associated with low trustworthiness. Thus, the denoising network can learn to avoid such edges, ensuring the high execution success rate of the generated micro-segmentations. Nonetheless, if the heuristic filters remove a large graph part, such as resource-oriented ones and *trust-3*, they may hinder LEGD's exploration of the zero-trust networks, which may affect the performance or even make the model fail into sub-optimal. In conclusion, incorporating heuristic expertise into graph diffusion can effectively refine models' graphical understanding ability and reduce the actual action space. Moreover, the LLM can efficiently generate and activate heuristic filters, thus achieving higher utility for micro-segmentation generations in most cases.

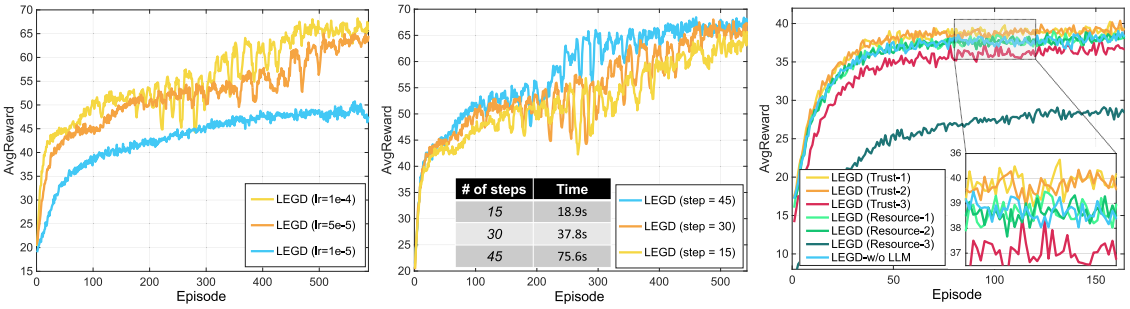
4) Micro-Segmentation Generation: We showcase two denoising processes to illustrate the effectiveness of the proposed LEGD algorithm in controllable micro-segmentation generation. As shown in Fig. 8, the initial micro-segmentation is randomly sampled, with only a few edges. With denoising being performed, more edges are generated and involved, thereby forming the final micro-segmentation. From networks 1 and 2, we can observe that LEGD realizes the optimal micro-segmentation generation through the following operations. First, the heuristic filters generated by the LLM-empowered agent help LEGD remove certain edges, such as the edges connected to type-*4* nodes. Then, the denoising process will generate new edges. Since newly generated edges are not guaranteed to make positive progress in optimizing the micro-segmentation, the reward function is utilized to train the denoising network for evaluating each action's desirability. Undesired edges are removed by the following steps (e.g., step 0 $\rightarrow$ step 9 of network 1). Finally, the generated edges, as well as the corresponding nodes, form the optimal micro-segmentation. Moreover, we track utility trends during the denoising process and observe that utility generally increases with more diffusion steps. This indicates the denoising network is well-trained and also validates our reward function.

C. Scalability of LEGD

Here, we perform extensive experiments to demonstrate the convergence and scalability of LEGD under heterogeneous NGN environments. Specifically, we construct varying scenarios by adjusting the following factors:

- **Network Scale**: We evaluate the convergence and scalability of LEGD by varying the network scale (i.e., the number of nodes). Considering that the existing experiments are performed in a nine-node network, we expand our evaluation to 6-, 16-, and 26-node networks.
- **Node Distribution Topology**: The existing experiments assume evenly distributed nodes. We adjust the node distribution to construct tree-typed, two-cluster, and US-Nobel [45] topologies. This group of experiments aims to evaluate LEGD's capability to adapt to diverse network topologies in practical deployments.
- **SFC Type**: To comprehensively evaluate LEGD's generalization capability, we consider three additional micro-segmentation settings in the 16-node network, namely {type-*2*, type-*3*, type-*1*}, {type-*1*, type-*3*, type-*4*}, and {type-*1*, type-*2*, type-*3*, type-*4*}. Note that in this last case, the network contains five types of servers.

Figs. 9–11 illustrate the performance of our proposed LEGD method across various experimental configurations. We can observe that LEGD converges stably in all test cases, exhibiting great robustness and scalability. Furthermore, LEGD consistently demonstrates superior performance compared to LEGD (w/o LLM), achieving a 16.3%-60.8% improvement in AvgReward. This significant performance increment is maintained regardless of network topology, node distribution pattern, or service chain types, demonstrating the efficiency of LLM enhancement in diverse operational scenarios.



(a) The impact of learning rate. (b) The impact of diffusion step. (c) The impact of filter activation.

Fig. 7. The impact of LEGD configurations on AvgReward. Note that for subfigure (c), trust-1, -2, and -3 refer to removing edges with mutual trustworthiness lower than 0.1, 0.3, and 0.5, respectively. In addition, resource-1, -2, and -3 refer to removing nodes with [computing, transmission] power less than [50GFLOPS, 20W], [70GFLOPS, 30W], and [90GFLOPS, 40W], respectively.

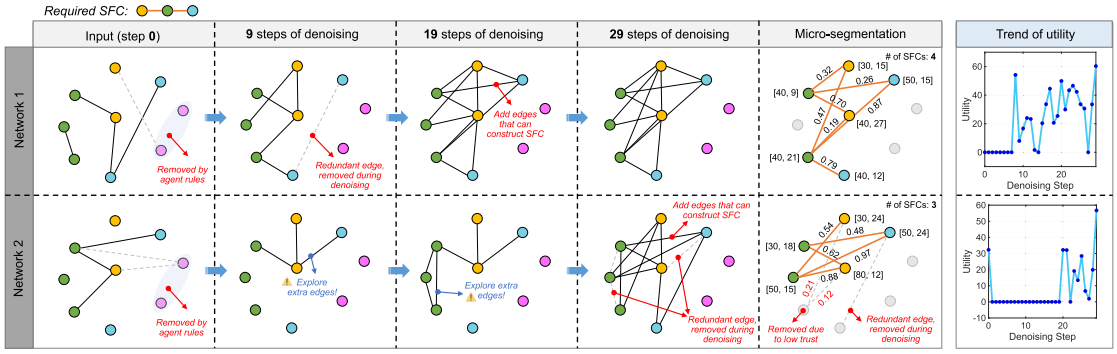
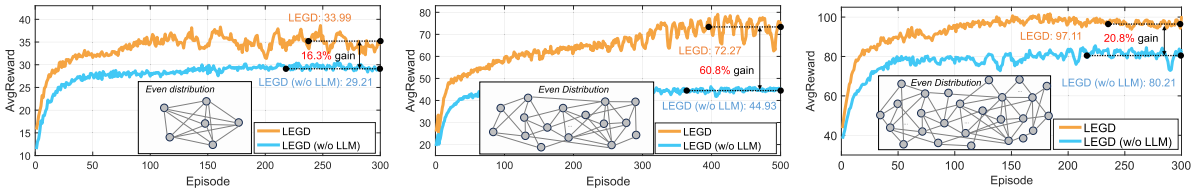
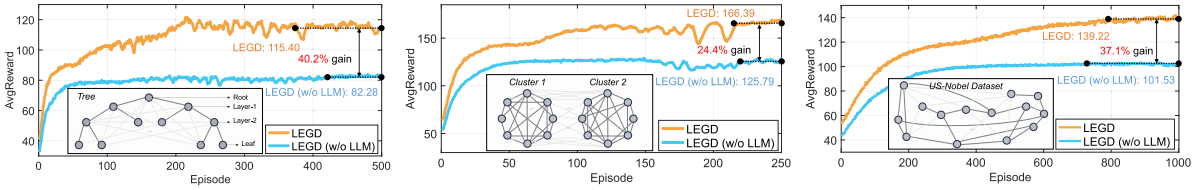


Fig. 8. The illustration of the micro-segmentation generation process. For each node in the generated micro-segmentations, the label [x, y] represents its [computing, transmission] resources. The units are GFLOPs and Watt, respectively.



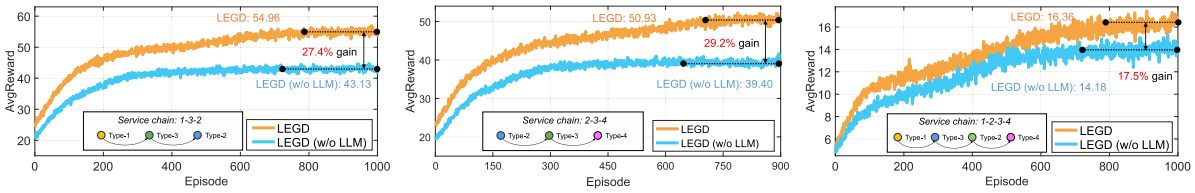
(a) The 6-node network. (b) The 16-node network. (c) The 26-node network.

Fig. 9. The impact of network scale on AvgReward.



(a) The tree-type node distribution. (b) The two-cluster node distribution. (c) The US-Nobel node distribution.

Fig. 10. The impact of network topology on AvgReward.



(a) {type-*2*, type-*3*, type-*1*}. (b) {type-*1*, type-*3*, type-*4*}. (c) {type-*1*, type-*2*, type-*3*, type-*4*}.

Fig. 11. The impact of required SFC types on AvgReward.

D. Ablation Study

To accomplish complicated micro-segmentation generation tasks, LEGD incorporates multiple innovations, including graph diffusion, graph transformer-based denoising network, and LLM enhancement. To demonstrate the effectiveness of each component, we perform the following ablation study. The experiments are conducted in the nine-node setting

TABLE I
THE ABLATION STUDY

Method	AvgReward
LEGD	*49.72*
LEGD (w/o LLM)	*35.87*
LEGD (w/o LLM & w/o diffusion)	*31.2*
LEGD (w/o LLM, w/o diffusion & w/o graph transformer)	*18.65*

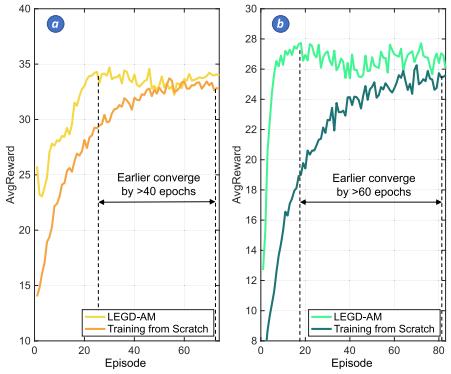


Fig. 12. LEGD-AM and LEGD training curves (i.e., training a model specific for the updated zero-trust NGN). Fig. (a) refers to trustworthiness update, in which a type-*2* node is removed from the current micro-segmentation; Fig. (b) refers to service upgrade, where the {type-*1*, type-*2*, type-*3*} service is required to evolve to {type-*1*, type-*2*, type-*3*, type-*4*}.

(described in Section VI-A). First, we remove the LLM enhancement from LEGD. Then, we further remove the graph diffusion structure. In this case, the algorithm becomes a graph transformer-aided policy gradient-based DRL. Finally, we remove the graph transformer and only use the MLPs for graphical states representation, resulting in a vanilla policy gradient-based DRL. We evaluate the performance of these four settings. TABLE I illustrates that the performance keeps decreasing as more components are removed. Hence, we can conclude that every component makes positive contributions to improving the generation capability of LEGD.

E. Efficiency of LEGD-AM

After the micro-segmentations are generated and operated, they should be maintained continuously to adapt to the dynamic zero-trust NGN. In this part, we validate the efficiency of the proposed LEGD-AM.

1) Fine-Tuning on LEGD: First, Fig. 12 illustrates the train curves of LEGD-AM, where Figs. 12 (a) and (b) correspond to trustworthiness updates and service upgrades, respectively. To prove the superiority of LEGD-AM, we compare them with training the corresponding LEGD models from scratch. We can observe that since the well-trained LEGD model for provisioning {type-*1*, type-*2*, type-*3*} services is leveraged as the checkpoint, LEGD-AM converges 40 and 60 epochs earlier for trustworthiness updates and service upgrades cases, respectively. Moreover, it maintains a similar or even better performance than training the model from scratch. Such experimental results demonstrate the capability of LEGD-AM to promptly adapt to the varying zero-trust NGNs and reduce the service outage time.

2) Micro-Segmentation Maintenance: Fig. 13(a) showcases the original and updated micro-segmentation once a type-*2* node is removed. We can observe that LEGD-AM involves another type-*2* node in the micro-segmentation and re-configures certain SFCs accordingly. The updated micro-segmentation holds high topological similarity to the original one and also maintains high utility. Similarly, Fig. 13(b) illustrates the micro-segmentation for supporting {type-*1*, type-*2*, type-*3*, type-*4*} services. Accordingly, LEGD-AM is trained to refine the micro-segmentation by extending SFCs while minimizing the costs. As shown in Fig. 13, LEGD-AM involves one type-*4* node to the original micro-segmentation, which supports the new service form while minimizing the costs.

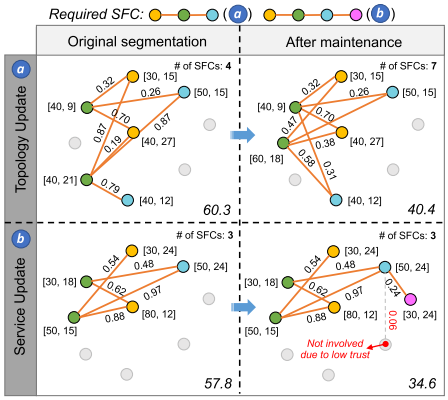


Fig. 13. Adaptive micro-segmentation maintenance. The cases of Figs. (a) and (b) are the same as those in Fig. 12.

F. Operation Performance

As shown in Figs. 14(a) and (b), in the beginning, the micro-segmentation generated by well-trained LEGD models can provision user services with high utility. When network trustworthiness or service states change, we only need to fine-tune the baseline LEGD model following the LEGD-AM paradigm. Compared with the conventional solutions that re-train the model from scratch, such a strategy can greatly reduce the time of service outage. In conclusion, the proposed framework and algorithms can effectively address two challenges in service provisioning for zero-trust NGN.

G. Discussions

1) Zero-Trust Security: This paper considers the most representative zero-trust mechanism, i.e., reputation [3]. Specifically, when a malicious node launches attacks (e.g., spoofing) or exhibits suspicious behavior, its reputation score is penalized accordingly, resulting in reduced trustworthiness. Once a node's trustworthiness falls below a predefined threshold, it is removed from the SFC. We emphasize that the specific reputation design is treated as a pluggable module. Additionally, the network security is directly affected by the reputation being adopted. For example, if the reputation values accurately fluctuate based on nodes' honest/malicious behaviors, attackers can be identified and removed promptly, ensuring robust network protection [3], [4]. Nevertheless, considering that various reputation designs have been developed, and their effectiveness in facilitating zero-trust security has been well studied in the literature, our experiments focus primarily on evaluating service provisioning efficiency.

2) Future Considerations: Introducing LLMs brings concerns about the computational complexity and inference latency. Specifically, at the beginning of denoising, LEGD calls the LLM to generate the filtering rule. Hence, the initialization complexity increases from $\mathcal{O}(m^2)$ to $\mathcal{O}(C_{\text{LLM}} + m^2)$.

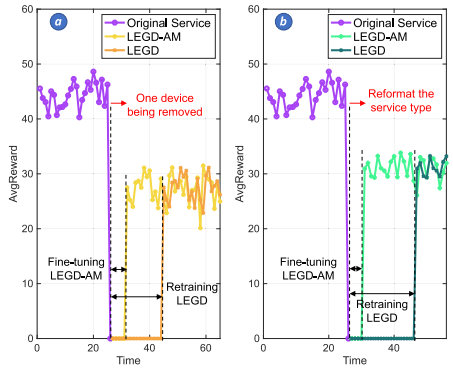


Fig. 14. The operational performance of our proposals. The cases of Figs. (a) and (b) are the same as those in Fig. 12.

Nonetheless, only the initialization stage calls the LLM, and the entire diffusion process utilizes the same filter. Hence, the computation complexity and latency brought by LLMs do not occupy the majority of the total training time. To further mitigate the effects brought by LLM, we can consider two promising directions. First, contributed to recent advancements, such as knowledge distillation and Mixture-of-Experts [46], LLMs can be lightweight and affordable for mobile devices. Second, in extremely resource-limited scenarios, we can cache filtering decisions from the most recent rounds and reapply them to future states with similar configurations, avoiding calling LLMs whenever possible.

VII. Conclusion

In this paper, we have presented a systematic method for the efficient provision of zero-trust services in NGN. Specifically, we have modeled zero-trust networks via hierarchical graphs, leveraging micro-segmentations for network organization and SFCs for service executions. Based on this framework, we have presented the LEGD algorithm, which leverages graph diffusion, policy optimization, and LLM enhancement to realize the utility-controlled micro-segmentation generation. Furthermore, we have proposed LEGD-AM, providing an adaptive way to perform task-oriented fine-tuning on LEGD to adapt to new environments/requirements. Extensive experimental results confirmed the effectiveness of our proposals.

References

[1] N. F. Syed, S. W. Shah, A. Shaghaghi, A. Anwar, Z. Baig, and R. Doss, "Zero trust architecture (ZTA): A comprehensive survey," *IEEE Access*, vol. 10, pp. 57143–57179, 2022.

[2] J. Guo, T. Sumi, Y. Kawashima, K. Parsons, Y. Nagai, and P. Orlik, "Minimizing route overlap for priority data delivery in next generation IoT networks," in *Proc. IEEE Global Commun. Conf.*, Dec. 2023, pp. 3664–3669.

[3] S. D. Okegbile, J. Cai, J. Chen, and C. Yi, "A reputation-enhanced shard-based Byzantine fault-tolerant scheme for secure data sharing in zero trust human digital twin systems," *IEEE Internet Things J.*, vol. 11, no. 12, pp. 22726–22741, Jun. 2024.

[4] Z. Tian, X. Gao, S. Su, J. Qiu, X. Du, and M. Guizani, "Evaluating reputation management schemes of Internet of Vehicles based on evolutionary game theory," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 5971–5980, Jun. 2019.

[5] W. Z. Khan, Q.-U.-A. Arshad, S. Hakak, M. K. Khan, and S.-Ur-Rehman, "Trust management in social Internet of Things: Architectures, recent advancements, and future challenges," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 7768–7788, May 2021.

[6] W. Jing, L. Peng, H. Fu, and A. Hu, "An authentication mechanism based on zero trust with radio frequency fingerprint for Internet of Things networks," *IEEE Internet Things J.*, vol. 11, no. 13, pp. 23683–23698, Jul. 2024.

[7] S. A. Khowaja et al., "Block encryption layer (BELA): Zero-trust defense against model inversion attacks for federated learning in 5G/6G systems," *IEEE Open J. Commun. Soc.*, vol. 6, pp. 807–819, 2025.

[8] Y. Liu et al., "A blockchain-based decentralized, fair and authenticated information sharing scheme in zero trust Internet-of-Things," *IEEE Trans. Comput.*, vol. 72, no. 2, pp. 501–512, Feb. 2023.

[9] (2024). *Microsoft Entra*. [Online]. Available: https://www.microsoft.com/en-us/security/business/microsoft-entra

[10] (2024). *IBM Maas360*. [Online]. Available: https://www.ibm.com/products/maas360

[11] (2024). *NIST Zero-Trust Standard*. [Online]. Available: https://www.nist.gov/publications/zero-trust-architecture

[12] Y. Li, Q. Zhang, H. Yao, R. Gao, X. Xin, and M. Guizani, "Next-gen service function chain deployment: Combining multi-objective optimization with AI large language models," *IEEE Netw.*, vol. 39, no. 3, pp. 20–28, May 2025.

[13] L. Bradatsch, O. Miroshkin, and F. Kargl, "ZTSFC: A service function chaining-enabled zero trust architecture," *IEEE Access*, vol. 11, pp. 125307–125327, 2023.

[14] N. Sheikh, M. Pawar, and V. Lawrence, "Zero trust using network micro segmentation," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2021, pp. 1–6.

[15] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, "Diffusion models in vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 9, pp. 10850–10869, Sep. 2023.

[16] Z. Li, Z. Yang, and M. Wang, "Reinforcement learning with human feedback: Learning dynamic choices via pessimism," in *Proc. ICML Workshop ILHF*, 2023, pp. 1–9.

[17] (2024). *Microsoft Zero-Trust Solution*. [Online]. Available: https://azure.microsoft.com/en-us/solutions/network-security

[18] (2024). *Illumio Micro-Segmentation*. [Online]. Available: https://www.illumio.com/

[19] M. Yousefi-Azar, M.-A. Kaafar, and A. Walker, "Unsupervised learning for security of enterprise networks by micro-segmentation," 2020, *arXiv:2003.11231*.

[20] M. Lee, G. Yu, H. Dai, and G. Y. Li, "Graph neural networks meet wireless communications: Motivation, applications, and future directions," *IEEE Wireless Commun.*, vol. 29, no. 5, pp. 12–19, Oct. 2022.

[21] C. Liu et al., "FERN: Leveraging graph attention networks for failure evaluation and robust network design," *IEEE/ACM Trans. Netw.*, vol. 32, no. 2, pp. 1003–1018, Apr. 2024.

[22] Y. Peng, J. Guo, and C. Yang, "Learning resource allocation policy: Vertex-GNN or edge-GNN?," *IEEE Trans. Mach. Learn. Commun. Netw.*, vol. 2, pp. 190–209, 2024.

[23] J. Hou, T. Tao, H. Lu, and A. Nayak, "An optimized GNN-based caching scheme for SDN-based information-centric networks," in *Proc. IEEE Global Commun. Conf.*, Dec. 2023, pp. 401–406.

[24] S. Zheng, Z. Ren, W. Cheng, and H. Zhang, "Performance analysis for subgraph isomorphism based embedding service function chains," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 6, pp. 3099–3113, Jun. 2023.

[25] F. Meng, M. Medo, and B. Buechel, "Whom to trust in a signed network? Optimal solution and two heuristic rules," *Inf. Sci.*, vol. 606, pp. 742–762, Aug. 2022.

[26] H. Du et al., "Attention-aware resource allocation and QoE analysis for metaverse xURLLC services," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 7, pp. 2158–2175, Jul. 2023.

[27] S. Edelkamp and S. Schrödl, "Constraint search," in *Heuristic Search*. Amsterdam, The Netherlands: Elsevier, 2012, ch. 13, pp. 571–631.

[28] C. Vignac, I. Krawczuk, A. Siraudin, B. Wang, V. Cevher, and P. Frossard, "DiGress: Discrete denoising diffusion for graph generation," in *Proc. ICLR*, 2023, pp. 1–22.

[29] Y. Fan et al., "DPOK: Reinforcement learning for fine-tuning text-to-image diffusion models," in *Proc. NeurIPS*, 2023, pp. 79858–79885.

[30] Y. Jiang, C. Li, W. Dai, J. Zou, and H. Xiong, "Variance reduced domain randomization for reinforcement learning with policy gradient," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 2, pp. 1031–1048, Feb. 2024.

[31] Y. Liu, C. Du, T. Pang, C. Li, M. Lin, and W. Chen, "Graph diffusion policy optimization," 2024, *arXiv:2402.16302*.

[32] V. P. Dwivedi and X. Bresson, "A generalization of transformer networks to graphs," in *Proc. AAAI Workshop DLG*, 2021, pp. 1–9.

[33] A. Vaswani et al., "Attention is all you need," in *Proc. NeurIPS*, vol. 30, 2025, pp. 5998–6008.

[34] E. Perez, F. Strub, H. D. Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *Proc. AAAI*, vol. 32, 2018, pp. 3942–3951.

[35] S. Palmer, *Vision Science: Photons to Phenomenology*. Cambridge, MA, USA: MIT Press, 1999.
[36] (2024). *ChatGPT-4*. [Online]. Available: https://openai.com/index/chatgpt/
[37] (2024). *LLaVA*. [Online]. Available: https://llava-vl.github.io/
[38] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan, "On the theory of policy gradient methods: Optimality, approximation, and distribution shift," *J. Mach. Learn. Res.*, vol. 22, no. 98, pp. 1–76, 2019.
[39] Z. Ankner, A. Renda, G. K. Dziugaite, J. Frankle, and J. Tian, "The effect of data dimensionality on neural network prunability," in *Proc. NeurIPS*, 2022, pp. 1–10.
[40] L. Xiao, "On the convergence rates of policy gradient methods," *J. Mach. Learn. Res.*, vol. 23, no. 1, pp. 12887–12922, 2022.
[41] Y. Cao et al., "Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 6, pp. 9737–9757, Jun. 2025.
[42] F. Che, Q. Z. Ahmed, F. A. Khan, and F. A. Khan, "Novel fine-tuned attribute weighted Naïve Bayes NLoS classifier for UWB positioning," *IEEE Commun. Lett.*, vol. 27, no. 4, pp. 1130–1134, Apr. 2023.
[43] J. Lv, L. Zhang, Y. Huang, J. Huang, and S. Chen, "Graph edit distance learning via different attention," 2023, *arXiv:2308.13871*.
[44] (2024). *Metis*. [Online]. Available: https://metis.readthedocs.io/en/latest/
[45] (2025). *Nobel-Us Network Structure*. [Online]. Available: https://sndlib.put.poznan.pl/home.action
[46] Z. Zhang et al., "MPMoE: Memory efficient MoE for pre-trained models with adaptive pipeline parallelism," *IEEE Trans. Parallel Distrib. Syst.*, vol. 35, no. 6, pp. 998–1011, Jun. 2024.

Yinqiu Liu (Member, IEEE) received the B.Eng. degree from Nanjing University of Posts and Telecommunications, China, in 2020, and the M.Sc. degree from the University of California, Los Angeles, USA, in 2022. He is currently pursuing the Ph.D. degree with the College of Computing and Data Science, Nanyang Technological University, Singapore. His current research interests include blockchain security, mobile AIGC, and generative AI.

Guangyuan Liu (Graduate Student Member, IEEE) received the B.Sc. degree from Nanyang Technological University, Singapore, in 2022, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Energy Research Institute @ NTU, under the Interdisciplinary Graduate Program. His research interests include GAI, agentic AI, and semantic communication.

Hongyang Du (Member, IEEE) received the B.Eng. degree from Beijing Jiaotong University, China, and the Ph.D. degree from Nanyang Technological University, Singapore. He is currently an Assistant Professor at the Department of Electrical and Electronic Engineering, The University of Hong Kong, where he directs the Network Intelligence and Computing Ecosystem (NICE) Laboratory. His research interests include edge intelligence, generative AI, and network management.

Dusit Niyato (Fellow, IEEE) received the B.Eng. degree from the King Mongkut's Institute of Technology Ladkrabang (KMITL), Thailand, and the Ph.D. degree in electrical and computer engineering from the University of Manitoba, Canada. He is currently a Professor with the College of Computing and Data Science, Nanyang Technological University, Singapore. His research interests include mobile generative AI, edge intelligence, decentralized machine learning, and incentive mechanism design.

Jiawen Kang (Senior Member, IEEE) received the Ph.D. degree from Guangdong University of Technology, China, in 2018. He was a Post-Doctoral Researcher at Nanyang Technological University, Singapore, from 2018 to 2021. He is currently a Full Professor at Guangdong University of Technology. His main research interests focus on blockchain, security, and privacy protection in wireless communications and networking.

Zehui Xiong (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from Nanyang Technological University (NTU), Singapore. He is currently a Full Professor with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, U.K. Prior to that, he was with Singapore University of Technology and Design, and NTU. His research interests include wireless networks, the Internet of Things, edge intelligence, semantic communications, and generative AI.

Dong In Kim (Life Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 1990. He was a Tenured Professor with the School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada. He is currently a Distinguished Professor with the College of Information and Communication Engineering, Sungkyunkwan University, Suwon, South Korea.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is currently an University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, the Internet of Things, AI for networks, and vehicular networks.