IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 6 July 2024, accepted 31 July 2024, date of publication 6 August 2024, date of current version 16 August 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3439262*

RESEARCH ARTICLE

# Privacy-Preserving Link Scheduling for Wireless Networks

**MARYAM ABBASALIZADEH, (Graduate Student Member, IEEE), JEFFREY CHAN, PRANATHI RAYAVARAM, YIMIN CHEN, (Member, IEEE), AND SASHANK NARAIN**

Miner School of Computer and Information Sciences, University of Massachusetts Lowell, Lowell, MA 01854, USA

Corresponding author: Sashank Narain (sashank_narain@uml.edu)

This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Grant N660012114033.

**ABSTRACT** Wireless communication is now a cornerstone of modern society, propelled by the widespread adoption of IoT devices and sophisticated wireless technologies. As wireless networks grow in complexity, there is an increasing need for efficient scheduling algorithms that can manage resources and adapt to evolving conditions effectively. Link scheduling is critical as it optimizes the use of wireless spectrum and bandwidth, prioritizing high-priority transmissions and minimizing interference. However, traditional link scheduling algorithms have primarily focused on performance, often overlooking the crucial aspect of privacy. This oversight poses significant risks in scenarios where privacy is paramount. To address this issue, we introduce PriLink, a novel link scheduling algorithm that prioritizes privacy without sacrificing performance. PriLink employs a least privilege model, sharing only essential links to protect critical topology details from potential adversaries. Our comprehensive evaluation shows that PriLink not only matches but occasionally surpasses the performance of established benchmarks like Greedy Maximal Scheduling and Local Greedy Scheduling. Moreover, it offers faster execution times and superior privacy protection. These results highlight PriLink's effectiveness as a robust solution for efficient, real-time, and privacy-preserving link scheduling in dynamic wireless networks.

**INDEX TERMS** Link scheduling, wireless networks, privacy, topology concealment.

## I. INTRODUCTION

In recent decades, wireless communication has become a fundamental and ubiquitous element of modern society. This growth has been driven by the increasing prevalence of Internet-of-Things (IoT) devices and the widespread adoption of next-generation wireless access points and base stations. Consequently, the size and complexity of these networks have significantly expanded.

Given their widespread use, optimizing wireless networks is now essential in today's technological landscape. Due to the increasing complexity and dynamic nature of these networks, there is a heightened demand for efficient scheduling algorithms that can effectively manage network resources and quickly adapt to changes [1], [2], [3]. Link scheduling is vital in this regard as it optimizes the use of wireless spectrum and bandwidth. It enhances network performance by prioritizing high-priority transmissions and minimizing interference, which is essential for maintaining optimal network operations under diverse loads and conditions. Effective link scheduling coordinates several critical factors, including link selection, transmission priority, and management of technical parameters such as power allocation and coding schemes [4], [5], [6]. This approach is indispensable in various applications such as Industrial and Agricultural IoT, smart grids, VANETs, and battlefield communications, where the efficiency and security of communications are paramount [7], [8], [9], [10].

Substantial research efforts have focused on developing efficient link scheduling algorithms for wireless networks. The optimal approach to link scheduling often involves solving the challenging NP-Hard Maximum Weighted Independent Set (MWIS) problem [11], [12]. However, the significant computational demands of MWIS solvers have

The associate editor coordinating the review of this manuscript and approving it for publication was Ding Xu.

led researchers to explore alternative approximate solutions. Early studies utilized greedy methods for link scheduling [5], [13], [14]. More recently, the focus has shifted to machine learning-based techniques, such as Graph Neural Networks (GNNs), Recurrent Neural Networks (RNNs), and Spatial Deep Learning [1], [15], [16]. Across these methodologies, the goal is to compute real-time link schedules in dynamic networks, enabling high-priority devices to transmit simultaneously without causing network interference.

Traditional efforts to improve scheduling performance have relied on a comprehensive understanding of the entire network topology [1], [2]. However, this extensive knowledge introduces significant privacy risks, necessitating protective measures for both network devices and individuals [17], [18]. In high-stakes scenarios, such as battlefield operations, where wireless communication is crucial for coordinating troops and performing maneuvers, the privacy of the network topology is paramount [10]. If adversaries were to access this information, they could easily identify critical network hotspots and disrupt communications through targeted attacks like jamming. Concealing the topology prevents adversaries from formulating such location-based strategies [19], [20]. Similarly, in civilian contexts such as Vehicle Ad-Hoc Networks (VANETs), where vehicles rely on networked communication for traffic management and safety, revealing the entire topology can lead to significant privacy breaches. Continuous exposure of topology across scheduling windows allows adversaries to track vehicles, deducing travel patterns, destinations, and frequent stops, thereby compromising individual privacy [21], [22]. By modifying these networks to conceal their topologies, we can effectively mitigate these risks, as adversaries will lack the necessary data to make privacy-sensitive inferences. To address such challenges, there is a pressing need for the development of privacy-focused link scheduling algorithms. These algorithms must manage real-time scheduling efficiently without exposing sensitive network topology information.

In response to the pressing need for privacy protection in wireless networks, we introduce PriLink, a novel link scheduling algorithm designed to prioritize the privacy of network topologies while ensuring high link scheduling performance and real-time execution. Unlike current greedy algorithms that rely on complete network topology and utilize graph-theoretic approaches, PriLink takes a unique stance by not requiring detailed topology information. Instead, it computes schedules directly from the links provided by individual devices. This is achieved through a ''least privilege'' model, where devices share only essential links with the scheduler, thus withholding critical topology details from potential adversaries. PriLink redefines link scheduling by eliminating the need for a global network view, which greedy algorithms in the literature rely on for optimal performance. This innovative approach enhances privacy and introduces a new paradigm that prioritizes data protection without sacrificing operational efficiency. This paper provides a detailed account of PriLink's design and implementation, complemented by practical simulations that demonstrate its performance. Our results show that PriLink's scheduling performance is comparable to that of established benchmarks such as Greedy Maximal Scheduling (GMS) [5], [23] and Local Greedy Scheduling (LGS) [13], [24]. Moreover, PriLink incorporates a privacy tolerance metric, allowing administrators to fine-tune the balance between performance and privacy. Lower tolerance settings improve topology concealment at a slight cost to scheduling performance, whereas higher settings maintain considerable privacy benefits while closely matching the scheduling performance of GMS and LGS. We believe that PriLink's unique combination of high scheduling efficiency, fast execution times, and strong privacy protection makes it a well-suited alternative to existing algorithms, meeting the increasing demands for both efficient link scheduling and robust privacy in wireless networks.

Our evaluation of the PriLink algorithm centered on its efficiency in link scheduling, execution times, and privacy across a range of simulated wireless network scenarios and densities. We used the Erdős-Rényi (ER) model [25] in our simulations to generate network topologies with controlled variations in size and density. The results indicate that PriLink is a viable alternative to current link scheduling algorithms, even in settings where privacy is not the primary concern. It achieves scheduling performance comparable to that of Greedy Maximal Scheduling (GMS) and Local Greedy Scheduling (LGS), coming within 0.05% of the highest performance benchmarks, while processing schedules nearly ten times faster than these benchmarks. When focusing on privacy, PriLink still maintains robust performance, achieving scheduling results within 3% of LGS in networks of 50 devices, and within 5.5% for larger networks containing 250 devices. Remarkably, it manages this performance while concealing up to 70% of the network topology in smaller networks and as much as 94.5% in larger networks. This capability demonstrates PriLink's effectiveness in balancing high scheduling efficiency with significant privacy protection, positioning it as a strong contender in the field of wireless network management.

Our contributions can be summarized as follows.

- We introduce PriLink, a novel link scheduling algorithm with built-in privacy protections. This represents the first implementation of a privacy-focused link scheduling algorithm tailored for dynamic wireless networks.
- We performed a comparative analysis of PriLink against benchmark algorithms, showing that it matches their scheduling performance and executes faster. In addition, it significantly enhances privacy by concealing a substantial portion of network links in various scenarios.

The structure of our paper is outlined as follows: Section II examines the issues of link scheduling and privacy in current algorithms. Section III reviews related research, setting the context for our study. In Section IV, we detail the design, implementation, time complexity, and privacy properties of our PriLink algorithm. Section V presents an evaluation of

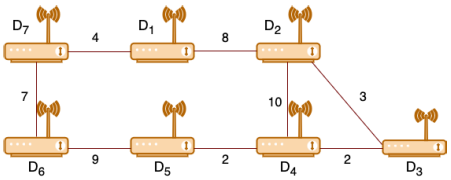



**FIGURE 1.** An illustration of a wireless network.

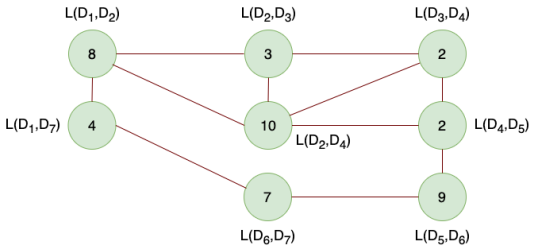



**FIGURE 2.** A conflict graph generated for the illustrated wireless network in Figure 1.

PriLink, focusing on its performance in link scheduling, execution time, and privacy benefits. We conclude our findings in Section VI.

## II. BACKGROUND

### A. LINK SCHEDULING FOR WIRELESS NETWORKS

Wireless link scheduling algorithms are designed to optimize the performance of wireless networks by efficiently allocating resources to maximize link utilization for packet transmission. By doing so, they effectively reduce interference, decrease the likelihood of packet collisions, and minimize the need for retransmissions. These algorithms are especially valuable in environments where collision-sensing protocols such as Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) are either not in use or require further optimization. The coordination between link scheduling and collision avoidance protocols significantly improves the utilization of network resources, enhancing overall network efficiency.

Link scheduling algorithms are utilized across various wireless topologies, such as point-to-point, point-to-multipoint, and multihop networks. The selection of a particular topology is influenced by the specific requirements and challenges of the network. Notably, multihop networks benefit significantly from link scheduling, which enhances data transmission efficiency and improves network reliability. Despite these specific advantages, link scheduling is universally valuable, offering crucial benefits to all types of networks where it is implemented. In this paper, our discussion focuses on link scheduling within wireless networks that use orthogonal frequency-division multiplexing (OFDM), a technology at the heart of modern broadband and mobile networks. This technology divides the communication channel into multiple orthogonal subchannels and time slots, optimizing the handling of multiple data streams simultaneously.

### B. ILLUSTRATING LGS LINK SCHEDULING ALGORITHM

This section provides an overview of the Local Greedy Scheduling (LGS) algorithm to illustrate how current greedy link scheduling algorithms operate. We chose the LGS algorithm because previous research [1], [13] has shown that it generates efficient link schedules that closely approximate the Maximum Weighted Independent Set (MWIS).

#### 1) CONFLICT GRAPH CONSTRUCTION

A conflict graph is a valuable tool in wireless networks for assessing interference between wireless devices. It simplifies the computation needed for scheduling links by allowing us to select a link for the schedule and then safely disregard all its neighboring links. This is crucial because activating any neighboring link would result in interference, compromising network performance.

To demonstrate a conflict graph construction, consider the wireless network example depicted in Figure 1. In this example, we represent the network as a graph labeled $G = (V, E)$. Here, the vertices $V = \{D_1, D_2, \ldots, D_7\}$ symbolize all the devices within the network. The edges $E = \{L(D_1, D_2, W_{1,2}), L(D_1, D_7, W_{1,7}), \ldots, L(D_6, D_7, W_{6,7})\}$ represent the links that connect these devices, where $W_{j,k}$ is the weight of the link between devices $D_j$ and $D_k$. These weights, which can be observed in Figure 1 with values like $W_{1,2} = 8$, $W_{1,7} = 4$, and $W_{6,7} = 7$, are crucial for scheduling as they dictate the link's priority. Weight values are typically determined based on the network's scheduling requirements. For instance, links with stronger signal strength might be assigned higher weights because they are more reliable and less prone to interference. Alternatively, in networks where Quality of Service (QoS) is a priority, links handling high-priority or QoS-compliant traffic may receive higher weights to ensure they are prioritized in the scheduling process.

From the original graph $G$, we can create a conflict graph $G_C = (V_C, E_C)$. In this conflict graph $G_C$, each vertex corresponds to an edge from $G$. For example, vertices $V_C^{1,2}$, $V_C^{1,7}$, and $V_C^{6,7}$ in $G_C$ represent the edges $L(D_1, D_2, W_{1,2})$, $L(D_1, D_7, W_{1,7})$, and $L(D_6, D_7, W_{6,7})$ in $G$, respectively. The edges $E_C$ in the conflict graph signify interference between these links. For instance, the vertex $V_C^{1,2}$ has interfering edges $I(V_C^{1,2}, V_C^{1,7})$, $I(V_C^{1,2}, V_C^{2,3})$, and $I(V_C^{1,2}, V_C^{2,4})$, where $I$ represents interference. Figure 2 illustrates this conflict graph for the wireless network shown in Figure 1. Essentially, if a link between devices $D_1$ and $D_2$ is active, other links such as $L(D_1, D_7, W_{1,7})$, $L(D_2, D_3, W_{2,3})$, and $L(D_2, D_4, W_{2,4})$ should not be activated to avoid interference and ensure effective link scheduling.

#### 2) LOCAL GREEDY SCHEDULING (LGS) OVERVIEW

The LGS algorithm utilizes the conflict graph $G_C = (V_C, E_C)$, which requires complete visibility of the entire network topology to construct. Initially, we define a set $r$ of remaining unvisited devices, starting with $r = V_C$

which includes all devices. The algorithm begins by sorting $r$ according to device weights. It then selects the device with the highest weight for inclusion in the schedule. For example, in our network, the algorithm first selects $V_C^{2,4} = L(D_2, D_4, W_{2,4})$ because it has the highest weight of 10. This device and its neighboring devices are marked as visited and added to the set $v$, which tracks visited devices. After the first iteration, $v$ includes $\{V_C^{2,4}, V_C^{1,2}, V_C^{2,3}, V_C^{3,4}, V_C^{4,5}\}$. In the subsequent iteration, the algorithm continues with the unvisited devices remaining in $r$. It next selects $V_C^{5,6}$, which has the second-highest weight of 9. Consequently, $v$ is updated to include $\{V_C^{2,4}, V_C^{1,2}, V_C^{2,3}, V_C^{3,4}, V_C^{4,5}, V_C^{5,6}, V_C^{6,7}\}$, and $r$ is reduced to $\{V_C^{1,7}\}$. The process repeats until all devices have been visited, culminating in the final schedule $[V_C^{2,4}, V_C^{5,6}, V_C^{1,7}]$ (i.e., $[L(D_2, D_4, 10), L(D_5, D_6, 9), L(D_1, D_7, 4)]$ with a total weight of 23.

The LGS algorithm's dependence on the conflict graph $G_C$ enforces full knowledge of the network topology to operate effectively. This requirement limits its use in environments where privacy concerns restrict the availability of complete network information.

## III. RELATED WORK

### A. EFFORTS ON WIRELESS LINK SCHEDULING

Considerable efforts have been invested in refining link scheduling algorithms for wireless networks. Initially, research primarily focused on using graph theory to approximate optimal link schedules while ensuring real-time operation. More recently, there has been a shift towards adopting machine learning techniques. These methods train models that often outperform traditional approaches in creating more efficient schedules. Generally, link scheduling strategies can be categorized into two groups:

#### 1) CENTRALIZED LINK SCHEDULING

This approach to link scheduling involves the use of a central server that acts as a coordinator. The server collects detailed information about all devices within the network, including the weights of their respective links. It then utilizes this information to calculate the link schedule, employing either traditional algorithmic methods or machine learning-based approaches [1], [12], [26]. For example, Leconte et al. [23] implemented the Greedy Maximal Scheduling (GMS) algorithm, which demonstrates high scheduling performance by approximating solutions close to the Maximum Weighted Independent Set (MWIS). This algorithm prioritizes local network graph data over global information, thereby achieving efficient scheduling solutions for wireless networks of varying sizes. In the realm of machine learning, Graph Neural Networks (GNNs) have gained attention for their effectiveness in centralized scheduling solutions. Zhao et al. [1] illustrate how GNNs, once adequately trained, can offer significant topological insights about the network, which in turn leads to more effective scheduling outcomes compared to those achieved with greedy methods.

#### 2) DISTRIBUTED LINK SCHEDULING

These schemes empower wireless devices to autonomously compute their own link schedules, thus eliminating the dependence on a central server. Typically, these distributed methods utilize iterative processes that involve local interactions among the vertices of the graph and their immediate neighbors [14], [27], [28]. These approaches can be algorithmic or based on machine learning techniques, similar to centralized methods. Several distributed greedy algorithms have been developed, such as Local Greedy Scheduling (LGS), Local Greedy Scheduling Enhancement (LGS-E), Local Greedy Scheduling with Two contention mini-slots (LGS-Two), and Greedy Coloring [13]. These methods aim to adapt the principles of the centralized Greedy Maximal Scheduling (GMS) algorithm for distributed environments, achieving scheduling outcomes that closely align with those of GMS. Recent advancements have also brought machine learning into distributed scheduling solutions. For instance, Zhao et al. have developed a distributed scheduler for the Maximum Weighted Independent Set (MWIS) problem using Graph Neural Networks (GNNs), which facilitates the learning of node topology to enhance graph information while maintaining generalizability and minimal complexity increase [29]. They have also introduced a delay-oriented distributed scheduler that utilizes Graph Convolutional Networks (GCNs) coupled with deep Q-learning, incorporating insights from network backlogs and previous scheduling decisions [30]. Additionally, Joo et al. have presented a distributed greedy approximation for MWIS scheduling in environments with fading channels, closely approximating the optimal max weight solutions [14]. Cui et al. have explored a Spatial Deep Learning approach for efficient link scheduling that leverages the geographic locations of transmitters and receivers to optimize network performance [16].

Our motivation stems from a desire to enhance real-time link scheduling in wireless networks, a goal shared by many in this field. Like previous algorithms, PriLink is designed to operate effectively in both centralized and decentralized settings, ensuring efficient and timely scheduling of network links. However, both centralized and decentralized approaches rely on having complete access to the network's topology. This common practice can lead to privacy concerns, as it involves extensive data exposure. PriLink sets itself apart by making privacy a priority and implementing a least privilege model. Furthermore, we have introduced a tolerance metric parameter that allows administrators to finely adjust the balance between link scheduling performance and the preservation of network topology privacy. This focus on privacy not only differentiates PriLink from conventional scheduling solutions but also addresses a crucial need in the management of modern wireless networks.

### B. EFFORTS ON PRIVACY IN WIRELESS NETWORKS

In this section, we explore current privacy research within the wireless networks domain. To the best of our knowledge,

no existing studies have specifically addressed privacy concerns in link scheduling. Thus, our discussion focuses on the general techniques used to protect privacy across other aspects of wireless networks.

Numerous studies have focused on enhancing privacy in wireless networks, introducing innovative defensive and offensive strategies [31], [32]. For example, Wang et al. proposed a probabilistic method to protect source location privacy in Wireless Sensor Networks (WSNs). Their approach modifies data packet transmission paths by introducing phantom nodes, fictitious sources, and weighted adjustments, which significantly reduces the chance of adversaries intercepting these transmissions [33]. Similarly, Koh et al. developed a privacy-focused routing algorithm designed to thwart route inference by global adversaries, whether they have lossless or lossy observations. They employ Bayesian maximum-a-posteriori estimation to lower the probability of detecting transmissions within WSNs [34]. Additionally, Chakraborty et al. introduced a technique to ensure temporal differential privacy in WSNs, aimed at preventing adversaries from learning about the timing of events at specific nodes. This is accomplished by strategically delaying traffic traces using differential privacy mechanisms [35].

Additionally, recent research has leveraged advanced Machine Learning techniques effectively to enhance privacy protection within wireless networks. For example, Mohamed et al. introduced a privacy model that employs federated learning over a wireless channel. This model includes user sampling and a novel wireless gradient aggregation scheme to enhance privacy [36]. Similarly, Liu et al. developed a gated recurrent unit neural network algorithm for accurately predicting traffic flow while preserving privacy. Their method integrates a federated averaging algorithm with a joint-announcement protocol in the aggregation process, effectively balancing privacy protection with prediction accuracy and reducing communication overhead [37]. Furthermore, Kim et al. investigated adversarial attacks against deep learning models used for modulation classification in wireless channels. They also developed a certified defense mechanism to counteract the effects of these attacks, enhancing the robustness of wireless communications [38].

Our work stands apart from the studies mentioned above, both in motivation and methodology. First, our research is centered on link scheduling, whereas prior studies have primarily addressed different aspects of wireless networks such as safeguarding communication paths, protecting the timing of events, and securing processes within federated learning frameworks. Second, our approach involves disclosing only essential information tailored to application needs by configuring a tolerance metric value for schedule computation. Techniques such as differential privacy, network obfuscation, and noise addition, while effective in some contexts, may not provide adequate solutions for link scheduling. Differential privacy, designed to protect individual data points within a dataset, may not be directly applicable to concealing a wireless network's topology because it generally relies on a large, static dataset for effective noise application–an unlikely scenario in dynamic networks where topology and parameters frequently change [1], [2]. Additionally, network obfuscation and noise addition can introduce overhead and degrade performance, which is particularly problematic in wireless networks where efficiency and low latency are crucial [19], [39]. Moreover, sophisticated adversaries may be able to reverse engineer obfuscated data or filter out noise, diminishing the effectiveness of these methods in critical privacy-sensitive scenarios [40], [41], [42]. Instead, we enhance privacy by limiting the exposure of information to potential adversaries, as detailed in Section IV.

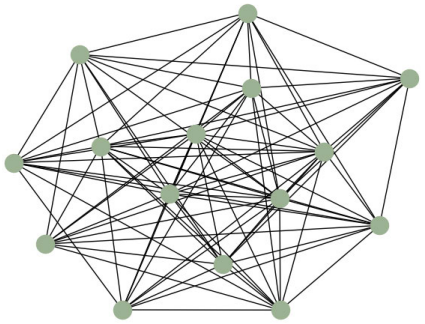


**FIGURE 3.** An illustrative dense network comprising 15 nodes and edges between devices able to communicate.

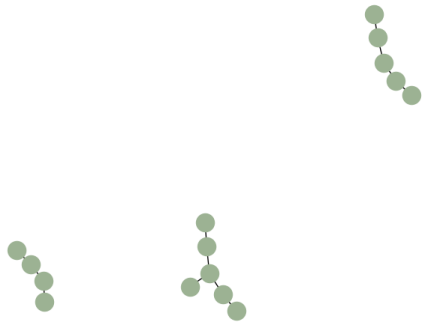


**FIGURE 4.** The network observed by an attacker when PriLink is employed at a tolerance value of 1, showing only 12% of the total links and concealing the rest.

## IV. PRILINK: ALGORITHM DESIGN

### A. SYSTEM MODEL

The PriLink algorithm offers a versatile alternative to current greedy link scheduling methods, tailored for wireless networks that need to safeguard their topology privacy. It is particularly effective in dynamic networks where the topology changes frequently, requiring adaptive real-time link scheduling. While PriLink is optimized for such dynamic environments, it can also perform well in static settings. However, in static networks, it is important to be aware that privacy protections may erode over time. This vulnerability arises because an adversary who can consistently monitor a static network over time may be able to deduce the network topology, a capability that is significantly diminished in dynamic networks where the topology changes frequently.

PriLink also provides administrators the flexibility to adjust a privacy tolerance metric for balancing link scheduling performance with the privacy of the network topology. At a tolerance value of 1, the algorithm provides maximum privacy protection. Low tolerance settings enhance privacy, which is vital in sensitive environments, though it may slightly reduce scheduling efficiency. Conversely, increasing the tolerance improves scheduling performance but at the expense of reduced privacy. This streamlined approach simplifies network management, allowing for quick adjustments without the need for complex strategies or technologies.

Figures 3 and 4 visually illustrate the privacy benefits of the above model. Figure 3 shows an original dense network with 15 nodes, where each device is connected to all its neighboring devices, creating a highly interconnected topology. In contrast, Figure 4 demonstrates the network as perceived by the adversary when PriLink is employed at a tolerance value of 1, concealing approximately 88% of the network links from the adversary in this graph. This drastic reduction in visible links underscores PriLink's tolerance effectiveness in preserving privacy by limiting the information available to the adversary. In dynamic networks, where the topology is continuously evolving, this makes it exceedingly challenging for an adversary to accurately identify the network topology, thereby significantly enhancing overall network privacy. These privacy protection capabilities are central to the evaluations discussed in Section V, showing how PriLink balances performance and privacy based on tolerance settings.

### B. ADVERSARY MODEL

We define our adversary as an entity capable of accessing the information shared by wireless devices for schedule computation. This adversary could be a passive observer, either within or outside the network, who can receive certain messages from devices to deduce the network's topology. If the adversary is within the network, they will have access to the control messages of their neighbors if they are tuned into the appropriate channel, gaining detailed knowledge about local network topology. However, they will remain unaware of the control messages shared in the rest of the network, severely limiting their ability to deduce the complete topology. If the adversary is outside the network, they will not have access to the control messages, thus completely restricting their ability to discover the topology. Alternatively, the adversary might take on an active role, such as a scheduler responsible for computing the network schedule. While we assume that the scheduler performs its task of scheduling accurately, its role provides it with access to comprehensive network data. This 'honest but curious' behavior scenario is concerning because the scheduler, while not deviating from its assigned tasks, may still analyze or use the sensitive data it accesses for purposes other than intended.

With access to network topology information, the adversary can carry out various privacy breaches, as previously discussed in Section I. For example, in the battlefield scenario mentioned earlier, an adversary would need the entire network topology to strategically position themselves for maximum impact through methods such as jamming. It is important to note that simply knowing the highest-weighted links does not aid the adversary significantly, as multiple links within the network may have the same weight regardless of their position in critical zones or on the network's periphery. Therefore, the ability of PriLink to conceal the network topology is crucial, as it prevents the adversary from gaining the comprehensive knowledge necessary to execute a strategic attack. In the context of Vehicle Ad-Hoc Networks (VANETs), an adversary with access to regular topology updates could effectively track vehicles and individuals over time. This detailed tracking would enable the adversary to deduce travel patterns, destinations, and frequent stops, posing significant privacy risks. However, PriLink is designed to counteract this threat by concealing regular topology updates. Its algorithm ensures that even if a vehicle consistently has high weight in the network, and thus transmits more often, the necessary links to infer its exact location remain concealed. By doing so, PriLink makes it difficult for adversaries to track targeted vehicles and individuals consistently, significantly enhancing privacy and security within VANETs.

Understanding the adversary model is crucial for appreciating the effectiveness of PriLink's privacy mechanisms, which we explore next in Section IV-C. By considering the capabilities and limitations of the adversary, we can better understand how PriLink's design addresses these threats, thereby validating the practical relevance of our privacy and scheduling performance results.

### C. THE PRILINK ALGORITHM

Our development of PriLink was inspired by a key observation: conventional link scheduling algorithms focus on identifying the highest-weighted links within a network, often overlooking other less prominent links. This focus is logical since the goal is to activate links that maximize the network's schedule from all possible combinations of link schedules. Building on this, we considered an alternative strategy where devices only reveal their highest-weighted link for the scheduling process, keeping all other links private. We hypothesized that this selective disclosure could still enable effective scheduling, though it might initially result in lower performance compared to current greedy algorithms like Local Greedy Scheduling (LGS). To enhance this approach, we proposed allowing devices to share additional links as needed, ordered by their importance. This strategy aims to balance privacy with scheduling efficiency. The culmination of these ideas led to the development of the PriLink algorithm, detailed in Algorithm 1. PriLink's design inherently improves communication performance by optimizing link scheduling, thereby minimizing channel congestion and interference without requiring explicit optimization of channel selection or the communication process. By prioritizing the most critical links, PriLink ensures that

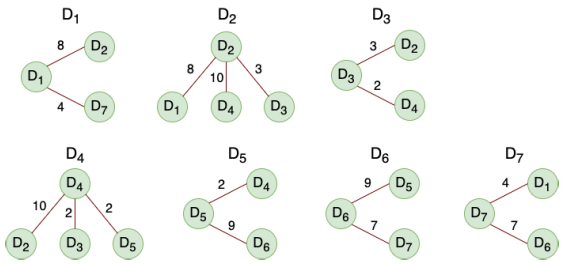



**FIGURE 5. Breaking down the wireless network shown in Figure 1 into individual graphs for each device. Each device maintains its respective link weight information.**

the communication channels are used efficiently, leading to improved overall network performance. This efficiency is achieved through algorithmic enhancements rather than changes to the channel selection mechanism. The subsequent sections will provide further insights into how PriLink operates and its advantages over existing methods.

### 1) NETWORK SETUP

In PriLink, we implement a least privilege methodology to model the network setup. This approach leads to a decentralized network configuration where each device maintains its own graph. This graph includes the device itself and its immediate neighbors as vertices, with edges representing the connections to these neighbors. Devices can readily construct this graph based on their knowledge of local communication patterns. Due to its design, PriLink can be effectively deployed in both centralized and decentralized configurations. In a centralized configuration, devices transmit their links to a central scheduler. Conversely, in a decentralized setup, each device broadcasts its links across the network. Regardless of the configuration, the fundamental steps of the algorithm remain consistent, ensuring reliable operation across different network setups.

The concept of decentralization in our network setup can be visually understood through the example illustrated in Figure 1, which displays a wireless network with devices labeled $D_1$, $D_2$, …, $D_7$. The decentralization process for each device is detailed in Figure 5. For example, Device $D_1$ maintains its own unique graph denoted as $G_{D_1} = (V_{D_1}, E_{D_1})$. Here, the vertices $V_{D_1} = \{D_1, D_2, D_7\}$ represent Device $D_1$ and its immediate neighbors. The edges $E_{D_1} = \{L(D_1, D_2, W_{1,2}), L(D_1, D_7, W_{1,7})\}$ specifically connect Device $D_1$ to these neighbors. Similarly, Device $D_3$ has its graph $G_{D_3} = (V_{D_3}, E_{D_3})$, with vertices $V_{D_3} = \{D_2, D_3, D_4\}$ and edges $E_{D_3} = \{L(D_2, D_3, W_{2,3}), L(D_3, D_4, W_{3,4})\}$ linking it to its neighbors. This decentralization is uniformly applied across all devices, including $D_2$, $D_4$, $D_5$, $D_6$, and $D_7$. Such a setup allows each device to manage its privacy effectively by disclosing only the links necessary for the computation of the schedule while keeping all other connections private.

### 2) ALGORITHM OVERVIEW

The PriLink algorithm, as detailed in Algorithm 1, requires two primary inputs: a set of *devices* denoted as $\{D_1, D_2, \ldots, D_N\}$ and a tolerance metric value denoted as $\tau$. The set of devices provides the scheduler with the necessary information to anticipate potential links from each device, facilitating the planning of link schedules. The tolerance value, $\tau$, specifies the maximum number of links each device is expected to contribute, which helps in managing the complexity and breadth of the network data processed during scheduling (line 1). For example, if the tolerance $\tau$ is set to 1, each device in the network will transmit only its single highest-weighted link. If $\tau$ is increased to 5, each device will then share its top five highest-weighted links. The algorithm produces a single output, *schedule*, which comprises all the links chosen by the algorithm for transmission during the current scheduling window (line 2).

**Algorithm 1** PriLink Link Scheduling Algorithm

1: **Inputs:** $devices \leftarrow \{D_1, \ldots, D_N\}$, $tolerance\,(\tau) \in \mathbb{R}$
2: **Output:** $schedule \leftarrow []$
3: $links \leftarrow \emptyset$
4: $visited \leftarrow \emptyset$
5: **for all** $D_i \in devices$ **do**
6: $\quad receive\, msg(m = sort(E_{D_i}))[0 : \tau])$
7: $\quad links \leftarrow links \cup \{m\}$
8: **end for**
9: $sorted\_links \leftarrow sort\_by\_weight(links)$
10: **for all** $(D_i, D_j, W_{i,j}) \in sorted\_links$ **do**
11: $\quad$ **if** $!(D_i \in visited \,||\, D_j \in visited)$ **then**
12: $\quad\quad schedule \leftarrow schedule + [(D_i, D_j, W_{i,j})]$
13: $\quad\quad visited \leftarrow visited \cup \{D_i, D_j\}$
14: $\quad$ **end if**
15: **end for**

*The algorithm initiates by creating two sets: links* to store the links received from the devices, and *visited* to keep track of all devices that have been processed (lines 3 and 4). The process of receiving links occurs in lines 5-8. Each device $D_i$ sorts its edges $E_{D_i}$ by link weight in descending order and then transmits the top $\tau$ links to the scheduler (line 6). These links are then added to the *links* set for the next phase of scheduling (line 7). To ensure there are no duplicates in *links*, the links from each device, such as $L(D_i, D_j, W_{i,j})$, are transmitted in a manner that ensures $i < j$ to maintain order and prevent redundancies efficiently.

After gathering all links for a specific contention window, the algorithm proceeds to sort these links by their weight, storing the result in a list named *sorted_links* (line 9). This sorting is crucial as it ensures that links with higher weights are prioritized for scheduling over those with lower weights, following a common practice in scheduling algorithms. In the subsequent step, the algorithm processes each link from the *sorted_links* list individually (line 10). If either of the devices involved in a link is already marked in the *visited* set, indicating prior scheduling, that link is skipped (line 11). If neither device has been visited, the link is added to the *schedule* list (line 12), and both devices are marked as visited by adding them to the *visited* set (line 13). This

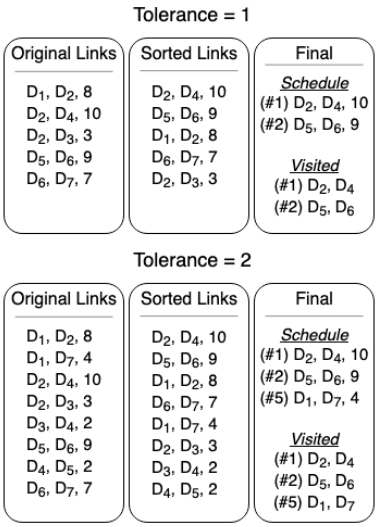



**FIGURE 6.** Visual representation of steps in the PriLink algorithm applied to the wireless network example in Figure 1.

prevents any further scheduling of links that would conflict with the previously scheduled ones. After iterating through all the sorted links (lines 10-15), the final link schedule is compiled in the *schedule* list. This finalized schedule is then disseminated, ensuring that all devices are aware of their scheduled transmissions.

### 3) ALGORITHM DEMONSTRATION ($\tau = 1$)

Here, we demonstrate the operation of the PriLink algorithm using device-based graphs as depicted in Figure 5, following the steps outlined in Algorithm 1. Initially, we set the tolerance parameter $\tau$ to 1 to illustrate the algorithm's behavior. Subsequently, we will increase the tolerance $\tau$ to 2 and examine the tolerance metric's impact on the scheduling performance and privacy. A visual representation of the algorithm's steps and outputs for these two tolerance settings is provided in Figure 6.

Setting the tolerance value $\tau$ to 1 instructs network devices to disclose only their highest-weighted link. Accordingly, in lines 5-8 of Algorithm 1, each device submits just one link to the scheduler. For example, as shown in the device graphs from Figure 5, Device $D_2$ transmits $L(D_2, D_4, 10)$, which is the highest-weighted link in its graph $G_{D_2}$. Likewise, Device $D_3$ sends $L(D_2, D_3, 3)$, representing the highest-weighted link in its graph $G_{D_3}$. It is important to note that Device $D_4$ transmits $L(D_2, D_4, 10)$ as well, the same link shared by $D_2$. Since this link is a duplicate, the algorithm discards it to avoid redundancy. These links are then stored and processed in subsequent steps and are visually represented in the 'Original Links' section of Figure 6.

After gathering links from the devices within the specified contention window, the PriLink algorithm sorts these links as outlined in line 9 of Algorithm 1. The sorted links are visually displayed in the 'Sorted Links' section of Figure 6. This sorting operation organizes the links in descending order of their weights, positioning the highest-weighted links at the top of the list. This prioritization is crucial as it allows the algorithm to focus on selecting the most significant links first in the subsequent scheduling step. In our example, with a tolerance value of 1, the link $L(D_2, D_4, 10)$ appears at the top, being the highest-weighted link, followed by $L(D_5, D_6, 9)$ and others in descending order.

After sorting the links, the PriLink algorithm iterates through each sorted link, as detailed in lines 10-15, to compute the final schedule. In this instance, with a tolerance value of one and five sorted links, the algorithm executes five iterations. Initially, no devices have been visited, so the condition in line 11 is true. The algorithm adds the highest-weighted link, $L(D_2, D_4, 10)$, to the schedule list (line 12), resulting in $schedule \leftarrow [L(D_2, D_4, 10)]$. It then marks $D_2$ and $D_4$ as visited (line 13), updating the visited set to $visited \leftarrow \{D_2, D_4\}$. The next highest link, $L(D_5, D_6, 9)$, is processed. Since neither $D_5$ nor $D_6$ has been visited, the link is added to the schedule in line 12, updating the schedule to $schedule \leftarrow [L(D_2, D_4, 10), L(D_5, D_6, 9)]$. The visited set is expanded to include these devices, becoming $visited \leftarrow \{D_2, D_4, D_5, D_6\}$. The algorithm then processes the links $L(D_1, D_2, 8)$, $L(D_6, D_7, 7)$, and $L(D_2, D_3, 3)$. Each of these iterations fails to add the link to the schedule because at least one involved device in each link is already in the visited set. The condition at line 11 is not satisfied for these links, so they are skipped, resulting in a final schedule of $[L(D_2, D_4, 10), L(D_5, D_6, 9)]$, as no further unvisited links are available to be added.

### 4) ALGORITHM DEMONSTRATION ($\tau = 2$)

The schedule computed with a tolerance $\tau = 1$ using PriLink does not achieve an optimal solution when compared to the results from the Local Greedy Scheduling (LGS) algorithm detailed in Section II-B. Specifically, the link $L(D_1, D_7, 4)$, included in the LGS schedule, was omitted in the PriLink run. The omission of this link in the schedule with $\tau = 1$ can be attributed to the link prioritization by the devices, as evident in the original links section of Figure 6. Both device $D_1$ and device $D_7$ did not expose the link $L(D_1, D_7, 4)$ because it was their second-highest weighted link, and thus was not shared with the scheduler under the $\tau = 1$ setting.

Executing PriLink with a tolerance value of $\tau = 2$ allows for the inclusion of previously omitted links in the schedule. Similar to other greedy algorithms, PriLink prioritizes links from devices with smaller index values when weights are equal. For example, device $D_4$ has potential second links such as $(D_3, D_4, 2)$ and $(D_4, D_5, 2)$. Given the choice, $D_4$ selects $(D_3, D_4, 2)$ because device $D_3$ has a smaller index than device $D_5$. By setting $\tau$ to 2, additional links are disclosed, notably $L(D_1, D_7, 4)$, $L(D_3, D_4, 2)$, and $L(D_4, D_5, 2)$, enhancing the schedule's completeness. These additions are visually documented in the 'Original Links' section of Figure 6.

After sorting the links in line 9 with setting $\tau = 2$, the PriLink algorithm executes eight iterations according to lines 10-15. The first four links remain the same as with $\tau = 1$, so the schedule and the set of visited devices

remain unchanged after the fourth iteration: *schedule* $\leftarrow [L(D_2, D_4, 10), L(D_5, D_6, 9)]$ and *visited* $\leftarrow \{D_2, D_4, D_5, D_6\}$. During the fifth iteration, the algorithm processes $L(D_1, D_7, 4)$. Since both $D_1$ and $D_7$ have not yet been visited, this link is added to the schedule, updating it to *schedule* $\leftarrow [L(D_2, D_4, 10), L(D_5, D_6, 9), L(D_1, D_7, 4)]$. Both $D_1$ and $D_7$ are then marked as visited, updating the visited set to *visited* $\leftarrow \{D_1, D_2, D_4, D_5, D_6, D_7\}$. The next three links, $L(D_2, D_3, 3)$, $L(D_3, D_4, 2)$, and $L(D_4, D_5, 2)$, are not added to the schedule because at least one involved device in each is already in the visited set. This results in a final schedule of *schedule* $\leftarrow [L(D_2, D_4, 10), L(D_5, D_6, 9), L(D_1, D_7, 4)]$, mirroring the outcome achieved by the Local Greedy Scheduling (LGS) for this example network.

In the demonstration using a tolerance value of $\tau = 2$, it is clear that the PriLink algorithm achieves the same scheduling outcome as the Local Greedy Scheduling (LGS) for our example network. In the subsequent sections, we will explore our comprehensive evaluation methodology and results, which include simulations of PriLink across various sizes and densities of wireless networks. We will also compare PriLink's performance against high-performance benchmark algorithms such as Greedy Maximal Scheduling (GMS) and Local Greedy Scheduling (LGS) to highlight its effectiveness and efficiency.

#### 5) TIME COMPLEXITY

The time complexity of the PriLink algorithm can be analyzed through its three main operational steps as outlined in the algorithm's description. Each step contributes distinctly to the overall computational load:

1) *Sorting Edges on Each Device (Lines 5-8)*: In this step, each device sorts its edges. If the number of edges each device handles is represented as $\hat{E}$, the sorting operation has a time complexity of $O(\hat{E} \log \hat{E})$. This complexity is specific to the edge count of each individual device, as the edges are processed locally within the device.
2) *Sorting Transmitted Links on the Scheduler (Line 9)*: In this step, the scheduler sorts all transmitted links. Each of the $N$ devices contributes up to $\tau$ links, leading to a total of $N \times \tau$ links. Therefore, the sorting operation for these links requires a time complexity of $O((N \times \tau) \log(N \times \tau))$, accounting for every link transmitted by each device.
3) *Computing the Schedule from the Sorted Links (Lines 10-15)*: In this step, the algorithm iterates over all the links. Throughout this iteration, it performs several constant-time operations, such as set operations, which each have a time complexity of $O(1)$. Consequently, the overall time complexity for this step is linear, amounting to $O(N \times \tau)$, directly correlating with every link transmitted by each device.

In scenarios common to large wireless networks, the number of edges $\hat{E}$ per device is usually much smaller than the total number of transmitted links $N \times \tau$. As a result, the sorting of links at the scheduler emerges as the most critical factor in the complexity analysis. Therefore, the worst-case time complexity of the PriLink algorithm is predominantly governed by this sorting step, which has a complexity of $O((N \times \tau) \log(N \times \tau))$.

#### 6) PRIVACY BENEFITS

In the PriLink algorithm, the privacy tolerance parameter $\tau$ imposes a limit on the maximum number of links that each device in the network, totaling $N$ devices, is allowed to transmit to the scheduler. This restriction is applied in lines 5-8 of the algorithm's implementation.

##### *a: MAXIMUM LINKS DISCLOSED*

The expression $N \times \tau$ sets the upper limit on the total number of network links that can be disclosed throughout the network.

$$max\_disclosed = N \times \tau$$

##### *b: EXPECTED LINKS COUNT*

The total number of distinct pairs of vertices in a graph with $N$ vertices is given by the binomial coefficient $\binom{N}{2}$. This coefficient counts all potential edges in the network, and the formula for this calculation is $\binom{N}{2} = \frac{N \times (N-1)}{2}$. According to the Erdős-Rényi (ER) model [25], which is detailed in Section V-A, each of these $\binom{N}{2}$ possible edges independently exists with probability $p$. Thus, the expected number of links in the network graph is calculated by multiplying the total possible number of edges by the probability $p$ that any given edge is included:

$$links\_count = p \times \binom{N}{2} = p \times \frac{N \times (N-1)}{2}$$

##### *c: PRIVACY COST*

We define privacy cost as the degree to which sensitive links in the network topology are disclosed by wireless devices. This disclosure is quantified by comparing the maximum number of links that can be disclosed, denoted as *max_disclosed*, with the expected total number of links, denoted as *links_count*. The formula to calculate the privacy cost is given by:

$$privacy\_cost = \frac{max\_disclosed}{links\_count} = \frac{N \times \tau}{p \times \binom{N}{2}}$$

which simplifies to:

$$privacy\_cost = \frac{2 \times \tau}{p \times (N-1)}$$

##### *d: HIGHEST PRIVACY CONDITION*

The minimum privacy cost, indicating maximum privacy, is achieved when $\tau = 1$. Under this condition, the privacy cost formula further simplifies to $\frac{2}{p \times (N-1)}$. This reduced value represents the highest privacy condition achievable by the PriLink algorithm, as it minimizes the extent of sensitive network topology information disclosed by wireless devices.

**TABLE 1. Simulation parameters employed for evaluation.**

| | |
|---|---|
| Methodology | Random topologies using ER model |
| Graph sizes | $[10, 50, 100, 150, 200, 250]$ |
| Probabilities ($p$) | $[0.2, 0.5, 0.8]$ |
| Number of runs | 250 runs per graph size and probability |

#### e: EXAMPLE ILLUSTRATIONS

To exemplify the privacy benefits afforded by the PriLink algorithm, we present a couple of scenarios demonstrating the extent of network topology disclosure. Consider a network with $N = 250$ devices. Under the highest privacy setting ($\tau = 1$), and assuming a densely connected graph modeled by the ER model with a connection probability of $p = 0.8$, the privacy cost calculates to 0.01. This indicates that only 1% of the total network links are disclosed, highlighting significant privacy benefits. If we increase the tolerance to $\tau = 10$, the privacy cost rises to 0.1, yet this still conceals about 90% of the network topology. These calculations are further substantiated in Section V-C4, where simulations in large dense networks show that disclosure can be as minimal as 1%, thus effectively safeguarding the network topology from the adversary with the use of PriLink.

## V. PERFORMANCE EVALUATION

### A. WIRELESS NETWORKS SIMULATION

#### 1) SIMULATION METHODOLOGY

To simulate wireless networks for our evaluations, we employed the Erdős-Rényi (ER) model, as described in [25] and it has been used in previous studies to analyze link scheduling performance [1], [29]. In this model, we define the number of nodes as $N$ and the probability of connecting any two nodes as $p$. The ER model constructs a network graph by randomly establishing connections between nodes, where each edge's presence is independent of other edges. This feature allows us to effectively manipulate the network size ($N$) and the probability of connections ($p$), creating a variety of network scenarios that mimic real-world conditions where link availability and network topologies can vary significantly. By adjusting these parameters, we can explore a broad spectrum of network densities and complexities, providing a robust framework for assessing the performance of our link scheduling algorithms under different conditions.

#### 2) SIMULATION PARAMETERS

For our simulations, we utilized parameters from Table 1 to evaluate our algorithm across a range of graph sizes, from 10 to 250 devices, including intermediary sizes. We employed the ER model with probabilities of 0.2, 0.5, and 0.8. Selecting a probability of 0.2 allows the ER model to generate sparsely connected networks by connecting a limited number of devices, which is ideal for evaluating the algorithms on networks with fewer links. A probability of 0.5 provides a medium connectivity scenario, offering a balanced environment that is neither too sparse nor too dense, which helps in assessing the algorithm's performance under moderate connectivity conditions. Conversely, a probability of 0.8 results in more connections between devices, suitable for assessing performance in denser network environments. Our choice of these specific probabilities was strategic, intended to showcase how PriLink's link scheduling performance, execution time, and privacy protection vary between sparse, moderately connected, and dense network conditions. For each graph size and probability combination (e.g., (10, 0.2), (100, 0.8), (250, 0.2)), we generated 250 unique graphs to ensure a thorough analysis of our algorithms and benchmarks across a spectrum of network sizes and densities.

#### 3) BENCHMARK ALGORITHMS

We selected benchmark algorithms for comparison based on their robust scheduling performance and operational efficiency. Algorithms that either underperformed in link scheduling or exhibited longer execution times were excluded. For example, Color Greedy Scheduling [13] and Distributed Greedy Scheduling [11] were not considered due to their lower scheduling performance compared to Local Greedy Scheduling (LGS). Furthermore, machine learning-based methods were excluded because their extensive training periods make them unsuitable for real-time execution in dynamic wireless networks. We also decided against implementing algorithms based on the Maximum Weighted Independent Set (MWIS) [12], [26] due to their NP-Hard complexity and the practical challenges they present in experimental settings. Ultimately, we chose the Greedy Maximal Scheduling (GMS) [5], [23] and Local Greedy Scheduling (LGS) algorithms for their near-optimal link scheduling performance and manageable worst-case complexities of $O(|V|)$ and $O(\log|V|)$, respectively, making them well-suited for real-world applications.

### B. EVALUATION METRICS

To assess the performance of the PriLink algorithm compared to high-performance greedy benchmark algorithms, we utilize the following crucial metrics:

#### 1) LINK SCHEDULING PERFORMANCE

This metric assesses the efficiency of an algorithm in selecting links for scheduling. We quantify it as a percentage by comparing the total weight of links chosen by the algorithm under evaluation ($W_a$) to the total weight of links selected by the best-performing benchmark ($W$), Local Greedy Scheduling (LGS), calculated as $\frac{W_a}{W} \times 100\%$. Evaluating scheduling performance is essential as it provides insights into how well an algorithm optimizes link selection to manage network resources effectively. We denote the high scheduling variant of the PriLink algorithm as "PriLink (Highest Schedule)." In this variant, we adjust the tolerance value to the maximum integer, exposing all network links. This adjustment allows us to benchmark PriLink against other high-performing algorithms in terms of link scheduling performance when the privacy features of the algorithm are not utilized.

### 2) ALGORITHM EXECUTION TIME

This metric measures the computational efficiency of each algorithm by timing how long it takes to compute the link schedule, with results recorded in seconds. Given the potential for significant variations in execution times across different algorithms, we use a logarithmic scale to present these times in our analysis for clearer comparison. It is important to recognize that although the worst-case time complexities for Greedy Maximal Scheduling (GMS) and Local Greedy Scheduling (LGS) are linear and logarithmic respectively, both algorithms involve a sorting process to prioritize the highest-weighted links, which can extend their execution times. Consequently, our evaluation focuses on the actual execution times of these algorithms in simulated wireless environments, providing a practical assessment rather than relying solely on theoretical time complexities. The implementations and setup of LGS and GMS used in this study have been derived from Zhao et al. [1].

### 3) PRIVACY COST

Privacy cost quantifies the extent of sensitive network topology information disclosed by wireless devices. It is calculated by taking the ratio of unique links disclosed ($L_d$) to the total number of links ($L$) in the network, expressed as $\frac{L_d}{L} \times 100\%$. A lower percentage signifies better privacy protection. This metric is important as it directly indicates the algorithm's effectiveness in safeguarding network topology from adversaries. In dynamic networks, where the topology changes frequently, an attacker would only be able to infer the topology based on the disclosed links indicated by this metric. A lower privacy cost makes it much harder for the attacker to gain a comprehensive understanding of the network. In our simulations, all benchmark algorithms and PriLink (Highest Schedule) reveal a privacy cost of 100%, as they access the complete network topology. For the highest privacy assessment, we designate this variation of the PriLink algorithm as "PriLink (Highest Privacy)," setting the tolerance value to 1. This minimal disclosure ensures at least one link per device is shared, enabling the algorithm to generate a fair schedule that meets the scheduling needs of all network devices.

## C. EVALUATION RESULTS

In this section, we discuss the results of our simulation evaluation, conducted according to the methodology detailed in Section V-A and using the metrics specified in Section V-B.

### 1) LINK SCHEDULING PERFORMANCE

Figure 7 presents an analysis of the link scheduling performance of the PriLink algorithm in sparse networks ($p = 0.2$), comparing it with Local Greedy Scheduling (LGS) and Greedy Maximal Scheduling (GMS). This comparison is essential as prior research has primarily focused on improving scheduling effectiveness. For clarity, the weights in this analysis have been normalized between 0 and 100, with 100 representing the average weight of the LGS algorithm

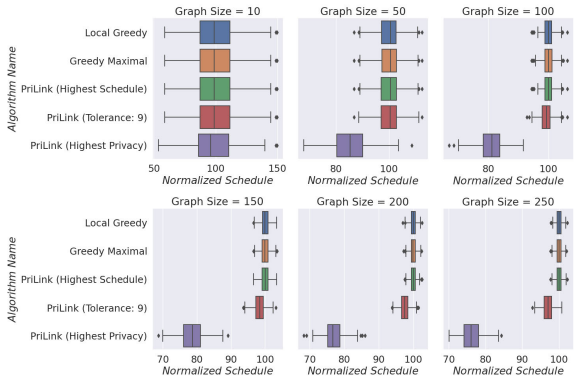



**FIGURE 7. Performance evaluation of scheduling comparing PriLink with benchmarks for sparse networks.**

for each graph size. The figure shows that the performance of LGS (blue) and GMS (orange) are nearly identical. Similarly, PriLink (Highest Schedule) (green) demonstrates comparable performance, staying within 0.05% of these benchmark algorithms, suggesting that PriLink is an effective alternative to current algorithms, even when network topology privacy is not a primary concern. However, PriLink (Highest Privacy) (purple) shows a decline in performance as the network size increases. For instance, at a graph size of 10, the algorithm achieves about 98% of LGS's performance. This figure drops to approximately 85% at a graph size of 50 and further decreases to around 76% for a graph size of 250. Despite this, in high-privacy scenarios, this setting can conceal nearly 99% of the network topology for larger graphs, as detailed in Section V-C4. Yet, if balancing performance with privacy is the goal, PriLink (Highest Privacy) might not be ideal due to its reduced scheduling effectiveness. To address this, adjusting the PriLink tolerance metric can significantly enhance performance. For example, setting the tolerance value to 9 (red) results in notable performance improvements, achieving approximately 99% of LGS's scheduling performance for a graph size of 50, and about 97% for a graph size of 250. This improvement comes with a slight increase in privacy costs, which we will explore in detail in the following sections.

Figure 8 introduces the analysis for moderately dense networks ($p = 0.5$), providing additional insights into PriLink's performance across different network densities. In these networks, PriLink (Highest Schedule) once again shows performance levels close to LGS and GMS, confirming its robustness across varying network densities. Specifically, PriLink (Highest Privacy) performs at about 80% of LGS's performance for a graph size of 50 and about 75% for a graph size of 250. This performance, while lower than in sparse networks, is higher than in dense networks, indicating a trend where network density inversely affects performance when high privacy is prioritized. By adjusting the tolerance value to 9, PriLink's performance improves significantly, achieving approximately 98% of LGS's performance for a graph size of 50 and about 96% for a graph size of 250.

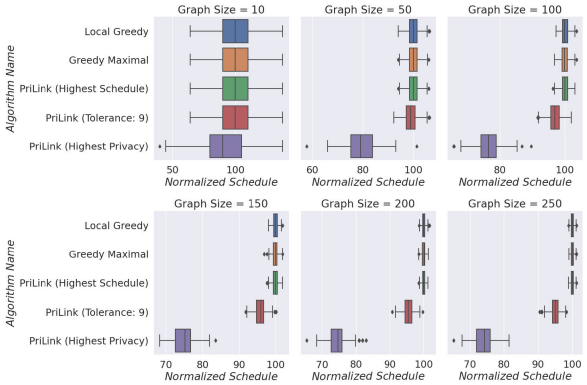



**FIGURE 8.** Performance evaluation of scheduling comparing PriLink with benchmarks for moderate networks.

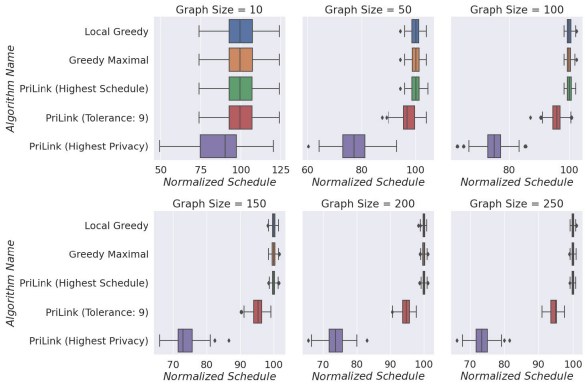
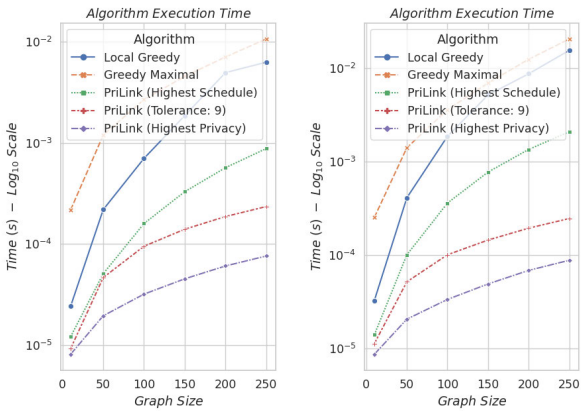
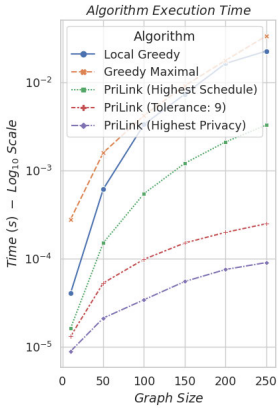



**FIGURE 9.** Performance evaluation of scheduling comparing PriLink with benchmarks for dense networks.




(a) Sparse networks (P=0.2). (b) Moderate networks (P=0.5).




(c) Dense networks (P=0.8).

**FIGURE 10.** Algorithmic execution times of link scheduling comparing PriLink with benchmark algorithms.

Figure 9 shifts our focus to dense networks ($p = 0.8$) and explores PriLink's performance in comparison to LGS and GMS. The results show that PriLink (Highest Schedule) continues to perform as efficiently as GMS and LGS. However, in dense networks with a graph size of 50, PriLink (Highest Privacy) sees a performance drop, achieving only 77% of LGS's performance, compared to 85% in sparse networks. For a graph size of 250, the performance further declines from 76% in sparse networks to about 73.5% in dense networks. This reduction highlights the challenges of maintaining high performance in denser settings. Conversely, increasing the tolerance value to 9 shows a less pronounced decrease in performance. In dense networks, it achieves approximately 97% of LGS's performance for a graph size of 50, and about 94.5% for a graph size of 250. These results suggest that PriLink, with a higher tolerance setting, remains effective in environments where a balance between scheduling performance and privacy is necessary.

In addition to the core performance metrics, it is important to consider the error margins, deviations, and outliers observed in these analyses. Across different values of $p$, error margins are represented by the shaded areas in the Figure 7 to 9, indicating variability in performance measurements. These margins are generally narrow, suggesting consistent algorithm performance. Although deviations and outliers exist, they are more pronounced in larger graph sizes and higher privacy settings. For instance, in the case of PriLink (Highest Privacy), variability tends to increase with graph size, reflecting challenges in maintaining consistent performance under strict privacy constraints. However, these variations do not significantly impact the overall performance of PriLink. The narrow error margins indicate that the algorithm performs consistently well across different scenarios, and observed deviations are within acceptable limits for practical applications. Even in higher privacy settings, where variability is slightly more pronounced, PriLink continues to deliver robust performance. The predictable nature of these deviations allows for informed adjustments to tolerance settings, ensuring high performance while balancing privacy requirements. Consequently, the impact of these error margins, deviations, and outliers is manageable and does not undermine the effectiveness of the PriLink algorithm.

### 2) ALGORITHM EXECUTION TIME

Figures 10a to 10c present the execution times (in $\log_{10}$ scale) for the PriLink algorithm relative to benchmark algorithms in sparse, moderately dense, and dense network conditions.

These times were recorded during the same experiments that evaluated scheduling performance. The benchmarks were conducted using a Dell Latitude desktop computer featuring 16 CPUs, 16 GB RAM, and an Intel Core i7 processor. From the analysis, we observe that all PriLink variants—Highest Schedule, Highest Privacy, and tolerance set to 9—consistently outperform both the Local Greedy Scheduling (LGS) and Greedy Maximal Scheduling (GMS) algorithms. The reason PriLink is faster is that it performs sorting on a much smaller quantity of links compared to LGS and GMS. For instance, for a graph size $N = 250$, $p = 0.8$, and tolerance $\tau = 9$, LGS and GMS need to sort about $p \times \binom{N}{2} = p \times \frac{N\times(N-1)}{2} = 0.8 \times \frac{250\times249}{2} = 24900$ links. Meanwhile, PriLink needs to sort only $N \times \tau = 250 \times 9 = 2250$ links, which is significantly more efficient to sort.

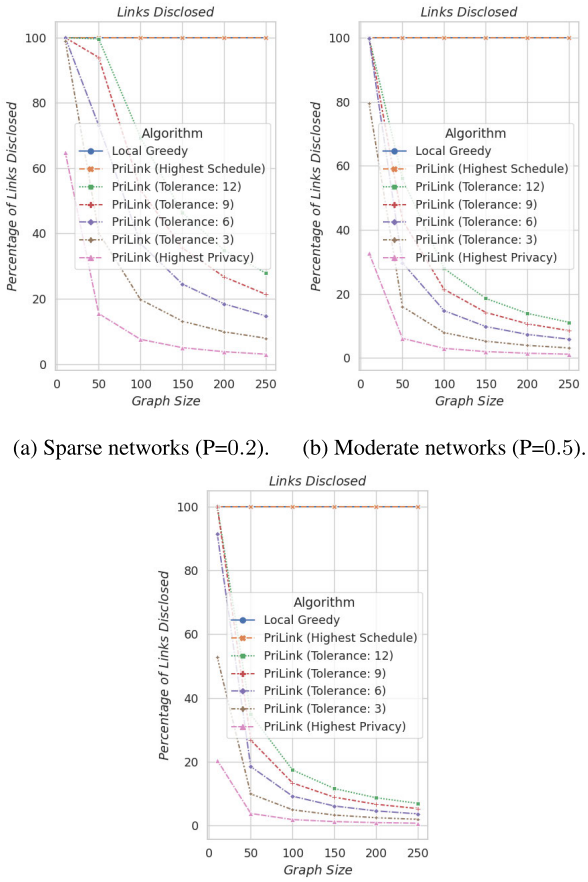
In dense networks ($p = 0.8$), GMS, while the slowest, remains effective for real-time applications with execution times averaging 0.0006 seconds for graph sizes of 10, increasing to 0.06 seconds for graph sizes of 250. LGS shows better efficiency, with times ranging from $9.9 \times 10^{-5}$ seconds for graph sizes of 10 up to 0.06 seconds for graph sizes of 250. PriLink significantly outperforms these benchmarks. For a graph size of 250, PriLink (Highest Schedule) averages only 0.007 seconds—nearly an order of magnitude faster. PriLink (Highest Privacy) and PriLink with a tolerance of 9 are even faster, at 0.00025 seconds and 0.00055 seconds, respectively. In moderately dense networks ($p = 0.5$), PriLink continues to demonstrate superior performance. For a graph size of 250, PriLink (Highest Schedule) records an average execution time of 0.0035 seconds, markedly faster than both LGS and GMS. PriLink (Highest Privacy) and PriLink with a tolerance of 9 exhibit even quicker times, averaging 0.00021 seconds and 0.00052 seconds, respectively. In sparse networks ($p = 0.2$), PriLink's performance is even more impressive. For a graph size of 250, PriLink (Highest Schedule) averages 0.0018 seconds, PriLink (Highest Privacy) averages 0.00018 seconds, and PriLink with a tolerance of 9 averages 0.00048 seconds. These fast execution times, combined with the high scheduling performance discussed in Section V-C1, further validate PriLink's effectiveness and efficiency. This makes PriLink an optimal choice for environments where both link scheduling and privacy are critical.

### 3) PRIVACY COSTS

This evaluation assesses the impact of varying PriLink tolerance metric values on privacy costs across different graph sizes and densities. A lower privacy cost indicates significantly enhanced privacy, allowing devices to withhold more network topology information from potential adversaries.

In Figure 11a, we evaluate the average privacy costs for PriLink under various tolerance settings in sparse wireless networks, characterized by a connectivity probability of $p = 0.2$. We analyzed the LGS algorithm and six different configurations to explore their impact on privacy costs: PriLink (Highest Privacy), and tolerance values of 3, 6, 9, and 12, along with PriLink (Highest Schedule). This range enables a comprehensive examination of the trade-offs between privacy and link scheduling effectiveness. LGS requires full knowledge of the network topology, resulting in a privacy cost of 100% across all densities. This dependence on complete topology information highlights the advantages of PriLink in scenarios where privacy is a significant concern. The findings indicate that with the Highest Privacy setting, only about 18% of links are disclosed on average for a network size of 50. This level of privacy improves as the network size increases to 250, with less than 4% of links being disclosed. Notably, even with a tolerance setting of 9—which offers scheduling performance comparable to that of GMS and LGS—the privacy benefits remain substantial. For instance, at a network size of 250, PriLink reveals only about 21% of the links, thereby concealing a significant portion of the network topology.

Figure 11b examines the privacy costs for PriLink in moderately dense wireless networks with a connection probability value of $p = 0.5$. The results reveal a balanced privacy-performance trade-off in these networks. For the Highest Privacy setting, approximately 35% of links are disclosed for a graph size of 10. This percentage decreases




(a) Sparse networks (P=0.2). (b) Moderate networks (P=0.5).

(c) Dense networks (P=0.8).

**FIGURE 11. Comparing the influence of privacy tolerance values on privacy costs in the PriLink algorithm.**

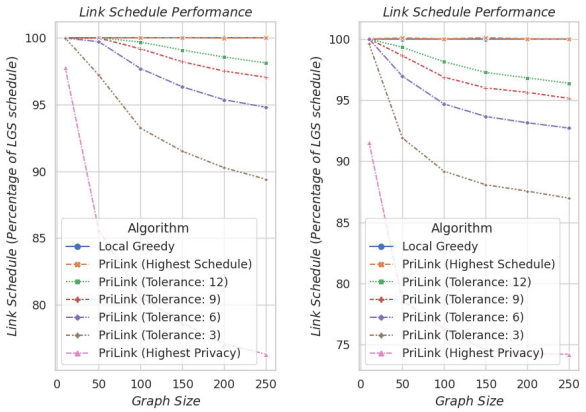
significantly as the network size increases, with around 4% of links disclosed for a graph size of 250. Even with a tolerance setting of 9, PriLink still provides substantial privacy protection. For example, for a graph size of 250, only about 13% of the links are disclosed, indicating a significant portion of the network topology is kept private. These results show that PriLink can effectively balance privacy and performance in moderately dense networks, making it a versatile solution for varying network conditions.

Figure 11c presents the privacy costs for PriLink in dense wireless networks with a connection probability of $p = 0.8$. Remarkably, privacy costs in dense networks are lower than in sparse networks, highlighting PriLink's substantial privacy benefits in such environments. For instance, under the Highest Privacy setting, privacy costs for a graph size of 10 are reduced from 65% in sparse networks to around 20% in dense networks. This improvement is even more significant in larger networks; privacy costs decrease from 4% in sparse networks to below 1% for a graph size of 250 in dense networks. Additionally, with a tolerance setting of 9, only about 5.5% of the links are disclosed for a graph size of 250, ensuring substantial concealment of the network topology. These findings demonstrate PriLink's effectiveness in enhancing privacy protection, making it a well-suited option for securing the topology of wireless networks in privacy-sensitive settings.

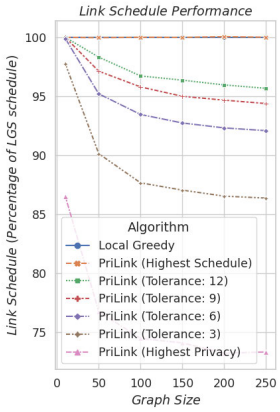
#### 4) PERFORMANCE IMPACT OF TOLERANCE VALUES

This evaluation examines the impact of various tolerance values in PriLink on link scheduling performance across different graph sizes and network densities. The objective is to assess the influence of these tolerance settings and to justify the choice of a tolerance value of 9 for comparison against benchmark algorithms, as discussed in Section V-C1. These analyses draw on the same simulation experiments previously used to evaluate privacy costs. We consider six distinct tolerance configurations for PriLink: the Highest Privacy setting, tolerance values of 3, 6, 9, and 12, and the Highest Schedule setting. Performance is quantified relative to the highest-performing benchmark, Local Greedy Scheduling, to provide a clear basis for comparing the effectiveness of each scheduling approach.

In Figures 12a to 12c, we explore how various tolerance settings affect scheduling performance in sparse ($p = 0.2$), moderately dense ($p = 0.5$), and dense networks ($p = 0.8$). The graphs demonstrate that scheduling performance generally improves as the tolerance values increase. This enhancement is due to the algorithm having access to a broader array of links, allowing for more optimized scheduling decisions. For sparse networks ($p = 0.2$), the performance leaps between the lower tolerance settings (such as Highest Privacy and 3) are significant—scheduling performance jumps by nearly 11% for a graph size of 50 and approximately 14% for a graph size of 250. The performance of LGS in these scenarios is used as a benchmark, demonstrating that PriLink with higher tolerance values (e.g., 9 and 12)




(a) Sparse networks (P=0.2). (b) Moderate networks (P=0.5).




(c) Dense networks (P=0.8).

**FIGURE 12. Comparing the influence of privacy tolerance values on link schedule performance in the PriLink algorithm.**

can achieve nearly equivalent performance. For moderately dense networks ($p = 0.5$), the improvement between the Highest Privacy setting and a tolerance value of 3 is about 9% for a graph size of 50 and approximately 12% for a graph size of 250. The LGS performance in these networks shows that PriLink, with a tolerance setting of 9, can achieve approximately 95% of LGS's performance for a graph size of 250, highlighting its effectiveness in balancing privacy and performance. Dense networks ($p = 0.8$) exhibit a similar pattern, with an improvement of about 10% for a graph size of 50 and approximately 13% for a graph size of 250. The performance of LGS in dense networks further underscores that PriLink with higher tolerance settings can closely match LGS performance. However, with higher tolerance values of 6, 9, and 12, the increments in scheduling performance become less marked, showing only about a 3% improvement for sparse networks and nearly 4% for dense networks for a graph size of 250. The difference in scheduling performance between tolerances 9 and 12 is marginal, less than 1%. These results indicate that performance plateaus beyond a tolerance value of 9, approaching a high scheduling performance.

Given these observations, we chose tolerance value 9 for detailed evaluations in Section V-C1 and V-C2 due to its

optimal balance between high scheduling performance and acceptable privacy costs. It is important to mention that the optimal privacy tolerance value will vary depending on specific network scenarios and privacy needs, which falls outside the scope of this study.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we introduce PriLink, a novel link scheduling algorithm designed specifically for dynamic wireless networks. PriLink enhances privacy by limiting the sharing of link details to only what is essential, effectively concealing a substantial portion of the network topology. Through comprehensive network simulations and comparisons with benchmark algorithms, we demonstrate PriLink's strong scheduling performance, rapid execution times, and robust privacy protection. These attributes confirm PriLink's suitability for real-world applications in environments that prioritize scheduling efficiency. Notably, PriLink can conceal nearly 80% to 95% of a wireless network's topology, making it highly reliable for privacy-sensitive applications. Looking forward, we plan to investigate the integration of Deep Learning techniques, such as Federated Learning, into link scheduling to further enhance PriLink's performance while adhering to its privacy-preserving principles. Our goal is to develop models that not only improve scheduling efficacy but also continue to protect network privacy.

## REFERENCES

[1] Z. Zhao, G. Verma, C. Rao, A. Swami, and S. Segarra, "Link scheduling using graph neural networks," *IEEE Trans. Wireless Commun.*, vol. 22, no. 6, pp. 3997–4012, Jun. 2022.

[2] S. Zhang, B. Yin, W. Zhang, and Y. Cheng, "Topology aware deep learning for wireless network optimization," *IEEE Trans. Wireless Commun.*, vol. 21, no. 11, pp. 9791–9805, Nov. 2022.

[3] L. Zhang, H. Yin, S. Roy, L. Cao, X. Gao, and V. Sathya, "IEEE 802.11be network throughput optimization with multi-link operation and AP coordination," 2023, *arXiv:2312.00345*.

[4] A. G. Marques, N. Gatsis, G. B. Giannakis, N. Zorba, C. Skianis, and C. Verikoukis, "Optimal cross-layer design of wireless fading multihop networks," in *Proc. Cross Layer Designs WLAN Syst.*, 2011, pp. 1–44.

[5] C. Joo, X. Lin, and N. B. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1132–1145, Aug. 2009.

[6] J. Tang, G. Xue, C. Chandler, and W. Zhang, "Link scheduling with power control for throughput enhancement in multihop wireless networks," *IEEE Trans. Veh. Technol.*, vol. 55, no. 3, pp. 733–742, May 2006.

[7] K. A. Abuhasel and M. A. Khan, "A secure industrial Internet of Things (IIoT) framework for resource management in smart manufacturing," *IEEE Access*, vol. 8, pp. 117354–117364, 2020.

[8] S. Jamali and M. Abbasalizadeh, "Cost-aware co-locating of services in Internet of Things by using multicriteria decision making," *Int. J. Commun. Syst.*, vol. 34, no. 16, Nov. 2021, Art. no. e4962.

[9] P. Kumar, Y. Lin, G. Bai, A. Paverd, J. S. Dong, and A. Martin, "Smart grid metering networks: A survey on security, privacy and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2886–2927, 3rd Quart., 2019.

[10] S. K. Sarkar, T. G. Basavaraju, and C. Puttamadappa, *Ad Hoc Mobile Wireless Networks: Principles, Protocols and Applications*. Boca Raton, FL, USA: Auerbach, 2007.

[11] P. Du and Y. Zhang, "A new distributed approximation algorithm for the maximum weight independent set problem," *Math. Problems Eng.*, vol. 2016, no. 1, pp. 1–10, Sep. 2016.

[12] S. Lamm, C. Schulz, D. Strash, R. Williger, and H. Zhang, "Exactly solving the maximum weight independent set problem on large real-world graphs," in *Proc. 21st Workshop Algorithm Eng. Exp. (ALENEX)*, Jan. 2019, pp. 144–158.

[13] C. Joo and N. B. Shroff, "Local greedy approximation for scheduling in multihop wireless networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 3, pp. 414–426, Mar. 2012.

[14] C. Joo, X. Lin, J. Ryu, and N. B. Shroff, "Distributed greedy approximation to maximum weighted independent set for scheduling with fading channels," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1476–1488, Jun. 2016.

[15] S. Zhang, W. Shen, M. Zhang, X. Cao, and Y. Cheng, "Experience-driven wireless D2D network link scheduling: A deep learning approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[16] W. Cui, K. Shen, and W. Yu, "Spatial deep learning for wireless scheduling," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1248–1261, Jun. 2019.

[17] Y. Tian, Z. Zhang, J. Xiong, L. Chen, J. Ma, and C. Peng, "Achieving graph clustering privacy preservation based on structure entropy in social IoT," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2761–2777, Feb. 2022.

[18] E. Rizk and A. H. Sayed, "A graph federated architecture with privacy preserving learning," in *Proc. IEEE 22nd Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Sep. 2021, pp. 131–135.

[19] N. Suri, M. Tortonesi, J. Michaelis, P. Budulas, G. Benincasa, S. Russell, C. Stefanelli, and R. Winkler, "Analyzing the applicability of Internet of Things to the battlefield environment," in *Proc. Int. Conf. Mil. Commun. Inf. Syst. (ICMCIS)*, 2016, pp. 1–8.

[20] N. K. Jadav, R. Gupta, and S. Tanwar, "Garlic routing-based privacy preserving framework for secure data exchange between IoMVs with 5G," in *Proc. Int. Conf. Netw., Multimedia Inf. Technol. (NMITCON)*, Sep. 2023, pp. 1–6.

[21] S. Tangade, S. S. Manvi, and P. Lorenz, "Decentralized and scalable privacy-preserving authentication scheme in VANETs," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8647–8655, Sep. 2018.

[22] T. Zhang and Q. Zhu, "Distributed privacy-preserving collaborative intrusion detection systems for VANETs," *IEEE Trans. Signal Inf. Process. over Netw.*, vol. 4, no. 1, pp. 148–161, Mar. 2018.

[23] M. Leconte, J. Ni, and R. Srikant, "Improved bounds on the throughput efficiency of greedy maximal scheduling in wireless networks," in *Proc. 10th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2009, pp. 165–174.

[24] C. Joo, "A local greedy scheduling scheme with provable performance guarantee," in *Proc. 9th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, May 2008, pp. 111–120.

[25] P. Erdős and A. Rényi, "On the strength of connectedness of a random graph," *Acta Mathematica Academiae Scientiarum Hungaricae*, vol. 12, nos. 1–2, pp. 261–267, Mar. 1964.

[26] K. Verhetsela, J. Pellerina, A. Johnena, and J.-F. Remaclea, "Solving the maximum weight independent set problem: Application to indirect hexahedral mesh generation," in *Proc. 26th Int. Meshing Roundtable, Res. Notes*, Barcelona, Spain, 2017, pp. 1–5.

[27] A. Douik, H. Dahrouj, T. Y. Al-Naffouri, and M.-S. Alouini, "Distributed hybrid scheduling in multi-cloud networks using conflict graphs," *IEEE Trans. Commun.*, vol. 66, no. 1, pp. 209–224, Jan. 2018.

[28] T. Ameen ur Rahman, M. S. Hassan, and M. H. Ismail, "A queue-length based approach to metropolized Hamiltonians for distributed scheduling in wireless networks," in *Proc. Wireless Telecommun. Symp. (WTS)*, Apr. 2020, pp. 1–6.

[29] Z. Zhao, G. Verma, C. Rao, A. Swami, and S. Segarra, "Distributed scheduling using graph neural networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 4720–4724, Jul. 2021.

[30] Z. Zhao, G. Verma, A. Swami, and S. Segarra, "Delay-oriented distributed scheduling using graph neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2022, pp. 8902–8906.

[31] S. Wang, Z. Chen, Y. Xu, Q. Yan, C. Xu, and X. Wang, "On user selective eavesdropping attacks in MU-MIMO: CSI forgery and countermeasure," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2019, pp. 1963–1971.

[32] X. Yang, Y. Feng, W. Fang, J. Shao, X. Tang, S.-T. Xia, and R. Lu, "An accuracy-lossless perturbation method for defending privacy attacks in federated learning," in *Proc. ACM Web Conf.*, 2022, pp. 732–742.

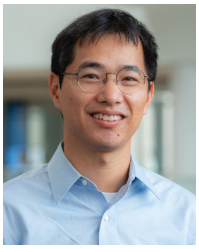
[33] H. Wang, G. Han, W. Zhang, M. Guizani, and S. Chan, "A probabilistic source location privacy protection scheme in wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 5917–5927, Jun. 2019.

[34] J. Y. Koh, D. Leong, G. W. Peters, I. Nevat, and W.-C. Wong, "Optimal privacy-preserving probabilistic routing for wireless networks," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 9, pp. 2105–2114, Sep. 2017.
[35] B. Chakraborty, S. Verma, and K. P. Singh, "Temporal differential privacy in wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 155, Apr. 2020, Art. no. 102548.
[36] M. S. E. Mohamed, W.-T. Chang, and R. Tandon, "Privacy amplification for federated learning via user sampling and wireless aggregation," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3821–3835, Dec. 2021.
[37] Y. Liu, J. J. Q. Yu, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A federated learning approach," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7751–7763, Aug. 2020.
[38] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek, and S. Ulukus, "Channel-aware adversarial attacks against deep learning-based wireless signal classifiers," *IEEE Trans. Wireless Commun.*, vol. 21, no. 6, pp. 3868–3880, Jun. 2022.
[39] N. B. Gaikwad, H. Ugale, A. Keskar, and N. C. Shivaprakash, "The Internet-of-Battlefield-Things (IoBT)-based enemy localization using soldiers location and gunshot direction," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11725–11734, Dec. 2020.
[40] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 274–283.
[41] A. Agarwal, M. Vatsa, R. Singh, and N. K. Ratha, "Noise is inside me! Generating adversarial perturbations with noise derived from natural filters," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 3354–3363.
[42] F. Khalid, M. Hanif, S. Rehman, J. Qadir, and M. Shafique, "FadeML: Understanding the impact of pre-processing noise filtering on adversarial machine learning," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, 2019, pp. 902–907.

**MARYAM ABBASALIZADEH** (Graduate Student Member, IEEE) received the B.S. degree in computer software engineering from the University of Mohaghegh Ardabili, Iran, in 2013, and the M.S. degree in computer software engineering from Islamic Azad University, Ardabil, Iran, in 2017. She is currently pursuing the Ph.D. degree in computer science with the University of Massachusetts Lowell (UMass Lowell). During the M.S. studies, her research centered on wireless networks and the Internet of Things (IoT). She has published a article on wireless optimization in the *International Journal of Communication Systems* (IJCS). Her research interests include optimizing wireless networks for enhanced privacy and security, particularly in scheduling for 5G and beyond technologies with the aim of integrating state-of-the-art machine learning techniques.

**JEFFREY CHAN** is currently pursuing the B.S. degree in computer science with the University of Massachusetts Lowell (UMass Lowell). He is focusing on domains, such as cybersecurity, data analytics, and full-stack development. Beyond his coursework, he actively participates in collaborative research projects. He contributes by collecting and aggregating data and developing software solutions that streamline and enhance the research process.

**PRANATHI RAYAVARAM** received the B.S. and M.S. degrees in computer science from the University of Massachusetts Lowell (UMass Lowell), in May 2020 and December 2022, respectively, where she is currently pursuing the Ph.D. degree in computer science, focusing her dissertation on cybersecurity education for K-12 students. She has expertise in machine learning and cybersecurity. Her research aims to clarify and engage young learners in AI and cybersecurity concepts. She is also investigating android security, specifically analyzing potential stalkerware threats. She has co-authored six peer-reviewed publications and actively contributes as a Reviewer at the ACM Technical Symposium on Computer Science Education (SIGCSE TS) and the IEEE Frontier in Education (FIE) Conferences. She is interested in wireless network security.



**YIMIN (IAN) CHEN** (Member, IEEE) received the B.S. degree in electrical engineering from Peking University, in 2010, and the Ph.D. degree, with a focus on security and privacy in mobile computing from Arizona State University, in 2018. After completing postdoctoral research with Virginia Tech, he joined the Department of Computer Science, University of Massachusetts Lowell (UMass Lowell), as an Assistant Professor, in 2021. Currently, his work focuses on the understanding and development of secure and privacy-aware machine learning models with applications on NLP, the IoT, and healthcare systems, including data space attack detection and privacy-aware NLP models for EHR. He has authored or co-authored papers in leading networking and security conferences (e.g., IEEE S&P and INFOCOM, ACM CCS, MobiCom, and NDSS) and journals (e.g., IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING). He is active in multiple societies, serving as a paper reviewer and a TPC member.

**SASHANK NARAIN** received the B.S. degree in information technology from the University of Mumbai, India, in 2007, and the combined M.S. and Ph.D. degrees from Northeastern University, Boston, USA, in 2018. He was a Software Engineer with multi-national software company. His doctoral research explored security and privacy challenges in mobile ecosystems. Then, he was a Postdoctoral Researcher with Northeastern University, for a year, before joining the Computer Science Department, University of Massachusetts Lowell (UMass Lowell), in 2019. His current research interests include the intersection of security and privacy in wireless networks, cryptography, mobile ecosystems, and K-12 cybersecurity education. He has co-authored papers presented at leading cybersecurity and educational conferences, including IEEE Symposium on Security and Privacy (IEEE S&P), USENIX Security Symposium, and the ACM Technical Symposium on Computer Science Education (SIGCSE TS). In 2021, he was a recipient of the prestigious DARPA Young Faculty Award (YFA). He actively contributes to the academic community by serving as a reviewer and a technical program committee (TPC) member for various conferences and journals.