IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

gShock: A GNN-based Fingerprinting System for Permissioned Blockchain Networks over Encrypted Channels

MINJAE SEO[1], JAEHAN KIM[2], MYOUNGSUNG YOU[2], SEUNGWON SHIN[2], and JINWOO KIM[3]

[1]ETRI, Daejeon, South Korea
[2]School of Electrical Engineering, KAIST, Daejeon, South Korea
[3]School of Software, Kwangwoon University, Seoul, South Korea

Minjae Seo and Jaehan Kim contributed equally.
Corresponding author: Jinwoo Kim (e-mail: jinwookim@kw.ac.kr).

ABSTRACT Blockchain technology has ushered in a transformative paradigm of decentralized and transparent systems, offering innovative solutions across diverse sectors. While these systems strive for unparalleled transparency and trustlessness in a fully distributed framework, permissionless blockchains, such as Bitcoin and Ethereum, encounter vulnerabilities due to their intrinsically public nature. Addressing these vulnerabilities, the emergence of permissioned blockchains presents a fortified alternative, incorporating rigorous access controls and authentication protocols to ensure participation exclusivity and transaction confidentiality. Nevertheless, a keen observation reveals that, despite encryption, the operational traffic within these blockchains manifests distinct time-series patterns and operational relations during sensitive data exchanges. Such patterns hold the potential to inadvertently expose critical details about the network, encompassing its topology and the operational dependencies among nodes. In light of this revelation, we introduce a pioneering blockchain fingerprinting mechanism, denoted as gShock. This system meticulously analyzes periodic patterns and the context of operational relations from the collected blockchain network traffic. It employs a Graph Neural Network (GNN)-based model, adept at capturing the intricate characteristics innate to specialized blockchain operations. Through empirical experiments conducted in a realistic permissioned blockchain environment, comprising various nodes, we ascertain that gShock demonstrates a remarkable proficiency in classifying blockchain operational traffic with an F-1 score of $\geq$ 96% and identifying individual dependencies with a macro F-1 score of $\geq$ 93%.

INDEX TERMS Blockchain Security, Fingerprinting, Graph Neural Network (GNN)

I. INTRODUCTION

Over the past decade, researchers have revealed that *permissionless* blockchains (e.g., Bitcoin and Ethereum) are susceptible to various attacks, primarily stemming from their public nature, which grants access without prior verification. One prominent example is the 51% attack where adversaries accumulate more than 50% of the network's mining power, thereby gaining the potential to control the blockchain [1], [2]. Moreover, adversaries who possess a significant number of nodes can orchestrate sybil attacks [3], manipulating the targeted nodes' perception of the blockchain by controlling the flow of information. Furthermore, adversaries may execute eclipse attacks [4], wherein they isolate specific nodes or a group of nodes from the network by enveloping them with malicious nodes under their control. Consequently, adversaries can disrupt communication channels or launch double-spending attacks.

In response to the security concerns raised by permissionless blockchains, the development of *permissioned* blockchains has emerged as a robust alternative. Permissioned blockchains incorporate access controls and authentication mechanisms, ensuring that only *authorized* participants can join the network. These participants are typically known entities, such as pre-approved organizations or individuals, who must meet specific criteria to gain access. This controlled access enables improved governance, as consensus mechanisms can be tailored to meet the requirements of the participating entities. As a result, permissioned blockchains have gained traction among enterprises and organizations seeking to leverage blockchain technology while maintaining

a higher degree of trust and control over the network [5]–[8]. For example, Hyperledger Fabric [6] and Quorum [7] provide features like private transactions and channels tailored for enterprise environments, which ensure that transaction details are only shared and accessible among the involved parties, thereby establishing confidentiality.

However, the exchange of sensitive data within a blockchain network raises significant security concerns. Recent findings have revealed that adversaries can fingerprint sensitive information from these networks. By employing machine learning-based classifiers trained on various features – such as transactions [9]–[11], addresses [12], and network traffic [13]–[17] – adversaries can infer user behavior, identity, and applications. Moreover, even when network traffic is encrypted, adversaries can extract valuable information by analyzing flow-level characteristics (e.g., packet sizes and timings), which remain unencrypted during transmission. These threats underscore that encryption alone is insufficient to secure blockchain networks. *We posit that an even more powerful fingerprinting attack may be possible in permissioned blockchain networks.*

In this paper, we demonstrate that confidential information within permissioned blockchain networks, such as *network topology* and *operational dependencies*, can be leaked. If adversaries acquire this information, they can launch severe attacks targeting critical nodes or links within the network, potentially causing failures or operational delays. However, fingerprinting this information presents several challenges: First, adversaries must distinguish permissioned blockchain traffic from the vast amount of general production traffic. Second, they must reconstruct the network topology and operational dependencies within the permissioned blockchain traffic, which is generated by multiple roles and links. Previous studies have predominantly focused on machine learning-based approaches [12]–[15], [17] or permissionless blockchain networks [10], [11], [16], making them unsuitable for addressing these challenges (Section VII).

To this end, we introduce gShock, an advanced deep learning-based system specifically designed for fingerprinting permissioned blockchain networks[1]. Our approach leverages the observation that permissioned blockchain traffic exhibits distinct time-series patterns and operational dependencies between multiple nodes. These patterns provide valuable insights into the temporal and relational aspects of blockchain operations, such as periodic synchronization, block creation, and transaction ordering. To effectively capture these inherent patterns in blockchain traffic, our classification models integrate a Long Short-Term Memory (LSTM) network [18] with a Graph Neural Network (GNN) [19]. Through an extensive series of experiments, we demonstrate that the LSTM-GNN models excel in capturing both time-series and operational relationships within permissioned blockchain traffic, outperforming other classification methods, including LSTM-based models [20].

We implemented a full prototype of gShock and conducted comprehensive experiments in a realistic private permissioned blockchain network comprising multiple nodes [6]. In our evaluation, we collected approximately 205,593,788 packets generated by the permissioned blockchain network, simulating real-world data traffic by deploying blockchain operation protocols and using CAIDA traces alongside various enterprise services and consensus protocols. We demonstrated gShock's effectiveness through an extensive set of experiments, comparing it with different baseline models. Our results show that gShock maintains robust performance even in the presence of several defense systems, underscoring its operational feasibility, particularly in realistic scenarios where adversaries can only capture portions of the traffic. We believe this work establishes a foundational understanding of permissioned blockchain architecture and enables network operators to assess its security during deployment, thereby enhancing security awareness in the blockchain industry.

Contributions: We make the following contributions:

- The in-depth exploration of a new attack vector in a permissioned blockchain network provides insight into an avenue through which adversaries can infer the internal architecture over encrypted channels.
- The proposal of a GNN-based fingerprinting method for a permissioned blockchain network, gShock, introduces a novel approach that shows significant outcomes even in the presence of multiple defense systems.
- The extensive experiments in a realistic permissioned blockchain network involve the utilization of the developed fingerprinting system to substantiate the effectiveness of the posed vulnerabilities.

It is imperative to acknowledge that the present study is an extension of our preceding research [20]. Within this study, we have contextualized the earlier methodology specifically for blockchain networks and have further enhanced its scope by incorporating GNN.

II. BACKGROUND AND MOTIVATION

A. PERMISSIONED BLOCKCHAIN NETWORK

Unlike permissionless blockchains, such as Bitcoin [21] and Ethereum [22], permissioned blockchain networks limit access and participation exclusively to authorized entities, thereby guaranteeing heightened levels of privacy and security. A prominent example of a permissioned blockchain network is Hyperledger Fabric [23]. Hyperledger Fabric is a highly configurable, open-source, modular blockchain platform optimized for enterprise applications and distributed deployment across the globe. Developed by the Linux Foundation, it is engineered with a suite of unique features tailored to modern business needs, including permissioned membership, customizable consensus protocols, and support for mainstream programming languages in writing smart contracts (chaincode). These capabilities ensure confidentiality, scalability, and flexibility, making it a preferred option for

[1]To the best of our knowledge, we are the first to fingerprint permissioned blockchain networks.

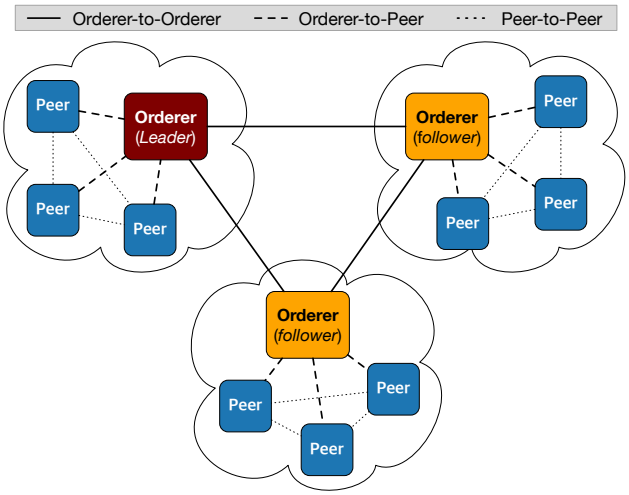

FIGURE 1. An example of a permissioned blockchain network deployed over a WAN. The nodes represent different types of blockchain nodes (including their roles for *Orderers*), and the links represent various operational dependencies within the blockchain network. Note that *Peer* nodes may belong to the same or different organizations.

enterprises seeking a robust and efficient blockchain solution [6], [24], [25].

B. NODES AND OPERATIONAL DEPENDENCIES

In a permissioned blockchain framework [6], [7], [23], both *peers* and *orderers* in Hyperledger Fabric play essential roles in maintaining the network. During its operation, specific *operational dependencies* emerge among the nodes, as illustrated in Figure 1. These operational dependencies represent the logical connections between blockchain nodes, reflecting the network's architecture. In what follows, we describe the roles of nodes within a permissioned blockchain network and introduce the operational dependencies that arise during its operation. Additionally, we discuss the inherent data exchange characteristics associated with each of these dependencies.

Peer Nodes: In the Hyperledger Fabric, there are three main types of peer nodes. *(i)* Endorser Peers endorse transactions by executing them against their copy of the ledger without actually updating it, and then provide a signed endorsement of the transaction's validity. *(ii)* Committing Peers, on the other hand, do not have endorsement capabilities but are responsible for validating endorsements on a transaction before committing it to the blockchain. Both endorsing and committing peers must commit transactions to the blockchain, but only endorsing peers generate endorsements. Finally, *(iii)* Anchor Peers are special, discoverable peers used for communication across different organizations; each organization in a channel has at least one anchor peer to facilitate inter-organizational communication and interaction. Thus, the messages exchanged between peer nodes (referred to as *Peer-to-Peer* dependency) include transaction proposals, endorsements, and, in the case of anchor peers, messages related to cross-channel communication. This exchange of messages ensures the proper endorsement, validation, and commitment of transactions, as well as the necessary communication between different organizations participating in the network.

Orderer Nodes: The orderer nodes are responsible for creating blocks of transactions and distributing them to peer nodes. The *default* ordering service uses the Raft consensus algorithm [26], which enables unanimity on the transaction sequence, preventing inconsistencies and ensuring the same transaction order across all nodes. During the data exchange among orderer nodes, *Orderer-to-Orderer* dependency is created while involving the exchange of Raft messages between orderer nodes to maintain consistency and agree on the transaction order. The Raft protocol includes the election of a *leader* among the orderer nodes, who then manages the log replication process to the *follower* nodes. This involves exchanging messages related to the Raft consensus algorithm, such as heartbeats, vote requests, and log entries. Additionally, the primary data exchanged between orderer nodes and peer nodes (referred to as *Orderer-to-Peer* dependency) consists of blocks of transactions. Peer nodes then validate these transactions and update their ledger accordingly.

C. MOTIVATION

Adversaries typically cannot access information about a permissioned blockchain network without participating in it. However, if they manage to obtain such information without being part of the network, they could exploit it for malicious purposes. For example, understanding the roles and operational dependencies within a permissioned blockchain network (Figure 1)—such as the relationships between peers and orderers or between leader and follower orderers—can provide adversaries with critical insights. While this information might seem general, if exposed, adversaries could use it to launch various critical attacks.

First, adversaries may launch targeted DDoS attacks. By analyzing the inferred internal structure, they could identify the links between leader and follower nodes and attempt to cut off these connections (i.e., *Orderer-to-Orderer* dependency) using stealthy DDoS attacks [20], [27], [28] or disrupt node operations through off-path TCP injection attacks [29], [30]. Given that the leader orderer is essential for consensus coordination and transaction block proposals, a disruption in its connection could lead to serious issues such as inconsistent transaction orders, network fragmentation, and transaction processing delays. Notably, the failure of even a single leader orderer has the potential to paralyze the entire permissioned blockchain network.

Second, adversaries can exploit known protocol vulnerabilities within the inferred operational dependencies. Notably, existing studies highlight the tendency of blockchain protocols to exhibit abnormal behaviors in response to unexpected environmental changes [31], [32]. For example, the Raft protocol, which governs the election of a leader orderer by assigning elevated *terms*, requires the current leader to step down if it encounters a node with a higher term. According to the Raft specification [26], the leader must relinquish

its role when faced with a node whose term surpasses its own. However, there are instances where two leaders may be elected simultaneously in different parts of the network, often due to network partitioning caused by link failures. Since Raft prohibits multiple leaders, this situation can create a complex loop where these leaders continuously compete for leadership, leading to disruptions in the entire blockchain network's operation.

III. FINGERPRINTING PERMISSIONED BLOCKCHAIN

A. THREAT MODEL

We consider a permissioned blockchain network that is geographically distributed and would be interconnected over the WAN. This is especially the case for global enterprises [6], [33], [34], which often utilize dedicated encrypted tunnels. Within individual sites of the permissioned blockchain network, we postulate that network operators adopt in-band management [35], meaning that both *blockchain traffic* (e.g., protocol-related messages, configuration, and management data) and *data traffic* (i.e., normal traffic) share the same communication link. Such a configuration is prevalently adopted in large-scale networks, owing to the reduced costs associated with constructing a dedicated control network and its significant simplification of network maintenance [36]–[38].

An adversary's goal is to fingerprint the internal architecture of the permissioned blockchain network without direct participation. Following the threat model proposed by prior fingerprinting studies [13], [16], [39], [40], we consider *local* and *passive* adversaries, whose presence is within the same network as the target blockchain network. The adversaries can intercept traffic by exploiting a (local) compromised router [41], [42], but are unable to access packet payloads due to the encryption. Instead, they can observe packet lengths and timings of encrypted packets. Additionally, we assume an *open-world* scenario, under which the scope of intercepted traffic is not confined to blockchain transactions alone. This broad capture introduces challenges for the adversaries to discern blockchain-related traffic among a wide spectrum of data transmissions. It is important to note that the scenario we present is challenging for adversaries, considering that they must possess the dual capability of i) classifying blockchain traffic from non-blockchain sources and ii) inferring confidential information from the classified blockchain traffic.

B. TECHNICAL CHALLENGES

Strawman solution: Adversaries could simply rely on traditional rule-based strategies [43], [44] to classify blockchain traffic based on empirical rules. However, as blockchain service operators might be able to use custom configurations (e.g., by changing the port numbers), this approach can lead to accuracy deterioration when processing unmatched traffic policies [45]–[47]. Unlike the traditional method, the state-of-the-art deep learning-based system can learn hidden patterns in the blockchain operation traffic itself, so the aforementioned problems can be effectively handled. To take advantage of a deep learning-based system in our work, we need to address the following three challenges:

C1. Pruning data traffic noise: Nowadays, data traffic is very dynamic and diverse due to the pervasive usage of many different applications and services within networks [48], [49]. Note that the blockchain traffic captured by adversaries contains not only control traffic but also data traffic; it may happen that the data traffic patterns accidentally overlap with the blockchain operation traffic patterns. Thus, the classification model may falsely detect the given traffic. In this context, the challenge is how to select the features to be used in the classifier such that false positives and false negatives are minimized considerably.

C2. Identifying operational dependencies: Upon differentiating between blockchain and data traffic, adversaries are faced with the subsequent challenge of identifying the specific operational dependencies between nodes. Existing studies [20], [50] predominantly concentrates on the directional attributes of individual flows in order to classify application-level protocols in the operational dependencies among nodes. Consequently, this approach falls short in encapsulating the comprehensive characteristics exhibited by the entirety of traffic flows generated by nodes.

C3. Determining the role of each node: After classifying the blockchain operational dependencies between nodes, the question of how to define the leader-follower roles of orderers in the permissioned blockchain network still remains. To infer the role of each orderer, we need to design additional deep learning models following the blockchain operational dependency classifier. However, the number of flows sent by the leader orderer is significantly lower compared to those sent by the follower orderer nodes, and handling sparse labels can make the model fall into overfitting [51]. Thus, models cannot be trained effectively to identify the role of each orderer.

C. OVERVIEW OF GSHOCK

Based on the given challenges, we devise an advanced automatic fingerprinting system—which we call gShock—to effectively address the three main issues stepwise, as shown in Figure 2.

Phase-1: Blockchain traffic classification. To address C1 in Section III-B, adversaries first need to prune data traffic noise from blockchain traffic traces. It is worth noting that blockchain operation traffic displays a periodical time-series pattern. To extract those features, gShock first runs a coarse-grained classification as shown in Figure 2. The *Packet Preprocessor* module arranges traffic traces into 2-tuple (`SrcIP, DstIP`) flows. The *Feature Extractor* module then generates time-series features of the given 2-tuple flows automatically. Finally, the *Sequence Classifier* module solves a 2-class classification problem (i.e., blockchain traffic or data traffic) based on the received features of the 2-tuple flows. Our implemented classification system incorporates the utilization of Long Short-Term Memory (LSTM), a neural network architecture renowned for its adeptness in capturing

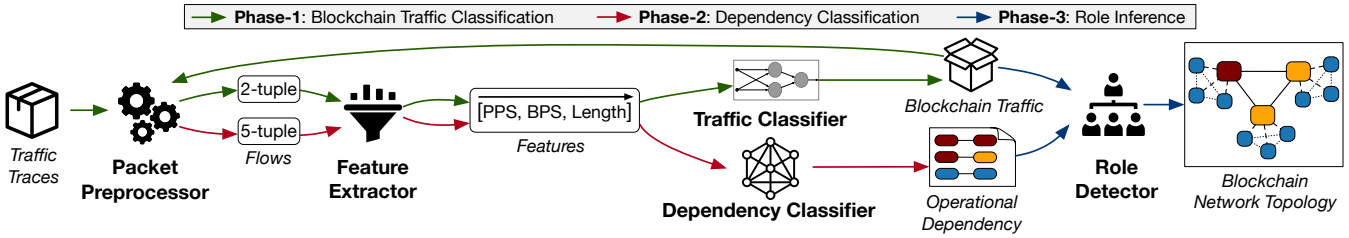

FIGURE 2. **gShock system overview and its workflow that includes three phases: ① Blockchain Traffic Classification, ② Depedency Classification, and ③ Role Inference.**

intricate time-series patterns inherent in network traffic. (see Section IV-B).

Phase-2: Operational Dependency Classification. To address C2 in Section III-B, adversaries need to extract more fine-grained features and adopt a specialized Graph Neural Network (GNN) model capable of capturing the comprehensive attributes inherent in blockchain operation traffic. This traffic includes various components such as transaction proposals, endorsements, Raft consensus protocol messages, and transaction blocks. In this phase, the goal is to effectively learn the dependencies between nodes and identify orderers, which will serve as inputs for the next phase. To achieve this, the *Packet Preprocessor* module organizes the blockchain operation traffic flows obtained in Phase-1 based on a 5-tuple (`SrcIP, SrcPort, DstIP, DstPort, Proto`). The *Feature Extractor* module then extracts time-series features from these 5-tuple flows. Finally, the *Sequence Classifier* module addresses a multi-class classification problem, distinguishing between operational dependencies using the features of the given 5-tuple flows. Specifically, links between nodes are classified as *Orderer-to-Orderer*, *Orderer-to-Peer*, or *Peer-to-Peer* dependencies (see Section IV-B).

Phase-3: Role inference. To address C3 in Section III-B, it is necessary to discover the role of each node, particularly identifying the leader and follower nodes. Conventionally, the leader node exhibits a higher volume of packet transmission and reception compared to follower nodes. Leveraging this distinction, adversaries can employ the standard *z-score* normalization technique to detect outliers that significantly deviate from the mean. The *Role Detector* module calculates the *z-score* based on the count of packets transmitted by individual `SrcIP` addresses. Subsequently, it determines inferred roles to each node, classifying them as leader or follower nodes, employing predefined *z-score* thresholds. By quantifying this process into a numerical metric, the procedure achieves computational efficiency of $\mathcal{O}(1)$. In contrast, certain approaches like a 1-layer GNN entail redundant inference computations with a computational complexity of $\mathcal{O}(NF^2 + |E|F)$, where N and F denote the hidden and embedding dimensions, respectively, and $|E|$ denotes the total number of edges [52].

IV. METHODOLOGY

In this section, we present pivotal structures of gShock. Specifically, we first address the problem of how to characterize time-series and operational dependency features by leveraging different types of deep learning layers (i.e., LSTM and GNN). We introduce two distinct deep learning models for Phase-1 and Phase-2 classification, respectively. Finally, we present key insight to infer the role of each node from the classified operational dependencies among nodes.

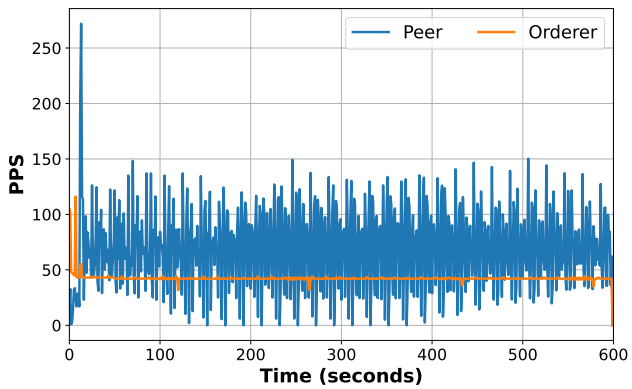

FIGURE 3. **An example to exhibit the consistent traffic patterns between peer and orderer nodes in a permissioned blockchain network.**

A. FEATURE EXTRACTION

A crucial component of gShock is the selection of appropriate features. Specifically, the accurate characterization of time-series patterns is essential; without it, the development of a robust classifier model becomes an exceedingly laborious task. To address this, we focus on the inherent traffic patterns of the blockchain system during its operation, aiming to gain insights into the composition of peer and orderer traffic. This, in turn, enables us to meticulously analyze the unique patterns retained in the block traffic flow.

We consider features like *the number of packets* (e.g., PPS) and *packet size* (e.g., BPS) since they are commonly used when fingerprinting network traffic [14], [20], [50], [53], [54]. Also, both features can represent the unique characteristics during the blockchain operation. By analyzing these sequential features, we discover that the blockchain traffic generate packets periodically. Specifically, nodes, such as peers and orderers, communicate with each other frequently to maintain consensus and synchronize the ledger. These communications and operations often lead to periodic traffic patterns. For example, orderer nodes, which are responsible for creating blocks and distributing them to peers, generate

traffic with regular intervals, especially in networks with high transaction rates. Similarly, peers also generate periodic traffic patterns while endorsing transactions, interacting with clients, or syncing with other peers. Furthermore, periodic maintenance operations, such as gossip protocol for state synchronization and leader election in case of failures, also contribute to the periodicity of the traffic. Hence, the combined effect of these regular operations and communications in the blockchain network results in traffic exhibiting periodic patterns. The noticeable *periodic* patterns of blockchain traffic are confirmed by the empirical results as shown in Figure 3.

B. BLOCKCHAIN TRAFFIC CLASSIFICATION

Now we present methodologies for classifying blockchain traffic and identifying operational dependencies among nodes in a permissioned blockchain network. We divide the classification procedure into two distinct phases: (i) pruning data traffic, and (ii) classifying operational dependencies.

Pruning data traffic. In order to identify blockchain traffic and eliminate irrelevant (data) traffic, we utilize time-series flow features extracted by *Feature Extractor*. The flow features are split into T time steps and averaged within each time step. By doing so, we organize time-series flow features at each time step $t \in \{1, 2, ..., T\}$ as $\mathbf{x}^t = [bps^t, pps^t, len^t]$ and a flow feature sequence for all the time steps as $\mathbf{x} = [\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^T]$. For each 2-tuple flow, we obtain a flow representation vector $\mathbf{h}$ by applying a LSTM and a fully-connected layers in turn to the flow feature sequence. The flow representation vector is passed to the output layer and *Sigmoid* activation function, and then the model finally predicts whether the 2-tuple flow is related to blockchain operations or not. For the training process, we optimize a binary cross entropy loss $\mathcal{L}$ for the binary classification task, which is defined as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i), \quad (1)$$

where $y_i \in \{0, 1\}$ denotes the ground truth label and $\hat{p}_i$ denotes the predicted probability that y_i is 1.

Classifying blockchain operational dependencies. To distinguish the blockchain operation dependencies, we first construct a traffic network graph $\mathcal{G}$ where a 5-tuple flow serves as an edge $e \in \mathcal{E}$ between network nodes (i.e., IP addresses) $v \in \mathcal{V}$. For each flow edge, we establish an edge feature sequence $\mathbf{x}_e = [\mathbf{x}_e^1, \mathbf{x}_e^2, ..., \mathbf{x}_e^T]$, where $\mathbf{x}_e^t = [bps^t, pps^t, len^t]$ is a flow feature at each time step $t \in \{1, 2, ..., T\}$. Then, we leverage both LSTM and GNN layers to represent time-series patterns and operational features on the network graph. This representation process is formally described in Algorithm 1.

In line 1 to 2, we adopt a Bidirectional-LSTM (Bi-LSTM) [55] layer to embed the input edge features, reflecting the temporal dependencies of the time-series features effectively. We take the concatenation of the last hidden state of forward and backward directions to obtain the edge feature embedding vector $\mathbf{h}_e$. In line 3 to 5, node v's representation vector is obtained by aggregating the edge feature embedding vectors of the ingress edges from node v, generated by applying the ingress edge function $\mathcal{I}(v)$. We devise a tailored aggregation function Agg for the traffic classification task. Specifically, it is beneficial to consider the relative importance of each ingress edge in a complex traffic network graph. To this end, we design the aggregation function based on the graph attention mechanism [56], which has been widely applied in various domains [57]–[59]. The attention coefficient is formulated as follows:

$$\alpha_v(e) = Softmax(\sigma(\mathbf{a}_{attn}^T(\mathbf{W}_{attn}\mathbf{h}_e))), \quad (2)$$

where $\mathbf{W}_{attn}$ denotes weight matrix for the attention mechanism, $\mathbf{a}_{attn}$ denotes a vector for parameterizing a scalar value, and $\cdot^T$ denotes the transpose of a given vector. Then, we linearly combine the edge embedding vectors of the ingress edges after multiplying each of them by the corresponding attention coefficient:

$$\mathbf{h}_{\mathcal{I}(v)} = \sum_{e \in \mathcal{I}(v)} \alpha_v(e)\mathbf{h}_e. \quad (3)$$

By multiplying $\mathbf{h}_{\mathcal{I}(v)}$ by the weight matrix $\mathbf{W}$ and taking the activation function σ, we obtain the node v's representation vector $\mathbf{h}_v$. In line 7, We normalize each node representation vector. In line 8, to obtain the edge representation vector $\mathbf{h}_{\tilde{e}_{uv}}$ between node v and u, we concatenate their node representation vectors and apply the weight matrix $\mathbf{W}_f$ and the activation function.

The edge representation vector $\mathbf{h}_{\tilde{e}_{uv}}$ is fed into the output layer and *Softmax* activation function, and then the model finally predicts whether each 5-tuple flow is related to a specific operational dependency. For a multi-class classification task, we opted to minimize a cross entropy loss function defined as follows:

$$\mathcal{L} = -\frac{1}{|\mathcal{E}|}\sum_{e_{uv} \in \mathcal{E}}\sum_{c} y_{e_{uv},c} log(\hat{p}_{e_{uv},c}), \quad (4)$$

where c denotes each class, $y_{i,c}$ is i-th sample which belongs to class c, and $\hat{p}_{e_{uv},c}$ is a predictable distribution of the sample. In this context, $\hat{p}_{e_{uv},c}$ would be computed by applying *Softmax* to $\mathbf{h}_{\tilde{e}_{uv}}$.

C. ROLE INFERENCE

The roles as the leader and followers can be distinguished through the total number of the packets orderer nodes transmit. Figure 4 illustrates the analysis of blockchain network flows concerning transmitted packets between orderer nodes. The `leader` orderer node typically handles a higher volume of packet transmissions compared to follower orderer nodes. This is because the leader is responsible for creating blocks of transactions and distributing them to the follower nodes. More specifically, the leader orderer node receives transactions from clients, orders them into a block, and then sends the block to the follower orderer nodes for validation and commitment to their ledgers. This process involves a higher volume of transmitted packets for the leader orderer node,

Algorithm 1 Edge representation

Input: Network traffic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$;
input edge feature sequence $\mathbf{x}_e = [\mathbf{x}_e^1, \mathbf{x}_e^2, ..., \mathbf{x}_e^T]$,
where $\mathbf{x}_e^t = [bps^t, pps^t, len^t], \forall t \in \{1, 2, ..., T\}$;
weight matrix $\mathbf{W}$;
weight matrix for edge representation $\mathbf{W}_f$;
activation function σ;
ingress edge function $\mathcal{I} : v \rightarrow 2^{\mathcal{E}}$;
edge aggregation function Agg;

Output: Vector representations $\mathbf{h}_{\tilde{e}_{uv}}, \forall \tilde{e}_{uv} \in \mathcal{E}$

$\mathbf{h}_{e,f}^T, \mathbf{h}_{e,b}^T \leftarrow \text{LSTM}(\mathbf{x}_e), \forall e \in \mathcal{E}$
$\mathbf{h}_e \leftarrow \text{Concat}(\mathbf{h}_e^{T,forward}, \mathbf{h}_e^{T,backward}), \forall e \in \mathcal{E}$
for $v \in \mathcal{V}$ **do**
  $\mathbf{h}_{\mathcal{I}(v)} \leftarrow \text{Agg}(\mathbf{h}_e, \forall e \in \mathcal{I}(v))$
  $\mathbf{h}_v \leftarrow \sigma(\mathbf{W} \cdot \mathbf{h}_{\mathcal{I}(v)})$
end for
$\mathbf{h}_v \leftarrow \mathbf{h}_v / \|\mathbf{h}_v\|_2, \forall v \in \mathcal{V}$
$\mathbf{h}_{\tilde{e}_{uv}} \leftarrow \sigma(\mathbf{W}_f \cdot \text{Concat}(\mathbf{h}_u, \mathbf{h}_v)), \forall \tilde{e}_{uv} \in \mathcal{E}$

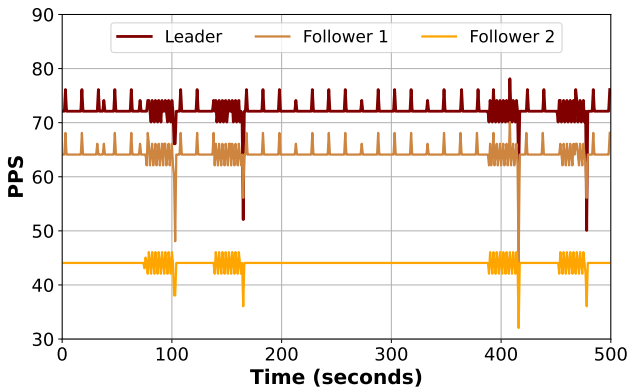

FIGURE 4. An example of disparate flow patterns corresponding to distinct roles of nodes.

as it is actively involved in the propagation of consensus messages and blocks to its follower orderer nodes, thereby facilitating the maintenance and updating of the blockchain ledger across the network.

Z-score Normalization. Based on this insight, we utilize the *z-score* normalization to define a role for each node to improve the efficiency of system flow. The utilization of *z-score* normalization has proved a significant effect on classifying with prominent pattern features [60]. Here, the distinct values of each node, which is organized from the classified operational dependencies, are normalized based on a mean and standard deviation. The *z-score* normalization value z is defined as $z = (x - \mu)/\sigma$, where x is the distinct values, μ is the mean of our data, and σ is the standard deviation. In this regard, we first arrange the total number of transmitted packets for each `SrcIP`. The total number of the packets of each `SrcIP` can be the distinct value x with which its mean value can be easily calculated. The calculated score enables us to infer the roles by defining a threshold (θ_z) in a heuristic manner.

V. EVALUATION

In this section, we evaluate the three phases mentioned above considering various environments to demonstrate their practicability. We first evaluate precision, recall, and F1-score to classify blockchain operation traffic. Then, we show the results of classifying operational dependencies. In addition, we demonstrate that our classification method effectively represents the sequential and operational features in comparison with several existing works' methods, such as different deep learning models. Also, we show the robustness of our classification model by testing the accuracy while adopting several defense systems. After that, we calculate *z-score* based on the total number of the packets for each refined `SrcIP` to infer one of the roles as a leader or a follower role for each node. Finally, we achieve blockchain network topology and operational dependencies among nodes.

A. EXPERIMENTAL ENVIRONMENT

In this study, we establish a private blockchain testbed encompassing multiple peer and orderer nodes, all under a dedicated organization. This testbed is hosted on several machines, each powered by an Intel Xeon Silver 4210R CPU (with 10 cores) and furnished with 64 GB of RAM. Notably, these machines collaboratively operate the distributed blockchain network, simulating a range of environmental conditions. While our methodology is adaptable to a variety of permissioned blockchain architectures, we utilize Hyperledger Fabric v2.2 [23] in our experimental setup, given its status as one of the leading open-source tools in this domain. This modular blockchain framework defaults to deploying the Raft consensus protocol. It is important to note that within the Raft protocol, nodes are categorized into distinct roles: the *leader* and the *followers*. With regard to potential adversaries, our underlying assumption is that they possess the capability to replicate our experimental blockchain environments. Such replication would allow them to achieve multiple configurations, secure labeled data, and consequently craft the necessary pre-trained models essential for fingerprinting blockchain networks.

We use NVIDIA GeForce RTX 3090 GPUs to train our deep learning models. To construct the input sequences, we align the time intervals of the traffic datasets by truncating them to match the shortest interval. We experiment with various sliding window sizes—60, 120, and 180—ultimately selecting 120 as it delivers the best performance. We configure the LSTM and GNN layers with dimensions of 32, 64, and 128, and test learning rates of 1×10^{-2}, 5×10^{-3}, and 1×10^{-3}, along with batch sizes of 4,096, 8,192, and 16,384. Through these experiments, we determine the optimal configuration to be 128 dimensions, a learning rate of 5×10^{-3}, and a batch size of 16,384.

B. DATASET DESCRIPTION

To our best knowledge, public permissioned blockchain traffic has not been released. Therefore, it is significant to collect blockchain traffic data as reliable as we can.

Blockchain traffic. We collect several sizes of blockchain network traffic deploying many containers in each machine by utilizing the container management tool Docker [61] because the sizes of blockchain network are numerous depending on demanding services. We use Hyperledger Fabric [23] to produce blockchain traffic. In the meantime, we execute several operations (e.g., chaincode operations, channel operations, transaction operations, block operations, etc.) on our testbed to reflect various events. Thus, we consider as many environments as possible extensively. It is important to note that all traffic we collect is entirely encrypted by SSL/TLS.

Data traffic. We utilize CAIDA backbone traffic [62]–[64] to provide a realistic network environment in our testbed. CAIDA passive trace dataset includes traces collected from high-speed monitors on a real backbone link. To evaluate our fingerprinting method fairly, we mix CAIDA traces with the blockchain operation traffic. Also, we play streaming videos from a secured VPN machine and Tor browser, and we utilize an email server and online-chatting platform (e.g., Skype) to improve the reality of our testbed. Furthermore, to evaluate the effectiveness of our system fairly, it is necessary to include the traffic that exhibits periodical patterns with blockchain traffic, such as the ones generated from commercial distributed systems. Therefore, we measure and store control traffic generated by distributed systems [20], which is developed to manage and control multiple networks comprised at multiple locations. This distributed system deploys the Raft protocol for the consensus algorithm, so that it can be proper control traffic to evaluate gShock's effectiveness.

Dataset collection methodology. We collect a total of 205,593,788 packets including both data and blockchain operation packets from our testbed which comprises 10 nodes to 20 nodes. Also, *Feature Extractor* module generates 1,142,233 data traffic flow dataset and 14,900 blockchain operation traffic flow dataset. The blockchain traffic flows including 10 nodes to 20 nodes in our testbed are used to emulate a realistic environment of the blockchain network having various sizes. Moreover, we create 5-tuple based dataset reflecting sequential and operational relations for the train and test dataset, which are labeled with three classes (i.e., Peer-to-Peer, Orderer-to-Peer, Orderer-to-Orderer). We consider the case where network adversaries can eavesdrop traffic on multiple sites.

C. ACCURACY OF BLOCKCHAIN TRAFFIC CLASSIFICATION

Table 1 describes how effectively blockchain traffic is distinguished from data traffic. We use an LSTM-based classifier, which is known to be well-suited for learning the sequential characteristics of network traffic due to its intrinsic ability to remember long-term dependencies and patterns within time-series data [18], [20], [50].

TABLE 1. Classifying blockchain operation traffic and data traffic.

Label	Precision	Recall	F1-score
Data	99.92	99.98	99.95
Blockchain	99.84	93.80	96.73

In our experimentation, the LSTM-based classifier exhibits outstanding performance in classifying both data traffic and blockchain operation traffic. Specifically, when evaluating the classification of blockchain traffic in our test scenarios, our model achieves an average F1-score of 96.73% over 100 executions, underscoring its robustness and efficacy. Due to the fluctuating patterns of traffic in real-world backbone links, distinguishing data traffic from blockchain operation traffic becomes notably distinct. Our LSTM-based classifier's accuracy and detail make it adept at recognizing these specific traffic characteristics, emphasizing its usefulness in real-world situations. Particularly, the high precision associated with the blockchain operation traffic label effectively guarantees the reduction of false positives. Consequently, this facilitates the acquisition of a refined set of blockchain operation traffic flows, which is necessary for the next phase of classification.

D. ACCURACY OF PROTOCOL CLASSIFICATION

In Phase-2, we show the effectiveness of our approach in identifying the operational dependencies between nodes. Table 2 presents a comparative performance among three classifiers. Notably, our LSTM-GNN classifier outperforms both the vanilla LSTM classifier and the state of the art LSTM-based classifier recently proposed for network traffic classification [20]. For clarity in our discussion, we utilize the nomenclature: Peer-to-Peer (P-P), Orderer-to-Peer (O-P), and Orderer-to-Orderer (O-O) to denote the respective operational dependencies.

In the case of our classifier, F1-score for each dependency of the P-P, O-P, and O-O is 99.58%, 86.17%, and 95.85%, respectively. It manifests a high performance because it can reflect the context of time-series patterns and operational relations sufficiently in the blockchain network. In addition, the F1-scores of the P-P dependency are higher than those dependencies of the O-P and O-O. This is because the P-P dependency includes a series of periodical operations. Specifically, Endorsement, validation, and commitment operations within Hyperledger Fabric inherently exhibit periodic patterns due to the structured process by which transactions are proposed, processed, and finally added to the blockchain. First, the endorsement phase involves a specified set of peers simulating and endorsing the transaction, producing a deterministic and predictable response pattern as transactions consistently go through this evaluation. Subsequently, the validation step, where transactions are assessed for compliance with the network's endorsement policies, follows a systematic check, leading to an observable pattern in traffic. Lastly, the commitment phase involves appending the approved transactions to the blockchain, an operation executed at a regular

TABLE 2. The performance of gShock and other solutions in classifying blockchain operational dependencies.

Traffic Type	LSTM			LSTM+Direction [20]			LSTM+GNN (Ours)		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Peer-to-Peer (P-P)	88.57%	96.98%	92.58%	94.78%	96.03%	95.40%	99.53%	99.62%	99.58%
Orderer-to-Peer (O-P)	99.99%	46.82%	63.78%	100%	50.00%	66.67%	99.97%	75.70%	86.16%
Orderer-to-Orderer (O-O)	98.97%	86.87%	92.53%	82.74%	92.19%	87.21%	92.04%	98.79%	95.30%

cadence as blocks reach their size or time threshold. Together, these sequential processes, operating in a steady flow of transactions, give rise to discernible, periodical patterns in the network traffic. In contrast, the performance results for the O-P dependencies are discernibly lower compared to other dependencies. The F1-scores recorded for O-P dependencies are 63.78%, 66.67%, and 86.17% for the vanilla LSTM classifier, the state-of-the-art LSTM-based classifier, and the LSTM-GNN classifier, respectively. This disparity in performance can be attributed to the relatively uniform nature of interactions between orderer and peer nodes. Primarily, the predominant data exchanges between these nodes revolve around the formation of transaction blocks and their subsequent distribution to the peer nodes, a process that is less varied than other operations.

In a practical setting, accurately identifying orderer nodes is of paramount importance to ensure successful outcomes during the role inference phase. The vanilla LSTM classifier faces challenges in this regard, given that the time-series patterns of certain O-O dependencies closely resemble those of P-P dependencies. While the state-of-the-art LSTM-based classifier alleviates this ambiguity by reflecting flow directional patterns, it still suffers from a low precision for classifying O-O dependencies. In contrast, our LSTM-GNN classifier identifies the majority of orderers by utilizing its GNN layers designed to capture operational relations.

E. ROBUSTNESS AGAINST DEFENSE SYSTEMS

Network operators may adopt several defense systems to hinder adversaries from learning unique patterns of network traffic. To demonstrate the robustness of our deep learning-based fingerprinting under the defense systems, we consider the case where network operators impose random noise and perturbation mechanism [65]–[68] on the outgoing traffic. We first add random noise, 30% extent from the root mean square (RMS) of the training inputs. In addition, we impose Fourier Perturbation Algorithm (FPA) [68] on the input traffic, which is a largely adopted method to protect sequence data. FPA is formalized by $\mathbf{FPA} = \mathbf{IDFT}(\mathbf{LPA}(\mathbf{DFT}(input), \lambda))$, where $\mathbf{DFT}$ and $\mathbf{IDFT}$ denote the Discrete Fourier Transform (DFT) and the Inverse DFT, respectively, and $\mathbf{LPA}$ denotes the Laplace Perturbation Algorithm. The λ is set to 30% of the RMS of the training inputs.

In Figure 5, the performance of our model during Phase-1 is notably robust, recording F1-scores of 93.02% and 93.57% for classifying blockchain traffic against random noise and

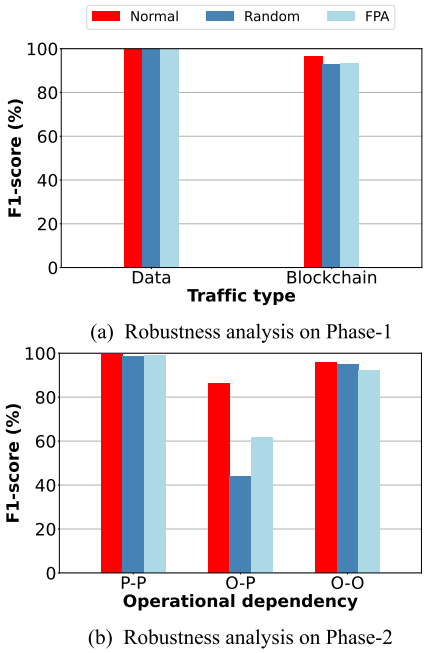

FIGURE 5. The accuracy of classifying blockchain traffic (Phase-1) and operational dependencies (Phase-2) while several defense mechanisms are adopted.

FPA, respectively. Meanwhile, Phase-2 of our model demonstrates commendable accuracy in classifying operational dependencies. Nevertheless, a noticeable decline in the F1-scores can be observed for the classification of Orderer-to-Peer dependencies under the aforementioned perturbation methods. This reduced performance can be primarily ascribed to the less diverse flow patterns observed during block formation and distribution processes. Despite the impact of the perturbation methods, our model exhibits resilience when classifying Orderer-to-Orderer dependencies, achieving F1-scores of 94.74% and 92.01% against random noise and FPA, respectively. Such robust classification capability underscores the potential of our framework to effectively pinpoint target nodes that could be susceptible to network attacks.

F. EFFECTIVENESS OF ROLE INFERENCE

We now evaluate how gShock correctly infers a role for each node using the *z-score* normalization based on the results derived from our LSTM-GNN classifier that shows the highest accuracy in the previous phases. In our analysis, the leader node is more likely to be located above the threshold (θ_z)

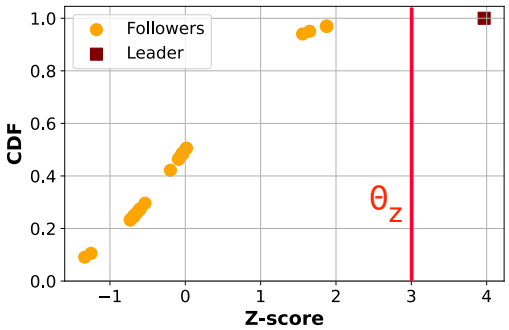

FIGURE 6. Node distribution for inferring orderer node roles based on blockchain operational dependencies and the result of role inference when $\theta_z = 3$.

in contrast with the follower nodes, which is computed by the total number of transmitted packets. Each execution time for calculating *z-scores* and detecting roles takes less than 2 seconds.

In Figure 6, it shows that *z-scores* for inferring leader role is 3.97 for our test case. Based on the aforementioned *z-scores*, role inferences in our test case is placed at our predictable domain. This is because the leader orderer node plays a pivotal role in the consensus mechanism, especially when Raft consensus is employed. The leader orderer is responsible for managing the log replication process across the network. When a new transaction or block is proposed, it first reaches the leader orderer node. This leader order node then logs the transaction or block and initiates the process of replicating this log entry to all follower orderer nodes. Consequently, the leader order node needs to transmit proposals, log entries, and various consensus-related messages to its follower orderer nodes, ensuring that all orderer nodes maintain a consistent and up-to-date ledger. This centralized coordination and log replication responsibility naturally result in the leader orderer node transmitting a significantly higher volume of packets or data compared to follower orderer nodes, which primarily act as passive recipients in this process.

G. GSHOCK USE CASE

Upon successfully fingerprinting the blockchain network topology and operational dependencies, an adversary can launch harmful attacks, thereby compromising the integrity of the blockchain's operation. Based on the achieved confidential information, an adversary can specify the crucial path, frequently traversed by pivotal blockchain operational messages. To illustrate the vulnerability arising from such potential exposures, we introduce a stealthy Distributed Denial-of-Service (DDoS) attack scenario. In this context, an adversary floods a specific path integral to the channel within an Orderer-to-Orderer dependency. However, it is worth noting that due to ethical considerations associated with generating enormous traffic volumes in a real-world WAN setting, we resorted to crafting our dedicated blockchain network testbed, as shown in Figure 7.

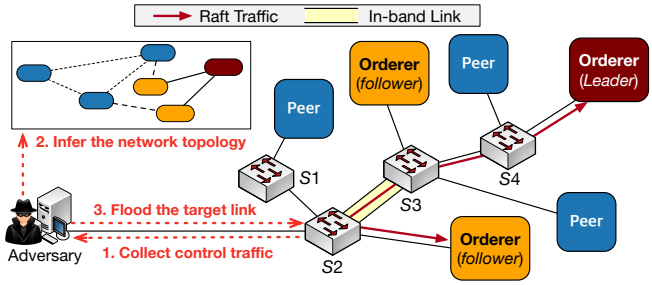

FIGURE 7. An example that shows how an adversary disrupts the Raft traffic on the in-band link *S2-S3* between the leader and follower.

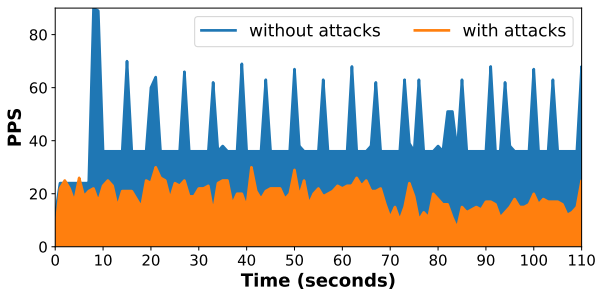

FIGURE 8. Measured throughput (PPS) of the cluster management protocol traffic for two cases; *(i) without attacks* and *(ii) with attacks.*

Our private testbed comprises four Pica 3297 switches and six Intel Xeon Silver 4210R CPUs, with three CPUs allocated for adversarial machines and the remaining three dedicated to running multiple nodes. We support multiple nodes (10 to 15) on each server, utilizing Docker [61]. Also, we deploy 20 containers, representing one-third of the total nodes, specifically configured to operate a variety of adversarial bots. This setup allows us to generate low-volume traffic across individual links while simultaneously directing a concentrated, high-volume traffic load toward the target path. Notably, to ensure the efficacy of the volumetric traffic and to clearly demonstrate the significant performance impact, all links within our testbed are equipped with 10 GbE interfaces.

To emphasize the implications of our proposed attack, it is important to note its potential to cause significant disruptions during the blockchain's operational phase. Empirical results indicate that our attack leads to an average performance degradation of 51%, as shown in Figure 8.

VI. DISCUSSION

This section introduces potential countermeasures designed to prevent adversaries from fingerprinting a permissioned blockchain network and discuss various aspects of gShock.

Multi-path routing. An effective protective measure involves limiting adversaries' access to sufficient traffic, thereby making it more difficult for them to construct an accurate network fingerprint of the blockchain. Fingerprinting becomes significantly more challenging when adversaries can only capture a minimal subset of the transmitted packets. Inspired by pioneering Internet frameworks like SCION [69],

multi-path routing emerges as a robust deterrent against attempts to map the blockchain network. By employing multi-path routing, blockchain traffic shared between any two points would traverse a diverse array of network routes, leveraging existing AS agreements. The key advantage of this approach is its ability to reduce traffic exposure to adversaries without requiring changes to node operations or core protocols. However, the effectiveness of this strategy depends on the assumption that adversaries have limited access to network traffic from multiple locations.

Obfuscation-based methods. Obfuscation-based strategies aim to suppress or significantly minimize the information adversaries can extract from communication patterns, even when encrypted. Common tactics include introducing delays to network packets, packet slicing, packet padding, or generating decoy packets [70]—all intended to disrupt the traffic patterns that facilitate fingerprinting. However, these methods come with their own set of challenges. For example, timely traffic exchange is essential for tasks like synchronization or membership validation, so introducing packet delays can compromise network efficiency, reliability, and security. Additionally, padding and dummy packet generation increase communication overhead, making them less attractive for practical use. Therefore, there is a growing interest in developing more streamlined obfuscation techniques that avoid these common issues, such as network performance degradation or bandwidth wastage [71], particularly when countering fingerprinting attacks like those posed by gShock.

Inefficiency of deploying fake nodes. Another potential countermeasure involves using a group of *fake nodes* (e.g., fake orderers and peers) to obscure important nodes and links, such as leader orderers, by generating large amounts of traffic—similar to obfuscation methods. While this approach is relatively easy to implement in any permissioned blockchain network, it incurs significant costs, as a large number of nodes would be needed to effectively mask the real ones. Additionally, this method does not address the unique time-series patterns exhibited by peer and orderer nodes, leaving them vulnerable to fingerprinting despite the added traffic.

Real-world applicability of gShock. While this paper focuses on Hyperledger Fabric as the target, gShock is flexible and can be applied to any permissioned blockchain network with a similar architecture. For example, Quorum [7], an Ethereum-based permissioned blockchain, uses a Raft consensus mechanism where peers assume leader or follower roles. A key advantage of gShock is that it does not require direct participation or detailed knowledge of the target blockchain's implementation. This flexibility allows gShock to be broadly adapted to fingerprint the internal architecture of Quorum, including identifying leader peers within the network. Moreover, gShock can be extended to different consensus mechanisms, provided it can train on sufficient blockchain traffic.

VII. RELATED WORK

Note that most previous studies have demonstrated the severity if unique traffic patterns are exposed to adversaries. Although the traffic is under SSL/TLS encryption, the threats can be applicable on both legacy network and blockchain networks.

Encrypted traffic analysis. Roei et al. [53] demonstrated that adversaries could perceive burst patterns of video streams to identify which video files are currently being accessed by users. Sun et al. [74] proposed a data-driven approach for passive fingerprinting of IoT devices based on the classification of encrypted SSL/TLS flows generated from several IoT devices. Apthorpe et al. [75] showed that analyzing the network traffic rate of smart home devices could reveal sensitive user interactions even when the traffic is encrypted. Some studies [65], [76], [77] de-anonymized end-hosts from encrypted traffic such as Tor. These prior researches motivate our approach and show that even though SDN control channels are secured by SSL/TLS, they can be in danger if their unique patterns are exposed.

Fingerprinting blockchain networks. Recent advancements in machine learning have developed to uncover confidential information through the analysis of blockchain data. Delgado *et al.* [9] posited that by examining orphan transactions, one could deduce the topology of the Bitcoin network, a premise that opens up new avenues for understanding the underlying structure of blockchain networks. Similarly, Shen *et al.* [13] proposed a method of feature fusion to improve the classification accuracy of algorithms such as KNN, SVM, and RF in the context of fingerprinting decentralized applications (DApps) on Ethereum, thereby enhancing the precision of application identification on the blockchain. Michalski *et al.* [12] presented a technique aimed at identifying the roles of nodes within the Bitcoin network, further enriching our understanding of network functionalities and participant interactions.

The scope of these studies extends into the realm of network security and privacy, with Rezaei *et al.* [14] developing classifiers capable of distinguishing Bitcoin traffic that is tunneled through encrypted channels. Wang *et al.* [15] demonstrated the efficacy of their proposed clustering-based quadruplet network (CQNet) in achieving high accuracy classification of 60 DApps, marking a significant stride in the application of machine learning to blockchain analysis. Shen *et al.* [10] introduced a methodology leveraging Graph Neural Networks (GNN) to infer identities within blockchain networks. A similar method was furthered by Shen *et al.* [16], who explored the use of GNN-based technique for identifying DApps visited by users from encrypted traffic. Wang *et al.* [17] illustrated the potential for identifying user behavior on DApps through encrypted traffic analysis. Zhou *et al.* [11] proposed a framework designed to learn behavioral patterns of Ethereum accounts, aiming at de-anonymizing the Ethereum network. Recently, Du *et al.* [72] proposed a graph-based framework (MixBroker) that leverages GNN to break the user-level anonymity of Ethereum mixing services, by correlating transaction addresses. Sun *et al.* [73] presented a

TABLE 3. Comparison of gShock with previous studies.
(SVM: Support Vector Machine, NN: Neural Network, DT: Decision Tree, RF: Random Forest, ET: Extra Trees, LR: Logistic Regression, KNN: K-Nearest Neighbor, LSTM: Long-Short Term Memory, GNN: Graph Neural Network)

Previous Studies	Target Blockchain Network	Classification Method	Input Dataset	Inferring Network Topology	Revealing Node Roles
Delgado-Segura *et al.* [9] (2019)	Bitcoin (Permissionless)	Orphan transaction analysis	Transaction	✓	✗
Shen *et al.* [13] (2019)	Ethereum (Permissionless)	KNN, SVM, RF	Encrypted Traffic	✗	✗
Michalski *et al.* [12] (2020)	Bitcoin (Permissionless)	SVM, NN, DT, RF, ET, LR, KNN	Address	✗	✓
Rezaei *et al.* [14] (2020)	Bitcoin (Permissionless)	NN	Encrypted traffic	✗	✗
Wang *et al*. [15] (2021)	Ethereum (Permissionless)	Clustering-based quadruplet network	Encrypted traffic	✗	✗
Shen *et al*. [10] (2021)	Ethereum and EOSIO (Permissionless)	GNN	Transaction	✗	✗
Shen *et al*. [16] (2021)	Ethereum (Permissionless)	GNN	Encrypted Traffic	✗	✗
Wang *et al*. [17] (2021)	Ethereum (Permissionless)	RF	Encrypted Traffic	✗	✗
Zhou *et al*. [11] (2022)	Ethereum (Permissionless)	GNN	Transaction	✗	✗
Du *et al*. [72] (2023)	Ethereum (Permissionless)	GNN	Transaction	✗	✗
Sun *et al*. [73] (2024)	Ethereum (Permissionless)	GNN	Transaction	✗	✗
gShock (our work)	Hyperledger Fabric (Permissioned)	LSTM+GNN	Encrypted Traffic	✓	✓

GNN-based phishing node detection by applying the adaptive attention convolution to the Ethereum transaction graph.

Overall, prior studies underscore an increasing focus on comprehending user-level behavior and interaction patterns within blockchain ecosystems. However, it is noteworthy that the majority of these studies have concentrated on inferring application and user-related information, often overlooking critical architectural aspects such as the topology and node roles within the networks. Furthermore, there appears to be a gap in research specifically targeting permissioned blockchain networks, which differ significantly from their public counterparts in terms of governance, privacy, and participant management. Table 3 summarizes the analysis on the comparison of previous works and ours.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we introduce gShock, a novel traffic analysis system designed to discern confidential information within blockchain networks. This system takes a mixture of encrypted traffic, capturing the intricate characteristics innate to specialized blockchain operations by leveraging a Graph Neural Network (GNN)-based model. We evaluate the efficacy of gShock in environments fortified with multiple defense mechanisms, offering a comparative assessment of its resilience and effectiveness. The results reveal that gShock can classify blockchain operational traffic with an F-1 score of $\geq$ 96% and identifying individual dependencies with a macro F-1 score of $\geq$ 93%. We posit that this research offers valuable insights into blockchain security, equipping operators with foresight regarding potential system vulnerabilities, thereby enabling proactive risk mitigation.

While gShock can effectively disclose blockchain operational patterns from encrypted traffic, there remains much room for further enhancements. First, the system could be extended to support a broader range of distributed consensus protocols [31], [78], [79]. Currently, gShock is tailored to the Raft protocol [31], which is one of the most widely used consensus protocols in permissioned blockchain networks. To achieve a more comprehensive traffic analysis, it is essential that the proposed system be adapted to support various consensus protocols. We further include an additional phase between the current phases 1 and 2. This new phase would analyze blockchain traffic to identify the underlying consensus protocols employed by the target blockchain network by capturing the unique characteristics inherent to each protocol.

Second, the system's capability for real-time traffic analysis requires further improvement. Our current design employs a GNN-based model that maps the target blockchain network

topology into a set of graphs. While this approach effectively captures hidden traffic patterns within encrypted blockchain traffic, it is not suitable for real-time analysis in large-scale blockchain networks due to the increased complexity in the GNN model. This is due to the fact that the computational time of a GNN model increases with the model's complexity. Although the model analysis is a one-time process, this limitation hinders real-time analysis, particularly when the target network undergoes frequent changes during runtime. To address this limitation, we further adopt various graph optimization techniques, such as node aggregation [80] and sampling [81], into our GNN model. These optimizations aim to reduce the model's complexity and overall analysis time, thereby enabling more agile analysis even when the network topology is subject to frequent modifications.

ACKNOWLEDGEMENT

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00118, Development of decentralized consensus composition technology for large-scale nodes, 70%), and the Research Grant of Kwangwoon University in 2024 (30%).

REFERENCES

[1] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Communications of the ACM*, vol. 61, no. 7, pp. 95–102, 2018.

[2] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. " O'Reilly Media, Inc.", 2014.

[3] J. R. Douceur, "The sybil attack," in *International workshop on peer-to-peer systems*. Springer, 2002, pp. 251–260.

[4] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *24th USENIX security symposium (USENIX security 15)*, 2015, pp. 129–144.

[5] D. Schwartz, N. Youngs, A. Britto *et al.*, "The Ripple Protocol Consensus Algorithm," *Ripple Labs Inc White Paper*, vol. 5, no. 8, p. 151, 2014.

[6] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger Fabric: a Distributed Operating System for Permissioned Blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[7] "Consensys Quorum," https://github.com/Consensys/quorum.

[8] "R3 Corda," https://github.com/corda/corda.

[9] S. Delgado-Segura, S. Bakshi, C. Pérez-Solà, J. Litton, A. Pachulski, A. Miller, and B. Bhattacharjee, "Txprobe: Discovering bitcoin's network topology using orphan transactions," in *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*. Springer, 2019, pp. 550–566.

[10] J. Shen, J. Zhou, Y. Xie, S. Yu, and Q. Xuan, "Identity inference on blockchain using graph neural network," in *Blockchain and Trustworthy Systems: Third International Conference, BlockSys 2021, Guangzhou, China, August 5–6, 2021, Revised Selected Papers 3*. Springer, 2021, pp. 3–17.

[11] J. Zhou, C. Hu, J. Chi, J. Wu, M. Shen, and Q. Xuan, "Behavior-aware account de-anonymization on ethereum interaction graph," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3433–3448, 2022.

[12] R. Michalski, D. Dziubałtowska, and P. Macek, "Revealing the character of nodes in a blockchain with supervised learning," *Ieee Access*, vol. 8, pp. 109 639–109 647, 2020.

[13] M. Shen, J. Zhang, L. Zhu, K. Xu, X. Du, and Y. Liu, "Encrypted traffic classification of decentralized applications on ethereum using feature fusion," in *Proceedings of the International Symposium on Quality of Service*, 2019, pp. 1–10.

[14] F. Rezaei, S. Naseri, I. Eyal, and A. Houmansadr, "The Bitcoin Hunter: Detecting Bitcoin Traffic over Encrypted Channels," in *Security and Privacy in Communication Networks: 16th EAI International Conference, SecureComm 2020, Washington, DC, USA, October 21-23, 2020, Proceedings, Part I 16*. Springer, 2020, pp. 152–171.

[15] Y. Wang, G. Xiong, C. Liu, Z. Li, M. Cui, and G. Gou, "Cqnet: A clustering-based quadruplet network for decentralized application classification via encrypted traffic," in *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part IV 21*. Springer, 2021, pp. 518–534.

[16] M. Shen, J. Zhang, L. Zhu, K. Xu, and X. Du, "Accurate decentralized application identification via encrypted traffic analysis using graph neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2367–2380, 2021.

[17] Y. Wang, C. Wang, G. Xiong, and Z. Li, "Multi-scene classification of blockchain encrypted traffic," in *Blockchain and Trustworthy Systems: Third International Conference, BlockSys 2021, Guangzhou, China, August 5–6, 2021, Revised Selected Papers 3*. Springer, 2021, pp. 329–337.

[18] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in *Neural Computation*, 1997.

[19] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv preprint arXiv:1609.02907*, 2016.

[20] M. Seo, J. Kim, E. Marin, M. You, T. Park, S. Lee, S. Shin, and J. Kim, "Heimdallr: Fingerprinting SD-WAN Control-Plane Architecture via Encrypted Control Traffic," in *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022, pp. 949–963.

[21] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, 2008.

[22] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, pp. 2–1, 2014.

[23] "Hyperledger: Advancing business blockchain adoption through global open source collaboration," https://www.hyperledger.org/, 2021.

[24] P. Helebrandt, M. Bellus, M. Ries, I. Kotuliak, and V. Khilenko, "Blockchain adoption for monitoring and management of enterprise networks," in *2018 IEEE 9th annual information technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 2018, pp. 1221–1225.

[25] D. Li, W. E. Wong, and J. Guo, "A survey on blockchain for enterprise using hyperledger fabric and composer," in *2019 6th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, 2020, pp. 71–80.

[26] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proceedings of the USENIX Annual Technical Conference*. USENIX, 2014.

[27] M. S. Kang, S. B. Lee, and V. D. Gligor, "The Crossfire Attack," in *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 2013.

[28] A. Studer and A. Perrig, "In Search of an Understandable Consensus Algorithm," in *Proceedings of the European Symposium on Research in Computer Security*. ESORICS, 2009.

[29] X. Feng, C. Fu, Q. Li, K. Sun, and K. Xu, "Off-path TCP Exploits of the Mixed IPID Assignment," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[30] Y. Gilad and A. Herzberg, "Off-path TCP injection attacks," *ACM Transactions on Information and System Security (TISSEC)*, vol. 16, no. 4, pp. 1–32, 2014.

[31] Y. Zhang, E. Ramadan, H. Mekky, and Z.-L. Zhang, "When Raft Meets SDN: How to Elect a Leader and Reach Consensus in an Unruly Network," in *Proceedings of the First Asia-Pacific Workshop on Networking*, 2017.

[32] C. Scott, V. Brajkovic, G. Necula, A. Krishnamurthy, and S. Shenker, "Minimizing Faulty Executions of Distributed Systems," in *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation*. USENIX, 2016, pp. 291–309.

[33] V. Ribeiro, R. Holanda, A. Ramos, and J. J. Rodrigues, "Enhancing key management in lorawan with permissioned blockchain," *Sensors*, vol. 20, no. 11, p. 3068, 2020.

[34] "BlockChain and Networking." https://www.rad.com/blog/blockchain-and-networking, 2018.

[35] W. Braun and M. Menth, "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices," *Future Internet*, vol. 6, pp. 302–336, 05 2014.

[36] P. Sköldström and K. Yedavalli, "Network virtualization and resource allocation in openflow-based wide area networks," in *2012 IEEE International Conference on Communications (ICC)*. IEEE, 2012, pp. 6622–6626.

[37] T. Zhang, P. Giaccone, A. Bianco, and S. De Domenico, "The role of the inter-controller consensus in the placement of distributed sdn controllers," *Computer Communications*, vol. 113, pp. 1–13, 2017.

[38] A. D. Ferguson, S. Gribble, C.-Y. Hong, C. Killian, W. Mohsin, H. Muehe, J. Ong, L. Poutievski, A. Singh, L. Vicisano *et al.*, "Orion: Google's {Software-Defined} networking control plane," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, 2021, pp. 83–98.

[39] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A systematic approach to developing and evaluating website fingerprinting defenses," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 227–238.

[40] T. Wang and I. Goldberg, "On realistically attacking tor with website fingerprinting," *Proceedings on Privacy Enhancing Technologies*, 2016.

[41] F. Lindner, "Cisco IOS Router Exploitation," *Black Hat USA*, 2009.

[42] K. Thimmaraju, B. Shastry, T. Fiebig, F. Hetzelt, J.-P. Seifert, A. Feldmann, and S. Schmid, "Taking control of sdn-based cloud systems via the data plane," in *Proceedings of the Symposium on SDN Research*, 2018, pp. 1–15.

[43] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1135–1156, 2013.

[44] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE network*, vol. 26, no. 1, pp. 35–40, 2012.

[45] B. Mao, Z. M. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "Routing or Computing? The Paradigm Shift Towards Intelligent Computer Network Packet Transmission Based on Deep Learning," *IEEE Transactions on Computers*, vol. 66, no. 11, pp. 1946–1960, 2017.

[46] B. Mao, F. Tang, Z. M. Fadlullah, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "A Novel Non-supervised Deep-learning-based Network Traffic Control Method for Software Defined Wireless Networks," *IEEE Wireless Communications*, vol. 25, no. 4, pp. 74–81, 2018.

[47] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE international conference on intelligence and security informatics (ISI)*. IEEE, 2017, pp. 43–48.

[48] H. Anttila, "Measurement and Analysis of Youtube Traffic Profile And Energy Usage With LTE Drx Mode," in *Tampereen Teknillinen Yliopisto Tampere University of Technology*, 2016.

[49] R. Coppola and M. Morisio, "Connected Car: Technologies, Issues, Future Trends," *ACM Computing Surveys (CSUR)*, vol. 49, no. 3, pp. 1–36, 2016.

[50] J. Cao, Z. Yang, K. Sun, Q. Li, M. Xu, and P. Han, "Fingerprinting SDN Applications via Encrypted Control Traffic," in *Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses*, 2019.

[51] Y. Guo and W. Xue, "Probabilistic Multi-Label Classification with Sparse Feature Learning." in *IJCAI*, 2013, pp. 1373–1379.

[52] D. Blakely, J. Lanchantin, and Y. Qi, "Time and space complexity of graph convolutional networks," *Accessed on: Dec*, vol. 31, 2021.

[53] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the Burst: Remote Identification of Encrypted Video Streams," in *Proceedings of the USENIX Security Symposium*. USENIX, 2017.

[54] M. Nasr, A. Bahramali, and A. Houmansadr, "Deepcorr: Strong flow correlation attacks on tor using deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1962–1976.

[55] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013.

[56] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks," *arXiv preprint arXiv:1710.10903*, 2017.

[57] J. Kim, M. Song, M. Seo, Y. Jin, and S. Shin, "PassREfinder: Credential Stuffing Risk Prediction by Representing Password Reuse between Websites on a Graph," in *2024 IEEE Symposium on Security and Privacy (SP)*, 2024.

[58] S. Peng, J. Nie, X. Shu, Z. Ruan, L. Wang, Y. Sheng, and Q. Xuan, "A Multi-view Framework for BGP Anomaly Detection via Graph Attention Network," *Computer Networks*, vol. 214, p. 109129, 2022.

[59] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song, "Heterogeneous Graph Neural Networks for Malicious Account Detection," in *Proceedings of the 27th ACM international conference on information and knowledge management*, 2018, pp. 2077–2085.

[60] B. Singh, "Investigation on Impact of Feature Normalization Techniques on Classifier's Performance in Breast Tumor Classification," in *International Journal of Computer Applications*, 2015.

[61] "Docker," https://www.docker.com/, 2016.

[62] "CAIDA Passive Monitor: Chicago B." https://www.caida.org/data/passive/trace_stats/chicago-B/2014/?monitor=20140320-130000.UTC, 2014.

[63] "CAIDA Passive Monitor: Chicago B." https://www.caida.org/data/passive/trace_stats/chicago-B/2015/?monitor=20150219-130000.UTC, 2015.

[64] "CAIDA Passive Monitor: Chicago B." https://www.caida.org/data/passive/trace_stats/chicago-B/2016/?monitor=20160121-130000.UTC, 2016.

[65] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective Attacks and Provable Defenses for Website Fingerprinting," in *Proceedings of the USENIX Security Symposium*. USENIX, 2014.

[66] J. Gong and T. Wang, "Zero-delay lightweight defenses against website fingerprinting," in *Proceedings of 29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 717–734.

[67] T. Wang and I. Goldberg, "Walkie-Talkie: An Efficient Defense Against Passive Website Fingerprinting Attacks," in *Proceedings of the 26th USENIX Security Symposium*, 2017.

[68] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, 2010.

[69] D. Barrera, L. Chuat, A. Perrig, R. M. Reischuk, and P. Szalachowski, "The SCION Internet Architecture," *Commun. ACM*, vol. 60, no. 6, pp. 56–65, 2017.

[70] R. Meier, V. Lenders, and L. Vanbever, "ditto: Wan traffic obfuscation at line rate," in *NDSS Symposium 2022*, 2022.

[71] M. Seo, M. You, T. Park, S. Shin, and J. Kim, "Poster: Towards a Secure and Practical System to Obfuscate Tor Network Traffic," in *2023 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2023, pp. 8–10.

[72] H. Du, Z. Che, M. Shen, L. Zhu, and J. Hu, "Breaking the anonymity of ethereum mixing services using graph feature learning," *IEEE Transactions on Information Forensics and Security*, 2023.

[73] H. Sun, Z. Liu, S. Wang, and H. Wang, "Adaptive attention-based graph representation learning to detect phishing accounts on the ethereum blockchain," *IEEE Transactions on Network Science and Engineering*, 2024.

[74] J. Sun, K. Sun, and C. Shenefiel, "Automated IoT Device Fingerprinting Through Encrypted Stream Classification," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2019.

[75] N. Apthorpe, D. Reisman, and N. Feamster, "A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic," *arXiv preprint arXiv:1705.06805*, 2017.

[76] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 605–616.

[77] K. Al-Naami, S. Chandra, A. Mustafa, L. Khan, Z. Lin, K. Hamlen, and B. Thuraisingham, "Adaptive encrypted traffic fingerprinting with bi-directional dependence," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016.

[78] R. De Prisco, B. Lampson, and N. Lynch, "Revisiting the paxos algorithm," *Theoretical Computer Science*, vol. 243, no. 1-2, pp. 35–91, 2000.

[79] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[80] K.-H. Lai, D. Zha, K. Zhou, and X. Hu, "Policy-gnn: Aggregation optimization for graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 461–471.

[81] Y. Bai, C. Li, Z. Lin, Y. Wu, Y. Miao, Y. Liu, and Y. Xu, "Efficient data loader for fast sampling-based gnn training on large graphs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 10, pp. 2541–2556, 2021.

MINJAE SEO is a Researcher at ETRI, Daejeon, South Korea. He received his M.S. degree from the Graduate School of Information Security at KAIST and his B.S. degree in Computer Engineering from Mississippi State University. His current research interests include Software-defined networking security, network fingerprinting, and deep learning-based network system.

JAEHAN KIM is a PhD candidate in the School of Electrical Engineering at KAIST. He received his B.S. degree and M.S. degree in School of Electrical Engineering at KAIST. His current research interests mainly focus on Cyber security, Machine learning Security, Large language model and Data mining.

MYOUNGSUNG YOU is a PhD candidate in the School of Electrical Engineering at KAIST. He received his M.S. degree from the Graduate School of Information Security at KAIST and his B.S. degree in Computer Science from Chungbuk National University. His research interests include programmable network data planes, cloud security, and distributed systems.

SEUNGWON SHIN is an Associate Professor in the School of Electrical Engineering at KAIST and an Executive Vice President at Samsung Electronics. He received his Ph.D. in Computer Engineering from the Electrical and Computer Engineering Department at Texas A&M University and his M.S. degree and B.S. degree from KAIST, both in Electrical and Computer Engineering. His research interests span the areas of Software-defined networking security, DarkWeb analysis, and Cyber threat intelligence.

JINWOO KIM is an Assistant Professor in the School of Software at Kwangwoon University, Seoul, South Korea. He received his Ph.D. from the School of Electrical Engineering at KAIST, his M.S. degree from the Graduate School of Information Security at KAIST, and his B.S. degree from Chungnam National University in Computer Science and Engineering. His research topic focuses on investigating security issues with software-defined networks and cloud systems.