Collaborative Learning for Cyberattack Detection in Blockchain Networks

Tran Viet Khoa, Do Hai Son, Dinh Thai Hoang, Nguyen Linh Trung, Tran Thi Thuy Quynh, Diep N. Nguyen, Nguyen Viet Ha, and Eryk Dutkiewicz.

Abstract—**This article aims to study intrusion attacks and then develop a novel cyberattack detection framework to detect cyberattacks at the network layer (e.g., Brute Password and Flooding of Transactions) of blockchain networks. Specifically, we first design and implement a blockchain network in our laboratory. This blockchain network will serve two purposes, i.e., to generate the real traffic data (including both normal data and attack data) for our learning models and to implement real-time experiments to evaluate the performance of our proposed intrusion detection framework. To the best of our knowledge, this is the first dataset that is synthesized in a laboratory for cyberattacks in a blockchain network. We then propose a novel collaborative learning model that allows efficient deployment in the blockchain network to detect attacks. The main idea of the proposed learning model is to enable blockchain nodes to actively collect data, learn the knowledge from data using the Deep Belief Network, and then share the knowledge learned from its data with other blockchain nodes in the network. In this way, we can not only leverage the knowledge from all the nodes in the network but also do not need to gather all raw data for training at a centralized node like conventional centralized learning solutions. Such a framework can also avoid the risk of exposing local data's privacy as well as excessive network overhead/congestion. Both intensive simulations and real-time experiments clearly show that our proposed intrusion detection framework can achieve an accuracy of up to 98.6% in detecting attacks.**

Index Terms—**Blockchain, deep learning, collaborative learning, cyberattack detection, intrusion detection.**

I. Introduction

This work is the output of the ASEAN IVO http://www.nict.go.jp/en/asean_ivo/index.html project "Cyber-Attack Detection and Information Security for Industry 4.0" and financially supported by NICT http://www.nict.go.jp/en/index.html. This research was supported in part by the Australian Research Council under the DECRA project DE210100651. This work was supported in part by the Joint Technology and Innovation Research Centre – a partnership between the University of Technology Sydney (UTS) and the University of Engineering and Technology, Vietnam National University, Hanoi.

T. V. Khoa is with the School of Electrical and Data Engineering, University of Technology Sydney (UTS), Sydney, Australia and the Advanced Institute of Engineering and Technology (AVITECH), University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam (e-mail: khoa.v.tran@student.uts.edu.au, khoatv.uet@vnu.edu.vn).

D. H. Son is with the VNU Information Technology Institute, Hanoi, Vietnam and the Advanced Institute of Engineering and Technology (AVITECH), University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam (e-mail: dohaison1998@vnu.edu.vn).

D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz are with the School of Electrical and Data Engineering, University of Technology Sydney (UTS), Sydney, Australia (e-mail: {hoang.dinh, diep.nguyen, eryk.dutkiewicz}@uts.edu.au).

N. L. Trung (corresponding author), T. T. T. Quynh, and N. V. Ha are respectively with the Advanced Institute of Engineering and Technology (AVITECH), the Faculty of Electronics and Telecommunications and the Insitute of Artificial Intelligence, at University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam (e-mail: {linhtrung, quynhttt, hanv}@vnu.edu.vn).

BLOCKCHAIN [1]–[3] has been emerging as a novel technology in storing and managing data with many advantages over conventional data management systems. In particular, unlike traditional centralized data management solutions, blockchain technology allows data to be stored in a distributed manner across multiple nodes. In this way, data can be accessed and processed simultaneously at multiple nodes, thus avoiding the problem of bottlenecks and single point of failure. More importantly, one of the most important features of blockchain technology is to enable data to be stored in blocks, and once a block of data is verified and placed in the chain, it cannot be modified and/or deleted. In this way, the data's integrity can be protected thanks to outstanding features of blockchain, e.g., decentralization, immutability, auditability, and fault tolerance [2]. As a result, there are more and more applications of blockchain technology in our lives including finance, healthcare, logistics, and IoT systems [2]–[5].

Due to the rapid success with a wide range of applications in most areas, especially in money transfer and cryptocurrency, blockchain-based systems have been becoming targets of many new-generation cyberattacks. For example, in September 2020, KuCoin, a crypto exchange based in Singapore, announced that its system was hacked and the hackers stole over $281 million worth of coins and tokens [6]. In May 2019, Binance, one of the biggest cryptocurrency exchange companies in the world, was reported to be hit by a major security incident. In particular, the hackers did break the exchange's security system and withdraw over 7,000 bitcoins from digital wallets, causing a total loss of approximately $40 million for the customers [6]. Most recently, in January 2022, Chainalysis reported that North Korean hackers performed seven attacks on cryptocurrency platforms and stole nearly $400 million from digital assets in 2021 [7]. Although most current attacks target on virtual money exchange systems, a number of blockchain applications in critical areas such as healthcare [8] and food supply chains [9] could be potential for attackers in the near future. If these attacks happen, they not only cause huge losses on our assets but can also lead to many serious issues related to human health and lives. Therefore, solutions to detect and prevent attacks in blockchain networks are becoming more urgent than ever.

In network security, authentication methods (e.g., two-factor authentication and biometric technology) are used to verify and identify authorized users. By implementing authentication methods, we can ensure that only authorized individuals or entities can gain access to their systems or resources. However, it is worth noting that authentication methods, despite their benefits, are inefficient in preventing attacks in blockchain networks. For example, authentication methods cannot detect and prevent Flooding of Transactions (FoT) and Brute Password (BP) attacks that are among the most common attacks in blockchain networks. In this case, intrusion detection systems can be employed to identify anomalous actions, including unauthorized commands or the execution of malicious scripts, even after a user has been authenticated. By incorporating intrusion detection systems alongside authentication methods, we can better protect the system's security against cyberattacks [10]. Several studies have been conducted to detect BP, FoT, or Man in the Middle (MitM) attacks on blockchain networks [11]–[13]. In [11], the authors performed BP attacks in their testbed with different devices (i.e., MacBook Pro, MacBook Air, Mobile phone, and Raspberry Pi). After that, they could detect BP attacks based on abnormally high memory and CPU consumption. In [12], the authors explored the ledger of Monero blockchain network over a period of one month. They analyzed network capacity, block size, portion of empty blocks, etc., to point out how attackers earn profit through an FoT attack. The authors in [13] proposed a method to detect MitM attacks based on blockchain technology for photovoltaic (PV) systems. They used a smart contract that stores control commands before and after transmission. They then compared these control command values to detect MitM attacks. However, these methods only focus on a specific type of attack or are applicable after attacks have already caused damage. Machine Learning (ML) has been being considered the most effective solution to detect cyberattacks in intrusion detection systems with high accuracies [14]–[16]. The main reasons for the outstanding advantages of using ML for intrusion detection problems compared with other conventional detection methods such as signature-based and abnormally-based are threefold. First, unlike conventional intrusion detection solutions which are usually designed to detect a specific type of attack (e.g., virus, trojan, spam, and botnet), ML solutions allow to detect many types of attacks at the same time with high accuracies. For example, Deep Learning (DL) allows to detect cyberattacks in industrial automation and control systems (e.g., denial of service, reconnaissance, naive malicious response injection, and complex malicious response injection) with an accuracy of up to 97.5% [17]. Second, while traditional solutions are often designed to detect known attacks, ML allows to detect attacks that have never been detected and reported before. For example, in [18], the authors showed that their DL model can detect attacks such as the DDoS attack of Mirai and BASHLITE botnets, even though such types of attacks have never been learned/trained before by this model. Last but not least, ML algorithms, especially DL, can be deployed effectively, quickly and flexibly. For example, after a deep neural network is trained, it can be deployed in different intrusion detection systems at the same time to detect cyberattacks quickly with high accuracies. In addition, when data about new types of attacks is available, we can easily update new versions of deep neural networks through transfer learning techniques [19].

As a result, ML has been being considered a highly-effective solution to detect cyberattacks for blockchain networks [20]. In particular, in [21], the authors proposed to use Random Forest and XGBoost to detect attacks in a blockchain-based IoT system. The results showed that this solution can identify different types of attacks and normal behaviors with an accuracy of up to 99%. However, they only tested their results on the BoT-IoT dataset that is not real blockchain traffic. Similarly, in [22], the authors proposed an ML-based method, called bidirectional long short-term memory (BiLSTM) to detect attacks in an IoT network before the data is stored in the blockchain network. Although the results also showed that they can detect different kinds of attacks with an accuracy of up to 99%, they were validated only on conventional network datasets such as UNSW-NB15 and BoT-IoT datasets. These datasets were collected in conventional computer networks and thus cannot reflect actual traffic in blockchain networks. In particular, these datasets have just general attacks in computer networks without specific attacks in blockchains, e.g., changes in blockchain transactions, incorrect consensus protocol, and breaking the chain of blocks.

To the best of our knowledge, there are only few works that consider using the real blockchain traffic, e.g., try to generate artificial data or try to create data to simulate an attack for blockchain networks to train ML models such as [23]–[25]. Specifically, in [23] the authors proposed a method to collect blockchain traffic data. First, they captured traffic samples from a public Bitcoin node and used them as the normal network data. Then, for the malicious traffic data, the authors performed DoS and Eclipse attacks on a target device (this device was created to become a node in the Bitcoin network). After that, the collected data was used to train an autoencoder deep learning model. This solution showed an accuracy of attack detection up to 99%. In [24], the authors used a public dataset and a private dataset from their testbed. Then, they proposed to use a Long Short-Term Memory Network (LSTM) to learn the properties of normal samples in the datasets. After that, they deployed a Condition Generative Adversarial Networks (CGAN) model to generate the artificial Low-rate Distributed DoS (LDDoS) attack samples for their blockchain dataset. The results showed the accuracy of classification up to 93%. In addition, in [25], the authors performed a DDoS attack namely Link Flood Attack (LFA) on a simulation Ethereum network and collected the traceroute records of the network in both normal and attack behaviors. After that, the authors used the Recurrent Neural Network (RNN) to analyze the traceroute records to identify the attacks in the network. The results showed that the attack detection rate can achieve nearly 99%. In [26], the authors developed a framework based on blockchain to detect only one specific attack, i.e., replay attacks, for a power system. Moreover, in [27], the authors proposed using blockchain and the Support Vector Machine (SVM) to detect cyberattacks for multimicrogrid systems. Thus, we develop a decentralized learning model that can detect different types of attacks (i.e., Denial of Service (DoS), BP, and FoT) for blockchain-based systems.

From all the above works and others in the literature, we can observe two main challenges for ML-based intrusion detection systems in blockchain networks which still have not been addressed. In particular, the first challenge is the lack of synthetic data from laboratories for training ML models. Most of the current works, e.g., [21] and [22] used conventional cybersecurity datasets (e.g., UNSW-NB15 and BoT-IoT) to train data. However, these datasets were not designed for blockchain networks, and thus they are not appropriate to use in intrusion detection systems in blockchain networks. Other works, e.g., [23]–[25], tried to build their own datasets for blockchain networks, e.g., by obtaining the normal samples from the Bitcoin network [23], creating simulation experiment to detect the LFA [25] and generating artificial attack samples by CGAN [24]. However, these methods have several issues. First, normal samples of transactions from the Bitcoin network may include attacks from the public blockchain network, but all collected data was classified and labeled to be normal data. Second, the simulation experiment in [25] was to generate traceroute records only for the LFA so they cannot extend to other attacks. Furthermore, it is difficult to evaluate the effects of artificial attack samples in [24] whether they can simulate a real attack on a blockchain network or not. Another challenge we can observe here is that all of the current ML-based intrusion detection solutions for blockchain networks are based on centralized learning models, i.e., all data is collected at a centralized node for training and detection. However, this solution is not suitable to deploy in blockchains as they are decentralized networks. Specifically, nodes in blockchain networks may have different data to train, and due to privacy concerns, they may not want to share their raw data with a centralized node (or other nodes) for training processes. It is noted that the raw data means the network traffic data of a local network. This data can be classified into normal or attack data, that will be used for the learning process. It usually contains sensitive information (e.g., cryptographic keys, usage ports, or local network bandwidth) that the node does not want to share with other nodes in the network. Moreover, sending a huge amount of data to the network will not only cause excessive network traffic but also risk compromising the data integrity of blockchain networks.

This article aims to address the aforementioned challenges by first introducing ***a novel intrusion detection dataset named BNaT which stands for Blockchain Network Attack Traffic***, created from a real blockchain network in our laboratory and then proposing an effective decentralized collaborative machine learning framework to detect intrusions in the blockchain network. Specifically, to develop BNaT, we first set up a blockchain network in our laboratory using Ethereum (an open-source blockchain software) and perform intensive experiments to generate blockchain data (including both normal and malicious traffic data). The main objectives of producing BNaT dataset are fourfold. Firstly, we collect the BNaT in a laboratory environment to have "clean" data samples (i.e., to ensure that the obtained data is not corrupted, error and/or irrelevant), which is especially important for training ML models. Secondly, the BNaT can be easily extended to include new kinds of blockchain attacks, e.g., 51% or double spending attacks. Thirdly, we perform experiments with real attacks in the considered blockchain network, and thus the BNaT can reflect better the actual attack behavior of the network than simulations or by artificial attack data generated by GAN in the literature, e.g., [24]. Fourthly, we collect the data in different blockchain nodes to have a complete view of the effects when the attacks are performed in a decentralized manner. After that, we develop a highly-effective collaborative learning model to make it more effective in deploying in blockchain networks to detect attacks. In particular, in our proposed learning model, working nodes in the blockchain network (e.g., fullnodes) can be used as learning nodes to collect blockchain network data (e.g., observing its incoming traffic and classifying data). Our proposed model aims to leverage knowledge learned from all the nodes in the network in a decentralized manner yet without revealing their raw data (i.e., training datasets with labels). To do so, we first design a framework in the decentralized blockchain network in which each participated learning node (i.e., fullnode in the blockchain network) deploys a deep learning model (we will explain more details in the next section) to learn from its collected data and then share its trained model with a Centralized Server (CS). The CS can be a bootnode or any fullnode in the blockchain network. The CS will then aggregate all the trained models and send the aggregated model (i.e., the global model) back to the participated learning nodes. By repeating this process, the learning nodes can gradually update their deep learning models and finally reach convergence (to the global training model). In this way, we can not only improve the accuracy of detecting cyberattacks in blockchain networks but also eliminate the risks of exposing local data of learning nodes over the network. Our proposed model can achieve an accuracy of up to 98.6% in detecting cyberattacks in the considered network. Moreover, in our proposed learning model, even though the nodes do not need to share their raw data, they still can learn useful information from other nodes in the network through extracting information from shared trained models. The main contributions of this paper can be summarized as follows.

- We set up experiments in our laboratory to build a private blockchain network with the aims of not only obtaining real blockchain datasets, but also testing our proposed learning model in a real-time manner. To the best of our knowledge, this is the first dataset obtained from a laboratory for studying cyberattacks in blockchain networks, and thus we expect that our proposed BNaT dataset can promote the development of ML-based intrusion detection solutions in blockchain networks in the near future. More details of the dataset can be found at the link [1].
- We build an effective tool named Blockchain Intrusion Detection (BC-ID) to collect data in the blockchain network. This tool can extract features from the collected network traffic data, filter attack samples in network traffic, and exactly label them in a real-time manner.
- We propose a collaborative decentralized learning model to not only improve the accuracy of identifying attacks, but also effectively deploy in decentralized blockchain networks. This model enables fullnodes in the blockchain network to effectively share their trained models to im-

[1]https://avitech-vnu.github.io/BNaT

prove cyberattack detection efficiency without the need of sharing their raw data.

- We perform both intensive simulations and real-time experiments to evaluate our proposed framework. Both simulation and experimental results clearly show the outperformance of our proposed framework compared with other baseline ML methods. Furthermore, our results reveal some important information in designing and implementing learning models in blockchain networks in practice, e.g., real-time monitoring and detecting attacks.

The rest of this paper is organized as follows. Section II provides fundamental backgrounds and our designed blockchain network together with the collaborative learning model. Section III presents our proposed collaborative learning model to detect cyberattacks in the blockchain network. The experiment setup, dataset collection, and evaluation method are described in Section IV. After that, the experimental results and performance evaluations are discussed more details in Section V. Finally, we conclude the paper in Section VI.

II. Blockchain Network: Fundamentals and Proposed Network Model

A. Blockchain

Blockchain is a digital ledger technology that provides a transparent, tamper-proof, and secure environment for transmitting data. This technology enables various parties to join, verify and record transactions without a trusted third party (e.g., a bank). In a blockchain network, multiple nodes are used to simultaneously process and store data. In particular, when a node in the blockchain network receives transactions (e.g., money exchange in the Bitcoin network), it will gather all the transactions and put them in a block. This node will then start a mining process to find a “nonce” value for this block. It is important to note that thanks to the feature of the hash function, there is only a small set of satisfying nonce values for a block, and these values can only be found through an intensive searching process [1]. This mining process is a special process of blockchain networks to provide proofs for validated blocks, and thus this tamper-proof can significantly enhance security for blockchain networks. After the node finds the nonce value for the mining block, this new block will be broadcast and verified by other nodes in the network. Finally, if this block is verified, it will be put into the chain (linked to the hash value of the previous block inside its header). After the block is added to the chain, it is nearly impossible to change information in this block, and thus this property can guarantee the immutability of the blockchain. Another aspect of blockchain is traceability due to the infeasible collision of the hash function, and thus any transaction or block can be tracked correctly. In summary, blockchain can be termed as a decentralization, immutable, traceable, and time-stamped digital data chain (ledger).

B. Designed Blockchain Network at our Laboratory

To launch a blockchain network, there are two main kinds of blockchain nodes namely fullnode and bootnode. Firstly, fullnodes take responsibility to store the ledger, participate

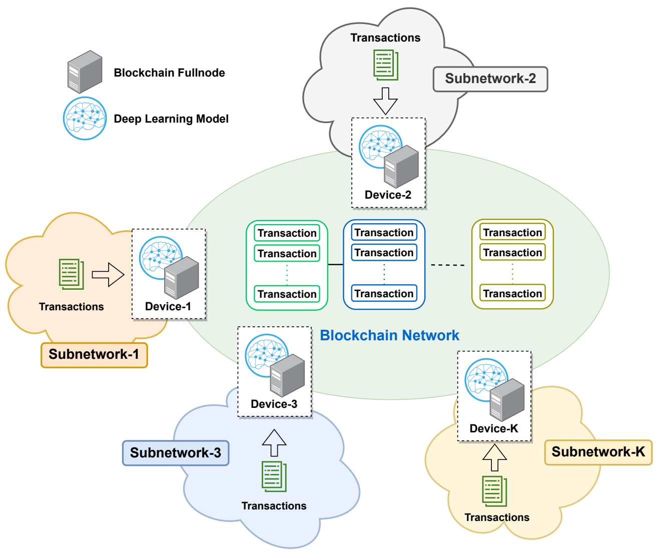


Fig. 1: Our proposed learning model for blockchain network.

in the mining process, and verify all blocks and states. Furthermore, they can be used to serve the network and provide data on request, e.g., netstats, which is a visual interface for tracking Ethereum network status (e.g., the block number, mining status, and the number of pending transactions). Secondly, bootnode is a lightweight application used for the Node Discovery Protocol. The bootnodes do not synchronize blockchain ledger but help other Ethereum nodes discover peers to set up Peer-to-Peer (P2P) connections in the network.

The system model together with essential components of our designed blockchain network is set up as illustrated in Fig. 1. Specifically, the system includes K fullnodes which are used to receive transactions, mining blocks, and keep the replica of ledger. These nodes continuously synchronize their ledgers together by the P2P protocol with equal permissions and responsibilities for processing data [1]. In order to connect them together, a management node, known as bootnode, is set up. The fullnodes connect and interrogate this bootnode for the location of potential peers in the blockchain network. After being connected, each fullnode can collect data (i.e., transactions) from its network. Transactions can come from different blockchain applications such as cryptocurrency, smart city, food supply chain, and IoT. First, when transactions are sent to a fullnode, they will be verified and packed into one block. After the node finds the nonce value for this block, it will broadcast the block together with this nonce value to other nodes in the network for verification. Finally, if the block is verified by majority of nodes in the network, it will be added to the chain.

At our laboratory, we design a private blockchain network based on the Ethereum blockchain network. This network also uses the Proof-of-Work (PoW) consensus mechanism, but the block confirmation time is significantly faster than the older version of Bitcoin. Furthermore, the smart contract layer of Ethereum is suitable for flexible purposes of decentralized

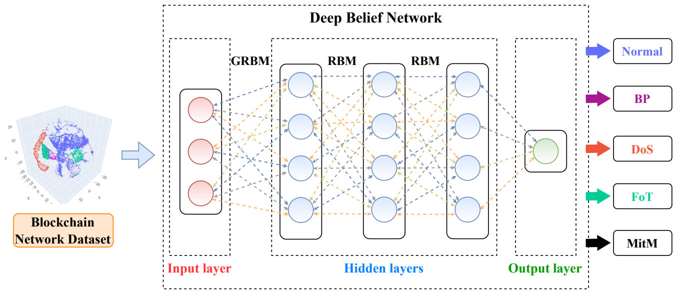


Fig. 2: The structure of classification-based for intrusion detection learning model in a blockchain network.

environments as mentioned above. In addition, at each node, several attacks, that can cause serious damage to the public blockchain network, will be considered. We then capture the traffic data to analyze their impacts on the blockchain network using BC-ID. Note that, in practice, there is no software that supports automatically capturing the blockchain network traffic so far. Thus, we analyze the blockchain network traffic data using a software named Wireshark [28] and build a new collection tool, namely BC-ID (more details will be explained in Section IV). In this way, we can observe the effects of these attacks on different nodes in the blockchain network.

III. Proposed Collaborative Learning Model for Intrusion Detection in Blockchain Network

Fig. 1 describes our proposed framework for intrusion detection in the blockchain network. In our proposed collaborative learning model, the fullnodes in the blockchain networks will be used as Learning Nodes (LNs) to learn knowledge from their collected data inside their subnetworks and share their learned knowledge to improve learning performance for the whole network. We also propose to use a deep neural network at each LN to learn useful information from its collected data. Then, the LNs will share their trained learning models with the CS. After that, the CS will calculate the aggregated model (i.e., the global model) and share this model back with the LNs. When a LN receives this aggregated model from the CS, it will integrate with its current LN and train its local dataset. This process will be repeated until convergence or reaching a predefined maximum number of iterations. In the end, we can obtain the global learning model for all the LNs.

In our proposed model, each blockchain node has a set of local collected data, and we propose a deep neural network (DNN) using Deep Belief Network (DBN) [29] to better learn knowledge from this data. The DBN is a type of deep neural network that is used as a generative model of both labeled and unlabeled data. Therefore, unlike other supervised deep neural networks which use labeled data to train the neural networks (e.g., convolutional neural networks [30]), the DBN has two stages in the training process. The first stage is the pre-training process where the DBN trains its neural network with unlabeled dataset. The second stage is the fine-tuning process where DBN uses labeled dataset to train its neural network. Thereby, the DBN can better represent the characteristics of the dataset, and thus it can classify the normal behavior and different types of attacks with high accuracies. In addition, the DBN includes multiple Restricted Boltzmann Machines (RBM) layers for latent representation [29]. In the DBN training process, the current layer generates latent representation by using latent representation of the previous layer as the input. Unlike other deep neural networks which also can process both labeled and unlabeled data (e.g., autoencoder deep learning network [30]), the DBN optimizes the energy function of each layer to have better latent representations of data on each RBM layer in each iteration. Thereby, the DBN is more appropriate to analyze the blockchain network traffic where the samples and features have relative coherence with each other.

The whole processes of DBN are illustrated in Fig. 2. Like other DNNs, the structure of DBN has three layers including an input layer, an output layer, and multiple hidden layers. As can be seen in Fig. 2, the Gaussian Restricted Boltzmann Machines (GRBM) layer, a type of RBM that can process real values of data, is the input layer to receive and transform the input data into binary values. We denote $k \in \{1, ..., K\}$ as the number of learning nodes in the collaborative learning model, $\boldsymbol{v}^k$ and $\boldsymbol{h}^k$ to be the vectors of visible and hidden layers of LN-k, respectively. In addition, M and N are the numbers of visible and hidden neurons of GRBM. We denote h_n^k and v_m^k as the hidden layer-n and visible layer-m of LN-k. As defined in [31], the energy function of GRBM of LN-k is calculated as follows:

$$\begin{aligned} E_G^k(\boldsymbol{v}^k, \boldsymbol{h}^k) = \sum_{m=1}^{M} \frac{(v_m^k - b_{1,m})^2}{2\epsilon_m^2} - \\ \sum_{m=1}^{M}\sum_{n=1}^{N} w_{m,n} h_n^k \frac{v_m^k}{\epsilon_m} - \sum_{n=1}^{N} b_{2,n} h_n^k, \end{aligned} \tag{1}$$

where $w_{m,n}$ is the weight between visible and hidden neurons; $b_{1,m}$ and $b_{2,n}$ indicate the bias of visible and hidden neurons, respectively; and ϵ_m represents the standard deviation of the neuron in the visible layer. From the result of equation (1), we can find the probability that is used in the visible layer of GRBM [31] as follows:

$$p_G^k(\boldsymbol{v}^k) = \frac{\sum_{\boldsymbol{h}^k} e^{-E_G(\boldsymbol{v}^k, \boldsymbol{h}^k)}}{\sum_{\boldsymbol{v}^k}\sum_{\boldsymbol{h}^k} e^{-E_G(\boldsymbol{v}^k, \boldsymbol{h}^k)}}. \tag{2}$$

Then, we use the probability in equation (2) to calculate the gradients of each GRBM layer with the expectation value $\langle . \rangle$ as follows [31]:

$$\begin{aligned} \nabla g_{G,m,n}^k &= \frac{\partial \log p_G^k(\boldsymbol{v}^k)}{\partial w_{m,n}} \\ &= \left\langle \frac{1}{\epsilon_m} v_m^k h_n^k \right\rangle_{dataset} - \left\langle \frac{1}{\epsilon_m} v_m^k h_n^k \right\rangle_{model}. \end{aligned} \tag{3}$$

Next, the gradient of GRBM layers can be calculated:

$$\nabla g_k^G = \sum_{m=1}^{M}\sum_{n=1}^{N} \nabla g_{G,m,n}^k \tag{4}$$

In the next stage, we need to calculate the energy function and the gradient of RBM layers. We denote M' and N' are the numbers of visible and hidden neurons of RBM layers. As defined in [31] the energy functions of RBM layer of LN-k

are defined as follows:

$$E^k_{RBM}(\boldsymbol{v}^k, \boldsymbol{h}^k) = -\sum_{m=1}^{M'} b_{1,m} v^k_m - \sum_{m=1}^{M'}\sum_{n=1}^{N'} w_{m,n} v_m h^k_n - \sum_{n=1}^{N'} b_{2,n} h^k_n, \quad (5)$$

The same as GRBM layers, we can calculate the gradient of each RBM layer as follows:

$$\nabla g^k_{R,m,n} = \left\langle v^k_m h^k_n \right\rangle_{dataset} - \left\langle v^k_m h^k_n \right\rangle_{model}. \quad (6)$$

And the gradient of RBM layers in LN-k:

$$\nabla g^R_k = \sum_{m=1}^{M'}\sum_{n=1}^{N'} \nabla g^k_{R,m,n} \quad (7)$$

After learning with multiple GRBM and RBM layers, we define $\boldsymbol{X}^{g,r}_k$ as the output of the last hidden layer of LN-k. In this paper, the output layer utilizes the softmax regression function to classify the data samples based on probability. We denote $\boldsymbol{W}^o$ and $\boldsymbol{b}^o$ as the weight matrix and bias vector between the output and the last hidden layer, respectively. We then can define the probability of the output Z belonging to Class-t as follows:

$$p^o_k(Z = t|\boldsymbol{X}^{g,r}_k, \boldsymbol{W}^o, \boldsymbol{b}^o) = softmax(\boldsymbol{W}^o, \boldsymbol{b}^o) \quad (8)$$

where $t \in \{1, .., T\}$ is a class of the output, and T refers to the total classes (including different types of attacks and normal behavior). The prediction $\boldsymbol{Z}_k$ of the probability p^o_k in LN-k can be calculated:

$$\boldsymbol{Z}_k = \underset{t}{\text{argmax}}[p^o_k(Z = t|\boldsymbol{X}^{g,r}_k, \boldsymbol{W}^o, \boldsymbol{b}^o)], \quad (9)$$

where Z is the output prediction. Then, we can calculate the gradient between the output layer and the last hidden layer from equation (8) as follows:

$$\nabla g^o_k = \frac{\partial p^o_k(Z = t|\boldsymbol{X}^{g,r}_k, \boldsymbol{W}^o, \boldsymbol{b}^o)}{\partial \boldsymbol{W}^o}. \quad (10)$$

After that, the results of equation (4), equation (7), and equation (10) are used to calculate the total gradient ∇g^t_k of DBN with multiple GRBM, RBM layers and the output layer of LN-k as follows:

$$\nabla g^t_k = \nabla g^G_k + \nabla g^R_k + \nabla g^o_k. \quad (11)$$

In the training process, the DBN first trains its neural network with unlabeled data for pre-training. Then, DBN uses its labeled data to fine-tune its neural network. At this stage, the DBN of LN-k calculates its gradient ∇g^t_k. After that, this gradient is sent to the CS to create an updated global model for all LNs as illustrated in Fig. 3. For example, at iteration i the CS receives gradients from all K LNs, the CS first performs the average gradient function [32] as follows:

$$\nabla g^* = \frac{1}{K}\sum_{k=1}^{K} \nabla g^t_k. \quad (12)$$

We then denote $\boldsymbol{\varphi}_i$ as the global model at iteration i which includes the weight matrix for all layers of the LN's deep learning model, and μ represents the learning rate. From the result of equation (12), the CS can update the global model at iteration $i + 1$ as follows:

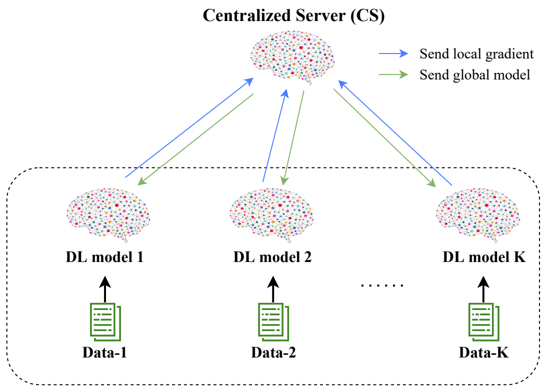


Fig. 3: The illustration of the collaborative learning between DL models and the CS.

$$\boldsymbol{\varphi}_{i+1} = \boldsymbol{\varphi}_i + \mu \nabla g^*. \quad (13)$$

Next, the CS sends the latest global model $\boldsymbol{\varphi}_{i+1}$ to the LNs to update their deep learning models. This process is repeated until it reaches convergence or gets the maximum number of iterations. At this time, we can find the optimal global model $\boldsymbol{\varphi}_{opt}$ that includes the optimal weights of all layers. We denote $\boldsymbol{W}^o_{opt}$ as the optimal weight matrix between the output layer and the last hidden layer from $\boldsymbol{\varphi}_{opt}$, the output prediction $\boldsymbol{Z}_k$ of LN-k thus can be calculated as follows:

$$\boldsymbol{Z}_k = \underset{t}{\text{argmax}}[p^o_k(Z = t|\boldsymbol{X}^{g,r}_k, \boldsymbol{W}^o_{opt}, \boldsymbol{b}^o)]. \quad (14)$$

Using equation (14), the softmax regression of each LN can classify its blockchain network samples to be a normal behavior or a type of attack. Algorithm 1 summarizes the process of our proposed collaborative learning model. In our proposed model, the learning model of each network can be trained by the dataset from its local network and exchange learning knowledge with those from other nodes in a blockchain network in an offline manner. In a practical blockchain network with a large number of learning nodes, we can schedule for nodes to exchange the learning knowledge in the offline training phase at appropriate times to avoid network congestion. In this way, each node can effectively learn knowledge from other nodes while avoiding the traffic congestion of the network. After the training process, the trained models can be used to help nodes to detect attacks in a real-time manner.

IV. Experiment Setup, Dataset Collection and Evaluation Method

This section will explain more details about experiment setup, data collection, and feature extraction over our designed blockchain system.

A. Experiment Setup

In our experiments, we set up an Ethereum blockchain network in our laboratory which includes three Ethereum

Algorithm 1 The classification-based collaborative learning algorithm

```
1: while i $\leq$ maximum number of iterations or the training process is not converged do
2:     for $\forall k \in K$ do
3:         DBN of LN-$k$ learns $\boldsymbol{X}_k$ and produces $\boldsymbol{Z}_k$.
4:         Calculate gradient $\nabla g_k^t$.
5:         Send $\nabla g_k^t$ to CS.
6:     end for
7:     CS calculates average gradient $\nabla g^*$ and global model $\boldsymbol{\varphi}_i$.
8:     $i = i + 1$.
9:     CS updates global model $\boldsymbol{\varphi}_{i+1}$.
10:    CS sends global model $\boldsymbol{\varphi}_{i+1}$ to all LNs.
11:    LNs update their DBNs.
12: end while
13: DBN of LNs predict and classify $\boldsymbol{Z}_k$ from the training dataset $\boldsymbol{X}_k$ with the optimal global model $\boldsymbol{\varphi}_{opt}$ .
```

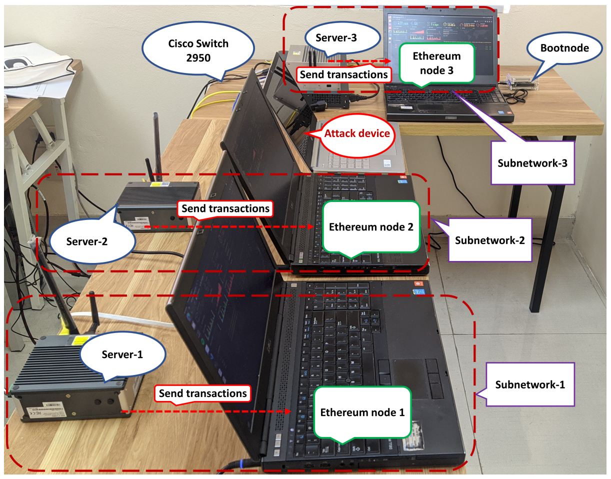


Fig. 4: Experiment setup.

fullnodes, an Ethereum bootnode, and a netstats server. All these nodes are connected to a Cisco Switch Catalyst 2950 as illustrated in Fig. 4. The details of these nodes are as follows:

- Ethereum fullnodes are launched by *Geth v1.10.14* [33] - open-source software for implementation of the Ethereum protocol. These nodes share the same initial configuration of genesis block, i.e., PoW consensus mechanism, 8,000,000 gas for block gas limit, initial difficulty 100,000. Each node is running on a personal computer with processor Intel® Core™ i7-4800MQ @2.7 GHz, RAM of 16 GB.
- Bootnode is also created by *Geth v1.10.14* and connected to the three Ethereum nodes.
- Ethereum netstats is launched by an open-source software named "eth-netstats" on Github [34].

The normal traffic data is configured with the three trustful servers, while an attack device will execute abnormal/malicious activities to the blockchain network traffic. Each trustful server takes responsibility for generating data and sending transactions to the corresponding Ethereum node in its subnetwork as visualized in Fig. 4. In summary, in the normal state, the following tasks are scheduled or randomly occur in the network:

- The servers are scheduled to send transactions.
- The users call functions in the deployed smart contracts to explore the ledger. Besides transaction-related functions, the users also send requests to the Ethereum nodes for tracking their balances or the status of miners. Both of these works are randomly made by HTTP requests to the Ethereum node API (Application Programming Interface).
- Ethereum nodes broadcast transactions and mined blocks to synchronize their ledgers. The packets of bootnode are also included in this field.
- WebSockets and JSON-RPC are used when netstats get information from *Geth* clients.
- HTTP requests and replies to view netstats interface and results of cyberattack detection.

B. Dataset Collection and Feature Extraction

In this section, we consider network layer aspects of the permissionless blockchain [35], [36] to detect cyberattacks in a blockchain network. In general, the goals of an adversary are usually the monetary benefit, e.g., chain splitting, and wallet theft, or stability of the network, e.g., delay and information loss. In this work, we focus on the attacks at the network layer. Attacks at the application layer, e.g., 51%, transaction malleability attacks, timejacking, and smart contract attacks, are out of the scope of this work and can be considered in future work. Specifically, we perform four typical types of network attacks that have been reported in blockchain networks, i.e., the BP for wallet theft; DoS and MitM for information loss; and FoT for consensus delay. These are the ubiquitous attacks in the network traffic layer that have caused a number of serious consequences for many years. More details are as follows:

- *Brute Password (BP) attack*: is derived from traditional cyberattack when hackers perform such attacks to steal blockchain users' accounts. In this way, the hackers can access the users' wallets and steal their digital assets. As mentioned in Section I, the BP attack on KuCoin caused the loss of up to \$281 million [6]. To perform this attack, the attacker retries passwords of an Ethereum public key until it finds out the correct login information.
- *Denial of Service (DoS) attack*: is also another common type of attack in blockchain networks as it can be easily performed to attack blockchain nodes. For such kind of attack, the attackers will launch a huge amount of traffic to a target blockchain node in a short period of time. Consequently, the target node will not be able to work as normal, i.e., mining transactions, and even be suspended. In the real-world, Bitfinex [37] was temporarily suspended due to such kind of attack. Thus, in our setup, a simple DoS attack is simulated, i.e., an SYN flood attack, by repeatedly sending initial connection request (SYN) packets to an Ethereum node.
- *Flooding of Transactions (FoT) attack*: targets delay the PoW blockchain network by spamming the blockchain

network with null or meaningless transactions. When the number of transactions per second in the Ethereum network suddenly hits the top, a mining node may face two following issues, i.e., too much traffic (similar as that of DoS), and the queue of pending transactions is full. It equates to the unnecessary time burden for mining process and block propagation [12]. In 2017, the Bitcoin mempool size was exceeded 115,000 unconfirmed transactions which led to $700 million worth of transaction stall [36]. In our work, FoT attack is implemented by continuously sending a large number of transactions to an existing smart contract.

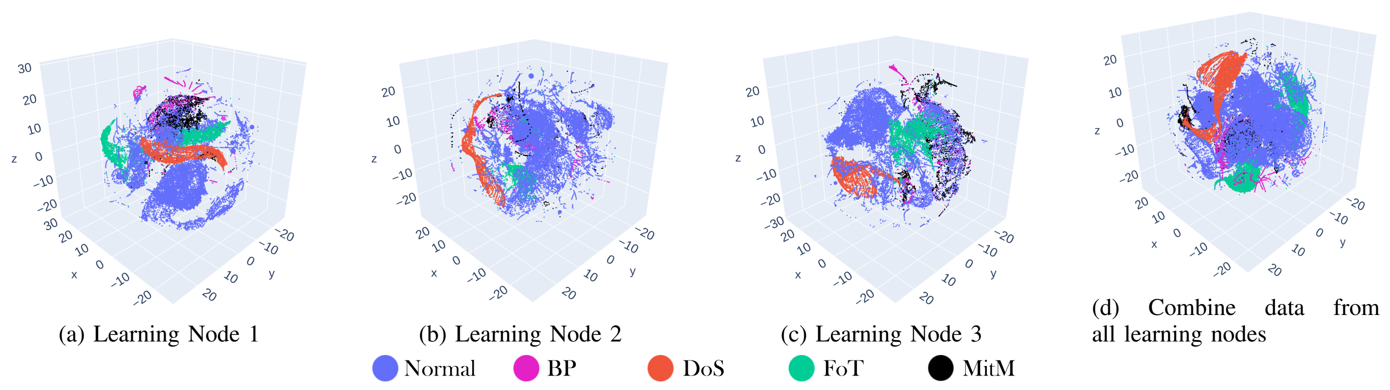
- *Man in the Middle (MitM) attack*: is another typical attack where an attacker places himself between the legitimate communicating parties and secretly relays and possibly modifies the information exchanged between them. In this way, the attacker can intercept, read, and modify the blockchain messages. For example, hackers can read the API messages between users and blockchain nodes to steal their wallet password [11]. To implement MitM, an attack device first filters '*eth_sendrawtransaction*' packets, which represent any transaction from users to blockchain nodes. Then, these packets' contents are randomly modified, leading to invalid transactions.

TABLE I: Features of the designed dataset.

#	Features name	T	Description
Basic features			
1	*duration*	C	length of the connection (seconds)
2	*protocol_type*	D	type of the protocol (i.e., tcp, udp, icmp)
3	*service*	D	network service (e.g., http, ssh, etc)
4	*src_bytes*	C	number of data bytes from source to destination
5	*dst_bytes*	C	number of data bytes from destination to source
6	*flag*	D	normal or error status of the connection
Statistical features			
Features refer to source IP-based Statistical			
7	*count*	C	number of connections to the same source IP as the current connection
8	*srv_count*	C	number of connections to the same service as the current connection
Features refer to these same source IP connections			
9	*serror_rate*	C	% of 'SYN' errors connections
10	*same_srv_rate*	C	% of same service connections
11	*diff_srv_rate*	C	% of different services connections
Features refer to these same service connections			
12	*srv_serror_rate*	C	% of 'SYN' errors connections
13	*srv_diff_host_rate*	C	% of different host connections
Features refer to destination IP-based Statistical			
14	*dst_host_count*	C	number of connections to the same destination IP as the current connection
15	*dst_host_srv_count*	C	number of connections to the same service as the current connection
Features refer to these same destination IP connections			
16	*dst_host_same_srv_rate*	C	% of same service connections
17	*dst_host_diff_srv_rate*	C	% of different services connections
18	*dst_host_same_src_port_rate*	C	% of same both source port and destination IP connections
19	*dst_host_serror_rate*	C	% of 'SYN' errors connections
Features refer to these same service connections			
20	*dst_host_srv_diff_host_rate*	C	% of different host connections
21	*dst_host_srv_serror_rate*	C	% of 'SYN' errors connections

In order to capture traffic data of these attacks, we build a dataset collection tool, named BC-ID, which inherits the core of an open-source utility named "kdd99_feature_extractor" [38] and our new designs to fit the considered Ethereum network, i.e., correct the service of packets related to Ethereum nodes, remove meaningless features, and automate label dataset samples based on some given properties. To do this, we first use the 'libpcap-dev' library of Linux to capture all the network data (including normal and different types of attacks) from the local network. Then, the BC-ID is used to extract features from the collected data, filter the attack samples, and label them as normal or a type of attack. In particular, the BC-ID starts by capturing raw traffic data based on 'libpcap-dev' package of Linux OS. Since each blockchain network has a few specific $ports$ for peer connections, client connections, and so on, BC-ID targets to filter and analyze traffic data in these ports. For example, the Ethereum blockchain network uses port 30303 for the TCP port listener, and port 8545 for JSON-RPC by default. As KDD99 dataset [39], BC-ID extracts features and then separates them into two categories, e.g, basic features (i.e., all the attributes can be extracted from a TCP/IP connection) and traffic features (i.e., statistics of packets with the same destination host or service in a window interval). Especially, our goal is to achieve a trained model that can be applied to our proposed real-time blockchain attack detection system, when the number of samples in a prediction frame is limited. Thus, the BC-ID collects the dataset frames in which each frame lasts for 2 seconds and extracts their features. The BC-ID then puts all collected data in this frame into a single file. Finally, we merge all single files together to make the full dataset. In summary, Table I shows 21 features in the designed dataset, which are separated into two types, i.e., discrete (D) and continuous (C). For continuous features, they are calculated in 2 seconds time window (similar to that of the famous KDD99 dataset [39]).

In each Ethereum node, the separated dataset is collected in five states (classes), i.e., normal state (Class-0), BP attack (Class-1), DoS attack (Class-2), FoT attack (Class-3), and MitM attack (Class-4). The normal state is captured in two hours, the rest of them in an hour through the designed BC-ID tool. As described above, when a node is attacked, the normal traffic still exists. Therefore, the attack samples can be filtered out by features-based two properties, i.e., the source and destination IP address of the attack device; *service*, *src_length*, and *dst_length* of the samples, which are analyzed by Wireshark [28]. To improve the diversity of the designed dataset, the normal traffic data in the attack state is mixed with traffic data in the normal state. In our experiments, a number of random samples in each state are selected to reduce the size of the bulk dataset as shown in Table II. In fact, we mix



(a) Learning Node 1 (b) Learning Node 2 (c) Learning Node 3 (d) Combine data from all learning nodes

Fig. 5: Visualization using *t*-SNE for collected datasets.

TABLE II: The number of samples in the designed dataset.

Ethereum node / Class	Node-1 (samples)	Node-2 (samples)	Node-3 (samples)
Normal	50,000	50,000	50,000
BP	5,000	5,000	5,000
DoS	5,000	5,000	5,000
FoT	5,000	5,000	5,000
MitM	5,000	5,000	5,000

normal traffic data in an equal ratio, i.e., 10,000 samples per normal state, normal traffic data at BP, DoS, FoT, and MitM, respectively.

Fig. 5 illustrates the visualization of our designed dataset using *t*-Distributed Stochastic Neighbor Embedding (*t*-SNE) [40] with three most important components. Although sharing the same configurations for *t*-SNE, the dataset of each LN has a different distribution in the output. In 3D view, the DoS and FoT attack samples show a fairly clear separation from normal state points. Otherwise, the BP and MitM attack samples collide with the normal state samples. This indicates that discriminating BP and MitM samples from the normal data points would be significantly challenging.

C. Evaluation Method

The confusion matrix with accuracy, precision, and recall proposed in [41] is widely used to evaluate the performance of machine learning algorithms. Let TP, FP, TN, and FN denote "True Positive", "False Positive", "True Negative", and "False Negative", respectively. The accuracy of the total system with T classes including normal behaviors and different types of attacks is as follows:

$$Accuracy = \frac{1}{T}\sum_{t=1}^{T}\frac{\mathrm{TP}_t + \mathrm{TN}_t}{\mathrm{TP}_t + \mathrm{TN}_t + \mathrm{FP}_t + \mathrm{FN}_t}. \quad (15)$$

The precision of class t is calculated as $P^t_{re} = \frac{TP_t}{TP_t + FP_t}$. In this paper, we use weighted average precision to evaluate the performance of the whole system. We denote S_t as the number of samples of class t and S as the number of samples of the whole dataset. The weighted average precision is calculated as follows:

$$Precision = \sum_{t=1}^{T}\frac{P^t_{re}\times S_t}{S}. \quad (16)$$

The recall of class t is calculated by $R^t_e = \frac{TP_t}{TP_t + FN_t}$. The weighted average recall that we use to calculate the performance of the total system is calculated as follows:

$$Recall = \sum_{t=1}^{T}\frac{R^t_e\times S_t}{S}. \quad (17)$$

In the next section, we also use accuracy, precision, recall to evaluate and compare the performance of our proposed Collaborative Learning model (proposed CoL) with two other baseline methods, i.e., Centralized Learning model (CeL) and Independent Learning model (IL).

V. Experimental Results and Performance Evaluation

In this section, we use the collected datasets of three nodes described in the aforementioned section for the corresponding LNs. The dataset of each LN is randomly split into training and testing dataset. All LNs use DBN with the same structure of neural network for learning and detecting attacks. However, the LNs have to work in different learning models and various scenarios. Each LN has itself training and testing dataset, and thus we can use these datasets to evaluate and compare the performance of the proposed CoL, the CeL and the IL in different scenarios.

A. Simulation Results

In this section, we present the simulation results with the dataset of LNs in different learning models. The details of datasets using for simulation are as follows:

- **Proposed Collaborative Learning Model (proposed CoL)**: Each LN learns its training dataset and performs collaborative learning with other LNs to generate the global model. Then, we use the global model to test the merged testing dataset of all participated LNs.
- **Centralized Learning Model (CeL)**: The centralized node (e.g., one of the mining node in the network) is assumed to be able to collect data from all the nodes in the network and train the deep learning model on

TABLE III: Simulation results.

Model	2 Learning Nodes (LNs)				3 Learning Nodes (LNs)				
	Proposed CoL	CeL	IL		Proposed CoL	CeL	IL		
			LN-1	LN-2			LN-1	LN-2	LN-3
Accuracy	**97.427**	97.330	97.036	96.865	**97.276**	97.270	96.827	96.731	96.825
Precision	**93.861**	93.620	92.793	92.000	**93.448**	93.249	92.209	91.343	92.798
Recall	**93.567**	93.324	92.590	92.162	**93.189**	93.176	92.067	91.829	92.063

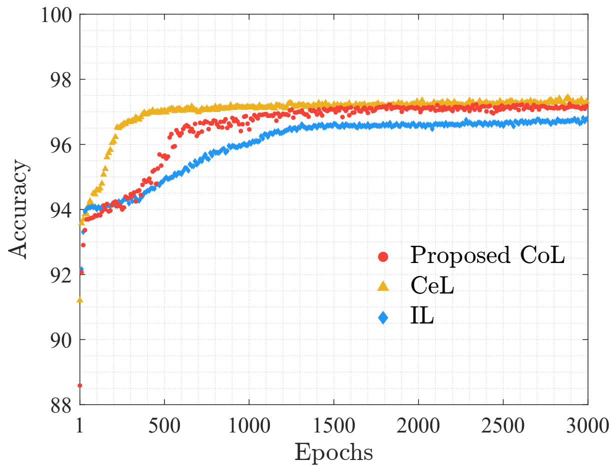


Fig. 6: Training process of considered learning models.

the collected datasets. Then, we use the trained model to test data based on the merged testing dataset of all participated LNs.

- **Independent Learning Model (IL)**: Each LN learns its training dataset without sharing knowledge with other LNs. Then, we use this model to test data based on the merged testing dataset of all participated LNs.

1) Convergence Analysis: Fig. 6 describes the convergence of the proposed CoL, the CeL and the IL (in terms of accuracy) of three LNs in the training process. The proposed CoL is obtained at the LN-1 after obtaining the global model. The CeL has a large number of training samples from three LNs so it can reach the convergence with around 97% accuracy after 400 epochs. Besides, the proposed CoL and IL converge after 800 epochs and 1300 epochs, respectively. After 3,000 learning epochs, we can observe that the proposed CoL has a higher accuracy compared with that of the IL (i.e., 97.2% vs 96.8%). The reason is that the proposed CoL can obtain the exchange knowledge from DL models of other LNs. Thereby, it can achieve similar performance as that of the CeL.

2) Performance Analysis: Table III presents the simulation results in two cases, i.e., two participated LNs, and three participated LNs. In both cases, we can observe the same trend when the accuracy, precision and recall of the proposed CoL are higher than those of the IL and nearly equal to those of the CeL. In particular, the accuracy of the proposed CoL is higher than that obtained by LN-1 in IL (approximately 0.5%), and the precision of the proposed CoL is about 2% higher than that obtained by LN-2 in IL in the case of three participated LNs. These results demonstrate that the proposed CoL can exchange knowledge with other LNs to improve its ability of detection, so it can achieve better performance in classifying

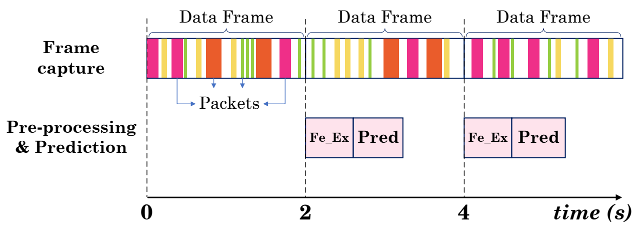


Fig. 7: Timeline of verification phase.

attacks in the blockchain network than those of the IL. It also demonstrates that the learning model of IL should not be used to classify the data of other LNs. In addition, without sharing LN's dataset with a central node for training (e.g., a cloud server), the proposed CoL can achieve nearly the same accuracy as those of the CeL in all the scenarios.

B. Experimental Results

In this section, we present the experimental results obtained through real-time experiments at our laboratory. In this experiment, each blockchain node is installed a learning model to become an LN. Each LN learns its local dataset and then performs real-time attack detection in the blockchain network. We consider the scenario of two LNs and three LNs with the proposed CoL and the CeL. In the training process, the proposed CoL and the CeL are fed with the similar datasets as explained in the previous section. We then implement the trained model to all the participated LNs to perform real-time attack detection for both learning models in the testing process.

1) Real-time capturing and processing: In a real-time system, the cyberattack detection system continuously receives a number of the Ethereum network traffic data. Therefore, the system has to perform capturing, collecting frames, extracting features, analyzing and predicting within a period of time, i.e, 2 seconds. Fig. 7 shows the timeline of the cyberattack detection program. The data frame is exploited by our feature extractor function (Fe_Ex) of BC-ID tool, and this is input for the trained model to predict (Pred) and classify packets to be normal or attack. All processes have to complete in 2 seconds before the next data frame of IP packets coming. To verify the predicted results from the trained model, all frames and prediction results are stored. These frames are merged into a full validation dataset and labeled by own designed BC-ID. After that, these ground truth labels are compared with the prediction results to obtain a validation report.

2) Performance Analysis: Table IV presents the experimental results of the proposed CoL and the CeL with different participated LNs. We obtain the same trends as those of the

TABLE IV: Real-time experimental results of 3 LNs models.

Model	2 Learning Nodes (LNs)				3 Learning Nodes (LNs)					
	Proposed CoL		CeL		Proposed CoL			CeL		
	LN-1	LN-2	LN-1	LN-2	LN-1	LN-2	LN-3	LN-1	LN-2	LN-3
Accuracy	**98.611**	**98.242**	98.464	98.097	**98.440**	**98.131**	**97.686**	98.503	98.192	97.771
Precision	**97.433**	**96.871**	97.146	96.634	**97.146**	**96.717**	**95.902**	97.138	96.679	95.864
Recall	**96.529**	**95.606**	96.159	95.243	**96.101**	**95.328**	**94.214**	96.258	95.481	94.427

simulation results. The accuracy results obtained by two and three learning models of both proposed CoL and CeL are slightly higher than those of the simulations at about 1%. This is because each type of attack has different distributions of attack samples in a period of time. Table V presents the number of samples of each class collected in 15 minutes. In this table, we can observe that Class-1 and Class-4 have small numbers of samples during this period, this can lead to low accuracy in statistics for these classes and reduce the total accuracy of the model. However, our proposed CoL still has better performance than those of the CeL in LN-1 in the case of two LNs (i.e., up to 0.2% accuracy, 0.3% precision and 0.4% recall). Overall, our proposed CoL always achieves the best performance with approximately 98.6% accuracy, 95.43% precision and 96.52% recall with two LNs and 98.44% accuracy, 97.14% precision and 96.1% recall with three LNs. These results demonstrate that our proposed CoL can detect attacks with nearly the same accuracies for all participated LNs as those of CeL.

3) Real-time Monitoring and Detection: Fig. 8 illustrates the real-time monitoring of our proposed CoL for normal state and three types of attacks in the network. Fig. 8(a) is the normal state (Class-0) of the network with a high number of normal samples and a low number of attack samples. Then, the BP and MitM attacks are performed. Fig. 8(b) and Fig. 8(e) show a slight increase in the number of BP attacks and MitM attacks. This is because the number of BP attack samples is much smaller than other states in a period of time as in Table V. In this case, the detection mechanism is activated and alarms the network under the BP attack (Class-1). Similarly, the DDoS attack in the network is described in Fig. 8(c) with a high increase in the number of samples of DoS attacks. Finally, Fig. 8(d) describes the FoT attacks. Unlike other attacks, the FoT attacks include a large number of samples, thus it increases both the number of normal and attack samples (above 200 traffic samples per 2 seconds) more than other attacks (about 100 traffic samples per 2 seconds). Thereby, in all the cases, we can observe that our proposed intrusion detection system can detect attacks effectively in a real-time manner.

4) Real-time Processing Capacity: In this experiment, we fix the number of input samples in our proposed model to find the maximum real-time processing capacity in capturing and detecting attacks. Fig. 9 illustrates the real-time processing capacity of our proposed model. The processing time τ is counted from the time when our proposed model reads the file containing the samples, until completing classification and producing the output. This work is repeated 20,000 times to determine the stability of the detection time of our proposed model. We vary the number of input samples multiple times to find the appropriate number that is adapted to the condition in Fig. 7. In most of the cases (98%), our proposed framework can classify 85,000 samples in less than 2 seconds. These results demonstrate that our proposed detection framework is efficient to deploy to detect attacks in real-world blockchain networks. It can not only detect attacks with high accuracy (up to 98.6%) but also quickly (up to 85,000 samples within 2 seconds).

TABLE V: The number of samples on LN-1 in five hours.

	Class-0	Class-1	Class-2	Class-3	Class-4
Number of samples	736,897	2,424	481,532	886,389	3,483
Portion (%)	34.912	0.115	22.814	41.994	0.165

VI. Conclusion

In this work, we have proposed a novel collaborative learning framework for a cyberattack detection system in a blockchain network. First, we have implemented a private blockchain network in our laboratory. This blockchain network is used to (1) generate data (both normal and attack data) to serve the proposed learning models and (2) validate the performance of our proposed learning framework in real-time experiments. After that, we have proposed a highly-effective learning model that allows to be effectively deployed in the blockchain network. This learning model allows nodes in the blockchain can be actively involved in the detection process by collecting data, learning knowledge from their data, and then exchanging knowledge together to improve the attack detection ability. In this way, we can not only avoid problems of conventional centralized learning (e.g., congestion and single point of failure) but also protect the blockchain network right at the edge. Both simulation and real-time experimental results then have clearly shown the efficiency of our proposed framework. In future, we plan to continue developing this dataset with other emerging types of attacks and develop more effective methods to protect blockchain networks.

References

[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, “Applications of blockchains in the Internet of Things: A comprehensive survey,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1676–1717, Dec. 2018.

[3] J. Xie, H. Tang, T. Huang, F. R. Yu, R. Xie, J. Liu, and Y. Liu, “A survey of blockchain technology applied to smart cities: Research issues and challenges,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2794–2830, Feb. 2019.

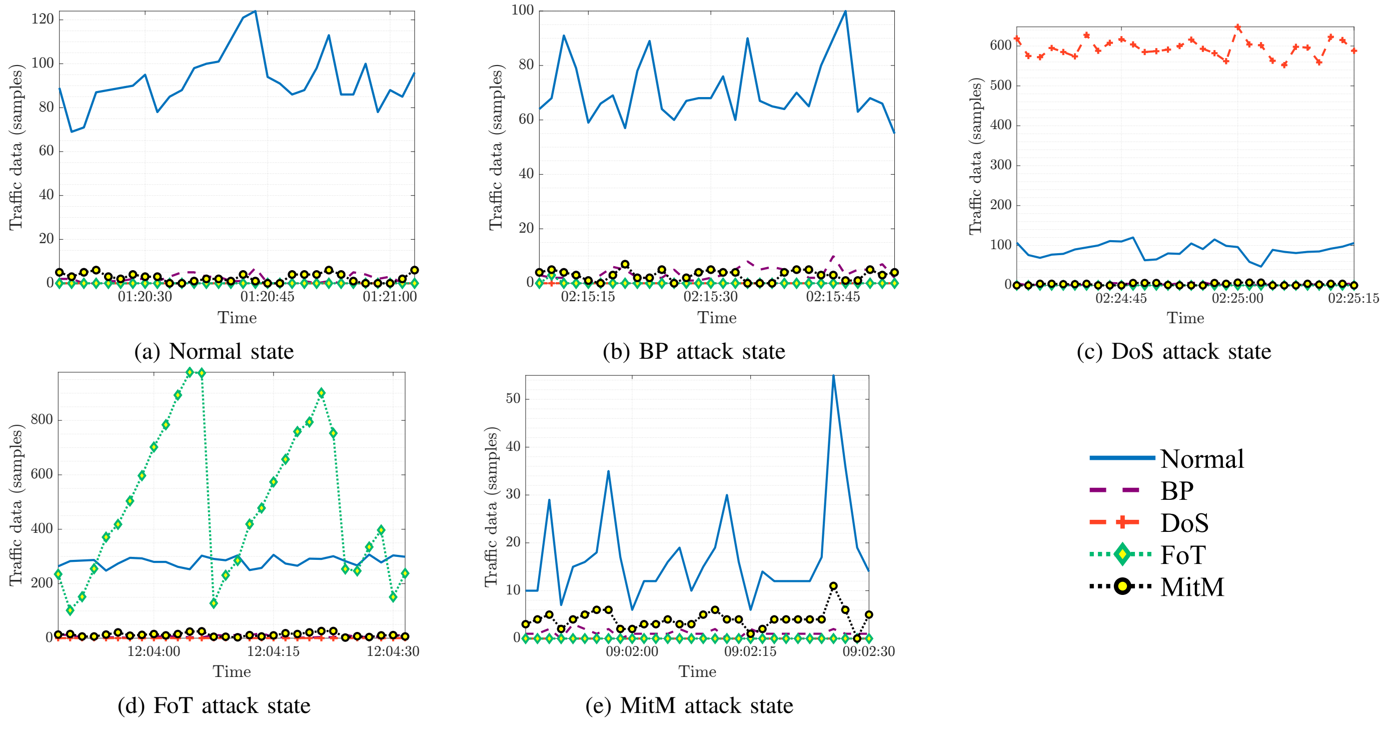


Fig. 8: Real-time blockchain cyberattack detection: proposed CoL model of 3 LNs in Ethereum node 1.

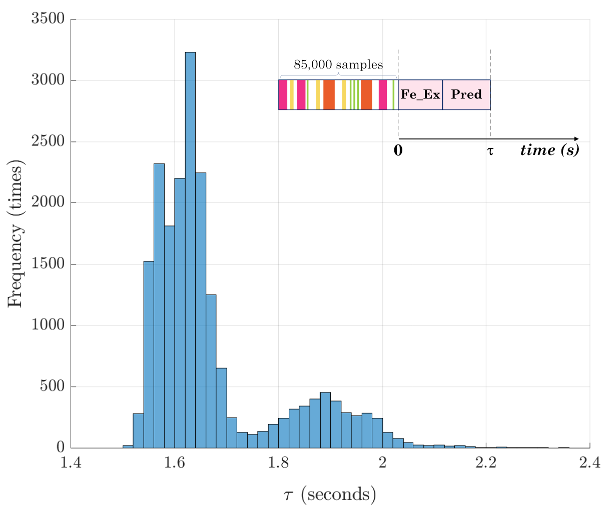


Fig. 9: Histogram of real-time detection duration.

[4] S. Biswas, K. Sharif, F. Li, Z. Latif, S. S. Kanhere, and S. P. Mohanty, “Interoperability and synchronization management of blockchain-based decentralized e-health systems,” *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 1363–1376, June 2020.

[5] Y. Yuan and F.-Y. Wang, “Blockchain and cryptocurrencies: Model, techniques, and applications,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1421–1428, July 2018.

[6] “The 10 Biggest Crypto Exchange Hacks In History,” Accessed: Feb. 14, 2022. [Online]. Available: https://crystalblockchain.com/articles/the-10-biggest-crypto-exchange-hacks-in-history

[7] “North Korean Hackers Have Prolific Year as Their Unlaundered Cryptocurrency Holdings Reach All-time High,” Accessed: Feb. 14, 2022. [Online]. Available: https://blog.chainalysis.com/reports/north-korean-hackers-have-prolific-year-as-their-total-unlaundered-cryptocurrency-holdings-reach-all-time-high

[8] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, “Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities,” *IEEE Access*, vol. 7, pp. 85 727–85 745, Jun. 2019.

[9] K. Salah, N. Nizamuddin, R. Jayaraman, and M. Omar, “Blockchain-based soybean traceability in agricultural supply chain,” *IEEE Access*, vol. 7, pp. 73 295–73 305, May 2019.

[10] S. Bu, F. R. Yu, X. P. Liu, P. Mason, and H. Tang, “Distributed combined authentication and intrusion detection with data fusion in high-security mobile ad hoc networks,” *IEEE transactions on vehicular technology*, vol. 60, no. 3, pp. 1025–1036, Dec. 2010.

[11] X. Wang, X. Zha, G. Yu, W. Ni, R. P. Liu, Y. J. Guo, X. Niu, and K. Zheng, “Attack and defence of ethereum remote apis,” in *2018 IEEE Globecom Workshops (GC Wkshps)*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–6.

[12] J. Otávio Chervinski, D. Kreutz, and J. Yu, “Analysis of transaction flooding attacks against Monero,” in *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, Sydney, Australia, May 2021, pp. 1–8.

[13] J. Choi, B. Ahn, G. Bere, S. Ahmad, H. A. Mantooth, and T. Kim, “Blockchain-based man-in-the-middle (mitm) attack detection for photovoltaic systems,” in *IEEE Design Methodologies Conference (DMC)*, July 2021, pp. 1–6.

[14] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, “Network intrusion detection for IoT security based on learning techniques,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2671–2701, Jan. 2019.

[15] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, “Survey on SDN based network intrusion detection system using machine learning approaches,” *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 493–501, Mar. 2019.

[16] M. Idhammad, K. Afdel, and M. Belouch, “Distributed intrusion detection system for cloud environments based on data mining techniques,” *Procedia Computer Science*, vol. 127, pp. 35–41, Jan. 2018.

[17] K.-D. Lu, G.-Q. Zeng, X. Luo, J. Weng, W. Luo, and Y. Wu, “Evolutionary deep belief network for cyber-attack detection in industrial automation and control system,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7618–7627, Nov. 2021.

[18] L. Vu, V. L. Cao, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, “Learning latent representation for iot anomaly detection,” *IEEE Transactions on Cybernetics*, pp. 1–14, Sep. 2020.

[19] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on deep transfer learning,” in *Int. Conf. on Artificial Neural Networks*. Rhodes, Greece: Springer, Oct. 2018, pp. 270–279.

[20] M. U. Hassan, M. H. Rehmani, and J. Chen, “Anomaly detection in blockchain networks: A comprehensive survey,” *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 289–318, Jan. 2023.

[21] P. Kumar, R. Kumar, G. Gupta, and R. Tripathi, "A distributed framework for detecting DDoS attacks in smart contract-based Blockchain-IoT systems by leveraging fog computing," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 6, pp. 1–31, June 2021.

[22] O. Alkadi, N. Moustafa, B. Turnbull, and K.-K. R. Choo, "A deep blockchain framework-enabled collaborative intrusion detection for protecting IoT and cloud networks," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9463–9472, June 2020.

[23] J. Kim, M. Nakashima, W. Fan, S. Wuthier, X. Zhou, I. Kim, and S.-Y. Chang, "Anomaly detection based on traffic monitoring for secure blockchain networking," in *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, Sydney, Australia, May 2021, pp. 1–9.

[24] Z. Liu and X. Yin, "LSTM-CGAN: towards generating low-rate DDoS adversarial samples for blockchain-based wireless network detection models," *IEEE Access*, vol. 9, pp. 22 616–22 625, Feb. 2021.

[25] W. Cao, Y. Huang, D. Li, F. Yang, X. Jiang, and J. Yang, "A blockchain based link-flooding attack detection scheme," in *IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, vol. 4, Chongqing, China, June 2021, pp. 1665–1669.

[26] P. Ramanan, D. Li, and N. Gebraeel, "Blockchain-based decentralized replay attack detection for large-scale power systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Aug. 2021.

[27] B. Hu, C. Zhou, Y.-C. Tian, Y. Qin, and X. Junping, "A collaborative intrusion detection approach using blockchain for multimicrogrid systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 8, pp. 1720–1730, Apr. 2019.

[28] "Wireshark Network Analysis," Accessed: Feb. 14, 2022. [Online]. Available: https://www.wireshark.org

[29] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, July 2006.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[31] G. E. Hinton, "A practical guide to training restricted boltzmann machines," in *Neural Networks: Tricks of the Trade, 2nd ed.*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Heidelberg: Springer, 2012, pp. 599–619.

[32] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, Oct. 2017. [Online]. Available: https://arxiv.org/abs/1610.05492

[33] Ethereum, "Official Go implementation of the Ethereum protocol," Accessed: Feb. 14, 2022. [Online]. Available: https://github.com/ethereum/go-ethereum

[34] M. Oance, "Ethereum Network Stats," Accessed: Feb. 14, 2022. [Online]. Available: https://github.com/cubedro/eth-netstats

[35] T. Neudecker and H. Hartenstein, "Network layer aspects of permissionless blockchains," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 838–857, Sep. 2019.

[36] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. Nyang, and D. Mohaisen, "Exploring the attack surface of blockchain: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1977–2008, Mar. 2020.

[37] "Bitfinex restored after DDoS attack," Accessed: Feb. 14, 2022. [Online]. Available: https://www.computerweekly.com/news/450431741/Bitfinex-restored-after-DDoS-attack

[38] "kdd99_feature_extractor," AI-IDS, Accessed: Feb. 14, 2022. [Online]. Available: https://github.com/AI-IDS/kdd99_feature_extractor

[39] "KDD Cup 1999 Data," The Fifth International Conference on Knowledge Discovery and Data Mining, Accessed: Feb. 14, 2022. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[40] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, Nov. 2008.

[41] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, June 2006.