IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 27, 2021, accepted June 18, 2021, date of publication July 2, 2021, date of current version July 26, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3094365

A Blockchain-Based Federated Forest for SDN-Enabled In-Vehicle Network Intrusion Detection System

IBRAHIM ALIYU[1], (Member, IEEE), MARCO CARLO FELICIANO[2], SÉLINDE VAN ENGELENBURG[3], DONG OK KIM[4], AND CHANG GYOON LIM[1], (Member, IEEE)

[1]Department of Computer Engineering, Chonnam National University, Yeosu 59626, South Korea
[2]Department of Electrical and ICT Engineering, University of Naples Federico II, 80138 Naples, Italy
[3]Faculty of Technology, Policy and Management, Delft University of Technology, 2628 Delft, The Netherlands
[4]National Innovation Cluster Support Center, Jeonnam Technopark, Suncheon, Jenonnam 58034, South Korea

Corresponding author: Chang Gyoon Lim (cglim@jnu.ac.kr)

This work was supported in part by the Ministry of Trade, Industry and Energy (MOTIE), and in part by the Korea Institute for Advancement of Technology (KIAT) through the National Innovation Cluster Research and Development Program under Grant P0016223.

ABSTRACT In-vehicle communication systems are usually managed by controller area networks (CAN). By broadcasting packets to their bus, the CAN facilitates the interaction between Electronic Control Units (ECU) that coordinate, monitor and control internal vehicle components. With no authentication mechanism for identifying the legitimacy and source of packets, CAN are vulnerable to cyber-attacks. An Intrusion Detection System (IDS) can detect attacks on CAN and machine learning can be used to create the models for the IDSs to detect non-linear attack patterns. However, car manufacturers and owners might want to keep the sensitive information required for training the models confidential. Therefore, we proposed a Blockchain-based Federated Forest Software-Defined Networking (SDN)-enabled IDS (BFF-IDS) to address the problem of data sharing the sensitive CAN data. To ensure scalability, we used InterPlanetary File System (IPFS) to host the models, and the blockchain is designed to store only a hash of the model and a pointer to its location. The SDN provides the dynamic routing of packets and model exchanges. We used Federated Learning (FL) to create a random forest model. Individuals provide partially trained models, allowing them to keep the underlying data confidential. Using Fourier transform, we decomposed the CAN IDs cycle from CAN bus traffic in the frequency domain for better generalization in multiclass detection of attacks. Multiple statistical and entropy features were extracted to handle the high complexity and non-linearity in CAN bus traffic. The proposed system allows manufacturers and car owners to contribute to the training of the models, as their sensitive data is protected. By storing hashes of the models on a blockchain, the risk of adversaries poisoning the models is reduced and a single point of failure is avoided. We evaluated the proposed system by conducting experiments on a testbed. We found that the proposed system has efficient use of memory and CPU resources and that the detection rate of closely related attacks was high. We recorded the highest model attack detection rate of about 0.981.

INDEX TERMS Blockchain, CAN, federated learning, intrusion detection system, in-vehicle network, random forest, SDN.

I. INTRODUCTION

In the modern transportation system, In-vehicle communication systems are managed by controllers know as controller area networks (CAN). The CAN facilitates the interaction of 20 to 100 Electronic Control Units (ECU) that coordinate, monitor and control loads of internal vehicle components such as engine system, brake system and telematics system through the exchange of information among the systems [1]. Nevertheless, the exchange of information within the CAN

The associate editor coordinating the review of this manuscript and approving it for publication was Lo'ai A. Tawalbeh.

bus system has exposed it to external threats which can harm the vehicle and car users [2], [3]. CAN works by broadcasting packets to its bus, which means all nodes and ECUs attached to the bus can receive all transmitted packets. However, nodes and ECUs have no authentication mechanism for identifying the legitimacy or source of packets. making them vulnerable to attacks. When CAN is compromised, one can easily inject malicious messages to ECUs that can trigger physical action such as steering and braking, or manipulation of speedometer display information [3], [4]. The attacks on the In-vehicular network can be analyzed in terms of attack surface and attack vector [5]. In the attack surface aspect, the attack is launched directly through the On-Board Diagnostic II (OBD-II) port or indirectly through firmware such as the media player. For example, a case was reported of a successful breach of CAN through OBD-II port and firmware (MP3) in a car thief and vehicle control [5], [6]. In terms of attack vector, attacks can be launched through the vulnerability of the Bluetooth protocol that supports in-vehicle audio video navigation (AVN) systems, or remotely by the exploitation of the communication channel between a telematics module and a smartphone application [5].

To protect the CAN, the IDS is considered one of the best solutions due to its simplicity and efficiency in detecting attacks [7]. Machine learning for the IDS would be adequate in learning non-linear attack patterns. However, the method of training the IDS is still a challenge as car manufacturers and owners are skeptical to share such sensitive information. Thus, the question remains on how to facilitate collaboration for the training of the IDS model by car manufacturers and owners to build a resilient model without risking any harm to their privacy and security. Besides, efficient Inter-Vehicle communications (IVC) in modern transportation is essential as it plays an integral role in enabling vehicles to share information in an ever-evolving network complexity.

IDS for CAN has been trained using local data samples (traditional learning, TL), i.e., the method proposed training of a model for a single car [4], [8], [9]. Even though [10] proposed continuous cloud service for smart vehicles that enables IDS for improving quality of service (QoS) and quality of experience (QoE), the IDS were locally trained on each vehicle. In contrast, this research addresses the drawback of training each vehicle's local model by using federated learning (FL) instead.

One main challenge of the local model is that local data samples generated by CAN are owned by each vehicle and automakers offer limited support for sharing the data with other automakers' vehicles. In addition, car owners too may be reluctant to share information due to privacy concerns. To keep training data private, Google proposed a federated learning where each device can exchange its local model update, i.e., weight and gradient parameters, without sharing a data sample or inferring the data sample from the local model updates. The federated learning utilizes a central server for the aggregation of the local model updates, yielding a global model update that can then be downloaded on devices/systems. However, due to the exchanges, tens of minutes in latency are procured and vulnerable to a single point of failure as a single server is dedicated for aggregation [11]. Also, the system is vulnerable to a single point of failure as a single server is dedicated to aggregate the models. To address these shortcomings, Hyenum Kim [12] proposed the integration of blockchain with Federated Learning (BC-FL). The BC-FL architecture enables the device to exchange their local model and updates while aggregation is conducted by each node after downloading the model's updates from the blockchain. However, this method is very expensive considering the size of the parameter being exchanged over the blockchain as more gas/ether (in Ethereum for instance) would be needed.

Hence, in this study, we proposed a blockchain-based federated forest SDN-enabled intrusion detection system (BFF-IDS) for an in-vehicle network to address the problem of sharing sensitive CAN data. The federated learning system creates a random forest model in a distributed manner by aggregating partially training models. The system provides a relatively cheaper solution compared to other systems as it uses IPFS to store the model by the hash of the model location is store and exchange over the blockchain. In addition, we proposed SDN to provide the dynamic routing of packets and model exchanges from IPFS through the blockchain. Since car manufacturers do not provide information regarding CAN actual identifiers and data, we utilized the CAN IDs sequence to detect intrusion [13]. CAN IDs cycle sequences were extracted and transformed using Fourier transform (FFT) to decompose the cycle in the frequency domain for better generalization in multiclass detection of attacks (fuzzy attack, DoS attack, Impersonation attack and attack-free state). We employed multiple statics and entropy features to handle the high complexity and non-linearity in CAN bus traffic. We evaluated the system using precision, recall F1-score and accuracy and compared our proposed system performance with other works based on accuracy. The main contributions of the proposed system are summarized as follows:

1. We proposed a blockchain-based federated forest for SDN-enabled intrusion detection in an In-vehicular network. To the best of our knowledge, this is the first system to utilize the integration of blockchain, SDN and federated learning for an In-vehicle IDS.

2. We created a testbed that enables the training and testing of the model using the Ethereum blockchain and Mininet emulator on a local environment.

3. We extracted statistical and entropy features that successfully provided distinct features for multiclass intrusion detection. Our model evaluation results showed a superior performance against the TL approach and other model machine learning models.

The remainder of this paper is organized as follows: In the next section, we discuss the related literature. In section 3, federated learning and problem formulation is presented. Section 4 presents the concept of blockchain-based SDN-enabled IDS, including the methodology, algorithms for

TABLE 1. Related works on the can IDS.

Year/Ref.	FFT	Features	Attack classes	Blockchain	F L	SDN
2021, [14]	×	Unique CAN IDs, Messages, DLC, bandwidth	4	×	×	×
2020, [13]	×	CAN IDs Sequence	2 (for each class)	×	×	×
2017, [24]	×		4	×	×	×
2011,[15]	×	Entropy (CAN IDs)	3	×	×	×
2016, [16]	×		2	×	×	×
2016, [17]	×	Time interval	3	×	×	×
2018, [18]	×		2	×	×	×
2018, [5]	×	Frequency	3	×	×	×
2015, [19]	×		2	×	×	×
2018, [20]	×		2	×	×	×
2016, [21]	×	Clock skew	2	×	×	×
2017, [22]	×	Remote Frame	3	×	×	×
2019, [23]	×	NTP	2	×	×	×
Our framework	✓	Statistics (CAN IDs), Entropy (CAN IDs)	4	✓	✓	✓

data preprocessing, feature extraction and FL training and testing. The results, discussion and conclusion are presented in sections 5, 6 and 7, respectively.

II. RELATED WORKS

The vulnerability of the in-vehicle network poses a great deal of threat not only to the driver's or passengers' safety but to the society at large. The surge in urbanization and the dependence on the intelligent system by society is increasing the urgency on which we need to act to ensure safety as we turn to the smart city to drive our societies in the coming era. Therefore, both industry and academia have been working to provide solutions to the security threat facing the In-vehicle network.

Since the use of CAN ID identifiers and data is not an option due to the security concern by the manufacturers making them unwilling to share, attack detection using other means such as CAN ID cycle, unique number of CAN ID, etc., as against the use of semantic features is the most effective means. Efforts have been made in the utilization of the transmission pattern of the ECU in the data link layer of the CAN bus. Recently, cosine similarity was proposed for the detection of three forms of an anomaly in CAN bus [14]. Lightweight feature vectors for real-time detection were designed and tested on different cars. In addition to CAN ID fields in the CAN message, the number of messages, sum of DLC and bandwidth of the CAN bus were used as features. Deep Convolutional Neural Network (DCNN) was proposed for the utilization of the CAN ID field to detect attacks through the analysis of CAN traffic patterns [13]. An optimized model was developed to solve the complexity problem while achieving a high accuracy rate. A study also proposed the detection of attacks based on the entropy of CAN bus traffic to distinguish normal traffic from malicious ones [15]. The proposed method shows significant performance for different attack scenarios. Similarly, another study utilized entropy to detect attacks based on the volume of forged CAN messages [16]. Meanwhile, time intervals of CAN messages were deployed to detect three forms of message injection attack [17]. Similarly, [18] used the time interval in detecting attacks based on the time changes in CAN bus traffic. Using unsupervised methods (ARIMA and Z-score) packet drop and packet injection attack types were effectively detected using the time interval. The CAN bus frequency-based methods have also shown significant results. For instance, another effort survival analysis of CAN IDs frequency was proposed for the detection of the three forms of attack against the CAN bus [5]. The method showed a better detection rate against CAN IDs with a short cycle. Another paper explored frequencies of packets to detect anomalies by sliding window to measure the inter-packets timing [19]. In addition, another set of researchers explored statistical analyses for the detection of anomalies from CAN IDs frequencies [20]. Besides, clockwise, remote frame, network time protocol (NTP) and ID sequence based features have also been proposed for intrusion detection in CAN [21]–[24]. The summary of the related works is as shown in Table 1.

III. FEDERATED LEARNING AND PROBLEM FORMULATION

A. DEFINITION OF FEDERATED LEARNING

The idea behind the innovation of the federated learning concept is to facilitate the building of a model based on a distributed data set across multiple devices while preventing data leakage [25]. Given data owners, $\{\mathbb{N}_1 \ldots \mathbb{N}_i\}$, who wish to build a strong model by the consolidation of their data $(\mathcal{D}_1 \ldots \mathcal{D}_i)$, conventionally train the model, $\mathcal{M}_{SUM}$, by combining the data, $\mathcal{D} = (\mathcal{D}_1 \cup \ldots \cup \mathcal{D}_i)$. However, in the federated learning concept, the owner collaboratively trains the model, $\mathcal{M}_{FL}$., in such a way that any data owner $\mathbb{N}_i$ does not reveal its data $\mathcal{D}_i$ to others while ensuring that its performance, $\mathcal{P}_{FL}$, is close to that of $\mathcal{M}_{SUM}$ model performance, $\mathcal{P}_{SUM}$. Formally, given δ as a non-negative real number, if

$$|\mathcal{P}_{FL} - \mathcal{P}_{SUM}| < \delta \tag{1}$$

the federated model is said to have δ -accuracy loss.

Federated learning can be categorized based on the distribution nature of the data [25]. Let matrix $\mathcal{D}_i$ defines the data held by owner i with each row in the matrix representing a sample. Let $\mathcal{F}$, $\mathcal{Y}$, and $\mathcal{J}$ denotes feature, label and sample ID space, respectively. The training dataset is thus constituted as ($\mathcal{F}$, $\mathcal{Y}$, $\mathcal{J}$). Based on how the data is distributed among the subsets, the feature and sample ID space, Federated learning can be classified as horizontal federated learning, vertical federated learning and federated transfer learning. In horizontal federated learning, the data sets have the same feature space but different samples, while in vertical scenario the datasets have the same sample ID but share different feature space. On the other hand, federated transfer learning designates a situation where the dataset differs in both samples and feature space.

B. PROBLEM FORMULATION

In this work, we focused on the horizontal federated learning aspect. The dataset $\mathcal{D}$ was distributed among K parties $\mathbb{N}_1 \ldots \mathbb{N}_k$. Only a subset of the data $\mathcal{D}_k \subseteq \mathcal{D}$ with N_k samples are utilized by k^{th} party, where $k \in [1, k]$. We assumed that the data points in $\mathcal{D}$ allocated to any distinct parties $\mathbb{N}_1 \ldots \mathbb{N}_k$ are disjoint, i.e, $\mathcal{D}_1 \ldots \mathcal{D}_k$ are partitions from $\mathcal{D}$. The main goal was to build a detector from the complete dataset while minimizing δ-accuracy loss.

We employed federated forest i in this study to build an accurate intrusion detection model. The model built was such that (1) a partial model forest model was built and held by each miner $\mathcal{M}_i$, $1 \leq i \leq K$; (2) complete model, $\mathcal{M}_{FL}$, was aggregated at each user end while minimizing δ-accuracy loss.

IV. BLOCKCHAIN-BASED SDN-ENABLED IN-VEHICULAR NETWORK INTRUSION DETECTION SYSTEM

Due to the immense global population growth in the cities, the architectural design of the network of smart cities is increasingly faced with challenges in terms of latency, scalability, network bandwidth usage data privacy and security [26]. These challenges need to be addressed for the sustainability of the smart city networks. Besides, the IoT network, which is part of the smart city's composition, is envisioned to provide an efficient and scalable trust management system with authentication and authorization based on the centralized and distributed concept for local and global infrastructure, respectively [27]. Inspired by [27] and [28], [26] proposed a hybrid architecture that guarantees scalability in smart city networks using blockchain and SDN. Meanwhile, connected vehicles, being one of the entities in the smart city, should be equipped with IDS that can intercommunicate through the hybrid system. Hence, we took the advantage of the hybrid architecture to propose and build a scalable IDS network for effective collaboration in ensuring resilient federated learning that benefits from the abundant island of data available across the transportation ecosystem.

Our proposed hybrid architecture with the scalable IDS network is as presented in Figure 1. Algorithm 1 is the implementation of the testbed. The architecture of the network is divided into three planes based on the SDN – the data plane consisting of the vehicles that host data, the control plane that manages the communication of the In-vehicle IDS through blockchain, and the application plane which consist of the authorities saddle with the system management. In the data plane, each vehicle is assumed to be a node with generated data that can be harness to train IDS models. Models are exchanged over a blockchain network managed by the SDN. The blockchain network consists of miner nodes with high computation and storage resources, which are responsible for creating blocks and verifying proof-of-Authority. We proposed the use of mobile network infrastructures as the mining nodes since the infrastructure is in place and has high computational and storage resources. Considering the cost of uploading models or model weight over the blockchain, we proposed the used of IPFS to upload the model while the hash is exchanged over the blockchain. This way, the only cost incurred will be for the cost of some bytes to the size of the hash. Authenticated nodes (users) who wish to use the model can download the hash to gain access to the models at the end of the mining process. The models are aggregated (federated) at the user end. The nodes at the data plane are SDN controller-enabled to reduce hardware management costs, and to ease deployment in the network infrastructure with high agility and security. The application plane consists of critical stakeholders such as network management agencies, identification/key providers for participants, and threat intelligent agencies that oversee the identification of attack trends and policies.

A. BFF-IDS FOR IN-VEHICLE NETWORK

For security reasons, car manufacturers do not provide information regarding CAN actual identifiers and data. Thus, the utilization of CAN IDs sequence pattern is the most effective way to detect intrusion since semantic features are unavailable [13]. Figure 2 presents the overview of the proposed BFF-IDS. The system consists of five main steps- data processing, feature extraction, model training, model exchange and model aggregation. The detail of each step is illustrated below.

1) CAN BUS DATASET

The attack scenarios considered in this study include threats that are capable of manipulating in-vehicle nodes remotely/physically by the injection of unauthenticated or malicious messages in the CAN bus. We used the CAN-intrusion dataset (OTIDS) that is publicly provided by the Hacking and Countermeasure Research Lab at Korea University [22]. It was created by logging CAN traffic on a real vehicle via the OBD-II port of KIA SOUL while message injection attacks were conducted. The classes of traffic provided in the dataset include fuzzy attack, DoS attack, Impersonation attack and attack-free state. DoS attack traffic was created by injecting messages in a short cycle using '0×000' CAN ID, Fuzzy Attack was generated by injecting messages

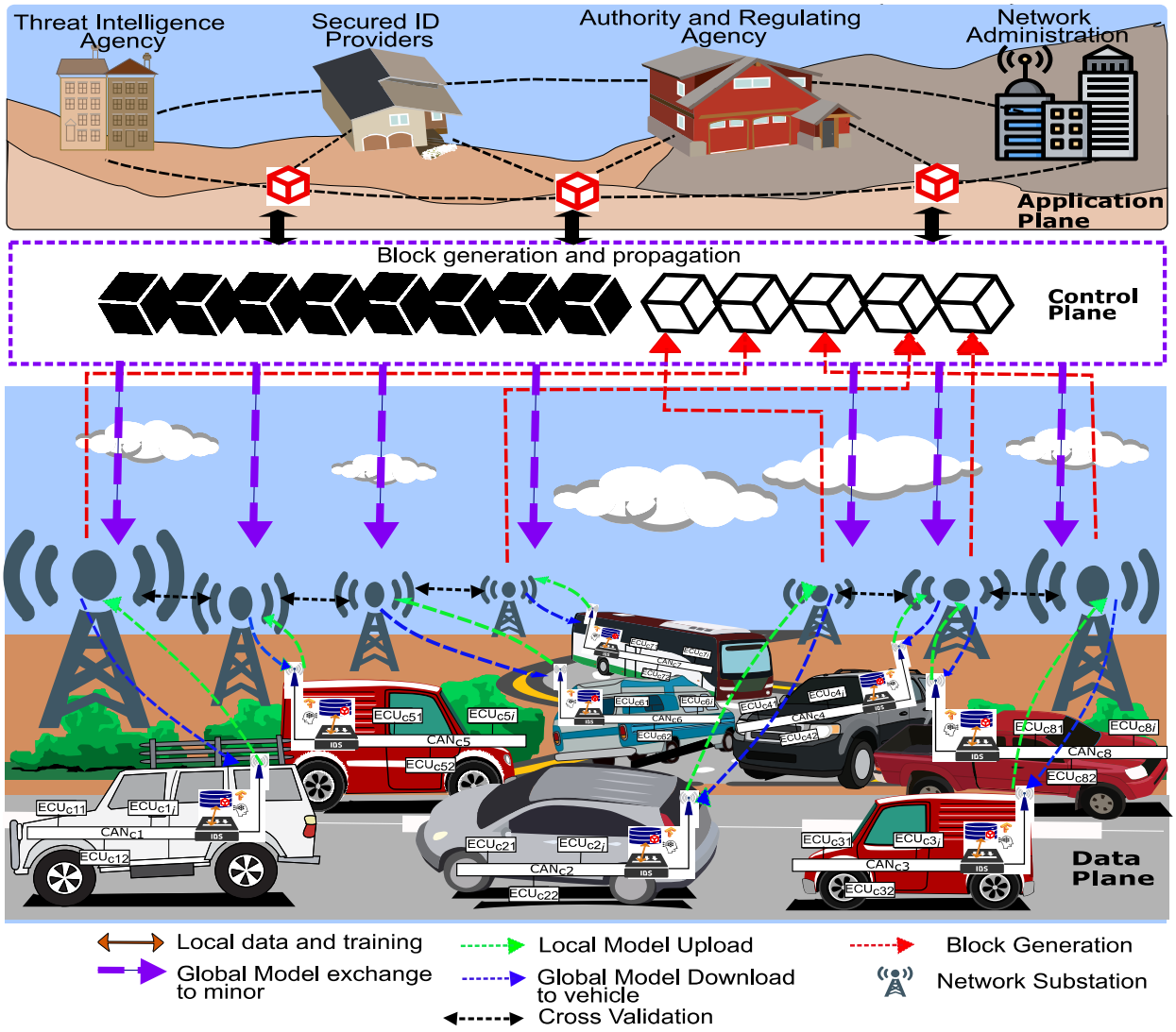


FIGURE 1. The proposed blockchain and federated learning framework for SDN-enabled in-vehicle IDS architecture.

of spoofed random CAN ID and DATA values, while an impersonation attack was created by injecting messages from an impersonating node with arbitration ID, '0×164'. Attack-free traffic was recorded from normal CAN messages.

These attack types are partly considered in this study because of the devastating consequence they have on the CAN. For instance, a simple Fuzzy attack can override normal functions or reset internal conditions of the vehicle by filling a whole range of functional CAN identifiers on the bus; the packet flooding by DoS attack can enable a compromised node to indefinitely declare dominant status, denying access to other legitimate nodes; impersonating attack maliciously takes the role of the target node and cause the vehicle to display unintended states with no clear causes [22].

2) DATA PREPROCESSING

In the data preprocessing, the CAN IDs sequences are first extracted and numbered accordingly to extract CAN ID cycle- A cycle is considered as the interval at which a particular CAN ID occurred again after its first appearance [5]. The data is then grouped by CAN IDs in order to calculate the cycle of each unique CAN ID using the assigned sequence numbers as a feature. The order of the data is then reset to the original sequence after calculating the cycle as presented in Algorithm 2. The CAN IDs cycle is used as input for the feature extraction step.

3) FEATURE EXTRACTION

The packets traffic stream continuously in the CAN bus, therefore messages are not explicitly segmented into sub-fragments associated with the transmission pattern of the CAN ID. Hence, we segment the cycle into several samples of equal length by sliding a window of fixed length through the entire traffic. The window is parameterized by window length, l, and step length, s. Fourier transformation is then applied to observe the cycle in the frequency domain and

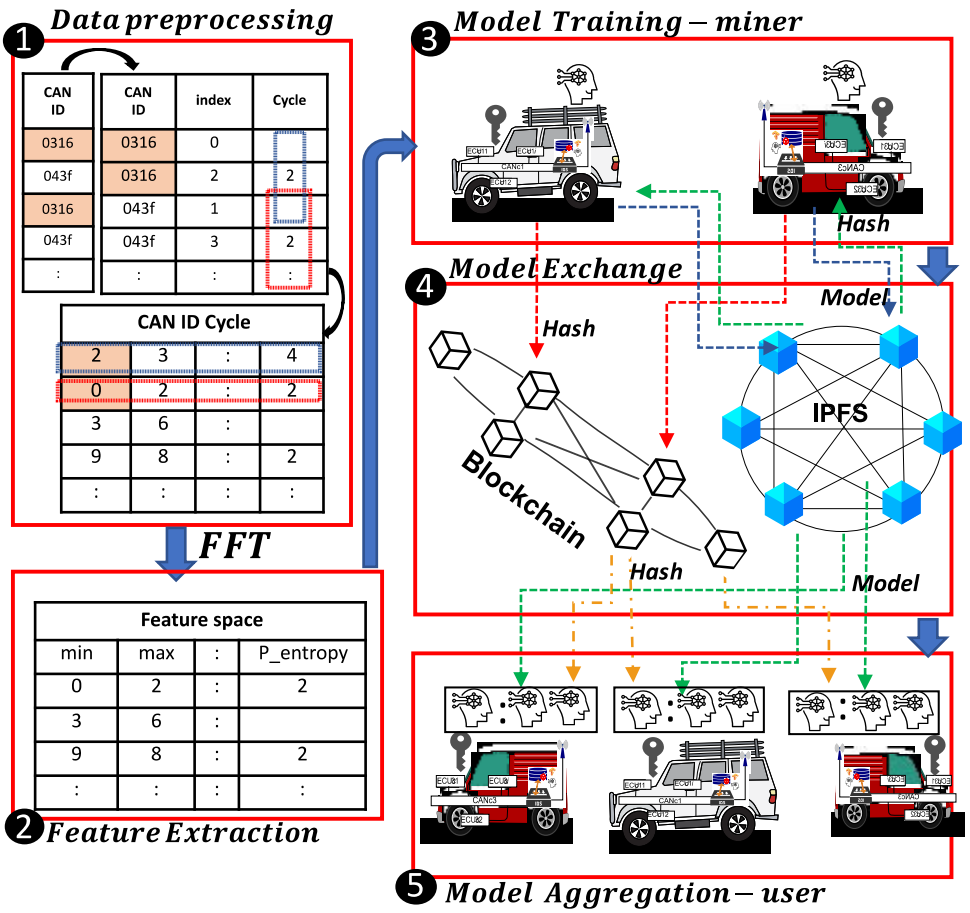


FIGURE 2. Overview of the federated forest-based intrusion detection scheme.

measure the sequence of occurrence in the traffic pattern for the various attacks. Very little information is lost when Fourier transformation is applied to the CAN IDs cycle as it uses all parts of the cycle waveform to translate it into the frequency domain [29]. The fast Fourier transformation, $y(k)$, of length N for the sequence x_n of the length, N, is given as:

$$y(k) = \sum_{n-1}^{N-1} x_n e^{-2\pi j \frac{kn}{N}} \tag{2}$$

Due to the high complexity and nonlinearity of the CAN bus datasets, we employed various statistical and entropy measures to reduce the dimensional space of the data. The combination of several feature extraction measures will provide the most distinctive and informative feature sets for effective detection. Algorithm 3 presents the feature extraction steps adopted in this study.

Statistical features provide the most characteristic values that define the distribution of the transform cycles. The statistical features extracted include minimum, maximum, mean, standard deviation and two high-order statistical (HOS) features. The HOS-based features (skewness and kurtosis) assist in quantifying the nonlinear behavior of the random cycle pattern in the CAN dataset [30]. The Skewness feature provides the normalized third-order moment of a random cycle in each window's distribution. It indicates the degree of asymmetry of the distribution around its mean and the results are given in terms of positively skewed, negatively skewed or not skewed. Meanwhile, the Kurtosis offers the normalized fourth-order moment of a random cycle in each window's distribution. A high kurtosis value in a distribution indicates that the data is heavily tailed (large number of outliers). A low kurtosis value on the other hand indicates a small number of outliers. In general, although the first-order and second-order are crucial, HOS are needed to provide a better characterization of the CAN IDs cycles. The measures are express as follows:

1. Minimum value in each cycle sub-fragment:

$$\mathcal{F}_{\min} = \min_{i \in y} y(k)_i \tag{3}$$

2. Maximum value in each cycle sub-fragment:

$$\mathcal{F}_{\max} = \max_{i \in y} y(k)_i \tag{4}$$

3. Mean of the cycle in each window sub-fragment:

$$\mathcal{F}_{mean} = \frac{1}{l} \sum_{i=1}^{l} y(k)_i \tag{5}$$

Algorithm 1 BFF-IDS (Testbed)

(I) Blockchain Initialization:
(*a*) *Network creation:*
Miner nodes, $\mathcal{M}n_1 \ldots \mathcal{M}n_i$ and user node $\aleph n_1 \ldots \aleph n_k$ in the peer-to-peer network
(II) SDN Initialization:
(*a*) *Create network topology based on the Blockchain network:*
Function NetworkTopology*(number_of_host):*
  for *i* **range** *of number_of_host* **do:**
    node ["$\mathcal{M}n_i$"] ← addhost($\mathcal{M}n_i$)
  end
  for *i* **range** *of number_of_host* **do:**
    node ["$\aleph n_i$"] ← addhost($\aleph n_i$)
  end
  for *i* **range** *of number_of_host* **do:**
    node ["Sw_i"] ← addwitch(Sw_i)
  end
  links ← ["$\mathcal{M}n_i$", "Sw_i"]
  links+ =["$\aleph n_i$", "Sw_i"]
  for *pair* **in** *links* **do:**
    addswitch(*node*[*pair*[0]
    node[*pair*[1])
  end
end
(III) IPFS initialization:
(*a*) *initialize IPFS nodes*
(*b*) *bootstrap IPFS nodes*
(*c*) *start instance of IPFS daemon*
(*d*) *connect to the instance in* (*c*) *via ipfsapi:*
$\boldsymbol{IPFS}_{instance} \leftarrow ipfs.connect()$
(IV) Federated Forest-Training:
(*a*) *Through SDN node:* lunch the blockchain nodes created in (I)
(*b*) *Build the models with the miners:*
Using **Algorithm 4:** *Federated Random Forest-Training (miner)*
(V) Upload Model to IPFS:
(*a*) *upload to IPFS the partial forest model,* $\mathcal{M}_i$*:*
$\mathcal{M}_{IPFS} \parallel IPFS_{instance}$.add($\mathcal{M}_i$)
(*b*) *Obtain the model IPFS location "Hash" and "Name":*
$\mathcal{M}_{hash} \parallel \mathcal{M}_{IPFS}$ ["Hash"]
$\mathcal{M}_{name} \parallel \mathcal{M}_{IPFS}$ ["Name"]
Repeat $\mathcal{M}_{hash}, \mathcal{M}_{name}$
(VI) Federated Random forest-Aggregation:
(*a*) *Download the partial federated model path from the blockchain via the SDN user node*
(*b*) *download the partial models from IPFS;*
(*c*) *aggregate partial models:*
Using **Algorithm 6:** *Federated Random Forest-Aggregation (user)*

4. Standard deviation of the cycle in each window sub-fragment:

$$\mathcal{F}_{std} = \sqrt{\frac{1}{l}\sum_{i=1}^{l}(y(k)_i - \mathcal{F}_{mean})^2} \tag{6}$$

5. Skewness of the cycle in each window sub-fragment:

$$\mathcal{F}_{skew} = \frac{\frac{1}{l}\sum_{i=1}^{l}(y(k)_i - \mathcal{F}_{mean})^3}{(\mathcal{F}_{std})^3} \tag{7}$$

6. Kurtosis of the cycle in each window sub-fragment:

$$\mathcal{F}_{kur} = \frac{\frac{1}{l}\sum_{i=1}^{l}(y(k)_i - \mathcal{F}_{mean})^4}{(\mathcal{F}_{std})^2} \tag{8}$$

Entropy features measure the degree of uncertainty in the CAN IDs cycles with higher entropy signifying a more chaotic system. The entropy can also be used in determining other parameters such as negentropy, mutual information and KulbackLeibler divergence nongaussianity. The randomness in various attack types under study differs from attack-free traffic. But the fact that we are dealing with 3 classes of attacks with a close association, different entropy measures are employed, i.e., Shannon, sample and permutation entropy. Despite the similarities in the entropy algorithms, the theoretical ideas behind them are different. Shanon entropy is based on the concept of entropy from information theory in which it quantifies the magnitude of uncertainty (randomness) in the dataset in a purely mathematical way, without any knowledge regarding the source of the data; Sample entropy determines the complexity of a series of data using an alternative statistic that quantifies the randomness of the series to correct the problems of bias and lack of relative consistency [31]. Lastly, permutation entropy offers a method of calculating the complexity of a chaotic system in the presence of dynamical or observable noise with high speed, simplicity and robustness, while ensuring invariance regarding nonlinear monotonous transformation [32]. The entropies are express as follows:

1. Shanon entropy of each window's sub-segment given $y(k)_i \in y(k)$ and $\mathrm{p}(y(k)_i)$ is the probability associated with the values, is defined as:

$$\mathcal{F}_{shan} = -\sum_{i=1}^{l}\mathrm{p}(y(k)_i)\log\mathrm{p}(y(k)_i) \tag{9}$$

2. Sample entropy of each window's sub-segment given embedding dimension i, tolerance r, number of data points l, and distance function $d[y(k)_i(a), y(k)_i(b)](a \neq b)$:

$$\mathcal{F}_{samp} = -\log\frac{A}{B}, \tag{10}$$

where A is the number of having $d[y(k)_{i+1}(a), y(k)_{i+1}(b)] < r$, B is the number of having $d[y(k)_i(a), y(k)_i(b)]$.

3. Permutation entropy of each window's sub-segment in normalized form, given D as the embedded dimension, an ordinal pattern associated with y(k) defined as the permutation $\pi = r_0 r_1 \ldots r_{D-1}$, can be express as:

$$\mathcal{F}_{perm} = -\frac{1}{\log l!}\sum_{i=1}^{l!}\pi_i \log\pi_i \tag{11}$$

4) MODEL TRAINING AND AGGREGATION

Partial models are trained and build by a federating unit identify as a miner. The miner is assumed to have access to the framework and contain data from which features were extracted as described in section IV. Each miner creates a

local random forest model through bootstrap. During the training, the miner uses some of its datasets for validation. Once the training and validation are complete, the model is uploaded to IPFS, which generates hashes and returns them to the miner. The miner then uploads the hash and its user name into the smart contract. For the actual mining of the model hash into Ethereum, we proposed the use of mobile communication infrastructure as it has the computing capacity to mine. The training algorithms and smart contract are presented in Algorithm 4 and 5, respectively. The use of IPFS is to reduce the cost of hosting the model in the Ethereum network by simply upload the hash.

In the Model aggregation process, users first obtained the hash of the models uploaded by miners via smart contract, then use them to download all the models available or as the user wishes from the IPFS. After the download is complete, aggregation is now conducted at the user end, eliminating the cost of conducting the aggregation in Ethereum. The aggregation of partial models into a complete federated forest is conducted as described in Algorithm 6. We built the models using Sklearn python library implementation of random forest. As indicated in the algorithm, two major parameters: estimators (the collection of fitted sub-estimators) and the number of estimators (the number of trees in the forest) of each of the partial models are aggregated into $\mathcal{M}_{FL}$.

5) PERFORMANCE EVALUTION

We evaluated the proposed system training and testing using precision, recall and accuracy. The accuracy of the proposed model was compared with other TL machine learning methods as well as other state of art proposed algorithms. The evaluation metrics are express as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{15}$$

where TP, TN, FP, FN are the number of true-positive cases, true-negative cases, false-positive cases, and false-negative cases, respectively.

V. EXPERIMENTAL RESULTS

A. TESTBED EXPERIMENTAL RESULTS

The testbed was run on a VM host that runs a Linux OS (Ubuntu). The SDN emulator, Mininet, was configured to simulate the WAN, CPU and RAM allocated by the VM. Table 2 presents the environmental parameters. The CPU and RAM (MEM) performances were investigated based on traffic between peers of many nodes connected in "Double" bus topology consisting of miner and user node using the procedure described in [33]. The measurements were taken

Algorithm 2 Data Preprocessing

Input : Dataset, $\mathcal{D}$
Data Features, {*Timestep*,CAN_{ID}, *RTR, Msg*}
Output : CAN ID cycle extraction, $CycleID_{toSamples}$,
CalculateCycle(*id*)
Function $cycle_{byID} \leftarrow (group_{byID}[\text{"}CAN_{ID}\text{"}]==id,$
"*indexAsFeature*").***diff***(*1*)
Return $cycle_{byID}$
Drop {*Timestep*,CAN_{ID}, *RTR, Msg*}
Insert *index* **as** *Column*
$\mathcal{D}$["*indexAsFeature*"]‖ $\mathcal{D}$ *index*
Group $group_{byID} \leftarrow$ (*groupby*("CAN_{ID}", "*indexAsFeature*").*sum*()
Factorize $uniqueCAN_{ID} \leftarrow group_{byID}$
$cycle\_dict \leftarrow \{\}$
for *id* **in** $uniqueCAN_{ID}$ **do:**
$cycle \leftarrow CalculateCycle(id)$
$cycle\_dict$[str(*id*)]$\leftarrow cycle(id)$
end
Flatten *cycle_dict* into list and **add** to the dataset:
$cycle_{ID} \leftarrow$[item for in $\text{cycle}_{\text{dict}}$ for item in $\text{cylce}_{\text{dict[k]}}$]
$\mathcal{D}$ ["*Cylce_ID*"] ←Cycle_ID
Sort dataset back to *original sequence*:
Column_names ← ["CAN_ID",
"indexAsFeature",
"Cycle_ID"].
$Cylce\_ID \leftarrow group_{byID}$.sort_values(by=
["indexAsFeature"])
Window over "*Cylce_ID*" to extract ID cycle into
dataset with 1000 features
$window_{size} \leftarrow 1000$
$CylceID_{toSamples} \leftarrow [\ ]$
for *window* **in** *Cycle_ID*["*Cycle_ID*"].*rolling*
($window_{size}$) **do:**
$CylceID_{toSamples}$.append((*window*)[None])
end
Return $CylceID_{toSamples}$

during block forging operation for a fixed period of 180 seconds.

To test the CPU and RAM usage, transactions to the blockchain nodes were generated in a round-robin fashion- 50 transactions per burst. The bursts occurred every 2 seconds (sleep time) and were repeated for all transactions. This means that if there are 3000 transactions in 200 seconds, then the TPS load is 15. In real operation, the round-robin fashion is clients' behavior and this allows each client to be connected to a different node at a time.

The volume of traffic on one host is a good estimation of the volume of traffic experienced by other hosts [33]. Based on this hypothesis, we measured the performance as follows: read operation on the blockchain node through the SDN node was generated to fetch the current height of the blockchain head; polling state was then attained to wait for the completion of height increase; after which each of the new

Algorithm 3 Feature Extraction

Input: Dataset, $\mathcal{D}$
CAN ID cycle extraction, $CycleID_{toSamples}$
Output: Extraction features, $\mathcal{F} \leftarrow \{\mathcal{F}_{\min}, \mathcal{F}_{\max}, \mathcal{F}_{\text{std}}, \mathcal{F}_{\text{kur}}, \mathcal{F}_{\text{skew}}, \mathcal{F}_{\text{shan}}, \mathcal{F}_{\text{minsamp}}, \mathcal{F}_{\text{perm}}\}$

Function *FourierTransform(cycleID):*
  N$\leftarrow$sample.size
  w$\leftarrow$blackman(N)
  ft$\leftarrow$blackman(*fft*(cycleID*w))
Return *ft*
Function *CalculateFeatures(tx, attack_type)*
  $\mathcal{F}_{\min} \leftarrow min(tx)$, $\mathcal{F}_{\max} \leftarrow max(tx)$, $\mathcal{F}_{\text{std}} \leftarrow std(tx)$,
  $\mathcal{F}_{\text{kur}} \leftarrow kurtosis(tx)$, $\mathcal{F}_{\text{skew}} \leftarrow skew(tx)$,
  $\mathcal{F}_{\text{shannon}} \leftarrow shannon\_entropy(tx)$,
  $\mathcal{F}_{\text{samp}} \leftarrow sample\_entropy\ (tx)$,
  $\mathcal{F}_{\text{perm}} \leftarrow permutation\_entropy\ (tx)$
Return $\mathcal{F}_{\min}, \mathcal{F}_{\max}, \mathcal{F}_{\text{std}}, \mathcal{F}_{\text{kur}}, \mathcal{F}_{\text{skew}}, \mathcal{F}_{\text{shannon}}, \mathcal{F}_{\text{samp}}, \mathcal{F}_{\text{perm}}$
Function *ExtractFeature(data, attack_type)*
  Extracted_features $\leftarrow$[]
  for *row in range(data.shape[0])* **do:**
    data_fft $\leftarrow$*FourierTransform(data[row])*
    feature $\leftarrow$ *CalculateFeatures*(*data_fft, attack_type*)
    Extracted_features.append(*feature*)
  End
Return
dataset $\leftarrow$ load_dataset($\mathcal{D}$)

Algorithm 4 Federated Random Forest- Training (*Miner*)

Input: Dataset on miner *i*, $\mathcal{D}_i$
Extracted features $\mathcal{F} \leftarrow \{\mathcal{F}_{\min}, \mathcal{F}_{\max}, \mathcal{F}_{\text{std}}, \mathcal{F}_{\text{kur}}, \mathcal{F}_{\text{skew}}, \mathcal{F}_{\text{shan}}, \mathcal{F}_{\text{minsamp}}, \mathcal{F}_{\text{perm}}\}$
Label- $\text{attack}_{\text{type}}$,
Output: Partial Federated forest model, $\mathcal{M}_i$
Procedure BuildRandomForest($\mathcal{D}_i$, $\mathcal{F}$, attack_type)
  bootstrap sample of size, N, from the $\mathcal{D}_i$
  build recursively random tree, $\mathcal{L}_i$
    select *M* variable at *random*
    pick the best variable/split-point in *M*
    split into two daughter *node*:
      $left_{subtree} \leftarrow$BuildRandomForest
      ($\mathcal{D}'_i$, $\mathcal{F}'_i$, $\mathcal{Y}i_{left}$ attack_type)
      $right_{subtree} \leftarrow$BuildRandomForest
      ($\mathcal{D}'_i$, $\mathcal{F}'_i$, $\mathcal{Y}i_{right}$ attack_type)
      unit minimum, η_j, is reached
    Return *tree node*,$\{\mathcal{L}_i\}$
  append $\{\mathcal{L}_i\}$ to forest, $\{\mathcal{M}_i\}$
Return Partial Federated forest model, $\mathcal{M}_i$

Algorithm 5 Federated Random Smart Contract

Procedure ***Federated Forest Contract***
structure Model
  $modelPath_{ipfs}$
  $miner_{address}$
end
structure Miner
  $miner_{name}$
  $miner_{address}$
end
event Upload ($miner_{address}$, *index*):
event Download ($user_{address}$, *index*)
  //save miner download address, model indexed
mapping(address => uint256) *miner_model_index*
  //model track all model from user
mapping (address => mapping(uint256 => Model))
allmodel
Function Upload_model($modelPath_{ipfs}$, $miner_{name}$)
  index$\leftarrow$*miner_model_index*[msg.sender]
  allModel[msg.sender][*index*]
  $\leftarrow$Model($modelPath_{ipfs}$, msg.sender)
  emit *upload* (msg.sender, *index*)
Return *true*
Function get_model_path($miner_{address}$, *index*)
Return *allModel*[$miner_{address}$][*index*].modelPath_{ipfs}
Function get_model_minerName($miner_{address}$, *index*)
Return *minerDetails*[$miner_{address}$].$miner_{name}$

Algorithm 6 Federated Random Forest- Aggregation (*User*)

Input : Models downloaded by the users, $\mathcal{M}_i \leftarrow [\mathcal{M}_1 .. \mathcal{M}_i]$
Output : Complete aggregated Federated Random forest, $\mathcal{M}_{FL}$

Function *FederateRandomForest*($\mathcal{M}_1$, $\mathcal{M}_1$)
  $\mathcal{M}_1.estimator\_ += \mathcal{M}_2.estimator\_$
  $\mathcal{M}_1.n\_estimator \leftarrow \text{len}(\mathcal{M}_2.n\_estimator)$ $\mathcal{M}_1$.
Return
$miner_{address} \leftarrow [\mathcal{M}_{\text{addr1}} \ldots \mathcal{M}_{\text{addr}_i}]$
$model_{downloaded} \leftarrow [\,]$
For *miner* in $miner_{address}$ **do:**
  //from smart contract
  $download_{path}$
  $\leftarrow$contract.*get_model_path*($\mathcal{M}_{\text{addr1}}$, *index*)
  // from IPFS
  $download_{model} \leftarrow$IPFS.*get*($download_{\text{path}}$, *index*)
  $model_{downloaded}$.append($download_{model}$)
end
$\mathcal{M}_{FL} \leftarrow$Reduce(*FederateRandomForest*, $model_{downloaded}$)
Return $\mathcal{M}_{FL}$

blocks was added to a counter until the actual height of the blockchain was reached.

Our testbed utilized Geth, the Go official implementation of Ethereum. To deploy on a private testnet, Puppeth was used to create a genesis block and modified thereafter to enable its usage in any new experiment. Clique Proof of Authority consensus algorithm was configured to always set h1 and h2

TABLE 2. Environmental parameters.

Environment	Parameter	value
VM system specs	Number of CPU cores	2
	CPU clock frequency	3.10 GHz
	CPU model	Intel(R) Core(TM) i5-8600
	RAM allocated	4096 MB
	RAM block size	128 MB
	RAM type	DDR4
	DRAM	1197.1 MHz
	Hard disk type	SSD
	Hard disk size allocated	105 GB
Mininet hosts' interfaces specs	Medium delay	50 ms
	Standard deviation of delay	10 ms
	Distribution of delay	Gaussian
	Correlation between consecutive packets	25,00%
	Probability of loss	10,00%
	Probability of packet loss burst	65,00%
	Probability of bit corruption	0.01%
	Probability of reordering	10,00%
	Probability of duplication	500%

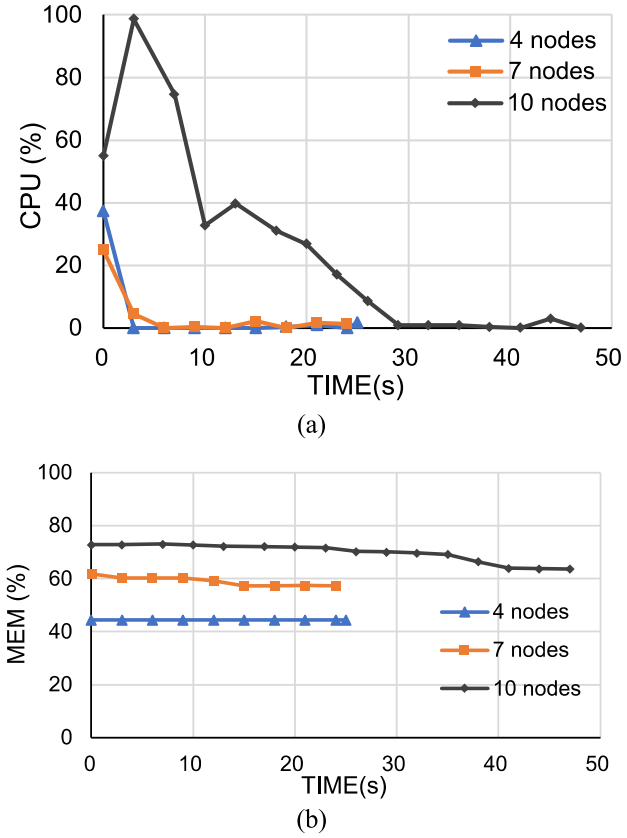


FIGURE 3. The testbed performance for the various number of nodes in the Double bus topology (a) CPU over time, (b) Main memory over the same time.

as signer nodes. Based on the algorithm, the nodes alternate in signing blocks. Before swap operations, only 10 nodes can run at the same time on the VM. As shown in Figure 3, good CPU and MEM usage can be achieved with more memory or reliance on swap operation.

B. FEDERATED FOREST MODEL EXPERIMENTAL RESULTS

1) DATA PREPROCESSING AND FEATURE EXTRACTION

The dataset entails CAN message frames that consist of CAN ID, DLC and data fields. Since ECU regularly broadcast messages, understanding the pattern/interval at the ECU sends a message can provide insight for our model to distinguish various forms of attack in the network. Therefore, we selected the CAN ID fields to extract our features as described in section IV. Table 3 provides the composition considered for our testbed.

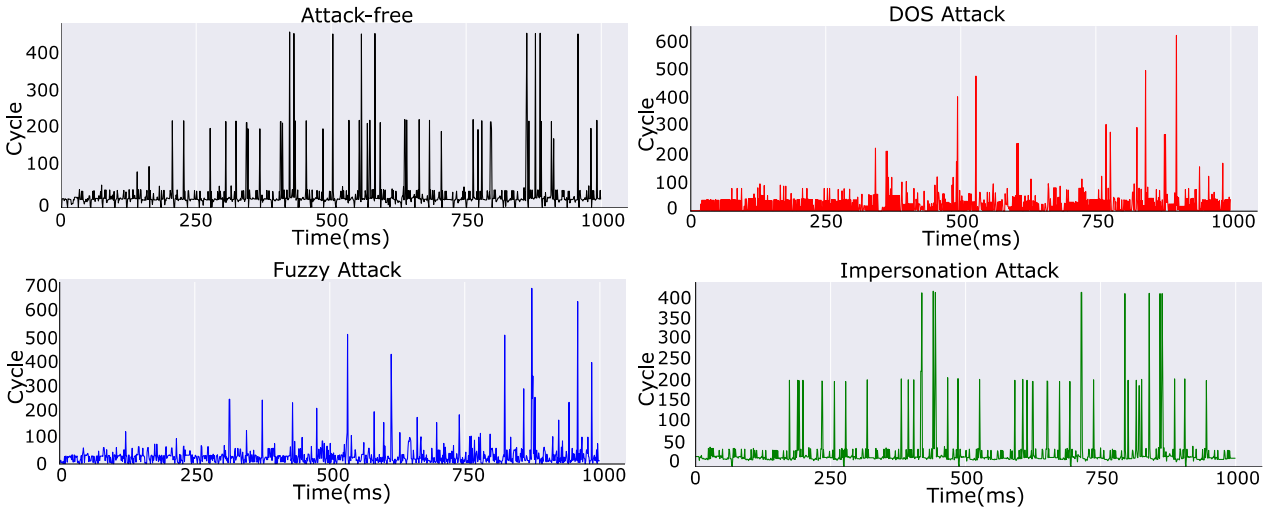


FIGURE 4. CAN ID cycles for the classes of attack under consideration.

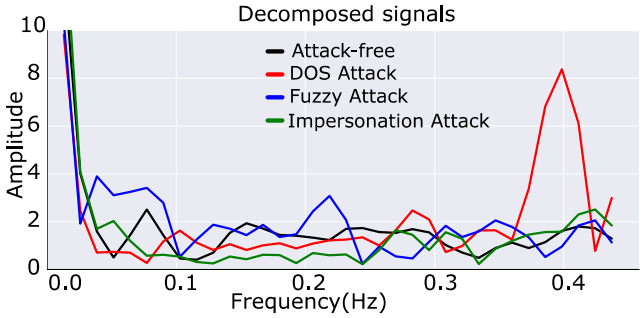


FIGURE 5. CAN ID cycles decomposed by fourier transform for the classes of attack under consideration.

As presented in Algorithm 1, we computed the cycle of occurrence of each ID in each class of the dataset. Figure 4 presents the pattern of the cycle for each attack class under consideration. From the plot of attack-free and impersonation attack packets, the IDs' cycle of occurrence can be grouped into 3- short, medium and long cycles- of which the short and long cycles have the most and least occurrence, respectively. The DoS and Fuzzy attack packets both have a predominantly short cycle that shows regular message injection to cause a denial of service. However, DoS shows a higher frequency in the cycles. Based on the close similarities between the attack classes, further transformation is needed to help render it more distinguishable.

We employed time-frequency analysis-Fourier transformation- to decompose the CAN ID cycle. The packets traffic stream continuously, therefore, are not explicitly segmented into sub-fragments associated with the transmission pattern of the CAN ID. Hence, we segment the cycle into several samples of equal length by sliding a window of fixed length through the entire traffic. The window is parameterized

TABLE 3. Composition of the training and test sets.

Attack type	No. of Message	Training set	Testing set
DoS Attack	50,000	60%	40%
Fuzzy attack	50,000		
Impersonation Attack	50,000		
Attack-free	50,000		

by window length l and step length s. Setting these parameters required a trade-off between accuracy and delay; and since our main objective is accuracy, we set the window length for 1000ms and step length to 1ms to effectively capture the subtle details considering the close association of the attack patterns. Figure 5 shows the time-frequency decomposition of the traffic by Fourier transformation. The transformation packet traffic shows more distinction between the classes of the packets.

Our dataset now contains a high-dimensional space of 1000 features after the windowing. To reduce the high-dimensionality of the data space into low-dimensional space and effectively harness useful insight, the statistics and entropies were deployed. The statistical and entropy values extracted include minimum, maximum, mean, skewness, kurtosis and Shannon, Sample, Permutation, respectively. Figure 6 presents the distribution of the extracted features. The extracted features are quite distinct from each other- the density of the statistical features is less than 1 range while that of the entropies is greater than 1.

2) FEDERATED FOREST INTRUSION DETECTION MODEL PERFORMANCE

The federated forest model training and testing performances are analyzed in this section. We split the dataset into equal sample sizes based on the number of miners. The number

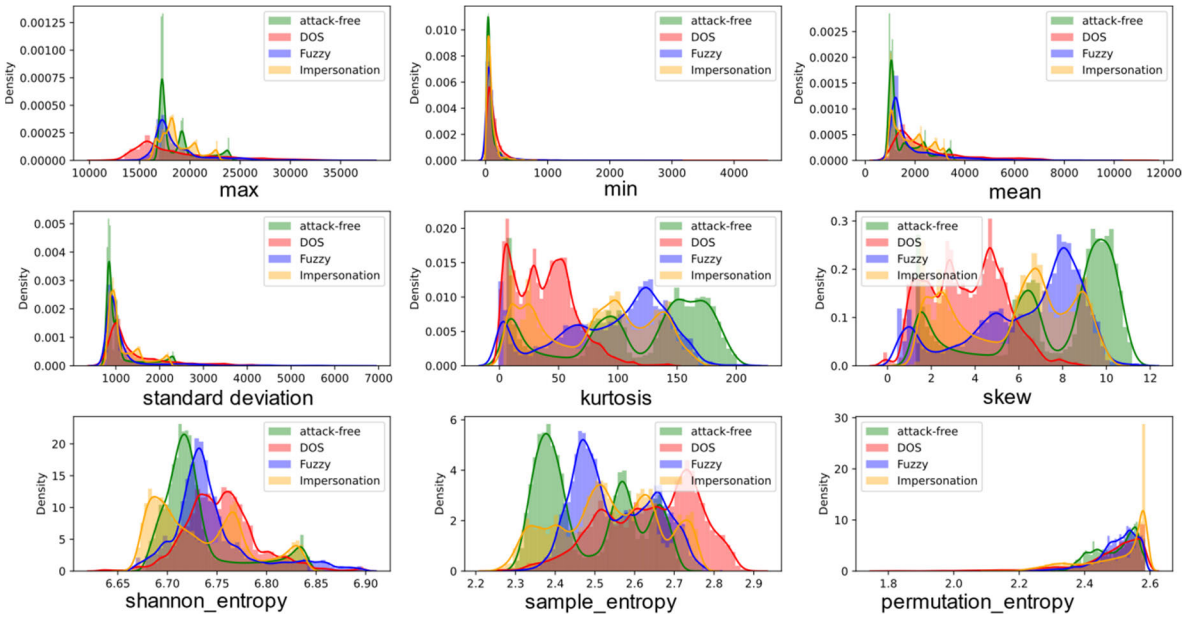


FIGURE 6. CAN ID cycles for the classes of attack under consideration (see eq 3-10).

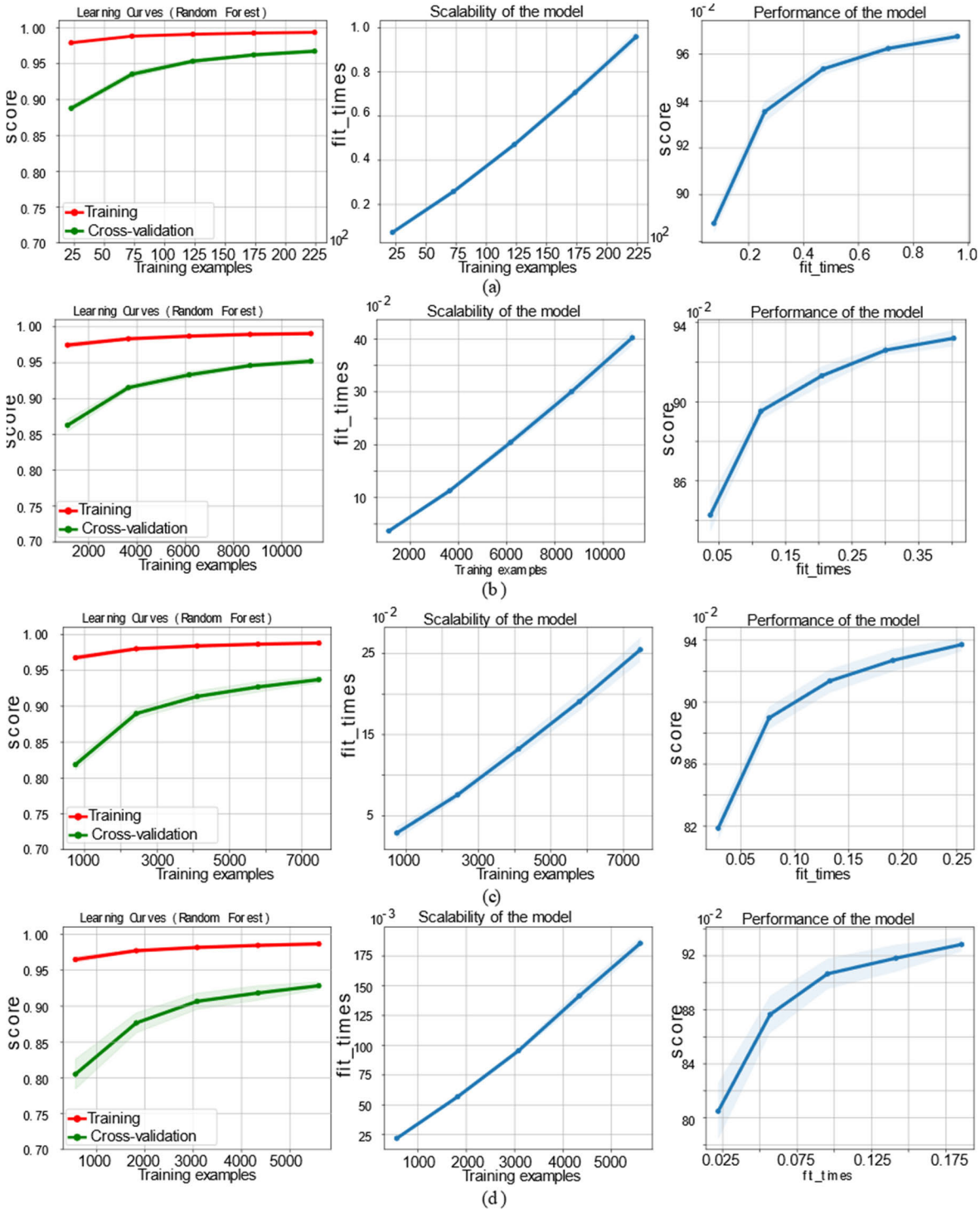
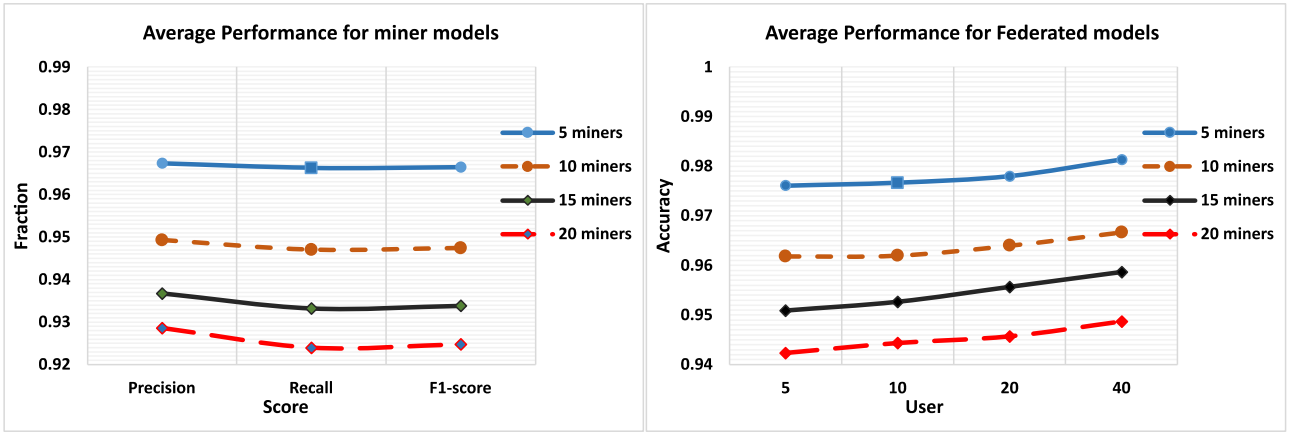


FIGURE 7. Miner nodes best training performances: (a) 5 miners (b) 10 miners (c) 15 miners (d) 20 miners.

of miners considered for the experiment includes 5, 10, 15 and 20 miners. Each miner trains its model on its data. At each miner, the dataset was split into training, testing and validation sets. The overall model was evaluated using 10-fold cross-validation. Figure 7 shows the best training and validation session alongside the model scalability and overall score for each miners' set. For the training session, the models across all the miners' sets approximately experience the same learning progress- the performance started at more than 0.95 accuracy and progressively improved to about 0.99. However, the validation performance varied, with 5 and 20 miners' sets reaching an accuracy of more than



(a) (b)

FIGURE 8. Test performance (a) Average testing performance by each miner on its partial model and local test data (b) Federated forest performance evaluation for different users and miners.

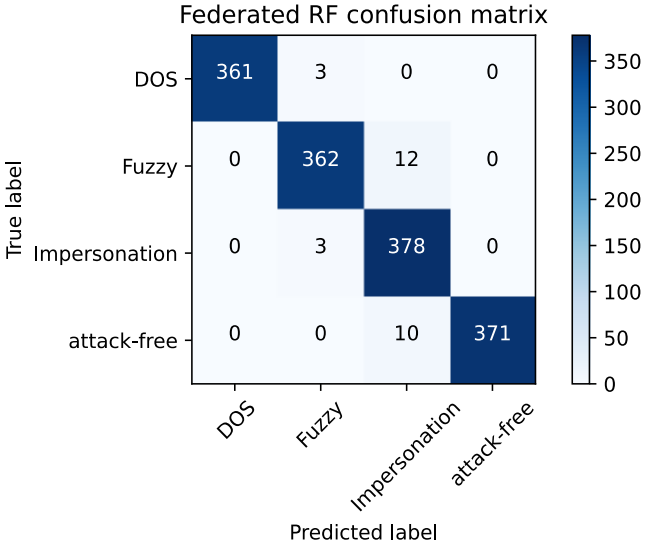


FIGURE 9. Confusion matrix of the overall best BFF-IDS model (5 miners' set FL model with 40 users).

0.95 and about 0.93 as the best and least performing models, respectively.

We conducted separate testing on each of the models. The average performance for each of the miners' sets in terms of precision, recall and F1-score was as presented in Figure 8 (a). 5 miners recorded the best performance with an average score of about 0.97 across the performance metrics. The least average performance of between 0.92 and 0.93 was recorded by the 20 miners. We associated the huge variation in performance with the size of training samples- 5 miners have the highest numbers of samples which provides enough examples for better generalization. Consequently, 20 miners' set has the least performance due to the smallest size of sample among the set of miners.

The train models are then uploaded into IPFS and the location hashes into the blockchain. The users access the location hash from the blockchain network and download the models from the IPFS using the hash. The aggregation of the downloaded models into the federated model is done at the user's node. Similar to the miners' training, the testing dataset is split into equal samples based on the number of users. The number of users under investigation were 5, 10, 15 and 20 users. The average accuracy performance of each set of users for each miners' sets was evaluated as shown in Figure 8 (b). The federated model from the 5 miners set recorded the highest performance ranging from 0.9761 to 0.9813. The least performance ranging between 0.9423 and 0.9457 was scored by the federated model of 20 miners set. The results reflect the amount of data used in training the model. The 5 miners

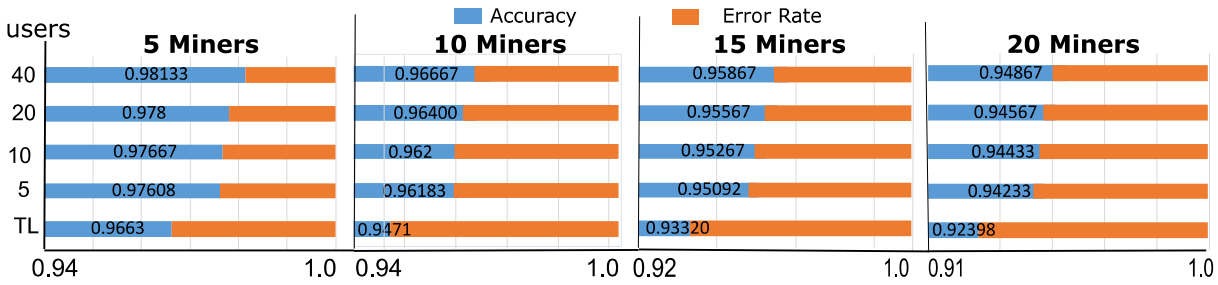


FIGURE 10. Federated Forest model and traditional Learning (TL) model.

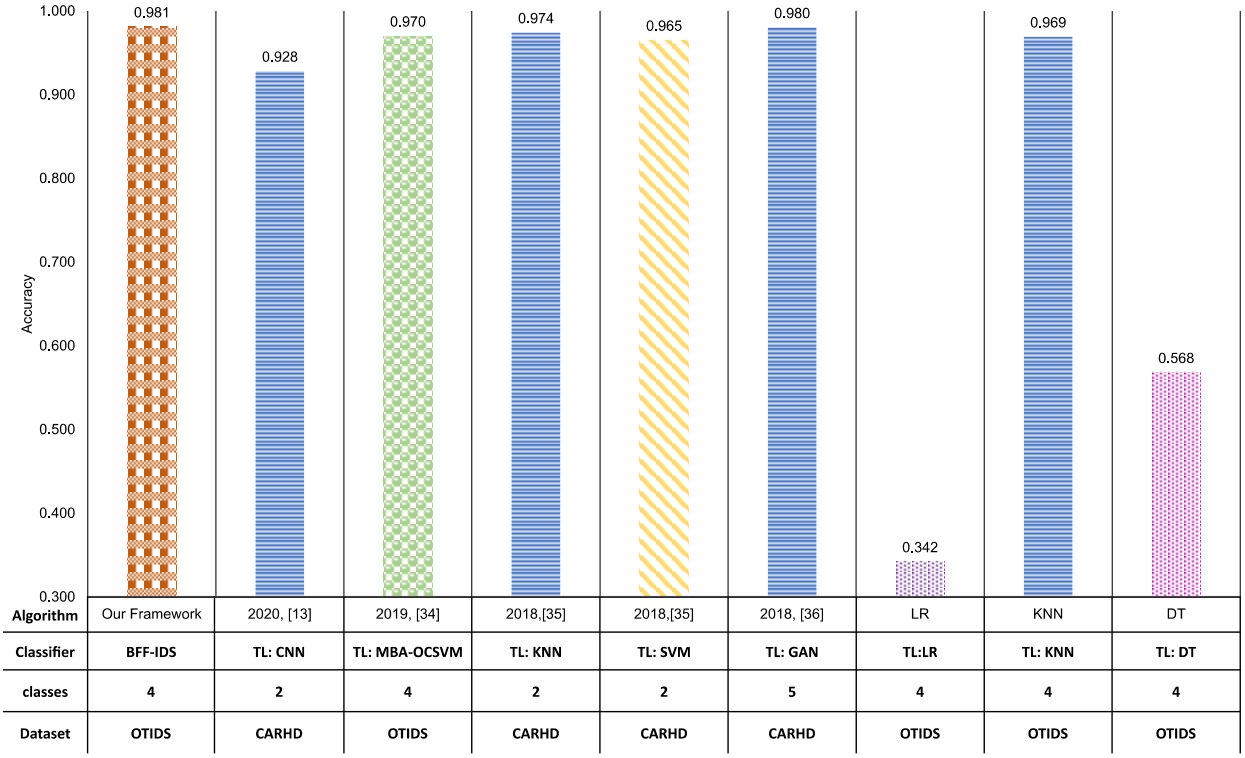


FIGURE 11. Performance evaluation of our proposed system against other detectors and related works in terms of accuracy.

set federated model appears superior to all other models due to the larger size of data that was used to train the model. Even though the federated model is the aggregation of the individual model in a set, the training performance is critical to how the model performs after aggregation.

Comparing the average accuracy performance of each model set (TL) and its corresponding federated model (FL), the federated model comes out superior across all the number of users. The best performing FL model from 5 miners' sets outperform the TL model in all cases with an average accuracy of 0.0073, the least performing FL model of 20 miners' sets outperformed its corresponding TL model with the average accuracy of 0.0012. Figure 9 presents the confusion matrix of the overall best model recorded by 5 miners set FL model with 40 users. From the figure, the highest misclassification comes from 12 fuzzy attack samples being classified as impersonation. Just 3 sample attacks from DoS and impersonation attacks were misclassified as fuzzy attacks, while about 10 attack-free samples were misclassified as impersonation attacks. Meanwhile, Figure 10 shows the comparison between the FL and TL models across all users' scenarios.

3) THE FEDERATED FOREST MODEL EVALUATION AGAINST OTHER MODELS

In this section, we evaluated our best model against other detectors and related works built using the TL method. As shown in Figure 11, our model outperformed other detectors such as Logistic Regression (LR), K-nearest neighbor (KNN), and Decision Tree (DT) that were trained using the TL approach. The LR, DT, K-NN and LR, recorded lower accuracies of 0.342, 0.969 and 0.568, respectively. Equally, our BFF-IDS outperformed other models proposed in [13], [34]–[36], which recorded average accuracies of 0.928, 0.970, (0.974, 0.965) and 0.980, respectively. Although only [34] used the same set of data as our study, others' datasets were equally collected and created from the same source. In addition, only [36] classified more labels than us with less accuracy.

The detectors trained using TL have more data during training when compared to FL where the datasets were splits among the miners. Despite the result (see figure 8) suggesting that split data across miners can reduce performance, our FL model was able to outperform other models and works-However, in practice, the FL model is expected to have more datasets than the TL model for training.

VI. DISCUSSION

The testbed provides a resilient network for inter CAN IDs federation with high security afforded by the blockchain. The proposed integration of IPFS to host models other than the blockchain reduces the cost (gas, ether) that may be incurred when using the blockchain.

The simulation shows that the testbed efficiently utilizes CPU and memory resources and will be adequate in actual application. Thus, the hybrid architecture leverages the benefit of SDN in providing network management flexibility, blockchain in providing security and FL learning in

harnessing the benefits of data available in individual vehicle or manufacturers. This is the first work, to the best of our knowledge to propose such a hybrid architecture for in-vehicle network security.

We detected three forms of attacks; fuzzy, DoS, and Impersonation, from attack-free traffic. Unlike other existing works, we applied Fourier transformation to observe CAN IDs cycle in the frequency domain and expose the distinction between traffic patterns in the attack class. Similarly, our proposed combination of statistical and entropy features proved effective in extracting the features and describing of the subtle difference in the pattern. Meanwhile, the FL method also proved effective for classifying the attacks with over 0.98 accuracy.

Comparing with some previous approaches such as [14], which utilized four features (CAN messages, number of messages, sum of DLC and bandwidth) from the CAN bus logs, we employed only the CAN ID cycle.

Although some works have used the CAN IDs field before as indicated in Table 1, this study applied FFT to transform the cycle. Similarly, unlike the previous studies, we combine the statistics and entropies to extract features from the transform data using a window. In addition, this is the first work that proposed the FL concept for CAN bus IDS.

VII. CONCLUSION

In this study, we proposed a blockchain-based Federated Forest SDN-enabled IDS that enables the training of models for IDSs that support the detection of intrusions in CAN in in-vehicular systems, while protecting the confidentiality of sensitive data. We used Federated Learning (FL) to create a random forest model, where manufacturers and car owners provide partially trained models that are integrated, allowing them to keep the underlying data confidential. We expect that this will improve their willingness to participate. We used blockchain technology to reduce the risk of poisoning the models. The testbed shows efficient use of memory and CPU resources for the proposed system. We applied Fourier transformation to CAN ID cycles and extracted statistical and entropies features. The extracted features resulted in a high detection rate of closely related forms of attack. To the best of our knowledge, this is the first work that proposed a blockchain-based federated learning framework via SDN for intrusion detection. In addition to protecting data confidentiality, willingness to participate can be further improved by providing flexible incentives to reward federating units based on their contribution to the building of the model. For sustainability in the market, the model will be on a pay-to-use basis. Consequently, the proposed system leverages the benefits of SDN in providing a flexible configuration for unforeseen network requirements, blockchain in providing security and FL learning in harnessing the benefits of data available in individual vehicle or manufacturers. Although the proposed system performed well in the testbed, more is needed to address the problem of privacy during training to ensure that the training data cannot be inferred from the model before exchange and subsequent federation. Furthermore, future research should focus on further evaluation of the system in practice.

ACKNOWLEDGMENT

The authors are grateful for the valuable feedback and comments provided by Prof. Muhammed Bashir Muazu of the Department of Computer Engineering, Ahmadu Bello University, Zaria, Nigeria, and the anonymous reviewers.

REFERENCES

[1] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," *Black Hat USA*, vol. 2014, p. 94, Aug. 2014.

[2] F. Sakiz and S. Sen, "A survey of attacks and detection mechanisms on intelligent transportation systems: VANETs and IoV," *Ad Hoc Netw.*, vol. 61, pp. 33–50, Jun. 2017.

[3] S.-F. Lokman, A. T. Othman, and M.-H. Abu-Bakar, "Intrusion detection system for automotive controller area network (CAN) bus system: A review," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, p. 184, Dec. 2019.

[4] F. Fenzl, R. Rieke, Y. Chevalier, A. Dominik, and I. Kotenko, "Continuous fields: Enhanced in-vehicle anomaly detection using machine learning models," *Simul. Model. Pract. Theory*, vol. 105, Dec. 2020, Art. no. 102143.

[5] M. L. Han, B. I. Kwak, and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," *Veh. Commun.*, vol. 14, pp. 52–63, Oct. 2018.

[6] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. 20th USENIX Secur. Symp.*, vol. 4, San Francisco, CA, USA, 2011, pp. 447–462.

[7] T. Hoppe, S. Kiltz, and J. Dittmann, "Applying intrusion detection to automotive IT-early insights and remaining challenges," *J. Inf. Assurance Secur.*, vol. 4, no. 6, pp. 226–235, 2009.

[8] I. Berger, R. Rieke, M. Kolomeets, A. Chechulin, and I. Kotenko, "Comparative study of machine learning methods for in-vehicle intrusion detection," in *Computer Security* (Lecture Notes in Computer Science), vol. 11387, S. Katsikas *et al.*, Eds. Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-030-12786-2_6.

[9] D. Kosmanos, A. Pappas, L. Maglaras, S. Moschoyiannis, F. J. Aparicio-Navarro, A. Argyriou, and H. Janicke, "A novel intrusion detection system against spoofing attacks in connected electric vehicles," *Array*, vol. 5, Mar. 2020, Art. no. 100013.

[10] M. Aloqaily, S. Otoum, I. A. Ridhawi, and Y. Jararweh, "An intrusion detection system for connected vehicles in smart cities," *Ad Hoc Netw.*, vol. 90, Jul. 2019, Art. no. 101842.

[11] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Res. Blog*, vol. 3, Oct. 2017. [Online]. Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[12] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020.

[13] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Veh. Commun.*, vol. 21, Jan. 2020, Art. no. 100198.

[14] B. I. Kwak, M. L. Han, and H. K. Kim, "Cosine similarity based anomaly detection methodology for the CAN bus," *Expert Syst. Appl.*, vol. 166, Mar. 2021, Art. no. 114066.

[15] M. Muter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 1110–1115.

[16] M. Marchetti, D. Stabili, A. Guido, and M. Colajanni, "Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms," in *Proc. IEEE 2nd Int. Forum Res. Technol. Soc. Ind. Leveraging*, Oct. 2016, pp. 1–6.

[17] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2016, pp. 63–68.

[18] A. Tomlinson, J. Bryans, S. A. Shaikh, and H. K. Kalutarage, "Detection of automotive CAN cyber-attacks by identifying packet timing anomalies in time windows," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops (DSN-W)*, Jun. 2018, pp. 231–238.
[19] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," in *Proc. World Congr. Ind. Control Syst. Secur. (WCICSS)*, Dec. 2015, pp. 45–49.
[20] T. Kuwahara, Y. Baba, H. Kashima, T. Kishikawa, J. Tsurumi, T. Haga, Y. Ujiie, T. Sasaki, and H. Matsushima, "Supervised and unsupervised intrusion detection based on CAN message frequencies for in-vehicle network," *J. Inf. Process.*, vol. 26, pp. 306–313, Dec. 2018.
[21] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 911–927.
[22] H. Lee, S. H. Jeong, and H. K. Kim, "OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame," in *Proc. 15th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2017, pp. 57–5709.
[23] X. Ying, S. U. Sagong, A. Clark, L. Bushnell, and R. Poovendran, "Shape of the cloak: Formal analysis of clock skew-based intrusion detection system in controller area networks," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 9, pp. 2300–2314, Sep. 2019.
[24] M. Marchetti and D. Stabili, "Anomaly detection of CAN bus messages through analysis of ID sequences," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 1577–1583.
[25] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol. (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.
[26] P. K. Sharma and J. H. Park, "Blockchain based hybrid network architecture for the smart city," *Future Gener. Comput. Syst.* vol. 86, pp. 650–655, Sep. 2018.
[27] H. Kim and E. A. Lee, "Authentication and authorization for the Internet of Things," *IT Prof.*, vol. 19, no. 5, pp. 27–33, 2017.
[28] P. K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park, "DistBlockNet: A distributed blockchains-based secure SDN architecture for IoT networks," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 78–85, Sep. 2017.
[29] S. Parsons, A. M. Boonman, and M. K. Obrist, "Advantages and disadvantages of techniques for transforming and analyzing chiropteran echolocation calls," *J. Mammal.*, vol. 81, no. 4, pp. 927–938, Nov. 2000.
[30] R. Alazrai, M. Momani, H. A. Khudair, and M. I. Daoud, "EEG-based tonic cold pain recognition system using wavelet transform," *Neural Comput. Appl.*, vol. 31, no. 7, pp. 3187–3200, Jul. 2019.
[31] A. Delgado-Bonal and A. Marshak, "Approximate entropy and sample entropy: A comprehensive tutorial," *Entropy*, vol. 21, no. 6, p. 541, May 2019.
[32] C. Bandt and B. Pompe, "Permutation entropy: A natural complexity measure for time series," *Phys. Rev. Lett.*, vol. 88, no. 17, 2002, Art. no. 174102.
[33] M. C. Feliciano. (2020). *Analysis of Blockchain Technologies and Benchmarking of NXT and Ethereum in Emulated Network Environment*. [Online]. Available: https://github.com/Shotokhan/blockchain-benchmarking-on-mininet
[34] O. Avatefipour, A. S. Al-Sumaiti, A. M. El-Sherbeeny, E. M. Awwad, M. A. Elmeligy, M. A. Mohamed, and H. Malik, "An intelligent secured framework for cyberattack detection in electric vehicles' CAN bus using machine learning," *IEEE Access*, vol. 7, pp. 127580–127592, 2019.
[35] A. Alshammari, M. A. Zohdy, D. Debnath, and G. Corser, "Classification approach for intrusion detection in vehicle systems," *Wireless Eng. Technol.*, vol. 9, no. 4, pp. 79–94, 2018.
[36] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN based intrusion detection system for in-vehicle network," in *Proc. 16th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2018, pp. 1–6.

IBRAHIM ALIYU (Member, IEEE) received the B.Eng. and M.Eng. degrees in computer engineering from the Federal University of Technology, Minna, Nigeria, in 2014 and 2018, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Engineering, Chonnam National University, Yeosu, South Korea, under the supervision of Prof. Chang Gyoon Lim. His research interests include federated learning, data privacy, network security, SDN, and BCI. He was a recipient of the 2017 Korean Government Scholarship Program Award.

MARCO CARLO FELICIANO received the B.Eng. degree from the University of Naples Federico II, Naples, Italy, in 2020, where he is currently pursuing the M.Eng. degree in computer engineering. His studies focus on cyber security, in particular system, network, and software security. He attended a cyber-security training program called "Cyberchallenge," in 2020, and following that he contributed to the creation of "pwnthenope," a team involved in online cyber security competitions. He also contributed to different open source projects.

SÉLINDE VAN ENGELENBURG received the master's degree in artificial intelligence from Utrecht University, with a focus on logic and intelligent systems, and the Ph.D. degree in the field of ICT from the Delft University of Technology. Her Ph.D. research was on designing large-scale context-aware architectures for information sharing to enhance security and safety in international container shipping. She is currently a Researcher with the Faculty of Technology, Policy and Management, Delft University of Technology. Her research interest includes the role of new technological developments in cyber security. Her research also focused on using distributed ledger technology to support information sharing in supply chains. In addition, she developed a new method for designing context-aware systems in complex multi-stakeholder environments.

DONG OK KIM received the Ph.D. degree from the Department of Control and Instrumentation Engineering, Chosun University, Gwangju, South Korea, in 2002. He previously worked in Korea Esen Company, South Korea, as the Technical Director. He also worked as a Researcher with the Department of Precise-Engineering Research, Tokyo Institute of Technology, Japan, and as a BK21 Research Professor with Chonnam National University, Gwangju. Since 2011, he has been consulting as a technology transfer agent. Since 2009, he has been working as a technology consultant for local and metropolitan governments. In 2008, he joined Jeonnam Technopark, where he is currently participating in five projects related to energy innovation as the Chief of Department of National Innovation Cluster Support Center.

CHANG GYOON LIM (Member, IEEE) received the Ph.D. degree from the Department of Computer Engineering, Wayne State University, USA, in 1997. Since September 1997, he has been working for major in computer engineering with Chonnam National University, Yeosu, South Korea, as a Professor. He was the Director of Home Robot Center, Gwangju Techno Park. He also leads various research projects of his interest areas. His current research interests include BCI, machine learning, soft computing, intelligent robot, the IoT, cloud computing, demand response, and embedded software. He is a member of Korean Society for Internet Information. Simultaneously, he acts as the committee member of several public and private regional institutions. He is the Chair of Gwangju-Jeonnam Cloud Computing Leaders Forum and an Editor of *Transactions on Internet and Information Systems (TIIS)*.