Blockchain-based Federated Forest for SDN-enable In-vehicle Network Intrusion Detection System

Ibrahim Aliyu¹, Marco Carlo Feliciano², Sélinde van Engelenburg³, Dong Ok Kim⁴ and Chang Gyoob Lim¹

¹Department of Computer Engineering, Chonnam National University, Yeosu, South Korea
²Department of Electrical and ICT Engineering, University of Naples Federico II
³Faculty of Technology, Policy and Management at Delft University of Technology
⁴National Innovation Cluster Support Center, Jeonnam Technopark, Haerong-myeon Suncheon, Jeonnam, South Korea

Corresponding author: First A. Author (e-mail: cglim@jnu.ac.kr).

This research was financially supported by the Ministry of Trade, Industry and Energy(MOTIE) and Korea Institute for Advancement of Technology (KIAT) through the National Innovation Cluster R&D program (P0016223).

ABSTRACT 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) which coordinate, monitor and control loads of internal vehicle components such as engine system, brake system and telematics system through the exchange of information among them. CAN operates by broadcasting packets to its bus. This means that all nodes and ECUs attached to the bus can receive the packets, without an authentication mechanism for identifying the legitimacy/source of packets. This makes it vulnerable to attacks. An Intrusion Detection System (IDS) can be used to detect attacks on CAN. Machine learning for the IDS, in particular, would be useful for creating models to detect non-linear attack patterns. However, car manufacturers and owners are might not be willing to just share the sensitive information required for training the models. In this paper, we propose a Blockchain-based Federated Forest Software-Defined Networking (SDN)-enabled Intrusion detection system (BFF-IDS) for an In-vehicle network to address the problem of sharing the sensitive CAN data. Due to the limited scalability of blockchain, InterPlanetary File System (IPFS) was used to host the models, while a hash of the model and a pointer to its location was stored and shared via the blockchain. The SDN provides the dynamic routing of packets and model exchanges from IPFS through the blockchain. In the detection model system, a Federated Learning (FL) method creates a radom forest model in a distributed manner by aggregating partially trained models that were trained by individuals with their data kept confidential during the process. 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. With this proposed system, manufacturers and car owners may be willing to contribute to the training of the models, as their sensitive data is protected due to the use of FL. 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. The evaluation was conducted by performing experiments in 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.

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) which coordinate, monitor and control loads of internal...
vehicle components such as engine system, brake system and telematics system through the exchange of information among them [1]. Nevertheless, the exchange of information within the CAN bus system has exposed it to external penetration threats which can harm drivers, the operating environment as well as other vehicles [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. Meanwhile, nodes and ECUs have no authentication mechanism for identifying the legitimacy or source of packets which makes them vulnerable to attacks. When the CAN is compromised, attackers can inject malicious messages to ECUs that can trigger physical action such as steering and braking, manipulate speedometer display information, etc [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 can be launched directly through the On-Board Diagnostic II (OBD-II) port or indirectly through firmware such as the media player. The breach of CAN through OBD-II port and firmware (MP3) was reportedly successful in a car thief and vehicle control [5, 6]. Concerning the 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, in particular, would be adequate in learning non-linear attack patterns. However, the method of training the IDS still requires attention 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 different entities 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 facilitate sharing of information among vehicles in ever-evolving network complexity.

In previous research, 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]. Although [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 training on each vehicle. This research addresses the problem 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. Besides, individuals (car owners) too may be reluctant to share information due to privacy concerns. To keep training data private, Google proposed 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 which can then be downloaded by devices after training. 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 the 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 paper, we proposed a blockchain-based federated forest SDN-enable Intrusion detection system (BFF-IDS) for an In-vehicle network to address the problem of sharing the sensitive CAN data. The federated learning system creates a random forest model in a distributed manner by aggregating partially training models. The system also addresses the problem of cost by using IPFS to store the model by the hash of the model location is store and exchange over the blockchain. In addition, SDN is also proposed in this work 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 cycles sequence were extracted and transform 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). Multiple statics and entropy features were employed to handle the high complexity and non-linearity in CAN bus traffic. The system was evaluated using precision, recall F1-score and accuracy. In addition, we 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-enable intrusion detection in an In-vehicular network. To the best of our knowledge, this is the first attempt 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. The detailed algorithms for implementation are present.

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

The remainder of the paper is organised as follows: In the next section, we discussed the related literature. In section 3, federated learning and problem formulation is presented. Whereas, section 4 presents the concept of blockchain-based SDN-enable IDS, including the methodology, algorithms for data preprocessing, feature extraction and FL training and testing. The Results, discussion and conclusion are presented in section 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 difficult due to the security concern by the manufacturers which makes 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. A lot of efforts have been made in the utilization of the transmission pattern of the ECU in the data link layer of the CAN bus. Very 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. However, 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. For the utilization of the CAN ID field to detect attacks through the analysis of CAN traffic patterns, Deep Convolutional Neural Network (DCNN) was proposed [13]. An optimized model was developed to solve the complexity problem while archiving a high accuracy rate. The detection of attacks based on the entropy of CAN bus traffic to distinguish normal traffic from malicious ones has also been proposed [15]. The proposed method shows significant performance for different attack scenarios. Similarly, entropy was utilized 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, the time interval was explored in detecting attacks based on the time changes in CAN bus traffic [18]. 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 he the CAN bus [5]. The method shows a better detection rate against CAN IDs with a short cycle. The frequencies of packets were also explored to detect anomalies by sliding window to measure the inter-packets timing [19]. In addition, statistical analyses were employed for the detection of anomalies from CAN IDs frequencies [20]. Besides, clock-wise, remote frame and network time protocol (NTP) based features have also been proposed for intrusion detection in CAN [21-23]. The summary of the related works in given in Table I.

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, \{N_i\}_{i=1}^n, who wish to build a strong model by the consolidation of their data \{D_i\}_{i=1}^n, conventionally train the model, \mathcal{M}_{\text{SUM}}, by combining the data, D = \bigcup_{i=1}^n D_i. However, in the federated learning concept, the owner collaboratively trains the model, \mathcal{M}_{\text{FL}}, in such a way that any data owner N_i does not reveal its data D_i to others while ensuring that its performance, P_{\text{FL}}, is close to that of \mathcal{M}_{\text{SUM}} model performance, P_{\text{SUM}}. Formally, given \delta as a non-negative real number, if

\begin{equation}
| P_{\text{FL}} - P_{\text{SUM}} | < \delta
\end{equation}

the federated model is said to have \delta -accuracy loss. Federated learning can be categorized based on the distribution nature of the data [25]. Let matrix 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 then 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 into 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.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/
The dataset $\mathcal{D}$ is distributed among $K$ parties $\mathcal{N}_1, \ldots, \mathcal{N}_K$. Only a subset of the data $\mathcal{D}_k \subseteq \mathcal{D}$ with $\mathcal{N}_k$ samples are utilized by $k^{th}$ party, where $k \in [1, K]$. We assume that the data points in $\mathcal{D}$ are disjoint, i.e., $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$ for all $i \neq j$. The main goal is to build a detector from the complete dataset while minimizing $\delta$-accuracy loss.

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

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

Due to the immense global population move to cities, the architectural design of the network of smart cities in urban areas is increasingly faced with challenges in terms of latency, scalability, network bandwidth usage data privacy and security [26]. These challenges faced by smart city network architecture need to be addressed for the sustainability of 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 will guarantee 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 wherein each vehicle IDS can interact through the hybrid system. Hence, we take the advantages of the hybrid architecture to proposed 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.

The hybrid architecture of the scalable smart city network for the IDS is presented in Figure 2. 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 which 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 of the vehicles is assumed to be nodes 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-work. 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 propose the used of IPFS to upload the model while the hash is exchange over the blockchain. This way the cost incur 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 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, ease deployment in the network infrastructure with high agility and security. The application plane consists of critical stakeholders such as network management agency, identification/key providers for participants, and threat intelligent agencies that oversee the identification of attack trends and policies. The Algorithm for the implementation of the testbed is given in Algorithm 1.

### TABLE I

<table>
<thead>
<tr>
<th>Year/Ref.</th>
<th>FFT</th>
<th>Features</th>
<th>Attack classes</th>
<th>Blockchain</th>
<th>FL</th>
<th>SDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021, [14]</td>
<td>×</td>
<td>Unique CAN IDs, Messages, DLC, bandwidth</td>
<td>4</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>2020, [13]</td>
<td>×</td>
<td>CAN IDs Sequence</td>
<td>2 (for each class)</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>2017, [24]</td>
<td>×</td>
<td>Entropy (CAN IDs)</td>
<td>3</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>2016, [16]</td>
<td>×</td>
<td>Frequency</td>
<td>3</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>2015, [19]</td>
<td>×</td>
<td>Clock skew</td>
<td>2</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>2016, [21]</td>
<td>×</td>
<td>NTP</td>
<td>2</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>2017, [22]</td>
<td>×</td>
<td>CAN IDs, Entropy(CAN IDs)</td>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2019, [23]</td>
<td>✓</td>
<td>Statistics(CAN IDs), Entropy(CAN IDs)</td>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

This work has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI: 10.1109/ACCESS.2021.3094365, IEEE Access
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 in the following.

1) **CAN BUS DATASET**
The attack scenario considered in the paper include threats that are capable of manipulating in-vehicle nodes remotely/physically by the injection of unauthenticated or malicious message in the CAN bus. CAN-intrusion dataset (OTIDS), which is publicly provided by the Hacking and Countermeasure Research Lab at Korea University, is used in this study [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 ‘0x000’ CAN ID, Fuzzy Attack was generated by injecting messages of spoofed random CAN ID and DATA values, while impersonation attack was conducted by injecting messages from impersonating node with arbitration ID, ‘0x164’. Attack-free traffic was recorded from normal can messages.
In the data preprocessing, the CAN IDs sequence 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.

Algorithm 1 BFF-IDS

(I) Blockchain Initialization:
(a) Network creation:
Miner nodes, \(\ldots, n_m, \ldots\) and user node \(n_u, \ldots, n_r\) created in the peer-to-peer network

(II) SDN Initialization:
(a) Create network topology based on the Blockchain network:

Function

For \(i\) range of \(\text{number of host}\):

\[\text{node}[\ldots, n_m, \ldots] \rightarrow \text{addhost}(\ldots, n_m)\]

For \(i\) range of \(\text{number of host}\):

\[\text{node}[\ldots n_u, \ldots] \rightarrow \text{addhost}(\ldots, n_u)\]

For \(i\) range of \(\text{number of host}\):

\[\text{node}[\ldots n_r, \ldots] \rightarrow \text{addhost}(\ldots, n_r)\]

(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:

\[\text{IPFS}_{\text{instance}} \leftarrow \text{ipfsapi.connect()}\]

(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, \(\mathcal{M}_i\) :

\[\text{IPFS}_{\text{instance}} \leftarrow \text{add}(\mathcal{M}_i)\]

(b) Obtain the model IPFS location “Hash” and “Name”:

\[\text{Hash} \leftarrow \text{ipfs} \{“Hash”}\]

\[\text{Name} \leftarrow \text{ipfs} \{“Name”}\]

Return \(\mathcal{M}_i, \text{Hash}, \text{Name}\)

(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 5: Federated Random Forest-Aggregation (user)

2) DATA PREPROCESSING

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

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

Due to the high complexity and nonlinearity of the CAN bus datasets, we employed several statical 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 for this paper.

Statitical features provide the most characteristic values that define the distribution of the transform cycles. The statistical feature extracted for this work includes 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 in order to determine whether the distribution is 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. High kurtosis value in a distribution indicates that the data is heavily tailed (contain 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:
FIGURE 2. Overview of the federated forest-based intrusion detection scheme.

1. Minimum value in each cycle sub-fragment:
\[ F_{\text{min}} = \min_{i} y(k)_i \]  

2. Maximum value in each cycle sub-fragment:
\[ F_{\text{max}} = \max_{i} y(k)_i \]  

3. Mean of the cycle in each window sub-fragment:
\[ F_{\text{mean}} = \frac{1}{l} \sum_{i=1}^{l} y(k)_i \]  

4. Standard deviation of the cycle in each window sub-fragment:
\[ F_{\text{std}} = \sqrt{\frac{1}{l} \sum_{i=1}^{l} (y(k)_i - F_{\text{mean}})^2} \]  

5. Skewness of the cycle in each window sub-fragment:
\[ F_{\text{skew}} = \frac{1}{l} \sum_{i=1}^{l} \frac{(y(k)_i - F_{\text{mean}})^3}{(F_{\text{std}})^3} \]  

6. Kurtosis of the cycle in each window sub-fragment:
\[ F_{\text{kur}} = \frac{1}{l} \sum_{i=1}^{l} \frac{(y(k)_i - F_{\text{mean}})^4}{(F_{\text{std}})^4} \]  

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 Kullback-Leibler divergence non-gaussianity. 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. Thus, the entropies extracted include Shannon, sample and permutation entropy. Despite the similarities between the different entropy algorithms, the theoretical ideas behind them are different. Shannon 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 generating 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]. Meanwhile, 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. Shannon entropy of each window’s sub-segment given \( y(k) \) is the probability associated with the values, is defined as:

\[
F_{\text{shann}} = -\sum \log_2 p(y(k_i)) p(y(k))
\]

(8)

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

\[
F_{\text{samp}} = -\log \frac{A}{B}
\]

(9)

where \( A \) is the number of having \( d[y(k), (a), y(k), (b)] < r \), \( A \) is the number of having \( d[y(k), (a), y(k), (b)] \).

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 \( \pi = r_0 r_1 \cdots r_{D-1} \), can be express as:

\[
F_{\text{perm}} = -\frac{1}{\log l!} \sum \log \pi_i
\]

(10)

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 4.1.3. Each miner creates a local random forest model through bootstrap. During the training, the miner uses part of its datasets for validation. Once the training and validation are complete, the model is uploaded to IPFS. The hashes of the model are then generated by IPFS and returned to the miner. When the miner gets the hash, it uploads the hash and its user name into the smart contract. For the actual mining of the model into Ethereum, we proposed that mobile communication infrastructure can be used 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 the 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 upload by miners via smart contract. The hashes are then used 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. This method of aggregation at the user end also eliminates 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 number of estimators (the number of trees in the forest) of each of the partial model are aggregated into \( M_{FL} \).

As indicated in the algorithm two major parameters: estimators (the collection of fitted sub-estimators) and number of estimators (the number of trees in the forest) of each of the partial model are aggregated into \( M_{FL} \).

5) PERFORMANCE EVALUATION
The proposed system training and testing is evaluated using precision, recall and accuracy. Whereas 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:

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
\]

(11)

\[
\text{Precision} = \frac{TP}{TP + FP}
\]

(12)

\[
\text{Recall} = \frac{TP}{TP + FN}
\]

(13)

\[
F_1 = \frac{2TP}{2TP + FP + FN}
\]

(14)
where TP, TN, FP, FN are the number of true-positive cases, true-negative cases, false-positive cases, and false-negative cases, respectively.

**Algorithm 4: Federated Random Forest - Training (miner)**

**Input**: Dataset D, Dij

**Output**: Partial Federated forest model, M

**Procedure** BuildRandomForest(Di, F, attack_type)

1. **bootstrap** sample of size, N, from the Di
2. **build** recursively random tree, Li
3. **select** M variable at random
4. **pick** the best variable/split-point in M
5. **split** into two daughter node:
   - **leftsubtree** — BuildRandomForest
   - **rightsubtree** — BuildRandomForest
6. **return** tree node(Li)
7. **append** {L} to forest, {M}

**Algorithm 5: Federated Random Smart Contract**

**Procedure** Federated Forest Contract

1. **structure** Model
2. **input** modelPath
3. **output** minerAddress
4. **function** FourierTransform(cycleID):
5. **input** cycleID:
6. **output** ft

**Function** CalculateFeatures(tx, attack_type):

1. **input** tx, attack_type
2. **output** ft

**Algorithm 6: Federated Random Forest - Aggregation (user)**

**Input**: Models downloaded by the users, M = [M₁, M₂, ..., Mₙ]

**Output**: Partial Federated forest model, M

**Procedure** BuildRandomForest(Di, F, attack_type)

1. For miner in minerAddress do:
2. **download** path
3. **contract** get_model_path(M, minerAddress, index)
4. **emit** upload(minerAddress, index)
5. **return** M

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/
V. EXPERIMENTAL RESULTS

A. TESTBED EXPERIMENTAL RESULTS

The testbed runs on a VM host that runs a Linux OS (Ubuntu). The SDN emulator, Mininet, is configured to simulate the WAN, CPU and RAM allocated by the VM. The CPU and RAM (MEM) performance 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 measurement is conducted during block forging operation for a fixed period of 180 seconds.

For testing the CPU and RAM usage, transactions to the blockchain nodes were generated in a round-robin fashion-50 transaction per burst. The bursts occurred every 2 seconds (sleep time) and are repeated for the total number of 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 measure the performance as follows: read operation on the blockchain node through the SDN node is 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 blocks is added to a counter until the actual height of the blockchain is 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 it thereafter to enable its usage for any new experiment. Clique Proof of Authority consensus algorithm was configured to always set h1 and h2 as signer nodes. Based on the algorithm, the nodes alternate in signing blocks. Before swap operations, just as few as 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 which consist of CAN ID, DLC and data fields. Since ECU regularly broadcast messages, understanding the pattern/interval at the IDS 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 4.1. Table II provides the composition considered for our testbed.

<table>
<thead>
<tr>
<th>Attack type</th>
<th>No. of Message</th>
<th>Training set</th>
<th>Testing set</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoS Attack</td>
<td>50,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fuzzy attack</td>
<td>50,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impersonation Attack</td>
<td>50,000</td>
<td>60%</td>
<td>40%</td>
</tr>
<tr>
<td>Attack-free</td>
<td>50,000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As presented in Algorithm 1, we compute the cycle of occurrence of each ID in each class of the dataset. Figure 5 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 said to be group 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 which 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 class, further transformation is needed to make rendered it more distinguishable.
We employed time-frequency analysis—Fourier transform—to decompose the CAN ID cycle. The packets traffic stream continuously, therefore, are not explicitly segmented into subfragments 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 \( s \). Setting these parameters required a trade-off between accuracy and delay. 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 transform. The transform packet traffic show more distinction between the classes of the packets.

Our dataset now contains a high-dimensional space of 1000 features after the windowing. In order 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 are greater than 1.

2) FEDERATED FOREST INTRUSION DETECTION MODEL PERFORMANCE

The federated forest model training and testing performances are investigated in this section. We split the dataset into equal sample sizes based on the number of miners. The number 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 8 shows the best training and validation session alongside the model scalability and overall score. The result shown in the figure is the best performing model in each of the miners’ sets. 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 progressive improve to about 0.99 accuracy. However, the validation performance varied, with 5 and 20 miners’ sets reaching an accuracy of more than 0.95 and about 0.93 as the best and least performing models, respectively.

Separate testing was conducted on each of the models. The average performance for each of the miners’ sets in terms of precision, recall and F1-score is presented in Figure 8. 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’. This huge variation in performance is associated 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 are safe into the blockchain as described in 4.1.4. 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 includes 5, 10, 15 and 20 users. The average accuracy performance of each set of users for each miners’ sets is evaluated as shown in Figure 9. The federated model from 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 test results reflect the amount of data used in training the model. The 5 miners 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’ set outperformed its corresponding TL model with the accuracy of 0.0012 on an average. Figure 10 present the confusion matrix of the overall best model recorded by 5 miners set FL model with 40 users. Meanwhile, Figure 11 shows the comparison between FL and TL models across all users scenario.

2) THE FEDERATED FOREST MODEL EVALUATION AGAINST OTHER MODELS

In this section, we evaluate our best model against other detectors and works build using TL method. As shown in Figure 12, our model outperforms 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 accuracy of 0.342, 0.969 and 0.568, respectively. Equally, our are BFF-IDS outperforms 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 us, others’ datasets were equally collected and created from the same source. In addition, only [36] classified more label than us will 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 evaluation in 4.2.2 suggests that split data across miners can reduce performance, our FL model was able to outperform other models and works- But 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 the models other than the blockchain reduces the cost (gas, ether) that may be incurred when using the blockchain.
Figure 7: Miner nodes best training performances: (a) 5 miners (b) 10 miners (c) 5 miners (d) 5 miners

The simulation shows that the testbed efficiently utilizes CPU and memory resources and will be adequate in practice. Thus the hybrid architecture leverage the benefit of SDN in providing network management flexibility, blockchain in providing security and FL learning in harnessing the benefits of data available that exist as Island 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 which include 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 to exposed the distinction between traffic patterns in the attack class.
Similarly, our proposed combination of statistical and entropy features for effective description of the subtle difference in the pattern proved effective in extracting the features. Meanwhile, the FL method of building the method also prove effective as it was able to classify the attacks with over 0.98 accuracy.

Comparing with some previous approaches such as [14], which utilized about four features (CAN messages, number of messages, sum of DLC and bandwith) from the CAN bus logs, we employed only the CAN ID cycle. Although some works have used the CAN IDs field as indicated in Table 1, FFT was not applied 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 paper, blockchain-based Federated forest SDN-enable IDS is proposed. The testbed shows efficient use of memory and CPU resources. Fourier transform was applied to CAN ID cycles and statistical and entropies features were extracted. 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. With this proposed system, manufacturer/car owners may be willing to federate as confidentiality of their data is safeguarded. In addition, flexible incentives can be enforced to reward federating units based on their contribution to the building of the model. Users who wish to use the model can equally be required to pay, thereby ensuring a sustainable market. Consequently, the proposed system leverage the benefit of SDN in providing flexible configuration for unforeseen network requirement, blockchain in providing security and FL learning in harnessing the benefits of data available that exist as Island in individual vehicle or manufacturers. Although the significant performance was recorded by our proposed system, 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

We 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


IBRAHIM ALIYU received his B.Eng. in 2020 and is currently pursuing (as of 2021) M.Eng in Computer Engineering at University Federico II, Napoli, Italia. He is focusing his studies on cyber security topics, 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 “pwnthego”, a team involved in online cyber security competitions. He also contributed to different open source projects.

MARCO CARLO FELICIANO received his B.Eng. in 2020 and is currently pursuing (as of 2021) M.Eng in Computer Engineering at University Federico II, Napoli, Italia. His research interest includes Federated Learning, data privacy, Network Security, SDN and BCI. Aliyu is a recipient of the Korean Government Scholarship Program Award in 2017.

SÉLINDE VAN ENEGBERG is a researcher at the faculty of Technology, Policy and Management at Delft University of Technology. Her research is on the role of new technological developments in cybersecurity. Previously, She obtained her PhD at Delft University of Technology in the field of ICT. Her PhD research was on designing large-scale context-aware architectures for information sharing to enhance security and safety in international container shipping. 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. Selinde obtained a master’s degree in Artificial Intelligence from Utrecht University with a specialization in logic and intelligent systems.

DONG OK KIM received his Ph.D. in Dept. of Control and Instrumentation Engineering at Chosun University, Korea, in 2002. He previously worked in Korea Esen Co., Korea as a technical director. He also worked as a researcher at the department of Precise-Engineering Research at Tokyo Institute of Technology in Japan, and as a BK21 research professor at Chonnam University, Korea. Since 2011, he has been consulting as Technology Transfer Agent. Moreover, since 2009, he has been working as a Technology consultant for local and metropolitan governments. He joined Jeonnam Technopark in 2008 and now he is participating in five projects related to energy innovation as the chief of the department of National Innovation Cluster Support Center, Jeonnam Technopark.
CHANG GYOOON LIM received his Ph.D. in Dept. of Computer Engineering in Wayne State University, U.S.A. in 1997. Since September of 1997, he has been working for Major in Computer Engineering, Chonnam National University, Yeosu, Korea, as a professor. He was a director of Home Robot Center in Gwangju Techno Park. He is a director of Korean Society for Internet Information. Simultaneously, he acts as the committee member of several public and private regional institutions. He plays an editor of Transactions on Internet and Information Systems (TIIS) and a chair of Gwangju-Jeonnam Cloud Computing Leaders Forum. He also leads several research projects of his interest areas. His current research interests include BCI, Machine Learning, Soft Computing, Intelligent Robot, IoT, Cloud Computing, and Embedded Software.