Edge Computing and Deep Learning Enabled Secure Multi-Tier Network for Internet of Vehicles

Article *in* IEEE Internet of Things Journal · April 2021

DOI: 10.1109/JIOT.2021.3071362

Vinay Chamola
Birla Institute of Technology and Science Pilani

Tejasvi Alladi
Birla Institute of Technology and Science Pilani

Edge Computing and Deep Learning Enabled Secure Multi-Tier Network for Internet of Vehicles

Harsh Grover, Tejasvi Alladi, *Senior Member, IEEE*, Vinay Chamola, *Senior Member, IEEE*, Dheerendra Singh, *Senior Member, IEEE* and Kim-Kwang Raymond Choo, *Senior Member, IEEE*
.

Abstract—**Internet of Vehicles (IoV) are fast becoming the norm in our society, but such a trend also comes with its own set of challenges (e.g., new security and privacy risks due to the expanded attack vectors). In this work, we propose an edge computing-based secure, efficient, and intelligent multi-tier heterogeneous IoV network. We first discuss the functionality and objectives of such an architecture. Then, we demonstrate how unsupervised deep learning techniques can facilitate the identification of suspicious vehicle behavior and ensure the security of such an architecture. The findings from our evaluations demonstrate the learning spatio-temporal information and parameter efficiency of the proposed stacked LSTM model over single LSTMs.**

Index Terms—**Internet of Vehicles (IoV), Vehicular ad-hoc networks (VANETs), Edge computing, Unsupervised learning, Anomaly detection, Security**

I. Introduction

In recent years, there have been rapid advances in vehicular and other communication technologies that can be used to support self-organizing vehicular ad-hoc networks (VANETs), which have partly contributed to the transitioning into next generation architectures such as the Internet of Vehicles (IoV). VANETs have dynamic network topology, and the communication in such networks is guided by the principles of mobile ad-hoc networks (MANETs) and their extensions. In other words, VANETs use vehicular wireless technology, which improves ease of communication between different entities in the system with the supporting VANET infrastructure and enables state awareness of the system.

Due to the high mobility, heterogeneity, and unstable connectivity of the nodes in VANETs, the network architecture of MANETs cannot be adopted to VANETs. There are several differences between MANETs and VANETs, for example vehicle motion is less random in VANETs compared to node movement in MANETs. Unlike smaller mobile nodes, vehicles may move at very high speeds (e.g., 70 mph or more) and consequently, the network topology is extremely dynamic / fluid. Due to the dynamic nature of the setting, connections are established for short periods making it difficult to secure the network. Further, the networks may disconnect due to the variable density of nodes as there can be extremely high and low node densities during traffic jams and suburban areas respectively. Given the time and latency-sensitive nature of VANETs (e.g., decision making required in nanoseconds or milliseconds), error tolerance is very low.

Edge-based VANETs are capable of minimizing communication and computation delays [1], and stabilizing the networks by facilitating load sharing [2], load balancing and localized decision making [3, 4]. In such a setting, different types of devices (e.g., multi-edge computing devices and cloudlets) are deployed at the edge of the network.

Given the open nature of VANETs, the attack surface is broad [5, 6]. This reinforces the importance of designing a secure network architecture for IoV. We also remark that significant volume of data can be generated in these vehicular networks such as environmental data (e.g., temperature and humidity), vehicle kinematics (e.g., velocity and driving patterns), and others (e.g., packet drop rate, packet modification rate, request to send flooding rate, channel status, packet interval, and packet size). These generated data can be used to identify malicious / misbehaving nodes in the network. There have been attempts to leverage artificial intelligence to defend the vehicular networks from malicious cyberattacks. Due to the lack of prior knowledge of the potential attack types and their nature, deep learning algorithms, in particular, provide the flexibility to adapt to different situations by recognizing patterns and extracting features constantly.

In this paper, we propose an edge computing-based secure multi-tier heterogeneous VANET architecture for IoV networks. We also present deep learning models for detection of malicious vehicle behavior in this model. The major contributions of this paper are highlighted below:

i. We propose an edge computing-based multi-tier heterogeneous vehicular network for IoV services.
ii. A stacked Long Short-Term Memory (LSTM) based unsupervised vehicle behavior detection algorithm for securing vehicular communication between participating vehicles and roadside units (RSUs) is presented.
iii. We evaluate and compare the performance of multiple LSTM models on a popular misbehavior detection dataset.

Manuscript received October 17, 2020; Revised Jan 26, 2021, and March 11, 2021; Accepted April 02, 2021. The associate editor coordinating the review of this article and approving it for publication was Dr. Wei Yu. (Corresponding author: Dr. Vinay Chamola.)

H. Grover, V. Chamola and D. Singh are with the Department of Electrical and Electronics Engineering & APPCAIR, BITS-Pilani, Pilani Campus, 333031, India. (e-mail: {f2015429, vinay.chamola, dhs}@pilani.bits-pilani.ac.in).

T. Alladi is with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada. (e-mail: talladi.carleton@gmail.com).

K-K. R. Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249, USA. (email: raymond.choo@fulbrightmail.org).

The rest of this paper is organized as follows. Sections II and III discuss the related works in vehicular networks and the concepts of multi-tier networking and deep learning, respectively. The proposed architecture is presented in Section IV, and the evaluation setup and findings are presented in Section V. Section VI concludes this paper.

II. Related Work

Several machine learning and deep learning-based techniques to address different security challenges in vehicular networks have been proposed in recent years. We discuss several of these recent approaches in this section. In light of the flexibility that the learning algorithms offer as compared to conventional methods, there have also been attempts to utilize hybrid models such as the Bayesian Neural Networks, combining deep learning with probabilistic modeling for intelligent decision making. Another work [7] applies convolutional neural networks (CNNs) to analyze traffic and to detect anomalies using a threshold-based method. This idea is further improved in [8] with the use of reinforcement learning to select the best thresholds. The traditional centralized control plane makes VANETs vulnerable to attacks. Another work [9] presents an artificial neural network paradigm (ANN) called a feed-forward ANN to detect misbehavior. Authors of [10] propose a Q-learning and blockchain-based method to distribute the control plane in VANETs. Blockchain has also been found to be beneficial in several other vehicular network security applications, in conjunction with artificial intelligence-based enabling technologies [11, 12].

Intrusion detection systems (IDSs) mitigate threats by detecting malicious nodes in the network. Collaboration between nodes to share information regarding malicious nodes in the network may help increase the accuracy of detection. A distributed machine learning framework for collaborative detection in VANETs is discussed in [13]. This method also preserves the privacy of information shared between nodes. While such a method requires access to private information, a raw traffic data-based method is proposed in [14] which uses a deep learning model to detect malware traffic. Location privacy is far more challenging in vehicular networks than the Location-based Services (LBSs) of mobile internet. Therefore, [15] solves this problem by using a prediction based model called LocJury and intent-based networking. The system learns and estimates the intent of location access and penalizes malicious access to a location. LBSs aim to protect the privacy of users and to prevent the leakage of vehicle movement trajectories. But there is a trade-off between privacy and quality of service. Wang et al. [16] present a reinforcement learning model to optimize user experience while maintaining the quality of services. The controller area network installed in vehicles to enable communication between all electronic control units is not sufficient for protection against suspicious network connections. In [17], generative adversarial networks (GANs) are applied to detect intrusions using normal data.

Another issue with VANETs is their inability to prevent fraudulent messages from legitimate nodes. Thus verifying the legitimacy of information of verified nodes becomes important. A reward-based system for drivers for vehicle-to-vehicle communication is presented in [18] which verifies the legitimacy of the messages sent from verified identities. To ensure the integrity of shared messages, the k-nearest vector (KNN) and support vector machine (SVM) are also feasible solutions, and a framework using these tools is suggested in [19]. The authors prove that it is possible to classify misbehavior once detected. KNN is also used by [20] where an IDS model is developed to detect Sybil attacks optimizing the runtime complexity of KNNs. While most models use general data such as traffic flow and vehicle density, [21] uses a machine learning approach with specific data like individual vehicle type and velocity. Such vehicle classification based on the radio fingerprint is cost-efficient and is independent of environmental factors. Rahim et al. [22] present a cost-effective and scalable method of driver authentication using data of driving patterns from GPS units in vehicles and hand-held mobile devices.

There have also been attempts to utilize edge computing in vehicular networks, as observed in the survey of De Souza et al. [23], as well as integrating both deep learning and edge computing in vehicular networks [24, 25]. Although several machine learning and deep learning approaches for securing VANETs have been proposed, they primarily focus only on the algorithmic implementations of the learning models ignoring the details of the network architecture. Hence, in this paper, we present a deep learning approach for anomaly detection for vehicular networks, and develop a secure network architecture customized for IoVs.

III. Background

A. Multi-tier Networking

Multi-tier computing networks generally combine cloud computing, edge computing, and/or IoT devices to support intelligent user-centric IoT services [26, 27]. This is because cloud computing alone will be inadequate to support large-scale applications of the IoT-drive future, like smart cities, environmental monitoring, etc, for example due to limited bandwidth, delay constraints, and connectivity issues. In other words, multi-tier networking addresses these issues and provides reliable, flexible, and scalable IoT application deployment.

The cloud is the top level. It has the highest processing power and data storage in this hierarchy. It is used for making global level decisions and data warehousing. These decisions generally include cross-region data analysis and tracing hidden problems. Hence it is the best option for performing computationally intensive tasks in the application. Edge is closer to the end-user. Thus it offers low latency communication and timely data processing. This makes it ideal for handling delay-sensitive tasks [28, 29]. Edge has lesser processing and data storage capabilities than the cloud. It is ideal for making more localized decisions. Communication infrastructure like RSUs and cellular base stations are the next layer. They connect the edge to the IoT endpoints and function as data forwarding devices. Finally, the IoT endpoints like smartphones, PC, etc. allow users to interact with the

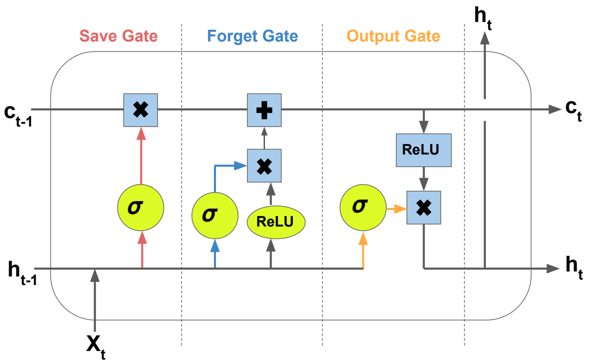

Fig. 1: Single cell of the LSTM variant used in this paper.

application. Each level in this multi-tier network fulfills a purpose and adds to the functionality of the application. Due to the different computational capabilities of each level, it is often called a multi-tier heterogeneous network.

Multi-tier networking when united with artificial intelligence (AI) and big data technologies can be used to build dynamic IoT applications. Such applications can automatically adapt to user preferences with large-scale deployment. Hence providing unparalleled services in various use-cases. In the back-end, multi-tier architecture combines the computing and storage resources at different levels through active communication between all the nodes in the IoT network. Not all the processes in an application require high resources. The network thus distributes processes to different levels based on their requirements. Hence significantly improving user experience while saving time, cost, and resources. This enables the network to optimally use the hardware and software capabilities at different locations and levels, to provide efficient, intelligent, and prompt services to the users.

B. Deep learning

Deep learning is the paradigm of machine learning which uses multi-layered (deep) neural networks to capture abstract non-linear features in the data [30]. Here we discuss Long Short Term Memory (LSTM) networks and intuition behind Stacked LSTMs which are used in this work.

1) Long Short Term Memory: Long Short Term Memory (LSTM) networks are a variant of Recurrent Neural Networks (RNNs). RNNs are used for sequential inputs where the order of the input sequence is important. Artificial Neural Network (ANNs) on the other hand cannot capture sequential information. This is because ANNs input the entire sequence at once, hence it loses any information about the order of the sequence. RNNs are required to keep the sequential information intact. It takes input one-by-one, in order. RNNs use ANNs as memory cells inside them. The learnable parameters of ANNs help RNNs to store information of data points from all of the previous steps in a sequence. This allows the output of the current step to be dependent on not only the current input but also on all the previous ones in the sequence. Thus RNNs perform drastically better than ANNs for sequential input. However, in practice RNNs suffer from vanishing gradient issues while training on long sequences. Hence it is not able to learn long-term information from the sequence. It is only able to preserve information from the previous few steps instead of the entire sequence. LSTMs, proposed by Hochreiter et. al. [31], have been shown to overcome the vanishing gradient problem and preserve information for a much longer term.

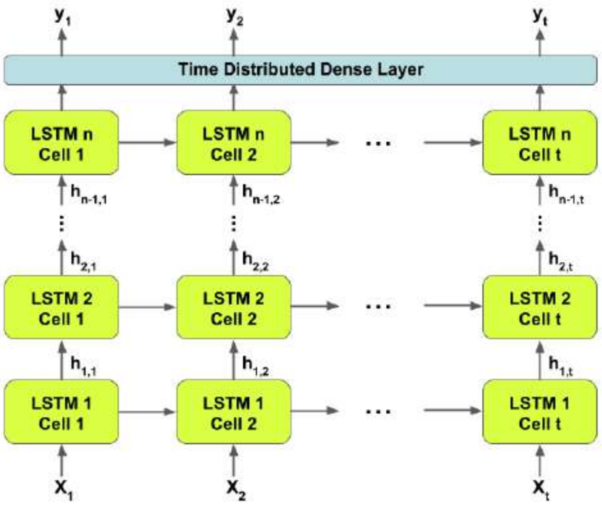

Fig. 2: Unrolled representation of Stacked LSTM with ***n*** layers and *t* time steps.

LSTMs use a gated structure of neural networks with a cell state and hidden state memory to allow the network to learn long-term dependencies. The cell state (c) in the LSTM is designed to preserve information across time steps without many changes. This acts as long-term memory. LSTM also has a working memory or a hidden state (h), which is relatively short-term and is updated more often. c_t and h_t are the cell state and the hidden state of the LSTM cell at time-step t respectively. h_t is also the output of this LSTM cell. Several gates control how the cell state and the hidden state change with each input – see Eq. 4 to Eq. 6. These gates are sigmoid neural networks. Their outputs decide which information from the cell state and hidden state is to be kept for the next time step and which information is irrelevant and deleted. Thus, through these gates, the network learns long-term temporal features in sequential data. The LSTM has the following three gates:

1) **Save gate:** Controls what information to add to the cell state from the current input (see Eq. 1).
2) **Forget gate:** Controls what information to forget from the cell state (see Eq. 2).
3) **Output gate:** Controls what information in the cell state to focus on (copy to the hidden state; see Eq. 3).

These neural network gates are defined as below:

$$i_t = \sigma(W_i[h_{t-1}, X_t] + b_i) \tag{1}$$

$$f_t = \sigma(W_f[h_{t-1}, X_t] + b_f) \tag{2}$$

$$o_t = \sigma(W_o[h_{t-1}, X_t] + b_o) \tag{3}$$

In the above equations, f_t is the forget gate, s_t is the save gate, and o_t is the output gate. X_t is the cell input. Both W and b are learnable neural network parameters. The memory

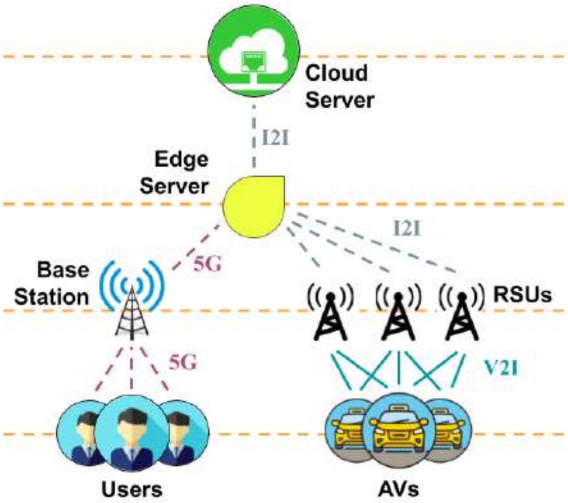

Fig. 3: Multi-tier network architecture proposed.

states are updated as:

$$\tilde{c}_t = \varphi(W_c[h_{t-1}, X_t] + b_c) \quad (4)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \quad (5)$$

$$h_t = o_t \cdot \varphi(c_t) \quad (6)$$

In the above equations, $\cdot$ and + are element-wise multiplication and addition, respectively. $\tilde{c}_t$ is the intermittent cell state for time step t and φ represents activation function. LSTMs traditionally use $tanh$ activation function for updating the states but we empirically found $ReLU$ function to perform better in the model presented in this paper. The variant of the LSTM cell used in this paper is shown in Fig. 1. In this figure, the circular shapes represent neural network layers and the boxes represent element-wise operations. The gates are color-coded and labeled.

Stacked LSTMs are multiple layers of LSTM one above the other, as shown in Fig. 2. In the figure, X is the input sequence, y is the output sequence, $h_{n,t}$ represents the hidden state of the cell t of LSTM n. Every LSTM cell in a stacked LSTM not only transfers its hidden state to the next cell in the same LSTM but also the cell corresponding to the same time step in the LSTM layer above it. In an ANN, the hidden layers recombine the learned features from prior layers and create new features at higher levels of abstraction. Similarly, in a stacked LSTM, the additional layers or depth adds levels of abstraction to input observations over time. Essentially, it chunks observations over time to extract features of different time scales. The time-distributed dense layer is used after the LSTM stack to make the output dimensions equal to the desired value. The time-distributed dense layer is essentially the same dense layer applied to all the time steps one by one.

IV. Proposed Architecture

Here, we propose a multi-tier heterogeneous vehicular network for IoVs. We use autonomous cabs as a case study to demonstrate how the proposed approach can be used to provide intelligent, secure, and efficient cab services to end-users. As shown in Fig. 3, this heterogeneous vehicular network consists of the following layers:

i. A cloud server at the top.
ii. Regional edge servers.
iii. RSU grid which interacts with the autonomous cabs and regional edge servers. The cellular base station connects a user with the regional edge server.
iv. Autonomous Cabs and IoT end-points.

The orange dashed lines in the figure depicts each level in the network.

The mobile application users interact with the edge servers using cellular data services and together they form the user side of the network. The autonomous cabs, RSUs, and edge servers form the vehicular side of the network. RSUs communicate with both the autonomous cabs (V2I) and the edge server (I2I). The edge server which is common to both the user-side and the vehicle-side has the software and hardware capabilities to function as a regional decision-maker. The cloud server acts as a global decision-maker and is connected to regional edge servers.

Fig. 4 is a representation of network deployment. The RSUs communicate with the autonomous cabs using Dedicated Short Range Communication (DSRC) protocol. RSU forwards cab states to the nearest edge server via wired internet. No computation takes place at RSUs, they function only as data forwarding devices. Along with receiving cab states from RSUs at regular intervals, the edge servers also receive user requests through the cellular network. It is assumed that the edge server is in the vicinity of the cellular base station. The user requests are filtered through IDS at the edge server for security against cyber attacks. Based on the cab allocation algorithm the user requests can be batched together or catered one by one. The edge server takes regional level decisions like cab allocation and flagging anomalous cabs. The cloud monitors logs and data sent by regional edge servers via the internet and takes global level decisions like demand prediction and reporting node failures to the authorities. textcolourblueCloud will routinely retrain the anomaly detection model and update it to all the edge servers. Table I gives a detailed overview of processes at each level of the network. These processes are key to the functioning of the network.

A. Objectives

The main objectives of the proposed approach are described next.

- **Secure Network**
  It is assumed that the network is secure from internal threats and any risk to the network is posed by external devices like autonomous cabs and user devices.
  - **User side:** IDS secures edge servers from cyber attacks like DDoS, DoS, Botnet, and Brute-Force attacks.
  - **Vehicle side:** We have leveraged Deep learning (DL) techniques to make the vehicle side of the network more secure and reliable. DL based anomaly detector in Section V finds malicious and faulty autonomous cabs and flags them. The flagged cabs are monitored by the

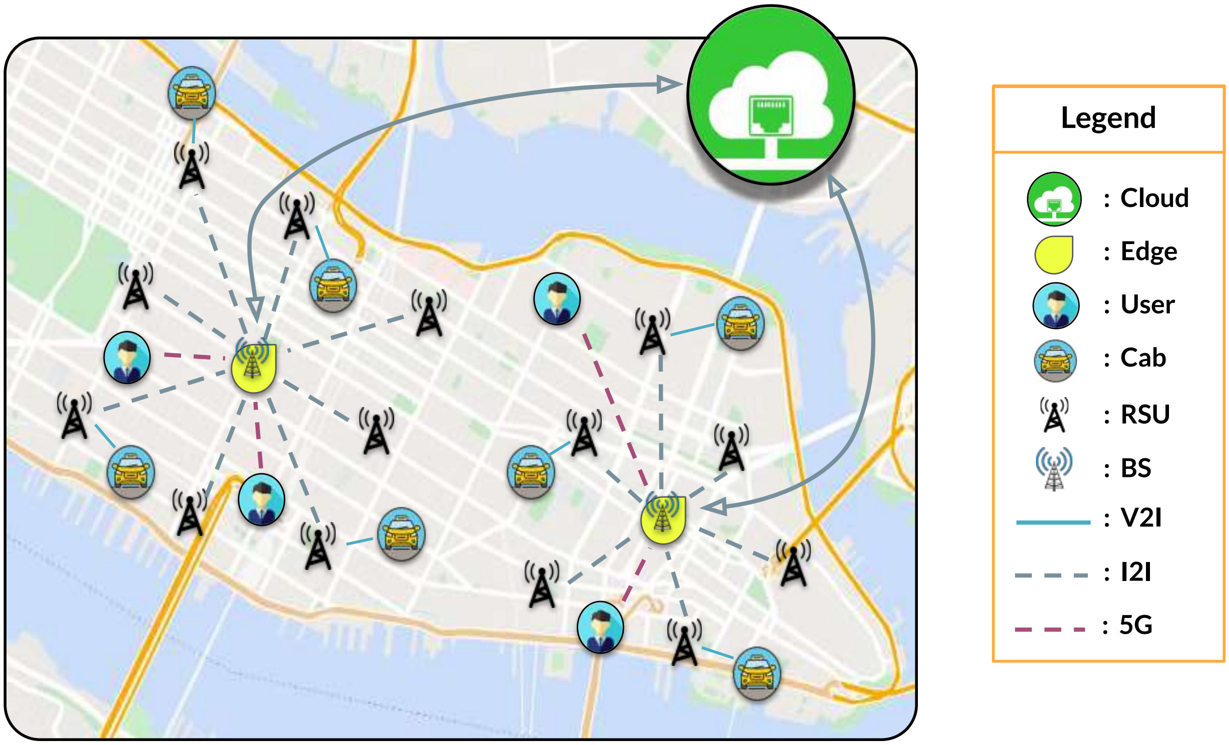

Fig. 4: Deployment of the proposed network. Table I gives a detailed overview of network functioning at each level.

Tier	Processes
Users	**Sends request for cab to the Edge server:** User requests are sent to the nearest Edge server via cellular data services.
Cabs	**Broadcasts cab state to nearest RSU:** Autonomous cabs broadcasts its state including cab ID, position, speed, sensor data, status(occupied/vacant) and charge level to the nearest RSU through DSRC.
BS	**Facilitates communication between end users and Edge server:** User requests are forwarded to Edge servers via 5G cellular network.
RSU	**Facilitates communication between autonomous cabs and Edge server:** Receives cab states and data through DSRC and forwards it to the nearest edge server via wired connection.
Edge	***User-triggered processes*** • **Receives user requests:** User requests are received by the nearest Edge server via cellular data services. The requests are filtered through intrusion detection system (IDS) to detect cyber attacks. • **Runs cab allocation procedure:** Edge server runs cab allocation algorithm to assign cabs to the users. • **Notifies the allocated cab and user:** Edge server notifies users via cellular and cabs via RSUs. Moreover, the cab's state is updated. ***Continuous processes (Performed at regular intervals)*** • **Gets cab states:** Edge server collects cab states for all the cabs in its region via RSUs. • **Updates cab states:** Based on some decisions from cloud processes, the cab states will be updated through the edge server via RSUs. • **Flags malicious and faulty cabs:** Edge server runs deep learning based anomaly detection to find abnormal behaviour in cabs. Anomalous cabs are flagged. • **Streams logs and data to the cloud.**
Cloud	Cloud processes uses the logs and data received from edge servers in different regions. • **Notifying faults in the network:** Notifies authorities of any faults in the network. For eg. faulty edge servers or autonomous cabs. • **Big data storage:** Cloud stores relevant data and logs for further analysis in future. • **Updating model weights:** Cloud routinely re-trains the model and updates the model weights at the edge. Data from misbehaving vehicles (identified either by the model itself or from user feedback) is not used for re-training.

TABLE I: Network processes for each tier in the multi-tier network.

  cloud and are reported to the authorities on continuous misbehavior.

  Further details about the DL based anomaly detection are covered in Section V.

- **Efficient Network**
  This proposed network not only uses hardware resources efficiently at each level but is also scalable and allows infrastructure sharing to increase return on investment.
  - **Scalability:** Multi-tier network segregates the area into smaller regions leading to faster data processing and low latency. This makes it easier to scale the network city-wide without much computational overhead, even with highly sophisticated allocation algorithms. This also allows other region-based smart algorithms like demand-prediction to run at a large scale.
  - **Infrastructure sharing:** No V2V communication is required in the vehicle side of our network. This allows cabs from different companies to be on the same network. Moreover, the same infrastructure can be used for other services in autonomous cabs like infotainment systems. Infrastructure sharing increases the return on investment for hardware and makes the network more cost-effective.
- **Intelligent Network**
  The proposed network is automated and state-aware.
  - **Automation:** The network functions in an automated way to receive user requests and to allocate cabs without any human intervention.
  - **State Awareness:** Data is gathered at the edge servers constantly, where such data defines the state of the network. Also, any changes in the network can be reported to the relevant authorities.

V. Evaluation Setup and Findings

As described above, the proposed approach comprises a user-side network and a vehicle-side network. Both networks face security threats from malicious users and misbehaving cabs respectively. In the user-side network, malicious users can launch cyberattacks to compromise the edge servers. An intrusion detection system (IDS) is hence a necessity for securing the edge servers.

On the vehicle-side of things, autonomous cabs can misbehave due to faults in the on-board sensors or due to getting hacked. In either scenario, it is important to flag such vehicles as anomalous and subsequently, remove them from the network if such behavior continues. An anomaly detection algorithm would be most suitable for such a scenario as abnormalities can occur due to a wide variety of reasons.

A good amount of work exists for IDS [32–34] and authentication at edge servers [35–37], but very few for anomalous cab detection in vehicular networks. We have hence used deep learning to create an anomaly detector for the vehicle-side of the network.

A. Dataset

We have used the VeReMi dataset [38] for misbehaviour detection. The dataset is simulated within the LuST scenario using the VEINS simulator, with a varying vehicle and attacker densities. The data contains five types of position-forging anomalous vehicles as shown in Table II. Fig. 5 visualizes the broadcasted positions in the dataset. We have shown the X, Y position coordinates as per the simulations in VeReMi. The first plot shows the true positions of vehicles in the simulations. The second plot shows the broadcasted locations by different types of vehicles.

Vehicle	**Description**
Type 0	Broadcasts correct location.
Type 1	Always broadcasts same location.
Type 2	Broadcasts location differing by a constant offset from real location.
Type 4	Broadcasts random location.
Type 8	Broadcasts location differing by a random offset from real location.
Type 16	Broadcasts normally for sometime and then broadcasts same location.

TABLE II: VeReMi dataset description.

We have pre-processed the VeReMi data to produce sequences of broadcasted positions, actual positions, and speed for each vehicle. The pre-processing of the dataset is described in detail below. The model will learn to reconstruct the positions from the input sequence. Only sequences corresponding to the normal behavior (type 0) are used for training. We randomly sampled 28000 sequences from the normal sequences to be used as training data. Test data of 4096 sequences were sampled randomly from the rest of the data. Refer Table III for number of sequences of each type. The sampling was done in such a way that the ratio of each type in the test set is the same as the ratio of each type in the entire dataset. Type 0 sequences from the test data were not used for training.

1) Pre-processing: We used packet types 3 and 4 from all the simulations in the VeReMi dataset, irrespective of vehicle and attacker densities in the simulation. These packets have information regarding the actual position, broadcasted position, time, speed, and attacker type of the vehicle. Each simulation data contains multiple such packets from different vehicles. For each vehicle in the simulation, we identify its corresponding packets and generate sequences of position and speed, sorted by time. We use the sliding window approach to generate these sequences for each vehicle in every simulation. The sequence length or the window size is taken as 24, while the slide length is 12. Sequences with a length less than 10 are ignored. This means if a vehicle broadcasts its position from 0 to 60 seconds in a simulation, we get 4 sequences from this vehicle, corresponding to 0-24 seconds, 12-36 seconds, 36-48 seconds, and 48-60 seconds. The sequence type is corresponding to the attacker type of the vehicle which broadcasted the sequence, i.e. a type 2 sequence will be from a vehicle that is attacker type 2. The total number of sequences for each type is shown in Table III.

Seq. Type	Total Seq.	Train Seq.	Test Seq.
Type 0	30947	28000	2410
Type 1	2712	-	210
Type 2	2746	-	213
Type 4	5644	-	438
Type 8	5664	-	440
Type 16	4951	-	385

TABLE III: Number of total, training and testing sequences, generated from VeReMi dataset.

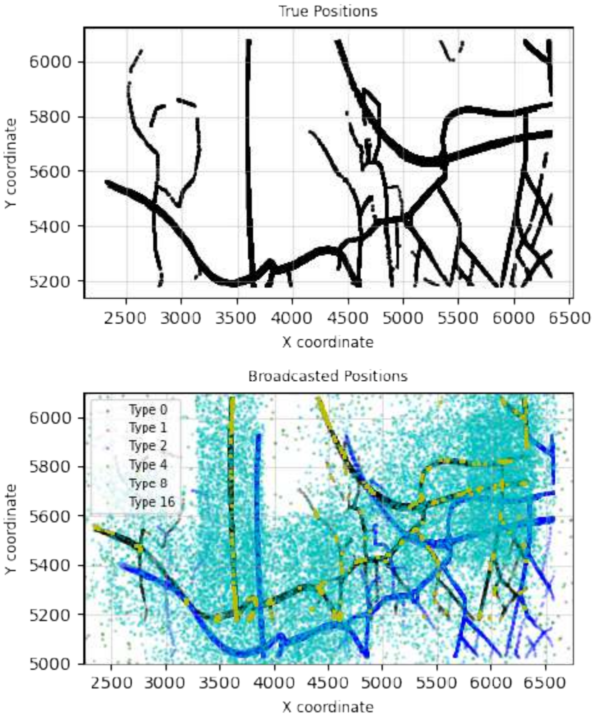

Fig. 5: VeReMi dataset visualisation.

The sliding window approach is used to increase the training data. The sequence length of 24 is used as it is long enough to capture important temporal features and short enough for LSTMs to learn easily. In choosing the sliding length there is a trade-off between quantity and repetition of training data. A shorter sliding length will result in more data but more repetitions, whereas a longer sliding length will result in lesser repetition but lesser data too. We want less repetition as useful information per sequence decreases with more repetitions. Hence sliding length of 12 was chosen as there is not much repetition in the training data but with an increase in the number of training samples.

The input sequences contain X, Y positions, and X, Y velocities corresponding to each time step. Sequences with a length less than 24 are padded with zeroes to make length equal to 24. The dimensions of an input sequence are thus 24×4. The positions are scaled down by a factor of 1000 for better optimization of the model while learning. The output sequences contain only X, Y positions. Similar to the input sequences, zero padding is used in the output sequences. The dimensions of output sequences are thus 24×2.

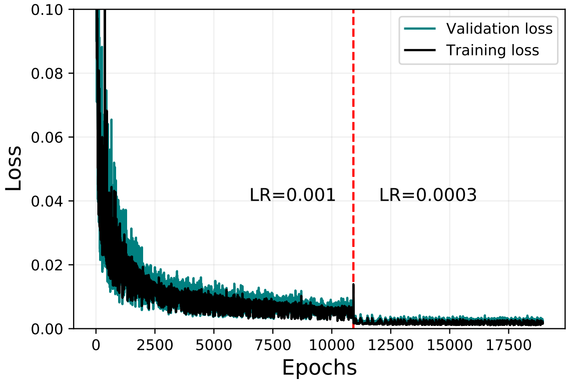

Fig. 6: Learning curve.

B. Deep Learning Model

Stacked LSTM and normal LSTM models are trained to learn reconstruction of the normal cab movement from the VeReMi dataset. After proper training on the normal data, the neural network should ideally be able to reconstruct the normal cab behavior but will not be able to reconstruct the anomalous behavior. This is a type of unsupervised learning. Reconstruction error is used to quantify reconstruction by the model. A reconstruction error threshold is finally used to classify normal and anomalous cab movement. As only normal data is used for training, this model should also be able to identify anomalies other than the ones present in the VeReMi dataset. All the codes are written in Python 3. We use Keras, an open-source neural network library, to build and train all the neural networks in our evaluations. Pandas library is used for data processing. The neural networks are trained on a Google Colaboratory environment with NVIDIA Tesla K80 GPU (16 GB GPU RAM) and 12 GB CPU RAM. Our best performing model is Stacked LSTM with 4 layers and 256 units in each layer. This model takes around 4.5 seconds to predict labels for 4096 sequences in our environment. The actual prediction time will vary depending on the edge hardware used and the number of sequences per server considered, which will be much less than 4096.

C. Training

Mean absolute error (MAE) is used as the training/reconstruction loss, refer to Eq. 7 where $\hat{y}$ represents predicted values.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y - \hat{y}| \quad (7)$$

The models are trained with batches of data using Adam optimizer in two stages. First, we use a learning rate of 0.001 for faster convergence, then finally learning rate of 0.0003 is used to reach the global minimum smoothly. A batch size of 2048 is used. Early stopping is used for each of these stages to prevent overfitting, with patience of 2000 epochs. Best weights are restored after each stage. ReLU activation is used for

Model	No. of	No. of	Parameters	Recall						Accuracy
	layers	units		Type 0	Type 1	Type 2	Type 4	Type 8	Type 16	
KNN[19]	-	-	-	-	.751	.771	.992	.772	.220	.879
GRU	1	256	202K	.968	.514	.343	1.0	1.0	.569	.882
LSTM	1	256	268K	.964	.967	.610	1.0	1.0	.748	.933
LSTM	1	512	1M	.978	.995	.648	1.0	1.0	.844	.953
LSTM	1	1024	4.2M	.982	.971	.807	1.0	1.0	.883	.967
Stacked LSTM	2	256	793K	.986	.829	.859	1.0	1.0	.795	.956
Stacked LSTM	3	256	1.3M	.994	1.0	.995	1.0	1.0	.930	.989
Stacked LSTM	**4**	**256**	**1.8M**	**.994**	**1.0**	**1.0**	**1.0**	**1.0**	**.966**	**.993**
Stacked LSTM	5	256	2.3M	.995	1.0	.995	1.0	1.0	.938	.991

TABLE IV: Results

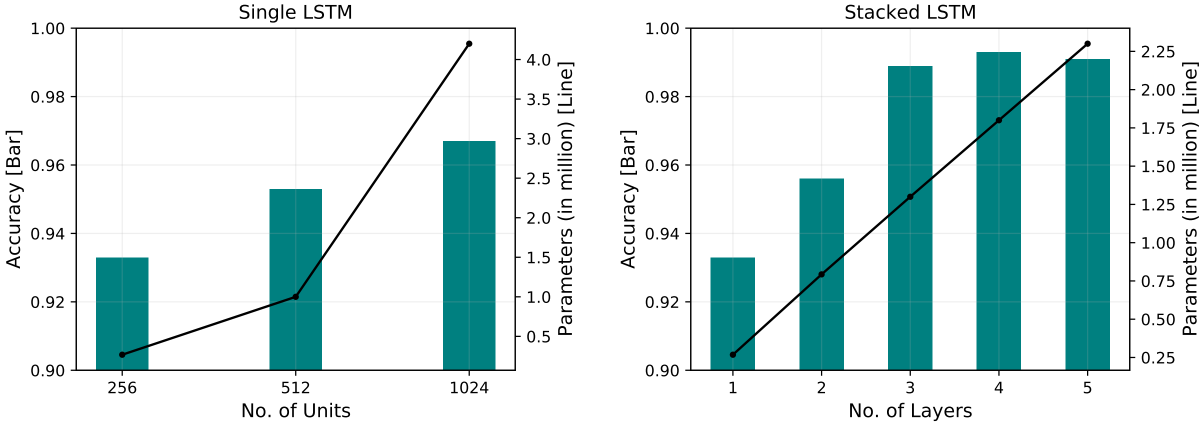

Fig. 7: Accuracy and number of parameters for different models.

LSTM. Unroll parameter was set to True for LSTMs as it tends to speed-up training. ReLU activation is also used for the time-distributed dense layer. The 4-layer Stacked LSTM model takes about 13 hours to train in our environment. The learning curve for the model is shown in Fig. 6

D. Comparison and Analysis

In Table IV we have compared our anomaly detection results of Stacked LSTMs with various depths and normal LSTMs with various widths. Classification with KNN as described in [19] is used as the baseline to compare the results with our unsupervised approach. As can be seen in the table, the performance of a single layer LSTM is shown to outperform that of a single layer Gated Recurrent Unit (GRU) on our dataset. Thus, we decided to use LSTMs in all further experiments. No. of units column depicts the number of units in each layer of the model considered. The performance is measured for a test-set of 4096 sequences. The threshold with the best accuracy is chosen for each model. Accuracy and recall is defined in Eq.8 and Eq. 9 respectively. In these equations, T means True; F means False; P means Positive; and N means Negative.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

We can see from the recalls in Table IV that all the LSTM models can detect anomalies of type 4, type 8, and type 1 very well, beating the baseline. This can be because these anomalies have very different temporal behavior than normal type 0 sequences. Single LSTM models struggle with detecting type 2 sequences that have similar temporal behavior as type 0. Relatively worse performance on type 16 for all the models can be because type 16 sequences follow normal behavior until the broadcasted positions get stuck to one coordinate. Therefore these sequences should have lower reconstruction error when compared to other anomalies. However, deeper stacked LSTM models are still able to detect these very well.

In Fig. 7 accuracy is depicted as bar plots and the number of parameters as line plots. In the first plot, it is seen that accuracy scales very poorly with several parameters in single LSTM models. The second plot shows better accuracy for Stacked LSTM models. We can also see that accuracy scales better with the number of parameters when compared with single LSTM models. This shows that the higher accuracy in Stacked LSTM models is not because of more number of parameters but due to better model architecture. Therefore the Stacked

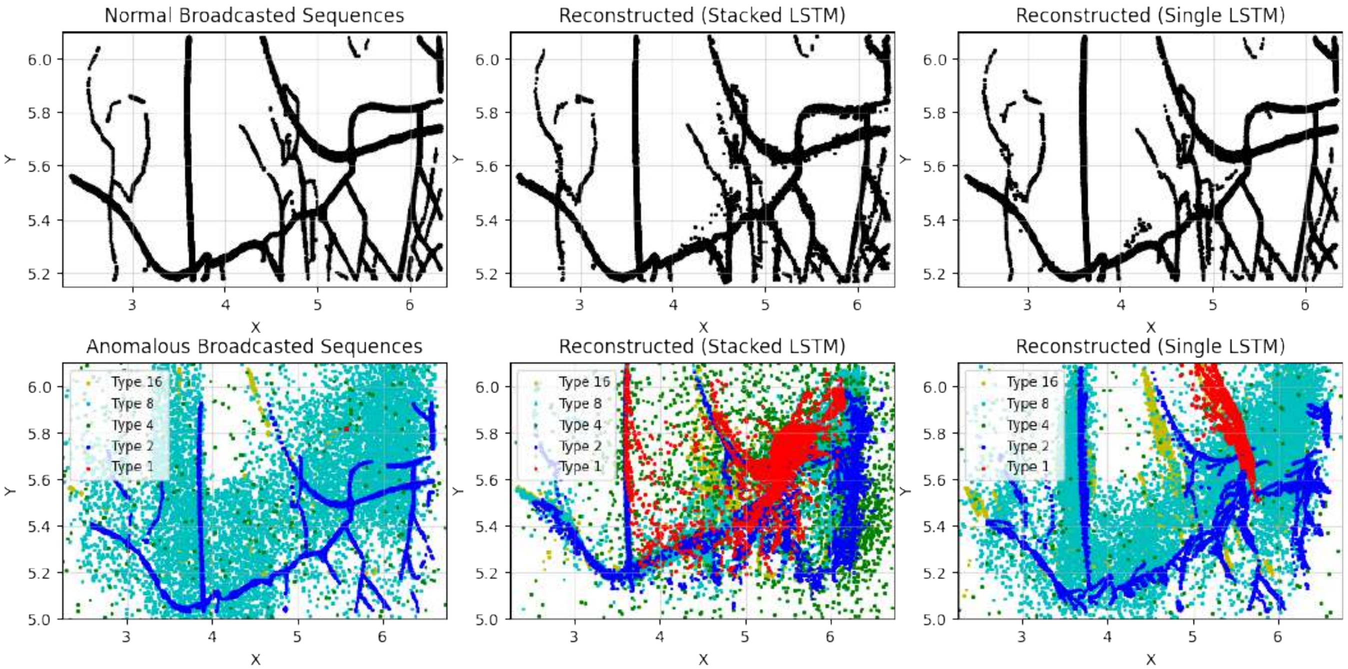

Fig. 8: Reconstruction of test sequences using the 4-layer Stacked LSTM model and 1024 units single LSTM.

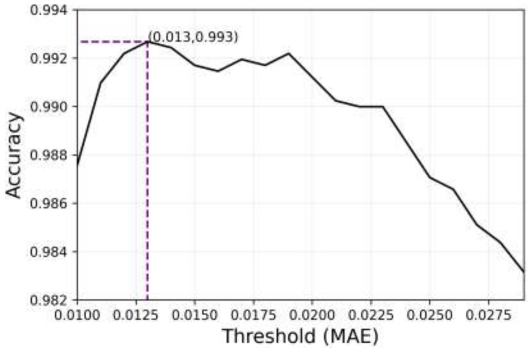

Fig. 9: Accuracy for different MAE thresholds.

LSTMs not only perform better than single LSTMs but also with much better parameter efficiency.

Fig. 8 shows the reconstruction of test sequences by the 4-layer Stacked LSTM model and 1024 units single LSTM. X and Y here are scaled X and Y coordinates from VeReMi simulations. It can be seen that both the models can reconstruct normal (type 0) sequences very well. The stacked LSTM model however is unable to reconstruct other sequence types. This is desirable as other types will thus have higher reconstruction loss and therefore easier to classify. It is to be noted that the stacked model is not even able to reconstruct type 2 sequences that follow similar temporal behavior as type 0 but differs in position coordinates by a constant offset. Also, it is observed that when trying to reconstruct anomalous sequences, the stacked model mostly gives out coordinates corresponding to type 0 sequences. This shows that the stacked model has also learned spatial information from normal behavior along with the expected temporal information. These spatio-temporal features are not observed in the reconstructions by normal LSTM model with 1024 units even with more than double the number of parameters.

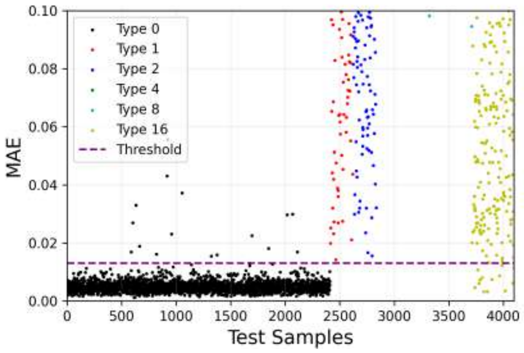

Fig. 10: Reconstruction loss distribution for test samples.

		True label	
		Normal	Anomaly
Predicted	Normal	2393	13
	Anomaly	17	1673

TABLE V: Confusion Matrix

We discuss the results of the 4-layer Stacked LSTM, which is the best performing model. The threshold for anomaly detection is determined from the distribution of reconstruction loss for test sequences. The best performance of 99.3% accuracy is observed for a threshold of 0.013 MAE as seen in Fig. 9. Fig. 10 shows reconstruction loss distribution for test samples of each type. The type 0 samples with reconstruction loss more than the threshold of 0.013 MAE are False Negatives (FN) and the other type (anomalous) samples with reconstruction loss lesser than the threshold are False Positives (FP). It is seen that anomalies of all types except type 16 have reconstruction errors more than the threshold. Also, very few FNs are observed. type 4 and type 8 samples have very high reconstruction loss and hence are not visible in the plot. The confusion matrix is shown in Table V.

VI. Conclusion

In this paper, we presented a secure, efficient, and intelligent multi-tier network architecture for the IoV networks. The Vehicle-side of this network is made secure using deep learning. Regional decisions are made by the edge servers, which communicate both with users and vehicles using user-side and vehicle-side networks respectively. It continuously receives user requests and vehicle traffic information. We trained a stacked LSTM model with unsupervised learning on a popular vehicular simulation dataset to demonstrate how it detects anomalous vehicle behavior in the network. We then summarized the performance and parameter efficiency for different LSTM models trained. The findings show that unlike normal LSTM models, the proposed stacked LSTM model can learn spatio-temporal information even with fewer parameters and hence perform better.

References

[1] A. H. Sodhro, S. Pirbhulal, G. H. Sodhro, M. Muzammal, L. Zongwei, A. Gurtov, A. R. L. de Macêdo, L. Wang, N. M. Garcia, and V. H. C. de Albuquerque, "Towards 5g-enabled self adaptive green and reliable communication in intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[2] A. H. Sodhro, J. J. Rodrigues, S. Pirbhulal, N. Zahid, A. R. L. de Macedo, and V. H. C. de Albuquerque, "Link optimization in software defined iov driven autonomous transportation system," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[3] J. Zhang, C. Xu, Z. Gao, J. J. Rodrigues, and V. Albuquerque, "Industrial pervasive edge computing-based intelligence iot for surveillance saliency detection," *IEEE Transactions on Industrial Informatics*, 2020.

[4] C. Wang, G. Yang, G. Papanastasiou, H. Zhang, J. Rodrigues, and V. Albuquerque, "Industrial cyber-physical systems-based cloud iot edge for federated heterogeneous distillation," *IEEE Transactions on Industrial Informatics*, 2020.

[5] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on iot security: application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82 721–82 743, 2019.

[6] T. Alladi, V. Kohli, V. Chamola, and F. R. Yu, "Securing the internet of vehicles: A deep learning based classification framework," *IEEE Networking Letters*, 2021.

[7] L. Nie, Y. Li, and X. Kong, "Spatio-temporal network traffic estimation and anomaly detection based on convolutional neural network in vehicular ad-hoc networks," *IEEE Access*, vol. 6, pp. 40 168–40 176, 2018.

[8] L. Nie, Y. Wu, H. Wang *et al.*, "Anomaly detection based on spatio-temporal and sparse features of network traffic in vanets," *IEEE Access*, vol. 7, pp. 177 954–177 964, 2019.

[9] F. A. Ghaleb, A. Zainal, M. A. Rassam, and F. Mohammed, "An effective misbehavior detection model using artificial neural network for vehicular ad hoc network applications," in *2017 IEEE Conference on Application, Information and Network Security (AINS)*. IEEE, 2017, pp. 13–18.

[10] D. Zhang, F. R. Yu, and R. Yang, "Blockchain-based distributed software-defined vehicular networks: A dueling deep q-learning approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1086–1100, 2019.

[11] T. Alladi, V. Chamola, N. Sahu, and M. Guizani, "Applications of blockchain in unmanned aerial vehicles: A review," *Vehicular Communications*, vol. 23, p. 100249, 2020.

[12] T. Alladi, V. Chamola, J. J. Rodrigues, and S. A. Kozlov, "Blockchain in smart grids: A review on different use cases," *Sensors*, vol. 19, no. 22, p. 4862, 2019.

[13] T. Zhang and Q. Zhu, "Distributed privacy-preserving collaborative intrusion detection systems for vanets," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 148–161, 2018.

[14] Y. Zeng, M. Qiu, D. Zhu, Z. Xue, J. Xiong, and M. Liu, "Deepvcm: A deep learning based intrusion detection method in vanet," in *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. IEEE, 2019, pp. 288–293.

[15] Y. Wang, Z. Tian, Y. Sun, X. Du, and N. Guizani, "Locjury: An ibn-based location privacy preserving scheme for iocv," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[16] W. Wang, M. Min, L. Xiao, Y. Chen, and H. Dai, "Protecting semantic trajectory privacy for vanet with reinforcement learning," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–5.

[17] E. Seo, H. M. Song, and H. K. Kim, "Gids: Gan based intrusion detection system for in-vehicle network," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2018, pp. 1–6.

[18] S. Tangade, S. S. Manvi, and S. Hassan, "A deep learning based driver classification and trust computation in vanets," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 2019, pp. 1–6.

[19] S. So, P. Sharma, and J. Petit, "Integrating plausibility checks and machine learning for misbehavior detection in vanet," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 564–571.

[20] P. Gu, R. Khatoun, Y. Begriche, and A. Serhrouchni, "k-nearest neighbours classification based sybil attack detection in vehicular networks," in *2017 Third International Conference on Mobile and Secure Services (MobiSecServ)*. IEEE, 2017, pp. 1–6.

[21] B. Sliwa, M. Haferkamp, M. Al-Askary, D. Dorn, and C. Wietfeld, "A radio-fingerprinting-based vehicle classification system for intelligent traffic control in smart cities," in *2018 Annual IEEE International Systems Conference (SysCon)*. IEEE, 2018, pp. 1–5.

[22] M. A. Rahim, L. Zhu, X. Li, J. Liu, Z. Zhang, Z. Qin, S. Khan, and K. Gai, "Zero-to-stable driver identification: A non-intrusive and scalable driver identification scheme," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 163–171, 2020.

[23] A. B. De Souza, P. A. Rego, T. Carneiro, J. D. C. Rodrigues, P. P. Rebouças Filho, J. N. De Souza, V. Chamola, V. H. C. De Albuquerque, and B. Sikdar, "Computation offloading for vehicular environments: A survey," *IEEE Access*, vol. 8, pp. 198 214–198 243, 2020.

[24] A. H. Sodhro, S. Pirbhulal, and V. H. C. de Albuquerque, "Artificial intelligence-driven mechanism for edge computing-based industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4235–4243, 2019.

[25] M. M. Hassan, M. R. Hassan, S. Huda, and V. H. C. de Albuquerque, "A robust deep learning enabled trust-boundary protection for adversarial industrial iot environment," *IEEE Internet of Things Journal*, 2020.

[26] T. Alladi, S. Chakravarty, V. Chamola, and M. Guizani, "A lightweight authentication and attestation scheme for in-transit vehicles in iov scenario," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 188–14 197, 2020.

[27] A. H. Sodhro, S. Pirbhulal, Z. Luo, and V. H. C. de Albuquerque, "Towards an optimal resource management for iot based green and sustainable smart cities," *Journal of Cleaner Production*, vol. 220, pp. 1167–1179, 2019.

[28] V. Chamola, C.-K. Tham, S. Gurunarayanan, N. Ansari *et al.*, "An optimal delay aware task assignment scheme for wireless sdn networked edge cloudlets," *Future Generation Computer Systems*, vol. 102, pp. 862–875, 2020.

[29] T. Alladi, V. Chamola, and S. Zeadally, "Industrial control systems: Cyberattack trends and countermeasures," *Computer Communications*, vol. 155, pp. 1–8, 2020.

[30] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[32] Z. Tian, C. Luo, J. Qiu, X. Du, and M. Guizani, "A distributed deep learning system for web attack detection on edge devices," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1963–1971, 2020.

[33] B. Naik, M. Obaidat, J. Nayak, D. Pelusi, P. Vijayakumar, and S. H. Islam, "Intelligent secure ecosystem based on meta-heuristic and functional link neural network for edge-of-things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1947–1956, 2019.

[34] X. An, X. Lü, L. Yang, X. Zhou, and F. Lin, "Node state monitoring scheme in fog radio access networks for intrusion detection," *IEEE Access*, vol. 7, pp. 21 879–21 888, 2019.

[35] H. Liu, P. Zhang, G. Pu, T. Yang, S. Maharjan, and Y. Zhang, "Blockchain empowered cooperative authentication with data traceability in vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4221–4232, 2020.

[36] X. Yao, H. Kong, H. Liu, T. Qiu, and H. Ning, “An attribute credential based public key scheme for fog computing in digital manufacturing,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2297–2307, 2019.

[37] C. Lu and H. Liu, “Generative adversarial networks based industrial protocol construction in the fog computing,” in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. IEEE, 2018, pp. 453–456.

[38] R. W. van der Heijden, T. Lukaseder, and F. Kargl, “Veremi: A dataset for comparable evaluation of misbehavior detection in vanets,” in *International Conference on Security and Privacy in Communication Systems*. Springer, 2018, pp. 318–337.