UNIVERSITY OF LEEDS

This is a repository copy of *Stacking Ensemble Models for Misbehavior Detection in Vehicular Networks*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/201470/

Version: Accepted Version

Proceedings Paper:
Fabiyi, S D orcid.org/0000-0001-9571-2964 and Ajibuwa, O E (Accepted: 2023) Stacking Ensemble Models for Misbehavior Detection in Vehicular Networks. In: Proceedings of 7th International Conference on Computing, Communication, Control and Automation. 7th International Conference On Computing, Communication, Control And Automation, 18-19 Aug 2023, Maharashtra, India. IEEE . (In Press)

This item is protected by copyright. This is an author produced version of a conference paper accepted for publication in Proceedings of the 7th International Conference on Computing, Communication, Control and Automation, made available under the terms of the Creative Commons Attribution License (CC-BY), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

Reuse
This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here: https://creativecommons.org/licenses/

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

White Rose university consortium
Universities of Leeds, Sheffield & York

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

Stacking Ensemble Models for Misbehavior Detection in Vehicular Networks

Opeyemi Emmanuel Ajibuwa
School of EECS
Oregon State University
Corvallis, USA
ajibuwao@oregonstate.edu

Samson Damilola Fabiyi
School of Computing
University of Leeds
Leeds, UK
S.D.Fabiyi@leeds.ac.uk

***Abstract*—Ensuring the safety and reliability of autonomous driving and vehicle-to-vehicle (V2V) communication in vehicular ad hoc networks (VANETs) requires effective misbehavior detection techniques. While threshold-based methods and various machine learning (ML) models have been widely used to identify and classify basic safety messages (BSMs) exchanged between vehicles, they have limitations in terms of their application and computational costs. In this paper, we propose a stacked ensemble tree-based ML model for detecting misbehavior of different types in VANETs. Our approach addresses the dynamic nature of VANETs while keeping computational costs low. Our experimental results show that the proposed stacked model outperforms base tree ML models in terms of F1-score and recall, while also reducing the model's overall execution time. Our work offers a promising approach for improving the performance and efficiency of misbehavior detection in VANETs.**

***Index Terms*—Vehicular Ad Hoc Networks (VANET), Vehicle-to-everything (V2X), Vehicle-to-Vehicle (V2V), Misbehavior Detection, Machine Learning, Cybersecurity, Ensemble Models**

I. Introduction

Vehicle-to-everything (V2X) communications have gained attention in recent years for their potential to support both safety and non-safety applications, such as forwarding collision warnings, route navigation, lane change warnings, and multimedia content sharing. In Vehicular Ad Hoc Networks (VANET), vehicles broadcast their current state (e.g., position, speed, heading, etc.) to establish collective awareness, but this also exposes the network to attacks that can disrupt traffic flow and cause accidents [1]. Misbehavior can take the form of exchanging false vehicles' kinematics or more sophisticated attacks such as denial-of-service (DoS) or Sybil attacks [2]–[5]. Identifying misbehaving entities and distinguishing between attacks and benign messages in VANET remain major challenges due to the dire consequences of propagating malicious messages. Cryptographic-based security techniques are often ineffective in detecting misbehavior because the attacks are often perpetrated by insider vehicles with valid credentials [4], [6].

Various misbehavior detection schemes (MDS) have been proposed in the literature to address this issue, but some of the existing techniques have limitations in real-world applications [7]. One limitation is that traditional rule-based (i.e., specification-based) detectors have difficulty adapting to changing traffic conditions and new types of misbehavior, and may have high false positive rates. Additionally, these methods can be difficult to scale as the number of rules increases with the number of vehicles in the network. Furthermore, MDS must be able to distinguish between misbehavior and anomalies caused by sensor faults. Machine learning/Deep learning-based (ML/DL) misbehavior detection techniques can overcome these limitations by allowing the system to learn from data and adapt to changing conditions. However, Machine learning/Deep learning-based schemes are limited by the availability of large labeled datasets for training, validation, and testing, as well as the complex, and computationally intensive training phase. Therefore, it is important to carefully evaluate the trade-offs when designing a misbehavior detection method for VANET.

In this work, we propose using tree and ensemble-based machine learning techniques to address the challenges and limitations of existing ML/DL techniques in misbehavior detection schemes for VANETs. We investigate various tree-based ML algorithms such as decision trees, random forests, extra trees, and extreme gradient boosting (XGBoost) to develop an MDS that can achieve high detection rates while minimizing computational overheads. To enhance the performance (measured in terms of F1-score) of the proposed system, we implement an ensemble learning model called stacking and apply feature selection techniques to reduce the computational time. We also evaluate our proposed solution using simulation data obtained from VEINs using the F^{2}MD framework [1]. Our results indicate that using a stacked ensemble ML misbehavior detection technique can effectively detect misbehavior in real time without compromising the computational cost for accuracy. Specific objectives include the following:

- Survey the literature for existing ML-based MDS for vehicular networks
- Implement tree-based and ensemble ML techniques for misbehavior detection in VANETs
- Evaluate performance of the proposed method for different attack types

The remainder of the paper is organized as follows: section II discusses related works. section III presents an overview of the proposed solution. section IV discusses the experiment and methods for evaluating the proposed solution. section V

presents and discusses the obtained results. Finally, section VI concludes the paper and provides some future scope of work.

II. Related Works

Various traditional approaches for detecting misbehaviour in vehicular networks have been proposed in the literature. For instance, Clavijo-Herrera et al. [9] investigated the effectiveness of three misbehavior detection mechanisms - threshold, behavioral, and cooperative - using the F^2MD framework. Their results showed that the threshold method outperformed the other two approaches in terms of accuracy and F1-score in the simulation scenarios tested. Begriche et al. [13] proposed and evaluated a reputation system based on Bayesian statistical filter to detect malicious nodes propagating blackhole and greyhole misbehavior in VANET.

Recently, machine learning models have found applications in misbehavior detection and achieved promising results. Kamel et al. [8] proposed an ML approach for detecting Sybil attacks in Cooperative Intelligent Transport Systems (CITS) and achieved a high accuracy (92.5%). Ghaleb et al. [7] presented a hybrid approach that combines statistical and ensemble learning-based classifiers to detect misbehaving nodes. Though the work in [7] achieved an accuracy of 92.8%, the method proposed in [7] relied on the assumption that mobility information obtained from neighboring vehicles is consistently accurate and reliable for creating dynamic thresholds. This assumption may not hold true in the presence of malicious nodes that distribute false or misleading data, potentially leading to traffic illusions. Furthermore, the size and complexity of their algorithm may restrict its practical application in large-scale traffic scenarios. Ercan et al. [11] proposed a ML-based approach for detecting position falsification attacks in vehicular networks. Their approach was based on using new features that they developed to train an ensemble of KNN and RF classifiers on the VeReMi dataset centrally. The trained model was then distributed to vehicles in the network for detecting misbehavior in a distributed fashion. Their results demonstrated the efficacy of their approach, which was extensively evaluated in terms of accuracy, F1-score, and computation time for each attack type at different traffic densities.

Sharma et al. [12] proposed a data-centric misbehavior detection framework that combines ML algorithms and plausibility checks to detect and classify misbehavior in vehicular networks using messages received from other vehicles in the network. Their approach was evaluated using six supervised ML algorithms on the VeReMi dataset, including voting and boosting-type ensemble models. The ensemble learning algorithms outperformed the others for most attacks, with an improvement of about 15% in the area under the curve (AUC). Integrating plausibility checks further improved precision and recall by 5% and 2% respectively. Gyawali et al. [4] proposed a hybrid method that combines ML and reputation-based misbehavior detection for 5G vehicular networks. The proposed MDS was trained using dataset generated through extensive simulation based on realistic vehicular network scenarios. To improve the accuracy of the proposed technique, Dempster-Shafer theory was employed for collaborative feedback combination, including a beta distribution-based reputation update and revocation scheme.

Some limitations can be noted when reviewing the misbehavior detection approaches in the literature. As can be seen in Table I, majority of the misbehavior detection approaches in the related work are only applicable on specific attack types and cannot be adapted to detect other types of misbehavior in vehicular networks. For instance, the focus of the work in [8] and [12] were limited to the Sybil and position forgery type of misbehavior respectively, overlooking other possible sources of attacks. Similarly, the work in [9] only considered four denial-of-service (DOS) attacks and its results may not be applicable under varying traffic scenarios. In another work [11], the proposed approach was deployed for detecting position falsification attacks. However, the method proposed in [11] may not generalize to large-scale vehicular networks where different attacks types may exist. Although the approaches proposed in [7], [10] and [14] were deployed to deal with more than one attack type, the computation latency was not considered when the effectiveness of the proposed misbehavior detection approaches was evaluated in those work. The computational complexity of ML based misbehavior detection approaches may pose a challenge with large-scale applications when they are not factored into the design process [**?**].

This paper therefore present a robust ML based misbehavior detection approach which can be applied to a range of attack types and also takes into account practical considerations such as the ML algorithm computation latency, which is essential for real-world deployment.

III. Proposed Misbehavior Detection System

The proposed MDS utilizes four popular tree-based algorithms, including Decision Tree, Random Forest, Extra Trees, and Extreme Gradient Boosting. This binary classification model aims to accurately identify whether a message is an attack or normal message.

Decision Tree (DT) uses a divide-and-conquer strategy that consists of decision nodes and leaf nodes to represent a decision test and the resulting class, respectively [15]. Random Forest (RF) is an ensemble learning classifier that leverages the majority voting rule to select the classification result from the class with the most votes from decision trees [16]. Extremely Randomized Trees (ET) is another ensemble model that builds a collection of randomized decision trees by analyzing various subsets of the dataset [17], [18]. Similarly, Extreme Gradient Boosting (XGBoost) is an ensemble learning algorithm that utilizes gradient descent to aggregate several decision trees for optimal speed and performance [18]. While other ML algorithms, such as K-Nearest Neighbor (KNN) and Support Vector Machines (SVM), were considered, their high computational costs were prohibitive factors. Additionally, the ensemble learning feature of tree-based models, coupled with their ability to calculate feature importance, made them ideal for our specific use case.

TABLE I
SUMMARY OF RELATED WORKS

References	Technique	Focus Area	Accuracy/F1-Scores	Computation time (ms)	Evaluation Method	Pub. Year
[4]	Machine-Learning and Reputation based	False alert and position falsification attacks	96%/-	2.01	Simulation dataset	2020
[7]	Statistical and Machine-Learning based	Different attack types	92.8%/-	n/a	Simulation dataset	2019
[8]	Machine-Learning based	Sybil Attacks	92.5%/96.4%	n/a	Simulation dataset	2019
[9]	Threshold, Behavioral and Cooperative	Denial of Service (DoS) attacks	97%/99.3%	n/a	Simulation	2021
[10]	Machine-Learning based	Different attack types	97%/-	n/a	Simulation dataset	2020
[11]	Machine-Learning based	Position falsification attacks	90.3%/-	10728.4	Simulation dataset	2022
[12]	Plausibility checks and Machine-Learning based	Position falsification attacks	69.84%/-	n/a	Simulation dataset	2021
[13]	Reputation based	Blackhole and greyhole attacks	-/-	n/a	Simulation	2020

$^{n/a}$not applicable

IV. METHODOLOGY

In this section, we provide a brief overview of the steps involved in developing the proposed solution. Figure 1 presents an overview of the framework for ensemble machine learning-based misbehavior detection. We describe each process of the proposed model below.

A. Simulation Settings and Scenarios

We created two separate data sets for training and testing purposes by simulating two different traffic scenarios based on the Luxembourg SUMO Traffic Scenario (LUST) network in F^{2}MD. F^{2}MD is an extension to VEINS [19], an inter-vehicular communication simulation framework that integrates OMNET++ [20] as a network simulator and a road traffic simulator (SUMO) for road traffic simulation [21]. The LUST scenario is based on actual traffic information within the city of Luxembourg and has been widely used for VANET simulations [22].

The training dataset contains 6,298 vehicles with 4,695,795 messages, while the testing dataset includes 4,369 vehicles with a total of 2,908,498 messages. The entire simulation process took about 8 hours, and an attacker rate of 20% was set. We performed the simulations on a workstation that featured an AMD Ryzen 9 5900HX @ 3.3 GHz with Nvidia RTX 3060 GPU and 32GBs of RAM. We tested all the implemented ML algorithms on the same workstation using the data obtained from the simulation with the parameters provided in Table II.

TABLE II
SIMULATION PARAMETERS

Parameter	Value
Mobility scenario	SUMO LUST
Vehicle density	4369, 6298
Simulation duration	8 hours
Attacker probability	0.2
Bit rate	6Mbps
Simulation area	2.3 $\times$ 2.3 km^2
Beacon size	64B
Beacon Interval	1s
Bandwidth	10Mhz
Channel Frequency	5.89Ghz
Sensitivity	-89dBm
Transmit power	20mW

B. Data Extraction

At this stage, simulated dataset were collected to train and test the machine learning algorithms of our proposed model. We briefly describe the details of the data extraction as follows.

1) The data extracted from the VEINS/F^{2}MD simulator is encoded in JSON file format and follows the naming convention `traceJSON-$vehicleID$-$A0-54$`, which represents the received messages for each vehicle in the simulation. For example, the name of an arbitrary file `traceJSON-50-A7` represents:
   - `$vehicleID$` denotes the receiver vehicle ID, in this case 50.
   - `$A0-15$` is the vehicles attack type. A0 means a receiver vehicle is genuine. Vehicles in the network can be either genuine or have any of the other 15 attack types described in [1]. In this example case, the receiver vehicle's misbehavior is `RandomSpeed`.
2) Every traceJSON file comprises two distinct types of messages:
   - The first type, denoted as `type:2` messages, refers to self-messages that represents information derived from the vehicle's own sensors, such as GPS. Specifically, `type:2` messages contain the position and speed

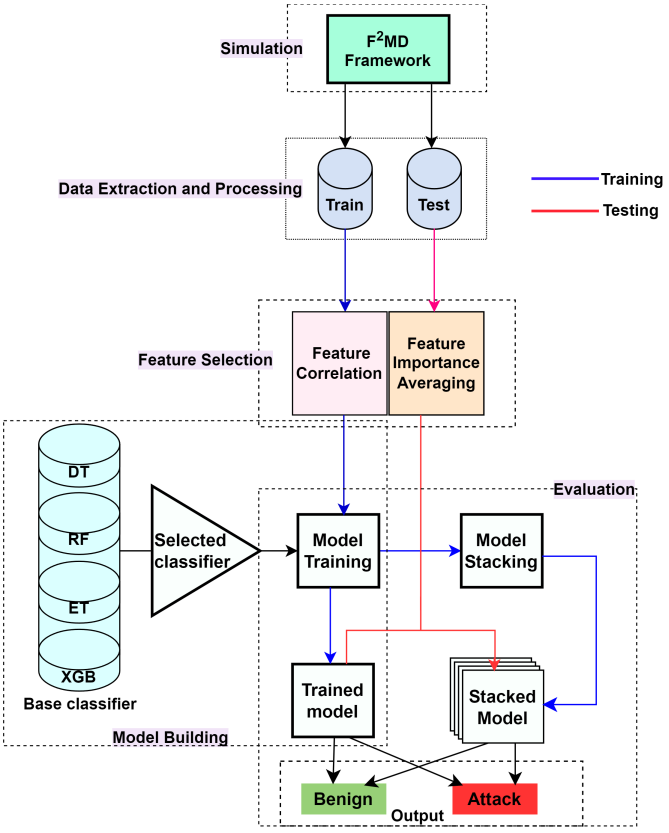

Fig. 1. The proposed ensemble misbehavior detection model

traces of the vehicle with ID 50.

- The second type, referred to as `type:3` messages, represents messages received from other vehicles and is annotated with the corresponding sender's identification (`senderID`). If a sender is denoted as an attacker, its message are deemed as an attack or a misbehavior. Each message possesses a unique identifier (`messageID`), which will be used to evaluate the performance of our proposed MDS.
- Both `type:2` and `type:3` messages include state information on vehicle position, speed, acceleration, and heading. The features are accompanied by the corresponding noisy components, such as position_noise, speed_noise, acceleration_noise and heading_noise, which represent the confidence of the receiver on the noise levels of the sender's message. Lower noise levels indicate more precise state information, and the noise can arise due to various factors such as communication channel disturbance or sensor measurement errors.

We solely collected `type:3` data from the JSON file because `type:2` data corresponds to sensor readings captured by individual vehicles, which are not considered misbehavior but are rather benign. Such messages are not shared with any other vehicle and remain confined to the originating vehicle. At the conclusion of this stage, we obtained a training dataset that consisted of 4,695,795 instances and 22 features, as well as a test dataset comprising 2,908,498 instances and 22 features.

C. Data Preprocessing

During the preprocessing stage, we found a class imbalance with more normal than attack messages due to the 20% attacker rate that was used in the simulation. Oversampling was considered but rejected due to increased dataset size and added computational overhead. Instead, we measured the proposed MDS performance with F1-score to adjust for the class imbalance. Preprocessing also included finding and eliminating duplicates, numerical feature normalization, and replacing null values with zeros.

D. Feature Selection

During the feature selection stage, the goal is to select the most important features to train the model while reducing the computational cost. To achieve this, two methods were employed. First, a correlation matrix was generated for the dataset, and features with the highest correlation to the target label were selected. Additionally, to avoid redundancy, features with a high correlation to each other were compared, and one of the two features with a correlation above 0.9 was removed. This step led to the elimination of several features such as `rcvTime`, `sendTime`, `sender` and `messageID`.

Secondly, an ensemble feature selection technique was employed, which involved calculating the feature importance lists generated by four tree-based machine learning models. Tree-based algorithms are useful for feature selection because they compute the importance of each feature based on each individual tree and average the output of the trees to make the result more reliable [18]. To select the most important features, the sum of the total feature importance was set to 1.0. Features were ranked based on their importance, and those with the highest importance were added to a list until the sum of importance reached 0.9. This allowed the selection of the top features while discarding the lesser important ones to reduce computational costs.

By employing this multilevel approach to feature selection, a final feature list of 9 was obtained, as shown in Table III, which resulted in the best performance for the proposed MDS.

TABLE III
SELECTED FEATURES FOR MODEL BUILDING

Features Used	Feature Discarded
sender	type, messageID, sendTime, recvTime
pos_x, pos_y	pos_noise_x, pos_noise_y
speed_x, speed _y	spd _noise_x, spd _noise_y
acl_x, acl_y	acl_noise_x, acl_noise_y
hed_x, hed_y	hed_noise_x, hed_noise_y

E. Model Building

After performing the necessary preprocessing steps on the dataset, four base models are trained. These models predict output labels, which are then used as input for the final

stacking ensemble model. The ensemble method, known as stacking, is employed to enhance the F1-score and reduce the computational latency. Stacking is a widely used ensemble technique comprising of two layers. The first layer contains a small number of trained base predictors whose outputs are utilized as inputs for a meta-learner in the second layer to construct a robust classifier. In this particular case, three of the four trained tree structure algorithms act as the base models in the first layer of the stacking ensemble method. The singular algorithm with the highest accuracy among these four base models is chosen to be the meta-classifier in the second layer. Ultimately, the trained stacked model is constructed to categorize each instance of test data as either an attack or normal message.

V. Results and Discussion

In this section, we present the evaluation results for the proposed MDS, which consists of four tree-based algorithms and a stacked model. Given our class-imbalanced dataset, we chose F1-score as the accuracy metric and also considered recall and total execution time (sum of training and inference time) as additional evaluation metrics. The dataset was divided into 80% for training and 20% for validation. We evaluated both the validation and test sets and discussed the performance in three-parts.

A. Before Feature Selection on Validation Set

The results of evaluating different algorithms on the validation dataset are summarized in Table IV. All four algorithms, namely DT, RF, ET, and XGBoost, achieved high classification results. RF and ET performed exceptionally well, reaching 99.8% in both recall and F1-score, indicating that they can accurately identify attacks (misbehavior) with up to 99% accuracy. However, RF had the worst performance in terms of computational overhead, taking approximately 434 seconds to train and infer on the validation set. This is partly because unlike the other three algorithms which were multi-threaded, RF is inherently single-threaded based on the python library used.

To address this issue, DT, ET, and XGBoost were selected as the base models for a stacked ensemble model, with DT serving as the meta-classifier. This approach significantly reduced the execution time to 1.7 seconds, a 99% improvement over the base DT model's execution time of about 185 seconds. The stacked model achieved 99.8% in F1-score and recall, matching the accuracy of the DT classifier while reducing execution time.

B. After Feature Selection on Validation Set

To speed up the algorithms, the feature selection methods from Section IV-C were used. This involved reducing the training set size based on the features in Table III, and then retraining the models on this modified dataset using the same 80-20% split. The classification accuracy for all 5 models remained the same, as shown in Table V. In addition, feature selection reduced the execution time of the models, with the Stacked model and DT providing the best savings. Specifically, DT reduced execution time by 63%, RF by 41%, ET by 38%, XGBoost by 56%, and Stacked by 82%.

TABLE IV
Evaluation of Validation set before Feature Selection

Method	F1-Score	Recall	Total Execution Time (s)
DT	0.997	0.997	184.5
RF	0.998	0.998	433.4
ET	0.998	0.998	227.7
XGBoost	0.997	0.997	237.6
Stacked	**0.998**	**0.998**	**1.7**

TABLE V
Evaluation of Validation set After Feature Selection

Method	F1-Score	Recall	Total Execution Time (s)
DT	0.997	0.998	68.1
RF	0.998	0.998	255
ET	0.998	0.998	140
XGBoost	0.997	0.997	105
Stacked	**0.998**	**0.998**	**0.3**

C. After Feature Selection on Test Set

The evaluation results of the trained models on the test data are presented in Table VI. Unfortunately, the classification performance of all 5 models decreased significantly on the test data. The best performing model was the stacked ensemble, achieving an F1-Score of 68.2% and recall of 53.7%. Despite the low F1-score and recall performance of the models on the test set, we consider the results promising. This is due to the low computational overheads of the stacked model and the broad range of attack types considered in the dataset. With further fine-tuning, these characteristics can make the proposed ensemble stacked model suitable for real-world deployments.

TABLE VI
Evaluation of Test set After Feature Selection

Method	F1-Score	Recall	Inference Time (s)
DT	0.602	0.617	0.4
RF	0.658	0.568	5.9
ET	0.67	0.514	7.4
XGBoost	0.63	0.588	0.6
Stacked	**0.682**	**0.537**	**0.1**

VI. Conclusion and Future Works

Misbehavior detection in VANETs is of absolute importance for realizing fully self-autonomous driving and V2V communication. In this paper, we propose a tree-based machine learning misbehavior detection system that can effectively detect misbehaviors of different types without incurring huge computational costs. We introduce feature selection by correlation and averaging and stack the tree-based algorithms into an ensemble model to save costs and improve the model's execution time. We implement the proposed MDS in Python

and test its performance using simulated dataset obtained using the F2MD framework. The results show that the proposed stacked ensemble model has a higher F1-score and recall and lower execution time than the base tree models, although by a small margin. We argue that with further optimization, the stacked ensemble model can offer a suitable balance in terms of classification accuracy and costs.

In future work, we will explore further feature engineering on the dataset, specifically creating new physics-based features to model each attack type in the dataset, which can further reduce the stacked model's overfitting. Although this idea has been explored in some existing literature, it has not been explored in the context of detecting all possible known attacks. Furthermore, we will extend the proposed work to a multiclass classification approach to correctly classify messages as either faults, attacks, or normal.

ACKNOWLEDGMENTS

For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission.

REFERENCES

[1] J. Kamel, M. R. Ansari, J. Petit, A. Kaiser, I. B. Jemaa, and P. Urien, "Simulation framework for misbehavior detection in vehicular networks," *IEEE transactions on vehicular technology*, vol. 69, no. 6, pp. 6631–6643, 2020.

[2] Y. Qian and N. Moayeri, "Design of secure and application-oriented vanets," in *VTC Spring 2008-IEEE Vehicular Technology Conference*. IEEE, 2008, pp. 2794–2799.

[3] Z. A. Biron, S. Dey, and P. Pisu, "Real-time detection and estimation of denial of service attack in connected vehicle systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3893–3902, 2018.

[4] S. Gyawali, Y. Qian, and R. Q. Hu, "Machine learning and reputation based misbehavior detection in vehicular communication networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8871–8885, 2020.

[5] Z. Yang, K. Zhang, L. Lei, and K. Zheng, "A novel classifier exploiting mobility behaviors for sybil detection in connected vehicle systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2626–2636, 2018.

[6] F. Ahmad, V. N. Franqueira, and A. Adnane, "Team: A trust evaluation and management framework in context-enabled vehicular ad-hoc networks," *IEEE Access*, vol. 6, pp. 28 643–28 660, 2018.

[7] F. A. Ghaleb, M. A. Maarof, A. Zainal, B. A. S. Al-rimy, A. Alsaeedi, and W. Boulila, "Ensemble-based hybrid context-aware misbehavior detection model for vehicular ad hoc network," *Remote Sensing*, vol. 11, no. 23, p. 2852, 2019.

[8] J. Kamel, F. Haidar, I. B. Jemaa, A. Kaiser, B. Lonc, and P. Urien, "A misbehavior authority system for sybil attack detection in c-its," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 2019, pp. 1117–1123.

[9] M. Clavijo-Herrera, J. Banda-Almeida, and C. Iza, "Performance evaluation in misbehaviour detection techniques for dos attacks in vanets," in *Proceedings of the 18th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, 2021, pp. 73–80.

[10] I. Mahmoudi, J. Kamel, I. Ben-Jemaa, A. Kaiser, and P. Urien, "Towards a reliable machine learning-based global misbehavior detection in c–its: Model evaluation approach," in *Vehicular Ad-hoc Networks for Smart Cities: Third International Workshop, 2019*. Springer, 2020, pp. 73–86.

[11] S. Ercan, M. Ayaida, and N. Messai, "Misbehavior detection for position falsification attacks in vanets using machine learning," *IEEE Access*, vol. 10, pp. 1893–1904, 2021.

[12] P. Sharma and H. Liu, "A machine-learning-based data-centric misbehavior detection model for internet of vehicles," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4991–4999, 2020.

[13] Y. Begriche, R. Khatoun, A. Rachini, and L. Khoukhi, "A reputation system using a bayesian statistical filter in vehicular networks," in *2020 Sixth International Conference on Mobile And Secure Services (MobiSecServ)*. IEEE, 2020, pp. 1–7.

[14] F. A. Ghaleb, F. Saeed, E. H. Alkhammash, N. S. Alghamdi, and B. A. S. Al-Rimy, "A fuzzy-based context-aware misbehavior detecting scheme for detecting rogue nodes in vehicular ad hoc network," *Sensors*, vol. 22, no. 7, p. 2810, 2022.

[15] L. Rokach and O. Maimon, "Decision trees," *Data mining and knowledge discovery handbook*, pp. 165–192, 2005.

[16] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, pp. 197–227, 2016.

[17] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, pp. 3–42, 2006.

[18] L. Yang, A. Moubayed, I. Hamieh, and A. Shami, "Tree-based intelligent intrusion detection system in internet of vehicles," in *2019 IEEE global communications conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[19] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved ivc analysis," *IEEE Transactions on mobile computing*, vol. 10, no. 1, pp. 3–15, 2010.

[20] A. Varga, "Discrete event simulation system," in *Proc. of the European Simulation Multiconference (ESM'2001)*, 2001, pp. 1–7.

[21] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of sumo-simulation of urban mobility," *International journal on advances in systems and measurements*, vol. 5, no. 3&4, 2012.

[22] L. Codeca, R. Frank, and T. Engel, "Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research," in *2015 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2015, pp. 1–8.

APPENDIX

Data Availability Statement: The data that support the findings of this study are available on request from the first author.