Computers & Security 142 (2024) 103881

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Fast and practical intrusion detection system based on federated learning for VANET☆

Xiuzhen Chen [a,b,*], Weicheng Qiu [a], Lixing Chen [a,b], Yinghua Ma [a,b], Jin Ma [a,b]

[a] Institute of Cyber Science and Technology, Shanghai Jiao Tong University, Shanghai, PR China
[b] Shanghai Key Laboratory of Integrated Administration Technologies for Information Security, Shanghai Jiao Tong University, Shanghai, PR China

ARTICLE INFO

Keywords:
VANET
Intrusion detection
Machine learning
DST
Federated learning

ABSTRACT

VANET facilitates communication between vehicles and roadside units, thereby enhancing traffic safety and efficiency. However, it also introduces several attack vectors, compromising security and privacy. To enhance the practicality and precision of existing VANET intrusion detection systems, we propose a novel hybrid algorithm named VAN-IDS. This algorithm combines packet analysis and physical feature analysis. Temporal properties of packets are analyzed using Bi-LSTM, while physical attributes of vehicles are evaluated using LightGBM. Predictive outputs from both models are merged using the Dempster–Shafer theory. Data is collected from the F2MD simulation environment, and experiments are conducted to evaluate the efficacy of VAN-IDS. Results show that VAN-IDS achieves an accuracy exceeding 98.5% in detecting DoS and Sybil attacks, surpassing existing VANET IDSs and meeting real-time detection requirements. Federated learning is introduced, with VAN-FED-IDS designed to accelerate model training and updates, while protecting the privacy of the model and dataset. Roadside units (RSUs) serve as edge computing nodes for local model training, while cloud services aggregate models for collaborative training of detection algorithms. In these experiments, the open-source federated learning frameworks Flower and FedTree are used to assess communication overhead and training duration in both single-machine simulation and multi-machine deployment scenarios. This demonstrates the viability of federated learning, which can protect local data privacy, expedite model training and updates within the IDS, and maintain the original model's detection accuracy.

1. Introduction

Vehicular ad hoc networks (VANETs) constitute wireless multi-hop networks wherein information is relayed among vehicles. Initially proposed as an application for vehicles within Mobile ad hoc networks (MANETs), this consideration took into account the rapid movement of vehicle nodes and the dynamic topology. VANETs play a crucial role in enhancing traffic safety and efficiency through collision avoidance, platooning, and emergency notifications (Rasheed et al., 2017), serving as an integral component of intelligent transportation systems. Classification includes vehicle-to-vehicle (V2V) and vehicle-to-roadside (V2R) communication. Typical standards for VANET include Dedicated Short-Range Communication (DSRC), Wireless Access in Vehicular Environments (WAVE), and IEEE 802.11p. These standards enable the exchange of information between on-board units (OBUs) in vehicles and roadside units (RSUs).

Despite rapid advancements in transportation applications, VANET encounters numerous security and privacy challenges. VANET needs to encompass all free spaces for vehicles. The design of VANET primarily focuses on communication efficiency to ensure fast and reliable transmission, without solely depending on sophisticated authentication and encryption schemes for protection against external and internal attacks. Intrusions, including DDoS and Sybil attacks, pose threats to availability, confidentiality, integrity, and authenticity (Lu et al., 2018). In VANET, vehicle trajectory and safety messages are transmitted, which attackers could use to compromise user privacy.

An Intrusion Detection System (IDS) serves as a network security technology used to detect attempts to compromise system confidentiality, integrity, or availability. Within the VANET context, it quickly identifies malicious activities during attacks, providing effective real-time protection for VANETs. Machine learning (ML) methods adeptly

☆ Foundation Items:Action Plan of Science and Technology Innovation of Science and Technology Commission of Shanghai Municipality, China (No. 22511101202), Project supported by the Joint Funds of the National Natural Science Foundation of China (No. U2003206).

* Corresponding author.
E-mail addresses: chenxz@sjtu.edu.cn (X. Chen), qwc011235@alumni.sjtu.edu.cn (W. Qiu), lxchen@sjtu.edu.cn (L. Chen), ma-yinghua@sjtu.edu.cn (Y. Ma), majin@sjtu.edu.cn (J. Ma).

https://doi.org/10.1016/j.cose.2024.103881
Received 14 January 2024; Received in revised form 16 April 2024; Accepted 26 April 2024
Available online 30 April 2024

model traffic and differentiate between malicious and normal traffic, thus eliminating the need for manual rules. Recently, they have been extensively applied in VANET IDSs. However, ML-based IDSs encounter numerous challenges. Firstly, the majority of ML methods, primarily crafted for Ethernet or wireless networks, overlook the features of VANET. Few VANET IDSs propose high-level designs to enhance practicability. Secondly, ML models require large datasets for training, necessitating substantial computational resources. To optimally utilize VANET's limited computational and storage capacities, lightweight intrusion detection algorithms and distributed model learning approaches are essential. Furthermore, data and model privacy protection must be integrated into IDS design. Moreover, evaluations of VANET IDSs regarding accuracy, model training time, and communication are limited, leaving their practicality unverified.

Federated learning, a broadly utilized distributed machine learning framework (Yang et al., 2019), has been applied in a multitude of machine learning domains. It effectively tackles the challenge of data isolation. By sharing model parameters rather than datasets, all participating entities can collaboratively train models. Given the nature of VANET as an edge computing system where Roadside Units (RSUs) and vehicles act as edge computing nodes, and the cloud oversees global computing tasks, like model aggregation, federated learning is highly suitable for VANET (Posner et al., 2021). It facilitates the development of a rapid and secure intrusion detection system (IDS) for VANET.

In this study, we introduce VAN-FED-IDS, a vehicular ad-hoc network (VANET) intrusion detection system leveraging federated learning. Our system is designed to collect and preprocess network traffic data on Roadside Units (RSUs). We have developed a novel intrusion detection algorithm, VAN-IDS, which employs the Dempster–Shafer Theory (DST) to integrate packet-based and physics-based intrusion detection systems (IDSs). This integration facilitates improved traffic analysis and more precise predictions. To guarantee rapid and privacy-preserving model training and updates, we implement federated learning, facilitating collaborative data learning across various RSUs. RSUs are the cornerstone of the system, serving both as intrusion detectors in the IDS and as data producers and learning clients in federated learning (FL). The cloud orchestrates model aggregation and conducts global analysis of detection results. By identifying anomalies, RSUs empower VAN-FED-IDS to react and initiate actions more swiftly than centralized IDS systems where the cloud handles all tasks.

In summary, the main contributions of the proposed system are as follows:

(1) **Practical high-level design** We introduce VAN-FED-IDS, a high-level design aimed at enhancing the practicality of IDS within VANET environments. Utilizing the edge computing framework within VANET, we incorporate federated learning to maximize the use of limited computational resources and expedite model training. Unlike previous VANET IDS approaches that primarily concentrate on detection algorithm accuracy, we detail the process of integrating an accurate IDS into an existing VANET. We offer an illustration depicting the roles of all parties involved in the system and the runtime workflow, encompassing model updates and inference. To more effectively address the challenges of diverse distribution and non-IID issues in machine learning applications that might diminish generalization, VAN-FED-IDS combines neural network-based and tree-based models to deliver robust detection results.

(2) **Fast and lightweight intrusion detection algorithm** Instead of solely emphasizing detection accuracy, we introduce VAN-IDS, a swift and efficient intrusion detection algorithm tailored for VANET, combining packet-based and physics-based IDSs through DST. The packet-based IDS extracts time-series features from sequential packets, while the physics-based IDS checks if the packet information, like location and speed, adheres to physical laws. Both IDSs utilize machine learning models of low computational complexity for traffic data classification. VAN-IDS merges predictions from both IDSs using a rule-based method to generate a final outcome. Federated learning is employed to optimize computational resource usage and speed up model training and updates.

(3) **Systematic experimental evaluations** We carry out extensive evaluations and experiments on VAN-FED-IDS to showcase the accuracy of VAN-IDS and the practicality and efficacy of incorporating FL into the system. Initially, we gather data using the F2MD simulation framework to evaluate the accuracy of VAN-IDS. We conduct ablation and comparative experiments to illustrate that the DST-based IDS surpasses both single packet-based or physics-based models, as well as state-of-the-art algorithms. Next, we simulate FL on a single machine to demonstrate that FL hastens model training, guarantees data privacy, and upholds the same accuracy level as centralized training. Lastly, we establish a prototype system across multiple physical machines to evaluate the communication cost and efficiency.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents the detailed design of VAN-FED-IDS. Section 4 shows experimental results, followed by conclusions in Section 5.

2. Related work

Intrusion detection is a crucial line of defense for ensuring the security of VANETs. Previous research has primarily concentrated on developing machine learning algorithms to enhance the accuracy of intrusion detection and implementing a distributed architecture for VANETs, thereby improving the practicality of VANET IDS.

2.1. VANET intrusion detection methods

Several studies have developed deep learning models to enhance the detection accuracy and generalization of VANET IDS against unknown attacks. Karthiga et al. (2022) trained a rule-based Adaptive Neuro Fuzzy Inference System to differentiate between known and unknown attacks. They also developed an enhanced LeNet CNN for identifying unknown attacks. The experiments conducted on i-VANET and CIC-IDS-2017 datasets demonstrate that the proposed IDS can achieve an accuracy of over 98% in detecting brute force, port scanning, denial of service, botnet attacks, and others. Huang and Ma (2023) proposed an incremental learning-based VANET IDS that detects unknown attacks through model updates. Uncertain data was stored on the blockchain during the inference stage, and this data can be utilized for model updates following expert annotation. The ISVM algorithm was employed to iteratively update the model using a subset of the dataset. Experimental results on the publicly available AWID2 dataset demonstrated that the proposed method significantly reduces computation and communication overhead while maintaining high detection accuracy. However, the aforementioned schemes, despite aiming to enhance the accuracy and generalization of VANET IDS, did not extract vehicle-related features and were solely evaluated on general IDS datasets instead of VANET IDS datasets, potentially compromising the practicality of their IDSs in VANET.

IDSs that extract vehicle-related features and consider distributed architecture are more suitable for VANET. Sontakke and Chopade (2022) proposed an intrusion detection and mitigation system (IDMS) that extracted flow features of vehicles, such as behavior similarity to nearby vehicles, and location features, such as sending location requests to other vehicles to verify position reasonableness. They utilized an enhanced particle swarm optimization algorithm to accelerate the training of multi-classification DNN. The effectiveness of the proposed method was evaluated through ablation and comparative experiments in the MATLAB simulation environment. Zhou et al. (2020) considered the delay and reliability requirements of VANET IDS and proposed a distributed collaborative IDS framework called DCDIV to optimize storage and computing resources. They devised a reputation-based collaborative communication method to establish a trusted and stable communication link, employed dynamic behavior analysis to detect

malicious behaviors, and ultimately employed a random Petri net to depict the system state and security status. Simulation experiments demonstrated that DCDIV attained a high accuracy and detection rate. While numerous studies have enhanced the applicability of VANET IDS, few of them addressed the deployment location or run-time workflow.

2.2. VANET simulation environments and datasets

To facilitate comparison, it is necessary to have a simulation environment or public dataset that incorporates traditional VANET communication standards as well as VANET attacks like DDoS and sybil attacks.

van der Heijden et al. (2018) provided an open-source dataset called VeReMi, which captures abnormal behavior in VANETs and includes attack patterns such as random location and random offset speed. Kamel et al. (2020) improved the original VeReMi dataset by incorporating a more realistic sensor error model, expanding the range of attacks, and collecting additional data. They also developed several basic detection algorithms as baseline models for further research and comparison. Kamel et al. (2019b) developed the F2MD system for VANET simulation and misbehavior detection, using the Veins simulation environment. They implemented sybil attacks, DDoS attacks, and fault message attacks. The authors proposed plausibility and consistency check features, converting them into continuous values to create a dataset that is more suitable for machine learning algorithms. The authors compared rule-based and MLP-based IDS using these features, with the latter outperforming the former. Kamel et al. (2019a) and Mahmoudi et al. (2020) conducted additional comparisons of various machine learning algorithms using the dataset generated by F2MD. These algorithms included nonlinear kernel support vector machines (SVC), long short-term memory networks (LSTM), XGBoost, DNN, and combinations of Bi-LSTM and DNN. In terms of binary classification, LSTM and XGBoost demonstrated higher detection accuracy, while the hybrid model combining DNN and Bi-LSTM achieved the best performance in multi-classification tasks.

2.3. Federated learning in VANET IDS

Introducing federated learning into VANET IDS enables collaborative model training in distributed systems while ensuring security and data privacy protection. According to Liu et al. (2021b), the limited computation resources in VANET hinder the rapid training and updating of IDS models. However, by offloading model training tasks to vehicles and roadside units, federated learning can address this issue. Additionally, they employed blockchain technology to safeguard the security of model parameters. Their federated learning-based IDS offers enhanced privacy and reduced computing and communication costs. Lv et al. (2021) proposed a similar IDS framework that integrates federated learning and blockchain. In addition to storing model parameters on the blockchain, smart contracts were utilized to incentivize client nodes to participate in global model aggregation. However, these studies only focused on developing a high-level framework for federated learning in VANET IDS and did not conduct in-depth research or systematic experiments on intrusion detection algorithms.

An ideal IDS for VANET should utilize a lightweight algorithm while maintaining high accuracy. Additionally, it is necessary to design a high-level IDS framework that takes into account the distributed architecture of VANET in order to enhance its practicality for deployment.

3. Federated VANET IDS: VAN-FED-IDS

This section provides an introduction to the system design of VAN-FED-IDS. Firstly, we will present the architecture of VAN-FED-IDS. This will be followed by a discussion of the DST-based intrusion detection algorithm VAN-IDS and the run-time design of the system.

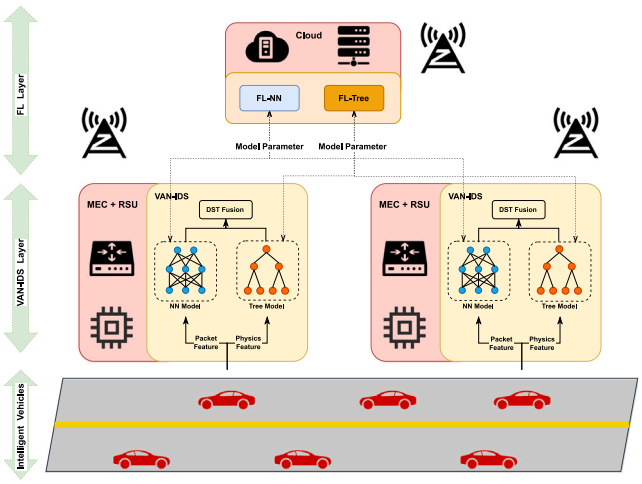


Fig. 1. Architecture of VAN-FED-IDS.

3.1. Architecture of VAN-FED-IDS

In this paper, our main focus is on Vehicle-to-Roadside communication in VANET, which is achieved through message exchange between Roadside Units (RSUs) and On-Board Units (OBUs). The VAN-FED-IDS system consists of two layers, namely the VAN-IDS Layer and the FL Layer, as illustrated in Fig. 1.

The VAN-IDS layer in our system refers to the intrusion detection algorithm that is responsible for detecting anomalies in Vehicle-to-Roadside (V2R) communication. Taking inspiration from DST-IDS (Dempster–Shafer Theory-based IDS), proposed in Qiu et al. (2022), we introduce a DST-based hybrid VANET IDS. This IDS combines packet-based and physics-based features to improve detection accuracy. DST-IDS is a lightweight intrusion detection framework known for its practicality, reduced data volume, and flexibility. It is designed to effectively handle diverse distribution and non-IID (non-independent and identically distributed) data. Experimental results demonstrate that DST-IDS achieves higher detection accuracy and lower time consumption compared to previous IDSs. This is particularly crucial in the design of VANET IDS.

The FL layer employs federated learning to expedite model training and ensure the privacy of security-relevant data. This study outlines the reasons for adopting federated learning, as follows: Firstly, the data for model training includes confidential user information, such as location data, vehicle identities, and historical intrusion instances. Data custodians, including vehicle owners and automotive manufacturers, are hesitant to share their data but are interested in collaborative learning and inference of IDS models without exposing local data. Second, an edge computing framework can be implemented in VANETs (Posner et al., 2021). Serving as intermediate nodes between vehicles and the cloud, base stations and Road Side Units (RSUs) are optimal for edge computing. Edge computing, combined with FL, can more effectively allocate global computing resources and optimize communication and computing efficiency, while meeting the IDS requirements of VANETs. Thirdly, IDS models require updates during runtime deployment. FL provides a more flexible and efficient approach, allowing selected nodes to collaboratively update the models.

The core hardware and infrastructure of VAN-FED-IDS consist of OBUs, RSUs, and MECs. RSUs and OBUs facilitate data exchange. OBUs, onboard units installed in intelligent vehicles, enable V2X communication. RSUs, roadside units, are typically installed on base stations. In traditional VANETs, RSUs collect and disseminate road condition information. Additionally, RSUs can gather vehicle-related information, including speed and location, from OBUs via the DSRC communication

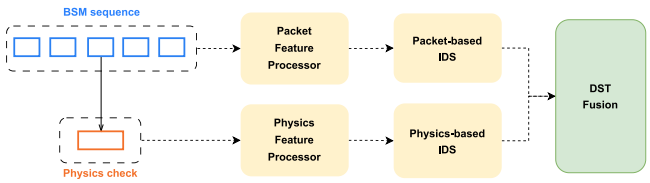


Fig. 2. Block diagram of VAN-IDS.

protocol. During this information collection, RSUs act as data exchange centers, akin to switches in VANETs. Within VAN-FED-IDS, upon receiving vehicle status from OBUs, RSUs are capable of detecting intrusions through VAN-IDS. Upon detecting an attack, RSUs can disseminate warnings to OBUs to enhance vehicle protection. MECs serve as core computational units. As intelligent applications in VANETs increasingly demand computing power, RSUs, with their limited computational resources, fail to meet these requirements. Drawing upon VANET designs that incorporate edge computing (Zhou et al., 2020), MECs are integrated with RSUs to augment their computational capabilities. In VAN-FED-IDS, MECs process the large volumes of data collected by RSUs and train machine learning models. By combining RSUs and MECs, models can be uploaded to the cloud to facilitate federated learning. Furthermore, RSUs equipped with MECs no longer need to upload raw vehicle-related data to the cloud or other RSUs; only the machine learning models produced by MECs are required. This significantly reduces communication costs and enhances system efficiency.

3.2. DST-based hybrid VANET IDS

3.2.1. Data flow of VAN-IDS

VAN-IDS is comprised of three integral modules: data collection and pre-processing, machine learning models for intrusion detection, and multi-tiered DST fusion.

VAN-IDS gathers Basic Safety Messages (BSMs) transmitted by vehicles. The BSM, a standard V2R communication format, encapsulates data such as speed, location, heading, among others. This vehicle-related information is parsed into raw features. Packet-based and physics-based features are subsequently extracted, representing the network flow pattern in V2R communications from diverse angles. Two distinct machine learning models are engineered to classify the messages based on their respective extracted features. DST is employed to amalgamate predictions for a single packet from the two models and to consolidate predictions for a single vehicle from various RSUs, culminating in a definitive outcome (see Fig. 2).

3.2.2. Data collection and pre-processing

The data collection and pre-processing module receive raw Basic Safety Messages (BSMs) and extracts features indicative of intrusions within Vehicle Ad-hoc Networks (VANETs). Feature extraction is a crucial component of IDS, with the feature quality directly influencing the detection outcomes. The VAN-IDS integrates two distinct feature types: packet-based and physics-based. Packet-based features analyze the sequential relationship of position, velocity, acceleration, and other data within the original BSMs to differentiate abnormal from normal traffic patterns. Sequences deviating from established norms of behavior are classified as intrusions. Physics-based features augment the original data with physical information, such as the proximity of obstacles to the vehicle, and assess whether the positional data in the message coincides with these obstacles. Additionally, the Kalman filter is employed to predict forthcoming vehicular parameters such as speed, acceleration, angle, and position from a physical perspective. A significant deviation between the observed and predicted values signals an intrusion.

The original BSMs, preserved as packet-based features, comprise 18 attributes, including GPS position, position confidence, speed, speed confidence, acceleration, acceleration confidence, and heading, along with their confidence values on both the x and y axes. Owing to the disparate physical dimensions and value ranges of each feature, standardization is essential prior to inputting them into the neural network to ensure rapid convergence. The standardized value of the jth feature x^j is x^j_{norm} in Eq. (1).

$$x^j_{norm} = \frac{x^j - \mu_j}{\sigma_j} \tag{1}$$

μ_j and σ_j represent the mean and standard deviation, respectively, of the feature across the entire dataset. These statistics are derived from substantial data collection to ensure that they accurately reflect the actual distribution of traffic in VANETs. Compared to maximum-minimum normalization, standardization demonstrates greater resilience to outliers and improved stability. Packet feature extraction proceeds as follows: Raw BSMs are segregated by the message sender ID, with the standardized features of the ith message sent by a vehicle being defined as the vector b_i. The physical quantities in a given message relate to the preceding and succeeding k and the next k messages, necessitating the processing of a total of $2k + 1$ messages as the packet-based feature for a single message, denoted by $b^t_i = \{b_{i-k}, \ldots, b_{i-1}, b_i, b_{i+1}, \ldots, b_{i+k}\}$. The dimension of b^t_i is $(2k + 1) \times 18$.

The extraction of physics-based features in this study is informed by the characteristics identified by Kamel et al. (2019b), which delineate the consistency between messages and physical laws. A quintessential example is the Intersection Check, which assesses whether the positions of two vehicles overlap. Should the messages indicate a collision between two vehicles without an actual accident, it suggests fabrication. To transform discrete values into concrete ones, vehicles are assigned a confidence range. Length, width, and angle measurements are employed to quantify the degree of intersection. Similar methods of consistency checks and quantification are applied to generate additional features. In examining the consistency among various physical quantities, such as speed and position, as well as speed and acceleration, Kalman Filtering is utilized to predict vehicle status for the subsequent moment and to compare it with the actual values.

In summary, 18 features identified by Kamel et al. (2019b) will be utilized in the VAN-IDS as physics-based features. Within VAN-IDS, each vehicle executes a Kalman filter algorithm concerning other vehicles, thereby making predictions for each received message to extract features. The physics-based feature of the ith message is denoted as w^t_i, with a dimension of 1×18.

3.2.3. ML-based intrusion detection

The machine learning-based intrusion detection module is designed to classify VANET traffic, utilizing a bidirectional LSTM model for packet feature learning and a LightGBM model for physical feature learning. The Bi-LSTM model is well-suited for analyzing time series data. It is used to model dependencies between a message and its antecedent and subsequent counterparts. A decision tree-based model is utilized to infer from physical features, and meticulously crafted feature engineering. Considering both classification efficacy and computational complexity, the LightGBM model was chosen. Similar to DST-IDS (Qiu et al., 2022), the machine learning models within VAN-IDS are modular and can be substituted with alternative algorithms to meet specific deployment requirements. The inputs to this module are the outputs from the data collection and processing module, namely the packet-based feature b^t_i and the physical feature w^t_i. The outputs from these models constitute the predictions made on the raw data by the packet-based and physical-based intrusion detection algorithms, respectively. VAN-IDS executes multi-class classification, assigning labels from 0 to K, where 0 denotes normal traffic and K corresponds to the total number of detectable attack types.

The packet-based IDS utilizes the Bi-LSTM architecture. The LSTM model (Hochreiter and Schmidhuber, 1997), a variant of the recurrent neural network (RNN), tackles the long-term dependency challenges of

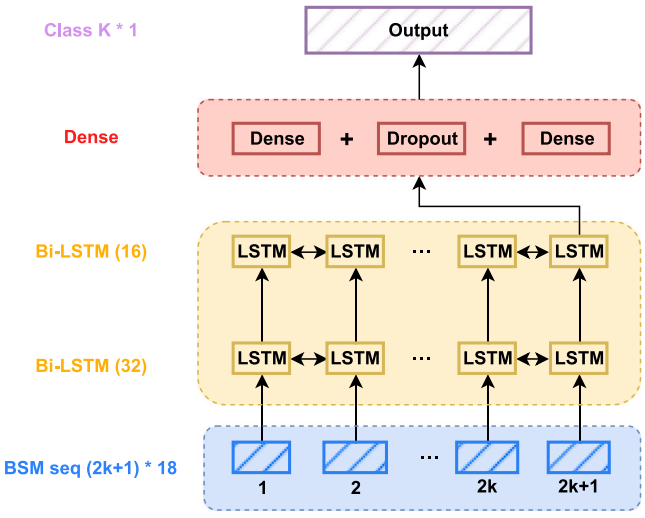


Fig. 3. Structure of Bi-LSTM in VAN-IDS.

RNNs using its gating mechanism and is widely used for processing time series data. The Bi-LSTM concurrently models both forward and backward sequence data, aligns with the requirements of intrusion detection by facilitating the identification of intrusions in temporal traffic contexts. In our packet-based IDS, the Bi-LSTM discovers intrusion patterns from historical and future traffic. The architecture of the Bi-LSTM employed in VAN-IDS is depicted in Fig. 3. A two-layer bidirectional LSTM layer is utilized for the extraction of sequential features. The intermediate layer has an output dimensionality of 32. Subsequently, the sequential features are aggregated via a fully connected layer, with a reduction in dimensionality from 32 to 16, utilizing a ReLU activation function. A dropout layer is incorporated to mitigate the risk of overfitting. Ultimately, a softmax activation function is applied to execute the classification and yield the final prediction p_b.

Physics-based IDS leverages LightGBM (Ke et al., 2017), which is recognized as an efficient advancement of XGBoost (Chen and Guestrin, 2016). LightGBM introduces several enhancements over XGBoost, which serve to further refine classification accuracy and operational efficiency. Noteworthy among these improvements is the implementation of a histogram-based decision tree approach, which significantly optimizes data processing efficiency. Additionally, LightGBM innovates with gradient-based one-side sampling, a technique that focuses on retaining data instances with larger gradients rather than examining all features. These strategic advancements contribute to LightGBM's status as the preferred classification model. LightGBM can effectively and efficiently discover relationships between the physics features generated through complex feature engineering techniques and the intrusion attacks. The resulting prediction from the model is denoted as p_w.

3.2.4. Multi-level DST fusion

Dempster–Shafer Theory (DST) (Shafer, 1976) is a mathematical framework used to integrate evidence from multiple sources, proficient at quantifying inherent uncertainty in evidence. This approach diverges from simply aggregating confidence for each hypothesis (normal or attack) provided by individual models. Uncertainty stems from models never being trained with 100% accuracy, leading to potential wrong predictions. We use the false positive and false negative rates of machine learning models to quantify this uncertainty. False positives and false negatives indicate that machine learning-based intrusion detection systems may fail to distinguish all attacks from normal activities. DST-based fusion enables correcting false predictions made by lower-quality IDS by incorporating more accurate ones. This capability enhances result stability, as demonstrated in our experiment.

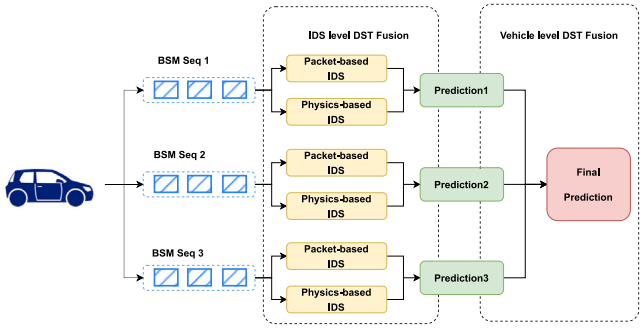


Fig. 4. Multi-level DST fusion.

The DST-IDS (Qiu et al., 2022) study demonstrated several advantages of DST fusion. Firstly, by using D-S fusion rules to integrate various features, it achieves prediction performance comparable to that of complex neural network methods but with lower computational complexity. Secondly, DST provides more stable prediction results across diverse distribution settings. Even when the testing set differs from the training set in distribution, DST-based IDS still predicts effectively. These advantages are in line with our aim of designing a fast and practical VANET IDS.

As implemented in VAN-IDS, a multi-level DST-fusion mechanism is provided, as illustrated in Fig. 4. For each BSM transmitted by the vehicles, DST synthesizes the insights from both the packet-based IDS and the physics-based IDS to produce a consolidated prediction for the BSM in question. Additionally, when a single vehicle sends multiple BSMs, DST integrates the individual predictions made on these BSMs to formulate a final judgment regarding the vehicle's behavior.

IDS level DST fusion is similar to DST-IDS (Qiu et al., 2022), where packet-based IDS (Bi-LSTM) and physics-based IDS (LightGBM) are regarded as two pieces of evidence. Symbolically we define the hypothesis space in DST as $X = \{S_0, S_1, S_2, \ldots, S_K, \Omega\}$. S_0 refers to the normal traffic, while S_k represents the kth attack type. Ω refers to the uncertainty on attack types, that is, $\Omega = \{S_0, S_1, \ldots, S_K\}$. The Basic Probability Assignment (BPA) function which projects the output of machine learning models (probability of each attack) to the probability of each hypothesis in DST can be written as Eq. (2), where p_k refers to the model output and $m(\Omega) = c$ symbolizes the false positive rate of machine learning models. After the training is completed, the value of c is confirmed.

$$m(A) = \begin{cases} (1-c)p_k, & \text{if } A = S_k, k \in \{0, 1, \ldots, K\} \\ c, & \text{if } A = \Omega \end{cases} \tag{2}$$

$$D(S_k) = \frac{1}{1-\Phi}(p_{b'_k} p_{w'_k} + p_{b'_k} p_{w'_\Omega} + p_{b'_\Omega} p_{w'_k} + p_{b'_\Omega} p_{w'_\Omega}),$$
$$\text{where } \Phi = \sum_{\substack{k_1 \neq k_2 \\ k_1, k_2 \neq \Omega}} p_{b'_{k_1}} p_{w'_{k_2}} \tag{3}$$

The output of BPA is collected in vector $p_{b'_i} = [p_{b'_0}, p_{b'_1}, \ldots, p_{b'_K}, p_{b'_\Omega}]$ or $p_{w'_{in}} = [p_{w'_0}, p_{w'_1}, \ldots, p_{p'_K}, p_{w'_\Omega}]$ with $\{p_{b'_k} \text{ or } p_{w'_k}\} = m(S_k)$ and $\{p_{b'_\Omega} \text{ or } p_{w'_\Omega}\} = m(\Omega)$. The output of DST fusion can be calculated by Eq. (3). The final prediction result of IDS level fusion is denoted by $D = \arg\max_{k \in [0,K]} D(S_k)$.

The application of vehicle-level DST fusion can significantly enhance the precision of detection. A single vehicle might traverse multiple zones that are under the surveillance of different RSUs. These RSUs collect and scrutinize BSMs, thereby enabling collaborative efforts to assess the behavior of the vehicle across various regions. BSM emitted from the same vehicle is independently evaluated by the VAN-IDS systems installed on each RSU, yielding a detection outcome D_{ij}, where this represents the result for the ith BSM as analyzed by the jth RSU.

Subsequently, these individual detection results are fed back into the DST framework. The uncertainty measure associated with the evidence, denoted as $m(\Omega)$ is informed by the historical false rate observed in the detection performance of each RSU. After this data undergoes DST fusion, a conclusive prediction is derived that categorizes the vehicle as either malicious or non-malicious.

By incorporating the reliability of each RSU into the DST fusion process, the system capitalizes on the collective intelligence garnered from multiple sources to arrive at a more accurate and trustworthy assessment of vehicle behavior. This holistic approach to evidence synthesis is pivotal in environments where the integrity and safety of vehicular communication networks are of paramount importance.

3.3. Run-time design of VAN-FED-IDS

Once the intrusion detection algorithms are deployed on RSUs, it becomes crucial to address the run-time operations, which include model training, updating, and inference, to ensure the system's practicality and efficiency in a real-world setting. To balance the needs of model training efficiency and data privacy, federated learning is employed as a strategy to train models without the need to centralize data, thereby preserving privacy.

In the context of VAN-FED-IDS, FedAVG (McMahan et al., 2017) is utilized to aggregate updates from neural network models. For tree-based models, which are particularly effective for certain types of structured data, VAN-FED-IDS employs FedTree (Li et al., 2020), a method designed to aggregate tree-based models in a federated learning setting.

3.3.1. Model training

An initial model is trained when the system is deployed in real VANET, provided that each RSU has the intrusion history of VANET or a partial dataset. Given N RSUs, $R_k(k = 1, 2, \dots, N)$ refers to the local dataset owned by RSU numbered k. Features and labels can be represented as (x_i, y_i). RSU uses the local dataset to train models and uploads the parameters to the cloud for model aggregation. Suppose $f(x;\omega)$ is a loss function and all RSUs have known the model structure. In our current implementation of VAN-FED-IDS, Bi-LSTM and cross entropy loss function is used. A instantiated definition of $f(x;\omega)$ is showed in Eq. (4). (Cross-Entropy loss function), where $p(x_{ic};\omega)$ refers to the probability of sample i belonging to class c by model prediction with parameters ω and y_{ic} is the ground-truth label. It is important to note that $f(x;\omega)$ represents an abstract loss function, which can be instantiated with various widely-used loss functions in neural network model training. Upon the deployment of VAN-FED-IDS, both the neural network model and the loss function can be replaced with alternative methods, tailored to the deployment environment, available computing resources, and other relevant factors.

$$f(x_i;\omega) = -\frac{1}{N}\sum_{i}\sum_{c=1}^{M} y_{ic} log(p(x_{ic};\omega)) \tag{4}$$

The steps to jointly train a neural network are:

Step 1 RSU splits the dataset to multiple batches $\{b_1, b_2, \dots, b_m\}$ with batch size B. The average loss in each epoch is given in Eq. (5).

$$F(b_j;\omega) = \frac{1}{B}\sum_{x_i \in b_j} f(x_i;\omega) \tag{5}$$

Stochastic Gradient Descent is used to locally update models as Eq. (6), where η refers to the learning rate.

$$\omega_{t+1} = \omega_t - \eta \nabla F(b_j;\omega) \tag{6}$$

After local models are trained E epochs repeatedly, the RSUs upload parameters to the cloud.

Step 2 Local models are weighted according to the proportion of the dataset size of the upload node to the total data. The cloud sums the weighted local model parameters as Eq. (7) to get the global model. R_k refers to the size of samples in local dataset owned by RSU numbered k, and R is the total size of N RSUs.

$$\omega_{t+1} = \sum_{k=1}^{N} \frac{|R_k|}{|R|}\omega_t^k \tag{7}$$

Step 3 The cloud distributes the aggregated model parameters to each RSU. RSU continues to train local models on the basis of new models. Repeat **Step1-3** C rounds according to the requirements of the cloud.

The steps to collaboratively train a tree model based on FedTree are as follows. The cloud initiates a request to obtain the hash value of each RSU's data through the Locality Sensitive Hashing algorithm (LSH), and calculates the similarity between the data of each node to produce a similarity matrix S_{ij}^m, which indicates the instance number in party P_j that is similar to the instance x_i in party P_m. Then the training begins.

Step 1 Assuming that the target RSU is P_m, each RSU P_i first gets the gradient matrix. More specifically, for each instance x_k^i, P_i calculates its first- and second-order gradients of the loss function g_k^i and h_k^i, and then searches for the most similar data $s = S_{km}^i$. P_i updates the gradient matrix G_{m*}^i, H_{m*}^i according to Eq. (8), and uploads the gradient matrix to the cloud.

$$G_{ms}^i = G_{ms}^i + g_k^i, H_{ms}^i = H_{ms}^i + h_k^i \tag{8}$$

Step 2 The cloud delivers the matrix G_{m*}^i to the RSU P_m.

Step 3 After RSUs receive the matrices required, they can calculate the loss and update trees as Eq. (9), where the weight matrix $W_{mk}^n = \{a|S_{am}^n = k\}$ indicates the number of instance in P_n that is similar to x_k^m in P_m. $\Omega(f)$ is the regularization function, e.g. l2 regularization $\Omega(f_\omega) = \sum \|\omega^2\|$. Repeat **Step1-3** C rounds according to the requirements of the cloud.

$$\begin{aligned} L_\omega &= \sum_{x_k^i \in R_k}\left[G_{mk} f(x_k^i;\omega) + \frac{1}{2}H_{mk} f^2(x_k^i;\omega)\right] + \Omega(f) \\ G_{mk} &= \sum_n \sum_{i \in W_{mk}^n} g_i^n, H_{mk} = \sum_n \sum_{i \in W_{mk}^n} h_i^n \end{aligned} \tag{9}$$

3.3.2. Model inference

Model inference refers to detecting malicious behaviors in real time. The cloud only carries out multi-level DST fusion while RSUs collect traffic from VANET and feed the machine learning model for real-time prediction. Suppose that the prediction result by packet-based IDS for BSM j from vehicle i is vector p_b^{ij} and p_w^{ij} by physics-based IDS, and the inference steps are:

Step 1 RSUs send local predictions p_b^{ij} and p_w^{ij} to the cloud.

Step 2 The cloud combines the prediction results through DST to obtain a prediction on a BSM $f_{ij} = DST(p_b^{ij}, p_w^{ij})$ for IDS level fusion, and a further vehicle level fusion result $F_i = DST(f_{ij_1}, f_{ij_2})$.

Step 3 The RSUs receive the results from the cloud and take corresponding measures against the vehicles judged to be attacked.

3.3.3. Model updating

After operating for some time, the system's models may degrade in performance in response to unknown attacks or the emergence of novel strategies for known attack types, necessitating retraining with new data. Additionally, models require updates when VANET's topology or standards change, altering the distribution of traffic data. VAN-FED-IDS offers a flexible and diverse update methodology. Consider a typical scenario: Suppose an RSU is proven incapable of recognizing an unknown attack type and requires updating through the following steps: First, the RSU sends a request to the cloud for model updates, indicating the occurrence of new attacks. Second, based on the current network status, the cloud selects certain nodes for federated learning and disseminates the updated models. Third, the selected RSUs or those with datasets of the unknown attack train models locally and upload them to the cloud. Finally, the cloud aggregates these models

and distributes them to all RSUs. In addition to selecting a subset of RSUs for training based on current network conditions and computing resources, it is unnecessary to retrain all models with each update. For instance, should the lightGBM model within VAN-IDS err in prediction, only the tree model requires updating, while the Bi-LSTM remains untouched. This approach further accelerates system updates. Moreover, even when updating all models is necessary, federated learning proves faster than centralized learning.

3.3.4. Role of OBU, RSU and MEC

During the VAN-IDS intrusion detection process, RSUs collect vehicle-related data, such as speed and location, from OBUs through the DSRC communication protocol. The MEC installed on RSUs processes a significant volume of data, inputs them into models, and merges the predictions using DST rules. In case of multi-level DST fusion, RSUs will additionally transmit the predictions to the cloud for a broader and more reliable detection outcome. Upon detecting any attack, RSUs will send alerts to the OBUs prompting vehicles to respond.

In the federated learning process of VAN-FED-IDS, RSUs collect data from OBUs in different vehicles and prepare for model training. The MECs equipped on RSUs act as the actual computation clients that carry out model training. RSUs will upload the updated model parameters to the cloud. The cloud aggregates models from all RSUs and offloads a global model back to the RSUs for MECs to continue training local models.

4. Experimental results

This section details a series of experiments designed to validate the accuracy of VAN-IDS and the efficacy of federated learning as applied in VAN-FED-IDS. Initially, the dataset and experimental environment are described in depth. Subsequently, ablation studies and comparative analyses are conducted to establish VAN-IDS's superiority over contemporary IDSs within VANET. Lastly, VAN-FED-IDS is deployed in both simulated and distributed modes to assess its training efficiency and communication overhead.

4.1. Experiment settings and datasets

The dataset was obtained from Veins, a VANET simulation framework that provides vehicle communication models such as IEEE 802.11p and IEEE 1609.4 DSRC/WAVE. Veins is based on two established simulators: OMNeT++, an event-based network simulator, and SUMO, a road traffic simulator. It offers a realistic and dependable VANET simulation environment, widely utilized by universities, governmental entities, and independent research institutes. F2MD (Kamel et al., 2019b) extends the functionality of Veins, focusing on simulating VANET attacks and evaluating IDS performance. A key feature of F2MD is its modularity and extensibility, facilitating the development of additional functions by users. F2MD's capabilities include simulating attacks, assessing detection algorithms, implementing new attack strategies, and gathering attack datasets. In this study, data collection and feature extraction were exclusively performed using F2MD. Utilizing F2MD allows us to concentrate on designing intrusion detection algorithms and comparing models. Furthermore, F2MD has been employed in previous studies (Kamel et al., 2019b,a; Mahmoudi et al., 2020) to assess their IDS, facilitating direct comparisons between our approach and theirs. The simulation utilized a map covering the area around the University of Saclay in France, spanning 2.68 square kilometers, with a traffic flow density of 7.83 vehicles per square kilometer. During network simulation, each vehicle broadcasts a Basic Safety Message (BSM) every second, which includes details like position, speed, heading, noise, time, and the sender's anonymized identifier.

This experiment concentrates on detecting two prevalent malicious activities in VANET: Denial-of-Service (DoS), aimed at incapacitating the server or system, and Sybil attacks, which involve forging identities to disseminate fraudulent messages. These types of attacks are not only easy to execute but also constitute the primary focus of detection in prior research. In particular, we implement four attack variants: random DoS, which involves sending messages at random to disrupt the system; disruptive DoS, characterized by message flooding; data replay Sybil attack, where an identity is forged to replay data; and grid Sybil attack, which entails forging an identity to send messages that simulate a traffic jam. These represent two tactics each for DoS and Sybil attacks. The labeling convention used is '1' for DoS attacks, '2' for Sybil attacks, and '0' for normal traffic. Each simulation runs for 600 s, yielding a dataset of approximately 70,000 packets per simulation, with malicious packets constituting between 10% and 40% of the total. The dataset's statistics are presented in Table 1.

Table 1
Statistics of data in VANET dataset.

Attack type	Normal	Malicious	Total	Malicious proportion
Random DoS	50512	18958	69470	22.88%
Disruptive DoS	50512	23692	74204	31.93%
Total DoS	101024	42650	143674	29.69%
Data replay sybil	50586	25955	76541	33.91%
Grid sybil	50528	7214	57742	12.49%
Total sybil	101114	33169	134283	24.70%
Total	202138	75819	277957	27.28%

The experiments with VAN-FED-IDS aim to determine whether a federated learning-based VAN-IDS can reduce or maintain training duration, protect data privacy, and preserve accuracy. Training for the federated learning (FL) model is bifurcated into two segments: neural network FL and tree model FL. After analyzing several existing federated learning frameworks, including FATE (Liu et al., 2021a), TensorFlow Federated, Flower (Beutel et al., 2020), and Fedtree (Li et al., 2022), we have cataloged their key features in Table 2. Flower was selected for Bi-LSTM federated model training due to its compatibility with all deep learning frameworks and its ability to support distributed mode with minimal hardware requirements. It efficiently facilitates communication between training clients and provides a flexible interface for existing deep learning frameworks such as Tensorflow or Pytorch. In federated learning of neural network models, after all clients finish training, only model parameters should be exchanged and aggregated across clients to obtain a global model. Flower provides state-of-the-art aggregation methods (e.g. FedAVG) and also interfaces for user-defined aggregation algorithms. As Flower does not support tree models, we use FedTree, an open-source federated learning project specialized in tree-based federated learning. In tree-based federated learning, not only model parameters like gradients, but also data sample information like similarity matrices should be exchanged among clients. FedTree is specifically designed for highly efficient, effective, and secure federated training of gradient-boosting decision trees. We use FedTree for lightGBM federated training.

Experiments related to VAN-IDS and Flower are conducted on a server running Ubuntu 18.04 64-bit OS, which is outfitted with an Intel Xeon E5-2680 14-core CPU and 200 GB of RAM. Experiments involving FedTree and evaluations of FL in distributed mode are carried out on Ubuntu 18.04 virtual machines, each with a 2-core CPU and 8 GB of RAM, hosted on a physical machine that boasts an Intel i5-7400 8-core CPU and 32 GB of RAM.

Evaluation metrics are accuracy, recall, precision and F1-Score, defined as $Acc = \frac{TP+TN}{TP+TN+FN+FP}$, $Rec = \frac{TP}{TP+FN}$, $Pre = \frac{TP}{TP+FP}$, $F1 = \frac{2 \cdot Rec \cdot Pre}{Rec+Pre}$, where TP be the number of S_i instances predicted as S_i, TN be the number of not S_i instances predicted not as S_i, FN be the number of S_i instances predicted not as S_i, and FP be the number of not S_i instances predicted as S_i.

Table 2
Comparison of the federated learning frameworks.

Framework	Applicable scenarios	Model	Deployment	Pros and cons
FATE	Industrial/Academic	NN/Tree	Simulated/Distributed	Complex architecture and high hardware requirements
TFF	Academic	NN	Simulated	Support Keras and Tensorflow
Flower	Academic	NN	Simulated/Distributed	Support all deep learning frameworks
Fedtree	Academic	Tree	Simulated/Distributed	Optimized boosting tree federated learning

Table 3
Results of Bi-GRU and lightGBM model training.

Model	Dataset	Accuracy	Precision	Recall	F1-Score
Bi-GRU	Training	$98.21\% \pm 0.20\%$	$96.33\% \pm 0.29\%$	$97.99\% \pm 0.20\%$	0.9710 ± 0.0033
Bi-GRU	Validation	$98.16\% \pm 0.35\%$	$96.18\% \pm 0.50\%$	$97.97\% \pm 0.42\%$	0.9705 ± 0.0053
lightGBM	Training	$98.50\% \pm 0.23\%$	$96.39\% \pm 0.34\%$	$98.80\% \pm 0.11\%$	0.9750 ± 0.0039
lightGBM	Validation	$98.48\% \pm 0.29\%$	$96.35\% \pm 0.53\%$	$98.70\% \pm 0.25\%$	0.9748 ± 0.0034

4.2. VAN-IDS evaluation

4.2.1. Model training

This study employs Bi-LSTM and LightGBM as machine learning (ML) models within the VAN-IDS framework. 80% of the packet-based and physics-based features are used for training, and the remaining 20% for testing (validation) in a stratified K-Fold cross-validation process.

During Bi-LSTM training, the Stochastic Gradient Descent (SGD) optimizer is employed, and hyperparameters such as batch size, dropout rate, and hidden layer size are carefully tuned, with the objective of optimizing prediction accuracy on the validation set. In the experiments conducted, the GRU model, which was tested as a replacement for LSTM, demonstrated improved accuracy. The GRU is a simplified variant of LSTM that utilizes only the forget and update gates, resulting in reduced computational complexity compared to LSTM. Post K-fold validation, optimal outcomes were obtained with a batch size of 50, a dropout rate of 0.1, and a hidden layer size of 16.

Fig. 5 displays the training loss and accuracy, which stabilize after 20 epochs, with an accuracy rate of 98%. In Fig. 5, the blue line denotes the training set, while the red line corresponds to the validation set; their convergence after 20 epochs suggests minimal overfitting.

The main parameters in lightGBM include the maximum depth of the tree max_depth, the number of leaves of a single tree num_leaves, the proportion of sampled features $feature\_fraction$, the proportion of sampled data $bagging\_fraction$, and regularization parameters λ and α. These parameters directly affect the training result and control over-fitting. After K-fold validation, the hyper-parameters are as follows: $max\_depth = 10$, $num\_leaves = 31$, $feature\_fraction = 0.7$, $bagging\_fraction = 0.7$, $\lambda = 2$, $\alpha = 0.1$. The learning rate is set to 0.1, the best result is obtained after 300 epochs.

The final training results of the two models are shown in Table 3, where the recall, accuracy, and F1-scores are averaged on four types of attacks. The number in each cell refers to the average metric of four attacks after 5 times shuffle split validation. Both models can achieve more than 96% recall and $97\% - 98\%$ accuracy. The standard deviation of all metrics is less than 0.6%, while the difference between results on the training set and the validation one is less than 0.3%. It indicates that the model is not over-fitting.

Following the training of the LightGBM model, it is possible to analyze feature importance to determine which features contribute most significantly to classification. The frequency with which a feature is used as a splitting attribute across all trees indicates its importance. Fig. 6 illustrates that position plausibility is the most impactful feature, signifying whether a vehicle's position is within a non-drivable area. During DoS or Sybil attacks, attackers disseminate falsified messages. Provided the message contains an incorrect location, it may be classified as an attack. Other significant features include position speed consistency, intersection check, and Kalman position and speed/acceleration consistency, which assess the plausibility of the position information and its correlation with other physical metrics.

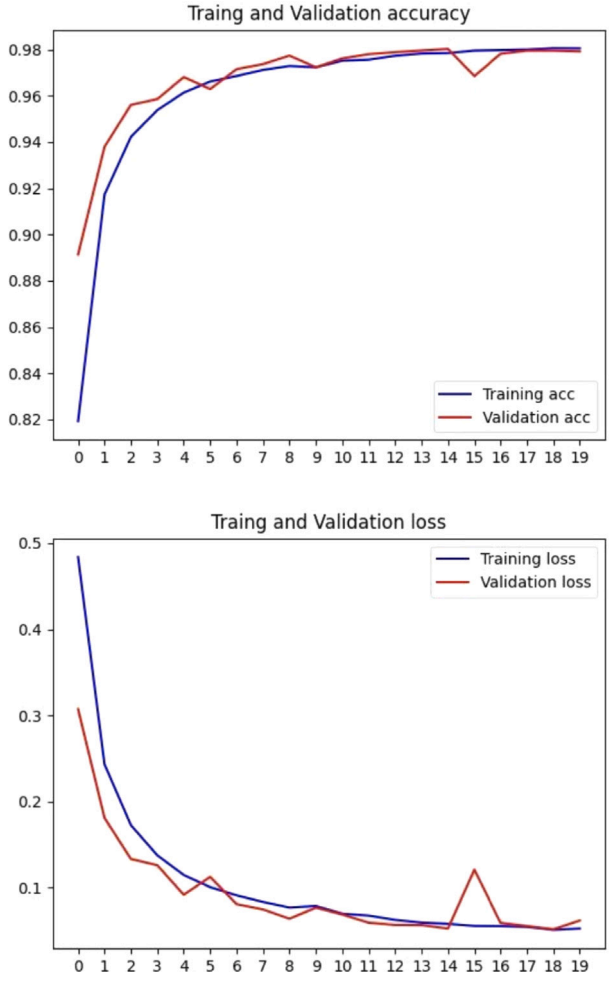


Fig. 5. Training loss and accuracy of Bi-GRU.

Consequently, the following conclusions can be drawn: The majority of the extracted features exhibit high importance and the final model demonstrates high classification accuracy, affirming the efficacy of feature engineering. Analysis of location plausibility aids in distinguishing between normal traffic and DoS or Sybil attack traffic. Thus, protecting location data remains a focal point for privacy, consistent with the perspective presented in the literature (Chen et al., 2021).

4.2.2. Ablation and comparison experiments

Prediction outcomes from the packet-based and physics-based IDS on the test set are fed into the IDS-level DST fusion module to derive

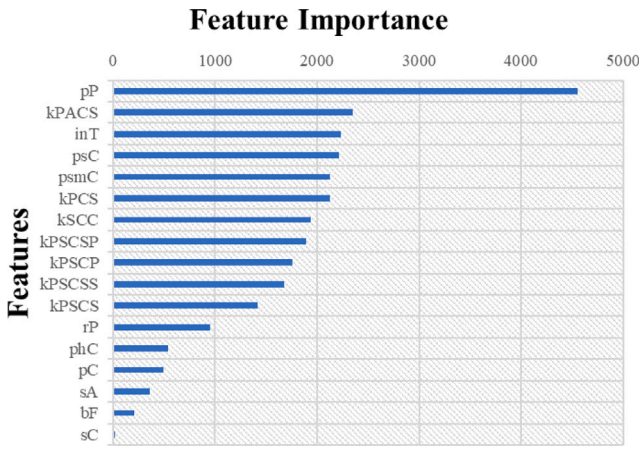


Fig. 6. Feature importance of lightGBM.

Table 4
Results of VAN-IDS ablation experiment.

Model	Attack	Accuracy	Recall	Precision	F1-Score
lightGBM	DoS	99.81%	98.56%	99.93%	0.9924
	Sybil	98.49%	90.77%	97.92%	0.9421
Bi-GRU	DoS	99.33%	98.39%	98.95%	0.9867
	Sybil	98.20%	90.18%	98.15%	0.9399
VAN-IDS	**DoS**	**99.90%**	**99.80%**	**99.88%**	**0.9984**
	Sybil	**99.55%**	**97.29%**	**99.65%**	**0.9845**

the final detection outcomes. Based on the average accuracy rates of Bi-LSTM and LightGBM, the uncertainty measure $m(\Omega)$ in DST is 0.02. Performance metrics pre- and post-DST fusion are presented in Table 4, where VAN-IDS's performance is juxtaposed with that of two standalone machine learning models. The table demonstrates that all performance indicators improve by over 1% after DST fusion, and specifically, the recall rate for Sybil attacks increases by 7%. Predictions from both packet-based and physics-based IDS can be amalgamated to achieve a more robust output, akin to the principles of ensemble learning. Performance metrics for DoS attacks exceed 98%. This indicates that DoS attacks are readily detectable due to the high volume of packets and the randomness of message content. The strategy behind Sybil attacks is more sophisticated. For instance, the grid Sybil attack involves sending messages that create the illusion of other vehicles near the target vehicle, based on actual road conditions, thereby simulating traffic congestion. The location information within these messages appears legitimate and is challenging for the model to distinguish, resulting in a recall rate for Sybil attacks by packet-based or physics-based IDS of approximately 90%, aligning with the findings from literature (Kamel et al., 2019b,a) that Sybil attacks are inherently more challenging to detect. Following DST fusion, the recall rate for Sybil attacks can exceed 97%. The detection speed can be evaluated on the test set. Detection speed is assessable using the test set. The corresponding detection time for a 600-second simulation in a VANET is 50 s, thus enabling real-time detection capabilities for VAN-IDS, meaning that detection outcomes for a given period can be generated before the subsequent period commences.

We selected several representative models from prior VANET IDS research for comparison with VAN-IDS, including prominent algorithms like DNN and XGBoost. Additionally, the hybrid DNN-LSTM model proposed by Mahmoudi et al. (Mahmoudi et al., 2020) demonstrates higher accuracy compared to other models, positioning it as the primary benchmark for comparison. This model processes original BSM data sequentially through a Bi-LSTM network, and feeds 18 features derived from plausibility checks into a DNN. The outputs from both networks are then merged and fed into a fully connected layer to perform classification. While VAN-IDS and the DNN-LSTM share similar feature processing methods, a key distinction is VAN-IDS's utilization of DST to fuse IDS results, as opposed to the joint training approach in neural networks. This strategy potentially mitigates overfitting issues often associated with complex neural network models. The comparative results, as outlined in Table 5, concentrate on average accuracy and F1-score. LightGBM demonstrates superior performance compared to the original XGBoost, while the Bi-GRU component of VAN-IDS surpasses both DNN and LSTM in terms of effectiveness. This validates the suitability of choosing Bi-GRU and LightGBM. Although the DNN-LSTM model outperforms other neural network-based models and is competitive with LightGBM, it still falls short of the performance achieved by VAN-IDS.

Table 5
Results of VAN-IDS comparison experiment.

Model	Macro-Acc	Macro-F1
Xgboost	96.20%	0.9302
lightGBM	98.48%	0.9748
DNN	94.34%	0.8937
LSTM	98.03%	0.9698
Bi-GRU	98.16%	0.9705
DNN-LSTM	98.40%	0.9730
VAN-IDS	**99.57%**	**0.9934**

At the IDS level, the DST fusion outputs predictions for each message. Subsequent vehicle-level fusion will yield a coarse-grained prediction for each vehicle. Following vehicle-level fusion, VAN-IDS can detect malicious vehicles with nearly 100% accuracy. Summarizing the above experiments, VAN-IDS has demonstrated enhanced detection performance attributable to its sub-models, with metrics surpassing those of existing works; moreover, the detection time satisfies the requirements for real-time detection.

4.3. Federated learning simulation results

Compared to VAN-IDS, VAN-FED-IDS substitutes centralized learning with federated learning. The introduction of FL into IDS accelerates the model training and updating processes, while concurrently protecting data privacy. We begin with a simulation experiment, where FL training clients are simulated using threads, and multi-node collaborative training is emulated with multi-thread training on a single physical machine. In simulation experiments, the number of clients can be unlimited, so it is easy to discover the relationship between the number of clients and training speed. However, communication costs cannot be analyzed in simulation mode, and the training speed may be restricted by the CPU and memory of the physical machine, especially when the number of clients exceeds 50.

4.3.1. Bi-GRU training

The dataset and the Bi-GRU model structure utilized herein are consistent with those employed in our preceding experiments. The dataset comprises 277,957 total samples. The training and testing sets are partitioned in an 8:2 ratio, and the dataset is evenly distributed among the predetermined number of clients. The objective is to document the training duration and pertinent parameters upon reaching an accuracy of 0.98 on the training set and when the discrepancy between the training and testing sets is less than 1%. During training, the SGD optimizer is employed, and the learning rate is set at 0.05. The results are presented in Table 6, wherein the Baseline model denotes the centralized learning approach. Utilizing the optimal parameter combination (M1, M3), federated learning can diminish the training time by nearly 40% in comparison to centralized learning. Variations in parameters influence the training velocity. Contrary to expectations, certain observations diverge from the conclusions drawn in FedAVG (Li et al., 2020). In FedAVG (Li et al., 2020), the local

Table 6
Federated learning results of Bi-GRU in VAN-IDS.

Model	Training time	Clients	Local epoch	Local batch size	Communication round
Baseline	1610	1	20	50	1
M1	**1100**	**3**	**2**	**50**	**20**
M2	1432	3	5	50	18
M3	**983**	**10**	**5**	**30**	**17**
M4	1500	30	10	30	20
M5	1542	35	10	30	22

Table 7
Federated learning results of XGBoost in VAN-IDS.

Model	Learning rate	Tree	λ	γ	Clients	Training time
Baseline	0.3	200	2	0.1	–	602
M1	0.1	50	2	0.1	3	1069
M2	0.3	50	1	1	3	943
M3	**0.5**	**30**	**2**	**0.1**	**5**	**589**
M4	0.3	40	2	0.1	5	812

epoch is described as inversely proportional to training time; however, this was not observed in our experiments (M2 and M3). This suggests that the impact of FL-related parameters on training velocity is predominantly contingent upon the data's distribution and innate characteristics. In conclusion, the model aggregation phase in federated learning resembles regularization, mitigating overfitting and expediting convergence.

4.3.2. LightGBM training

The datasets utilized are identical to those in our prior experiments. VAN-IDS employs lightGBM, which boasts certain engineering optimizations over XGBoost. Given that FedTree exclusively supports XGBoost, we have adapted the baseline model to XGBoost to enable a straightforward comparison. The objective is to document the training duration and associated parameters upon the training set achieving an F1-score of 0.95, and the difference between the training and the testing set is less than 1%. The results are shown in Table 7.

The training duration for federated learning (M3) using the optimal parameter set is comparable to that of centralized learning. This is attributable to the federated learning process of FedTree. In neural network (NN) federated learning, a single participant is required to compute only local data and discern the data distribution of other participants by interacting with the cloud and aggregating model parameters. Given that only model parameters are exchanged and communication overhead is minimal, training duration is substantially decreased. However, in tree-based federated learning (FedTree), clients must compute local data and also perform additional computations to aid the training of other participants, which may not lead to reduced training speed. The efficient implementation of FedTree ensures that privacy protection measures for local data do not impede training speed. Considering the CPU and memory constraints of the virtual machine, the competition for computing resources among multiple processes could result in slower training and restrict the feasibility of experimenting with FL using a larger number of clients (limited to 3–5 in this experiment).

In conclusion, federated learning safeguards the privacy of local training data for each Roadside Unit (RSU) in the Vehicular Ad-hoc Network (VANET) Intrusion Detection System (IDS). Neural network federated learning can significantly enhance training efficiency, while decision tree federated learning can match the centralized approach in training speed. During deployment, the system must identify FL-specific parameters, which can be dynamically adjusted based on the training conditions and deployment contexts.

4.4. System prototype evaluation

4.4.1. Training time and communication cost evaluation

When conducting simulations on a single machine, the training speed can be influenced by the strategy used for scheduling resources across multiple threads. This situation may not accurately reflect the actual deployment environment. In an effort to minimize the discrepancies between simulation and real-world deployment, we employed the cluster modes of Flower and FedTree to construct a prototype system. This prototype consists of three nodes, enabling us to perform a multi-machine federated learning test. For the purposes of our experiment, we utilized virtual machines as nodes, each provisioned with 2 CPU cores and 2 GB of memory. Within this setup, one node was designated to function as the cloud server, while the remaining two nodes represented Roadside Units (RSUs). Our experimental objective was to meticulously document both the communication overhead and the training time required to achieve the predetermined accuracy target.

Initially, the cluster mode of FedTree was evaluated, with outcomes presented in Table 8. Models M1-4 employed FedTree's secure aggregation algorithm, denoted as sa, which is designed to safeguard the privacy of gradients during communication. On the contrary, Model M5 did not utilize this secure aggregation feature. The results indicated by M1-4 demonstrate that the multi-machine environment facilitates parallel training of each local model, leading to a training speed that is approximately 60% faster compared to the single-machine training referenced in Table 7.

It is also observed that increases in both the depth of the decision tree and the number of training iterations contribute to extended training durations. FedTree monitors and reports on communication data throughout the training process. Consequently, the communication overhead escalates with the increment of training rounds and tree complexity. Specifically, the communication cost for the most efficient model, M1, reached 1 GB. This is attributed to the requirement for bidirectional data exchange between participants in each round of local training. Reducing the volume of communication data remains a primary area of research within the domain of federated learning for decision trees.

Opting to forgo the use of secure aggregation algorithms can lead to a reduction of nearly 50% in both training time and communication costs. However, this approach exposes the tree-based federated learning to heightened risks of adversarial attacks, such as membership inference or model poisoning attacks.

Subsequently, the cluster mode of Flower was put to the test, with results chronicled in Table 9. The communication data volume was meticulously recorded using the Linux-based network traffic monitoring tool, VnStat. The tabulated data reveals that the training velocity in cluster mode is broadly on par with that observed in the simulation environment (referenced in Table 6). This performance is significantly quicker than that of the centralized training approach, which underscores the effective utilization of distributed parallel computing in expediting the training process.

The communication data associated with neural network-based federated learning is relatively minimal. The communication overhead is dramatically reduced – nearly by a factor of 100 – when compared to the communication costs incurred during the training of equivalent data using FedTree.

Table 8
Results of FedTree training in cluster mode.

Model	Learning rate	Tree	Max_depth	Privacy Protection	Training time	Communication cost
M1	**0.3**	**50**	**10**	**sa**	**156.907**	**1062.77MB**
M2	0.1	50	10	sa	180.507	1557.48MB
M3	0.3	100	10	sa	220.513	1192.31MB
M4	0.3	50	12	sa	192.943	2055.64MB
M5	**0.3**	**50**	**10**	**none**	**84.8236**	**562.99MB**

Table 9
Results of Flower training in cluster mode.

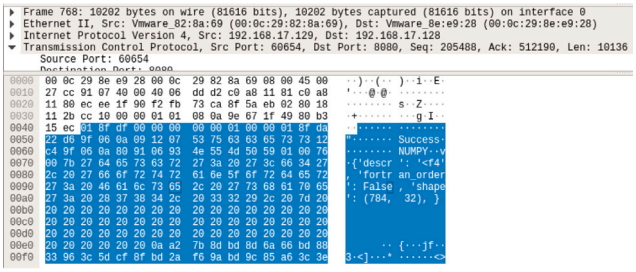
Model	Training time	Local epoch	Local batch size	Communication round	Communication cost
M1	769.41	5	32	3	2.55MB
M2	1152.54	10	32	2	2.33MB
M3	626.38	5	50	3	3.54M
M4	1135.62	10	50	3	3.55M

In summary, federated learning across multiple machines enhances the utilization of parallel training, substantially increasing training speed. The implementation of federated learning for neural networks by Flower incurs lower communication costs. FedTree's performance is constrained by its model aggregation method, resulting in high communication costs, which could pose a bottleneck during real-world deployment.

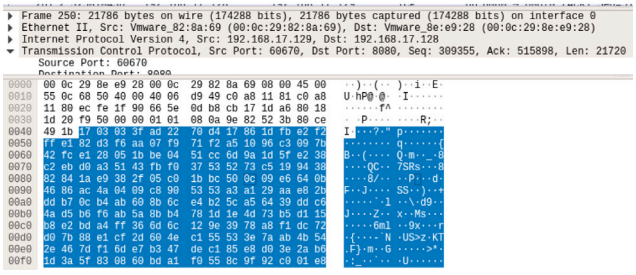
4.4.2. Privacy protection effect

This section empirically demonstrates the privacy protection effect of the system. Key data in VAN-FED-IDS requiring protection encompasses datasets for model training and model parameters shared during federated learning (FL). Firstly, from a theoretical perspective, the datasets are not transmitted over the network, ensuring there is no risk of dataset leakage. The neural network federated learning framework Flower and the tree-based framework FedTree utilize SSL (Secure Socket Layer) encryption and secure aggregation functions, respectively, to safeguard the privacy of model parameters. In the SSL protocol, participants generate and exchange keys during the handshake phase upon establishing a connection with the cloud for encrypting subsequent data communications, ensuring that the neural network's parameters are not transmitted in plaintext over the network. The secure aggregation function preserves the privacy of the tree model by facilitating key exchanges between the participants and the cloud prior to the commencement of training, subsequently using these keys to encrypt the tree model parameters with added noise.

The privacy protection efficacy of Flower is assessed from the perspective of potential attackers. Attackers commonly attempt to intercept and analyze critical data transmitted between participants and the cloud. As the dataset is not transmitted over the network, it remains inaccessible to an attacker. Without encryption, model parameters could potentially be inferred by an attacker. In the experiment, Wireshark is employed to capture packets for analyzing the communication data in distributed federated learning. Training parameters may be deduced from these packets, as illustrated in Fig. 7(a). The model's input data consists of a 784*32 matrix. Consequently, the input could represent images of size 28*28, with a potential batch size of 32 during training. During federated learning, data exchanged in each round is susceptible to capture and analysis, allowing an attacker to discern the model structure and parameters from plaintext. Following SSL encryption, these parameters are rendered in ciphertext, as depicted in Fig. 7(b). Consequently, SSL-secured Flower demonstrably enhances the privacy protection of datasets and models. The privacy protection impact of FedTree can be similarly assessed using this methodology, leading to analogous conclusions.



(a) Flower communication without SSL



(b) Flower communication with SSL

Fig. 7. Packet analysis on Flower with SSL encryption.

4.5. Discussion

4.5.1. Advantages of VAN-FED-IDS

After experiments and analysis, here we conclude some advantages of our VAN-FED-IDS.

Initially, VAN-FED-IDS stands out as a practical IDS framework, meticulously designed to accommodate the unique characteristics of VANETs. VANETs demand both high communication efficiency and minimal communication costs. In our VAN-FED-IDS, we clearly define the roles of core VANET components: RSUs and OBUs serve as communication channels, while MEC units act as the edge computing hardware. These design choices not only cut down on data or model communication costs but also decrease detection and response times, facilitating easy deployment in existing VANETs. Such strategic designs underscore the practicality of our approach, setting it apart from previous VANET IDS frameworks that primarily concentrate on detection accuracy alone. Given the high-speed movement of vehicles in VANETs, a rapid intrusion detection and response capability is essential for any IDS. We introduce and adapt the DST-IDS, a lightweight intrusion detection framework renowned for its practicality, reduced data volume requirements, and flexibility. DST-IDS achieves real-time detection, crucial in the low-delay environment of VANETs. Moreover, DST-IDS effectively addresses the challenges of diverse data distribution and non-IID data, prevalent issues in deploying machine learning models. To further minimize communication overhead and enhance privacy, we designed a federated learning scheme that offloads model training to VANET's edge computing nodes and shifts from data communication to model sharing.

Secondly, to ensure the timeliness required by VANET IDS, a swift intrusion detection algorithm is imperative. Adopting the DST-IDS design philosophy, we chose machine learning models with minimal computational complexity: a shallow RNN for packet-based IDS and LightGBM for physics-based IDS. These detection results are then merged using a straightforward rule-based approach. Federated learning is demonstrated to enhance computational resource utilization and speed up model training and updating by as much as 60%, according to our experimental findings.

It is worth noting that VAN-FED-IDS attains a detection accuracy of over 99% for both recall and precision. The results in Tables 4 and 5 demonstrate that VAN-IDS achieves prediction accuracy equal to or superior to methods presented in prior studies. Precision acts as the cornerstone and essential requirement for various other advantages. In parallel with upholding precision, we have devised a fast and practical VANET IDS utilizing federated learning.

4.5.2. Disadvantages and future works

Here we also list some shortcomings of VAN-FED-IDS and example some possible future work.

The attack surface of VAN-FED-IDS includes adversarial attacks towards machine learning models and attacks against federated learning security and privacy.

Intrusion Detection Systems (IDS) are particularly vulnerable to adversarial attacks, chiefly because the benefits are higher compared to other machine learning applications. These attacks could lead to the theft of sensitive information or disrupt communications between connected vehicles. Ibitoye et al. (2019) pointed out that techniques such as Generative Adversarial Networks (GANs) or the Fast Gradient Sign Method (FGSM) can be employed to craft adversarial samples, potentially significantly diminishing the accuracy of IDS. Deep learning models, such as LSTM networks employed in VAN-IDS, and tree-based algorithms like LightGBM, frequently stand as the focal points of these attacks. Although certain countermeasures like gradient masking, input randomization, or adversarial training have shown effectiveness against adversarial threats, they fall short of providing absolute security to IDS. Moreover, these defensive strategies tend to prolong model training durations and introduce extra computational burdens.

Mothukuri et al. (2021) surveyed the security and privacy challenges in federated learning. A classic example is data poisoning, wherein attackers generate contaminated samples to train the global model, thereby producing falsified model parameters that are sent to the server. IDS with poisoned models are ineffective at identifying attacks. Furthermore, data privacy in federated learning may be compromised through inference attacks. Secure aggregation methods or differential privacy could be viable solutions; however, these defenses significantly extend model training time. As shown in Table 8, secure aggregation using SSL encryption doubles both training time and communication costs.

Although various defensive methods have been proposed, a balance between security and cost must be achieved in VANETs. The question of how to safeguard the security of IDS itself, while maintaining acceptable communication delays and computational costs in VANETs, remains unanswered.

While the viability and validity of VAN-FED-IDS have been tested in a simulation environment, further efforts are needed to implement it in practical applications. Initially, engineering tasks must be completed to optimize the entire system. In simulations, the main focus lies in the accuracy and time efficiency of the core detection algorithm. Furthermore, software engineers are responsible for ensuring efficient data exchange in VAN-FED-IDS to reduce system latency. Secondly, the modularity of VAN-FED-IDS enables the substitution of the current machine learning algorithms (Bi-LSTM and lightGBM) with more recent models. Depending on the deployment environment and data volume, users can choose from different detection methods. For example, in scenarios where high detection accuracy is crucial for VANET, users might choose advanced and precise models, including foundational ones. Lastly, the federated learning aspect of VAN-FED-IDS can be incorporated using commercial federated learning frameworks such as FATE, which provide engineering enhancements and have been effectively utilized in real distributed systems with numerous nodes. Introducing a commercial FL framework into VANET presents a challenge for future research.

5. Conclusion

This study introduces VAN-FED-IDS, an Intrusion Detection System (IDS) for Vehicular Ad-hoc Networks (VANET) employing federated learning. Initially, a hybrid detection algorithm, VAN-IDS, is proposed, based on Dempster–Shafer Theory (DST) fusion. Packet-based and physics-based features are amalgamated using multi-tier DST fusion. Datasets for experimental evaluation were acquired from the simulation environment F2MD. Ablation and comparative experiments demonstrate that VAN-IDS achieves a detection accuracy of 99% for both DoS and Sybil attacks, surpassing current state-of-the-art IDSs for VANET, while maintaining real-time detection speed. Building upon the detection kernel, VAN-FED-IDS is architected with detailed explanations of runtime operations, including model training, updating, and inference. Simulation experiments in federated learning, wherein multiple nodes are emulated by multiple threads, affirm that federated learning can expedite model training and updating while ensuring prediction accuracy and data privacy. Additionally, a system prototype with three nodes was deployed, with experiments indicating that VAN-FED-IDS's communication costs and training times are within acceptable parameters. Distributed model learning has been shown to accelerate training by nearly 60%. SSL encryption alongside the security aggregation function can effectively safeguard privacy. These findings substantiate the benefits of integrating federated learning into VAN-IDS, rendering VAN-FED-IDS a viable IDS solution for VANET.

CRediT authorship contribution statement

Xiuzhen Chen: Writing – review & editing, Validation, Project administration, Methodology, Conceptualization. **Weicheng Qiu:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Conceptualization. **Lixing Chen:** Writing – review & editing, Writing – original draft, Methodology. **Yinghua Ma:** Writing – review & editing, Methodology. **Jin Ma:** Writing – review & editing, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Beutel, D.J., Topal, T., Mathur, A., et al., 2020. Flower: A friendly federated learning research framework. arXiv preprint arXiv:2007.14390.

Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining. pp. 785–794.

Chen, X., Zhang, T., Shen, S., et al., 2021. An optimized differential privacy scheme with reinforcement learning in vanet. Comput. Secur. 110, 102446.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Huang, Y., Ma, M., 2023. ILL-IDS: An incremental lifetime learning IDS for VANETs. Comput. Secur. 124, 102992.

Ibitoye, O., Abou-Khamis, R., Shehaby, M., et al., 2019. The threat of adversarial attacks on machine learning in network security–A survey. arXiv preprint arXiv: 1911.02621.

Kamel, J., Jemaa, I.B., Kaiser, A., et al., 2019a. Misbehavior detection in C-ITS: A comparative approach of local detection mechanisms. In: 2019 IEEE Vehicular Networking Conference. VNC, IEEE, pp. 1–8.

Kamel, J., Kaiser, A., ben Jemaa, I., et al., 2019b. CaTch: A confidence range tolerant misbehavior detection approach. In: 2019 IEEE Wireless Communications and Networking Conference. WCNC, IEEE, pp. 1–8.

Kamel, J., Wolf, M., Van Der Hei, R.W., et al., 2020. VeReMi extension: A dataset for comparable evaluation of misbehavior detection in VANETs. In: ICC 2020-2020 IEEE International Conference on Communications. ICC, IEEE, pp. 1–6.

Karthiga, B., Durairaj, D., Nawaz, N., et al., 2022. Intelligent intrusion detection system for VANET using machine learning and deep learning approaches. Wirel. Commun. Mob. Comput. 2022.

Ke, G., Meng, Q., Finley, T., et al., 2017. Lightgbm: A highly efficient gradient boosting decision tree. Adv. Neural Inf. Process. Syst. 30.

Li, Q., Cai, Y., Han, Y., et al., 2022. Fedtree: A fast, effective, and secure tree-based federated learning system.

Li, Q., Wen, Z., He, B., 2020. Practical federated gradient boosting decision trees. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, (no. 04), pp. 4642–4649.

Liu, Y., Fan, T., Chen, T., et al., 2021a. FATE: An industrial grade platform for collaborative learning with data protection. J. Mach. Learn. Res. 22 (226), 1–6.

Liu, H., Zhang, S., Zhang, P., et al., 2021b. Blockchain and federated learning for collaborative intrusion detection in vehicular edge computing. IEEE Trans. Veh. Technol. 70 (6), 6073–6084.

Lu, Z., Qu, G., Liu, Z., 2018. A survey on recent advances in vehicular network security, trust, and privacy. IEEE Trans. Intell. Transp. Syst. 20 (2), 760–776.

Lv, P., Xie, L., Xu, J., et al., 2021. Misbehavior detection in VANET based on federated learning and blockchain. In: International Conference on Algorithms and Architectures for Parallel Processing. Springer, Cham, pp. 52–64.

Mahmoudi, I., Kamel, J., Ben-Jemaa, I., et al., 2020. Towards a Reliable Machine Learning-Based Global Misbehavior Detection in C–ITS: Model Evaluation Approach. In: Vehicular Ad-hoc Networks for Smart Cities, Springer, Singapore, pp. 73–86.

McMahan, B., Moore, E., Ramage, D., et al., 2017. Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics. PMLR, pp. 1273–1282.

Mothukuri, V., Parizi, R.M., Pouriyeh, S., et al., 2021. A survey on security and privacy of federated learning. Future Gener. Comput. Syst. 115, 619–640.

Posner, J., Tseng, L., Aloqaily, M., et al., 2021. Federated learning in vehicular networks: Opportunities and solutions. IEEE Network 35 (2), 152–159.

Qiu, W., Ma, Y., Chen, X., et al., 2022. Hybrid intrusion detection system based on Dempster-Shafer evidence theory. Comput. Secur. 117, 102709.

Rasheed, A., Gillani, S., Ajmal, S., et al., 2017. Vehicular ad hoc network (VANET): A survey, challenges, and applications. In: Vehicular Ad-Hoc Networks for Smart Cities: Second International Workshop, 2016. Springer Singapore, pp. 39–51.

Shafer, G., 1976. A Mathematical Theory of Evidence. Princeton University Press.

Sontakke, P.V., Chopade, N.B., 2022. Optimized deep neural model-based intrusion detection and mitigation system for vehicular Ad-Hoc network. Cybern. Syst. 1–29.

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

Yang, Q., Liu, Y., Chen, T., et al., 2019. Federated machine learning: Concept and applications. ACM Trans. Intell. Syst. Technol. 10 (2), 1–19.

Zhou, M., Han, L., Lu, H., et al., 2020. Distributed collaborative intrusion detection system for vehicular Ad Hoc networks based on invariant. Comput. Netw. 172, 107174.

Xiuzhen Chen is an associate professor at Shanghai Jiao Tong University, Shanghai, China. She received the Ph.D. degree in Network Security from Xi'an Jiao Tong University. Her research interests mainly include intrusion detection, security evaluation and vehicular network security.

Weicheng Qiu received the BE and ME degrees from Shanghai Jiao Tong University. His research interests include intrusion detection system and machine learning.

Lixing Chen is an assistant professor at Shanghai Jiao Tong University. He received the Ph.D. degree in Electrical and Computer Engineering from the University of Miami in 2020, and the BS and ME Degrees from the College of Information and Control Engineering, China University of Petroleum, Qingdao, China, in 2013 and 2016, respectively. His primary research interests include mobile edge computing and machine learning for networks.

Yinghua Ma received a B.Sc. degree in Power System Automation (1993), and a M.S degree in Electronics (1996) from Shangdong University and a Ph.D. degree in Communication and Information System (2004) from Shanghai Jiaotong University. In 2005, she worked in Burgundy University as a postdoc researcher working mainly on semantic network. She is currently a lecture in the School of Network Security Engineering in Shanghai Jiaotong University. Her research interests are on information semantic analysis and related topics.

Jin Ma is the associate dean of Institute of Cyber Science and Technology and a senior engineer at Shanghai Jiao Tong University, Shanghai, China. She received the Ph.D. degree in EE from Shanghai Jiao Tong University. Her research interests mainly include artificial intelligence applications and vehicular network security.