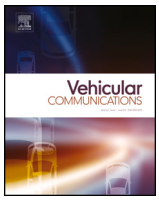
Vehicular Communications 48 (2024) 100787

Contents lists available at ScienceDirect

Vehicular Communications

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



# VAIDANSHH: Adaptive DDoS detection for heterogeneous hosts in vehicular environments

Amandeep Verma [a,d], Rahul Saha [a,b,*], Gulshan Kumar [a,b], Mauro Conti [b], Joel J.P.C. Rodrigues [c]

[a] School of Computer Science and Engineering, Lovely Professional University, Punjab, India
[b] Department of Mathematics, University of Padua, Padua, Italy
[c] Amazonas State University, Manaus - AM, Brazil
[d] Centre of Research Impact and Outcome, Chitkara University, Punjab, India

ARTICLE INFO

*Keywords:*
Vehicle
Intrusion
Security
DDoS
VANETs
Machine learning

ABSTRACT

Vehicular networks are vulnerable to Distributed Denial of Service (DDoS), an extension of a Denial of Service (DoS) attack. The existing solutions for DDoS detection in vehicular networks use various Machine Learning (ML) algorithms. However, these algorithms are applicable only in a single layer in a vehicular network environment and are incapable of detecting DDoS dynamics for different layers of the network infrastructure. The recently reported attacks on transport networks reveal the fact that a research gap exists between the existing solutions and the multi-layer DDoS detection strategy requirements. Additionally, the majority of the current detection methods fail in the consideration of traffic heterogeneity and are not rate-adaptive, where both the mentioned parameters are important for an effective detection system.

In this paper, we introduce a comprehensive ML-based Network Intrusion Detection System (NIDS) against DDoS attacks in vehicular networks. Our proposed NIDS combines a three-tier security model, traffic adaptivity, and heterogeneity traffic provisions. We call our model *Vehicular Adaptive Intrusion Detection And Novel System for Heterogeneous Hosts (VAIDANSHH)*. As mentioned earlier, VAIDANSHH has a three-tier security system: at RSU's hardware, communication channel, and RSU application level. VAIDANSHH combines the Adaptive Alarming Module (AAM) and the Detection Module (DM) for data generation, collection of generated data, flow monitoring, pre-processing, and classification. We use the NS3 simulation tool for our experiments to generate synthetic data and apply ML with WEKA. We run a thorough set of experiments, which show that VAIDANSHH detects UDP flooding, a form of DDoS attack, with 99.9% accuracy within a very short time. We compare VAIDANSHH with other state-of-the-art models; the comparative analysis shows that VAIDANSHH is superior in terms of accuracy and its multi-tier workflow.

## 1. Introduction

The progress of vehicular technology and Internet systems' integration leads to the significant applications of the Internet of Things (IoT) [1] in vehicular networks. The progress of vehicular technologies leads to Intelligent Transportation Systems (ITS) that continues to evolve. The main elements of ITS are intelligent cars, innovative infrastructure, electric vehicles (EVs), and electric grids. ITS is a framework incorporating various innovative technologies in the classical transport system to make it more clever, safer, convenient, and congestion-free. Vehicular Ad hoc Networks (VANETs) emerged with an identical objective. [2] [3]. In VANETs, communication occurs between vehicles and various other infrastructural units [4].

The EVs and their futuristic developments with intelligent vehicular operations benefit the sustainable growth of VANETs and ITS [5]. The vehicle industry is focusing on EVs as its future. EVs use smart charging to charge their batteries, sharing the data connection with the charging device and the charging operator. Smart grid technology is used by certain systems to allow EVs to exchange surplus battery power [6]. Vehicle-to-vehicle (V2V) charging is also suitable for better charge utilization among vehicles [7]. Therefore, vehicles on the road maintain a connection to a nation's vital resources, including its power grid. Thus, the security requirements of VANETs are crucial.

* Corresponding author at: Department of Mathematics, University of Padua, Padua, Italy.
*E-mail addresses:* rsahaaot@gmail.com, rahul.saha@unipd.it (R. Saha).

https://doi.org/10.1016/j.vehcom.2024.100787
Received 16 September 2023; Received in revised form 28 December 2023; Accepted 1 May 2024
Available online 10 May 2024

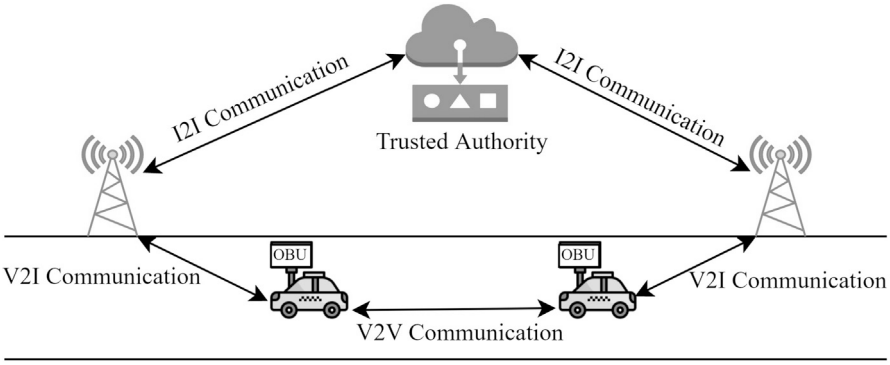



**Fig. 1.** Architecture of vehicular ad hoc network.

### 1.1. VANET architecture

We can describe VANET architecture based on components, communication, and a hybrid approach. VANETs use vehicles such as electric, nonelectric, or hybrid. VANETs use cellular networks and Dedicated Short Range Communication (DSRC) for communication. The component-based architecture of VANETs uses various components such as infrastructural components, mobile components, internet, and other related components. In the case of communication-based architecture, VANETs concentrate more on how communication takes place between vehicles than network components. In the hybrid approach, the elements and communication methodology are equally important. We show a generic architecture of VANET in Fig. 1.

The main components of VANETs include On-Board Units (OBU), Road-Side Units (RSU), and Trusted Authority (TA) [8]. All these units communicate with each other using DSRC. VANETs use OBU for communicating with other vehicles, infrastructure units/RSUs, and TA. OBU is a GPS-based device having multiple components like a processor, sensors, storage, and interface for communication. The RSUs and cellular towers are Infrastructural Units (IU) [8]. RSUs have antennas, processors, sensors, and storage systems and connect to infrastructural units using computing capabilities for intelligent systems. These towers receive and forward messages among vehicles, trusted authorities, and other cellular networks. TA is a centralized authority responsible for various activities such as registration of vehicle users, OBUs, and RSUs. TAs serve in such a place where all the traffic is easily manageable. TAs have systems with high computational power and large storage capacity and consume large amounts of energy without interruption. Vehicles and RSUs use cellular towers for communication. In contrast, some towers may communicate with other networks using a wireless communication link of IEEE 802.11p, a standard for communication in VANETs. VANETs use DSRC (IEEE 802.11p) and Wireless Access in Vehicular Environment (WAVE) for communication within RSUs, TAs, and other vehicles and other methods like 3G, 4G, or LTE cellular networks for communication with other networks.

### 1.2. DDoS problem in VANETs

Distributed Denial of Service (DDoS) is an acute and multi-dimensional attack in VANETs due to its varsity of attack vectors. DDoS attacks in VANETs are multi-dimensional because they encompass various attack vectors, target diverse network components, exploit vulnerabilities at different protocol layers, and disrupt dynamic traffic patterns, making them complex and challenging to mitigate. In DDoS, multiple attackers simultaneously send multiple traffic packets toward a target. Thus, the target gets exhausted and stops its regular operations. The attackers try to identify some vulnerabilities in the target host/system or the network and attempt to inject or propagate malware. The attackers try to create several zombies using the method mentioned above to have a significant impact while obscuring their true identities. The attacker uses command and control on the zombie systems [9]. When attackers launch an attack after compromising various nodes as zombies, its impact increases because of the large amount of fake traffic toward the victim node or network. These attacks are reflective botnet-based flooding attacks [10]. An attacker may launch a DDoS attack in two ways: basic DoS attacks and extended DoS attacks. Basic DoS attacks deplete the resources of vehicles or any infrastructure unit. In contrast, an extended attack forwards a large volume of data and blocks communication between multiple vehicles or infrastructure units and vehicles.

In a sleep deprivation attack, the attacker keeps the victim node alive until all its battery power is consumed [11] [12]. Flooding attacks forward many spoofed packets towards the victim, and the victim exhausts its resources [11]. In synchronization-based DDoS, the attacker broadcasts fake/dummy packets at a particular time when the corresponding provider starts sending service messages [13]. These service messages and dummy messages are synchronized and collide due to their same arrival time. Legitimate messages are unreachable due to the collision. Jammers work by sending radio frequency signals. In a jamming attack, the attacker jams the complete communication among all nodes within the jammer's range. In a jellyfish attack, the attacker enters the network and becomes a part of the network [14]. Then after receiving all the messages in the network, the attacker inserts some delays in forwarding the packets. The delays create more waiting time and thus, the packets are dropped; ultimately it leads to the unavailability of services to the victims. In an intelligent cheater attack, a malicious node misbehaves at a sudden instant of time and creates a negative impact on the packet unavailability [15].

### 1.3. Motivation and contribution

The existing solutions address the issue of DDoS attacks in vehicular environments, but these solutions use only single-layer architecture for attack detection. The single-layer architecture uses parameters from a single tier which yields less promising results. Single-layer architecture for DDoS detection in vehicular environments yields less promising results because it focuses on a limited set of parameters, which may lead to incomplete or inaccurate detection of attacks that span multiple layers and use diverse attack vectors. To overcome the shortcomings of the previous studies, we propose VAIDANSHH, which is an ML-based DDoS detection solution in VANETs. Specifically, VAIDANSHH is the first ML-based model to consider multi-tier architecture and adaptive thresholds of the traffic for efficient DDoS attack detection in vehicular environments. VAIDANSHH demonstrates the following contributions for this purpose.

- **Three-tier architecture:** VAIDANSHH provides a three-tier security architecture organized into physical and logical tiers. These three tiers are the hardware tier (physical tier), the interface tier (communication channel), and the application tier. The advantage

of this three-tier architecture is that it provides a safe vehicle environment by applying rigorous security checks. These checks include analysis of flow parameters, hardware consumption parameters, and unique features of a dataset.

- **Adaptive IDS:** Existing solutions are static but our proposed system VAIDANSHH is a dynamic IDS that collects real-time packet information and adapts the thresholds of traffic according to traffic load. Adaptation generates more accurate and reliable results.
- **Heterogeneity:** VAIDANSHH provides flexibility to incorporate vehicles of different vendors, standards, protocols, and technologies. Traffic is also heterogeneous in VANETs. As a result, the heterogeneity of vehicles does not create any problems related to compatibility or interoperability.
- **Dataset:** As DDoS datasets in VANET scenarios are rare, we generate a new VANET-based DDoS dataset. The dataset includes records of both benign traffic and DDoS attack traffic of UDP and TCP flood attacks. The inclusion of various attack types makes this dataset universally acceptable.
- **Classification:** We pioneer the BayesNet classification algorithm in the vehicular environment to detect DDoS attacks. BayesNet provides an approach to deal with the missing data and enables data to be combined with the domain knowledge. The algorithm promotes learning about links between variables and provides a mechanism for preventing data over-fitting. In ML, when a statistical model fails to produce reliable predictions on test data, it is said to be over-fitted. BayesNet can forecast accurately even with a small sample size. Another advantage of using BayesNet is that this algorithm can be easily coupled with decision analysis tools to improve data analysis and management.

### 1.4. Paper organization

We have organized the rest of the paper as follows. Section 2 defines the work of various researchers in DDoS security and displays their results. Section 3 shows the proposed work and defines the functionality of VAIDANSHH. Section 4 shows the results obtained from implementing our proposed intrusion detection system. In Section 5 we have compared our results with results of Naive Bayes family algorithms and results of other algorithms. Similarly, we have also compared our results with the solutions of other researchers from the same domain. Finally, Section 7 concludes our work.

## 2. Related work

In this section, we review some contemporary ML models used in VANETs for intrusion detection and analyze the parameters and performance of the models. Note that, we emphasize the research works, which attempt to detect DDoS in vehicular networks.

Grover et al. (2011) propose an ML-based solution for misbehavior detection in vehicular networks [16]. It uses tools like NCTUns-5.0 and WEKA for DDoS detection and shows the efficiency of IDS using random forest, Naive-Bayes, IBK, J-48, and Ada-boost1. Li et al. (2015) propose a context-aware ML-based solution. The SVM-CASE framework is a context-aware security framework that uses SVMs to detect security threats in VANETs. It uses the GloMoSim 2.03 tool and takes into account the context of the network, such as the location, velocity, and acceleration of vehicles, as well as the traffic conditions, to make more accurate security decisions [17]. Similarly, Ghaleb et al. (2017) suggest a model that uses ANN to detect different types of misbehavior in VANETs such as black holes, gray holes, wormholes, and Sybil attacks. The authors have used various datasets to train the ANN and evaluate its performance. The model's performance is evaluated in terms of accuracy, precision, recall, and F1-score [18]. Kim et al. (2017) propose a collaborative security attack detection mechanism that is an extension of their previous work on Software-Defined Vehicular Cloud (SDVC) [19]. It uses Multi-class SVM in MATLAB and the KDD CUP-1999 dataset to predict the attacks. The authors evaluate the performance of the proposed mechanism using simulations, and the results show that it detects security attacks with high accuracy and low false-positive rates. In the same domain, Yu et al. propose an SDN-based system for detecting DDoS attacks in vehicular environments [20]. The system uses differences in traffic flow and position as parameters for attack detection. It uses the Mininet network simulator tool and applies SVM.

D. Karagiannis and A. Argyriou (2018) propose a new IDS against spoofing attacks in connected EVs. The proposed IDS uses the R-Studio tool and unsupervised learning with clustering (k-means). The system uses Relative Speed Variations (RSV) as the detection parameter [21]. Similarly, Liang et al. (2018) suggest an Improved Growing Hierarchical Self-Organizing Map (I-GHSOM) that uses differences in traffic flow and position as parameters for attack detection. It uses the Mininet network simulator tool and applies the Support Vector Machine (SVM) algorithm [22]. Kosmanos et al. (2019) propose a new solution to handle the spoofing attacks in electric vehicular networks with an ML approach [23]. It employs a clustering algorithm to group vehicles according to their charging patterns and then uses a decision tree to classify the vehicles as normal or abnormal. Additionally, the IDS uses machine learning algorithms to analyze the spatial and temporal characteristics of the vehicles' charging patterns, to detect any anomalies that may indicate a spoofing attack. Kaur et al. (2019) provide an enhanced approach of an adaptive neuro-fuzzy system for attack detection in Vehicular networks [24]. The proposed approach uses an ANFIS, which is a type of artificial intelligence system that combines the strengths of neural networks and fuzzy logic to make decisions. The ANFIS is trained using a dataset of normal and attack traffic, and it uses a combination of related features. Aloqaily et al. (2019) propose an intrusion detection system for connected vehicles in smart cities [25]. It uses the MATLAB tool and applies the decision tree algorithm to the NSL-KDD dataset. Kolandaisamy et al. (2019) introduce a solution with an integrated Markov-chain and ant-colony optimization scheme [26]. The solution uses NS2 and an Exploratory Based Ant Colony Approach (EBACA). Similarly, Manimaran et al. (2020) propose NDNIDS, an intrusion detection system for NDN-based VANETs [27]. It uses heuristic-based adaptive IDS and generates a dataset. It also uses multiple sensor values and the driver's heartbeat rate as a detection feature. Schmidt et al. (2020) propose spline-based IDS and work with knot flow classification [28]. Adhikary et al. (2020) show a hybrid algorithm to detect DDoS attacks in VANETs [29]. It uses AnovaDot and RBFDot in SVM for classification. The generated/synthesized dataset uses jitter, delay, packet drop, throughput, and collision for attack detection. Recently, Kadam et al. (2021) proposed a Hybrid KSVM scheme for intrusion detection using parameters like protocol, source and destination IP, and port no. [30]. Türkoğlu et al. (2022) introduce SD-VANET architecture for intrusion detection in vehicular networks [36]. It works using the Minimum Redundancy Maximum Relevance (MRMR) based feature selection algorithm for classification. In addition to the above-mentioned works, the other recent works in [37], [38], [39], and [40] have provided similar technological advancements in the field of DDoS detection, QoE, QoS, and network technologies. A summary of all these solutions is visible in Table 1.

## 3. Proposed model: VAIDANSHH

VAIDANSHH is the first three-tier Network Intrusion Detection System (NIDS) that incorporates three layers (tiers) of security, adaptive traffic thresholds, and minimal packet features for attack detection. This layered approach makes a more efficient and effective IDS by reducing false positives, providing multiple layers of protection, and allowing for more advanced incident response. The first tier of security checks the hardware resource consumption like CPU and RAM and suggests abnormality. The second tier monitors the flow parameters at the communication channel level to find attack patterns in traffic, and the third

**Table 1**
Summary of related work.

| Year | Author(s) of Paper | Method | Parameters | Claims of accuracy | Advantage | Limitations |
|---|---|---|---|---|---|---|
| 2011 | Grover et al. [16] | ML with RF, NB, IBK, J-48 | Packet related features | 92.5% | Predicts misbehavior in different applications | Not applicable for temporal attacks |
| 2017 | Ghaleb et al. [18] | Artificial neural network (ANN) | Mobility message prediction error, Appearance position based feature, overlaying check etc. | 99.7% | Phases-wise detection system works more efficiently | Detects only misbehavior of driver |
| 2017 | Kim et al. [19] | Multi-class SVM | Packet related features, RTS flooding rate, Wireless channel status | 85%, | Method can identify the type of attack | Extra usage of network bandwidth |
| 2018 | Yu et al. [20] | Support vector machine | pkin count, srcIP, dstIP, srcport, dstport, protocol, packet, byte count | 97% | Quick detection and low false alarm | Suitable for software-defined VANET only. |
| 2018 | D. Liang et al. [22] | I-GHOSM | FlowR and PositionR | 99.69% | Novel feature extraction method and classifier | More computational overheads |
| 2019 | Kosmanos et al. [23] | Random Forest and k-NN algorithm | Position verification using relative speed (PVRS) | 91.3% | Perfect for spoofing attacks | Suitable for EVs only |
| 2019 | Aloqaily et al. [25] | Decision Tree | Number of TTPs, maximum speed, packet size, traffic type, no. of vehicle | 99.4% | Meet users' QoS and QoE requirements | Not tested with VANET dataset |
| 2020 | Schmidt et al. [28] | Spline-based IDS | Jitter, delay, packet drop, throughput, and collision. | 73% | Works with knot flow classification | Validation required in different environments |
| 2021 | Kadam et al. [30] | Hybrid KSVM scheme | protocol, source destination IP amd port no. | 92.5% | Benefits of both KNN & SVM algorithms | Limited types of attack detection |
| 2021 | Zang et al. [31] | Random Forest | protocol, source destination IP amd port no. | 95% | Improved accuracy rate and lightweight | Handover processes cause problems |
| 2021 | Bangui et al. [32] | Random Forest and K-Means | Signature detection | 96% | Post detection Phase for new threats | Additional memory overheads |
| 2021 | Goncalves et al. [33] | RF, J48 and MLP | protocol, source destination IP amd port no. | 96% | High accuracy & Low false positives | Performance is not good in certain attack types |
| 2022 | Soni et al. [34] | Particle Swarm optimization | Vehicle mobility and trust features | 89% | High recovery rate | Performance is not checked in High speed vehicles |
| 2022 | Malik et al. [35] | Detection and Prevention of a BHA (DPBHA) | RREPs, RREQs, Hop count etc. | 95% | High PDR and throughput with less delays | Only suitable for Black-hole attacks |
| 2022 | Türkoğlu et al. [36] | SD-VANET architecture | MRMR based features | 99.4% | Tested multiple times for validation | Only for software defined networks |

tier does ML detection at the application level. Security checks at all three tiers make it an efficient attack detection model. The VAIDANSHH can be deployed at an RSU or TA to monitor VANET traffic. These locations are efficient because their components have more computational power than in-vehicle components. The additional computational power and memory help handle large data bursts. We show the overall architecture of VAIDANSHH in Fig. 2.

In the 3-tier security system of VAIDANSHH, the Adaptive Alarming Module (AAM) and the Detection Module (DM) collaborate to enhance the security of VANETs. AAM, situated at the second tier, proactively monitors hardware resource consumption and flow parameters, adapting to changing network conditions. In the context of UDP flooding, AAM sets thresholds for relevant flow parameters and generates an alarming signal upon detecting potential threats. This signal triggers the Activation of DM at the third tier, which utilizes machine learning algorithms to analyze network traffic patterns. Trained on a synthesized dataset, DM distinguishes between benign and malicious packets in real time. The continuous adaptability of DM ensures effective threat detection and response, creating a multi-layered defense mechanism for VANET security. The collaborative interaction between AAM and DM is crucial for achieving high accuracy rates. AAM's early detection capabilities and adaptability provide timely alerts to potential UDP flooding, while DM's machine learning-based analysis ensures precise and efficient classification of packets during real-time operation. This multi-layered approach, combining proactive monitoring and sophisticated analysis, contributes to the robustness and accuracy of VAIDANSHH in detecting UDP flooding within the VANET environment.

We describe the proposed model in three subsections. The First subsection 3.1 is an experimental setup that defines the hardware/software used in experimentation and also defines how we have set the vehicles in VANET and how the attacker launches the DDoS attack in the network. The second subsection 3.2 defines the functionality of the AAM incorporating Tier-1 and Tier-2. The third subsection 3.3 defines the working of the 1DM that uses Tier-3 for attack detection using the ML process.

### 3.1. Experimental setup

Experimental setup refers to the details of the experimental design, the procedures followed, and the conditions under which the experiment is conducted. The experimental setup is crucial to any experiment as it determines how the data will be collected, measured, and analyzed.

#### 3.1.1. System configuration

In this research, we use Windows 11 as the primary operating system for our experimentation. We install Oracle Virtual Box Manager on the Windows operating system to facilitate the installation of other virtual operating systems. We then installed Ubuntu 20.04.2 LTS as the virtual operating system for our simulations. Additionally, we employ the use of Network Simulator 3 (NS3), Weka, Python, and Wireshark as other software tools in our research. The system's hardware configuration consisted of a specialized 4 GB graphic card, "LLVM 11.0.0, 256 bits," to support the simulation, an Intel i7 9th generation processor, 8 GB of RAM, 256 GB SSD, and 1 TB HDD. Hardware configuration is suitable for simulating realistic scenarios and experimenting with multiple nodes, incorporating all the necessary features required for our research. This hardware provides the necessary computational power and storage capacity for our research. The combination of hardware and software tools enables us to conduct our simulations with a high degree of realism, making our research results more accurate and reliable.

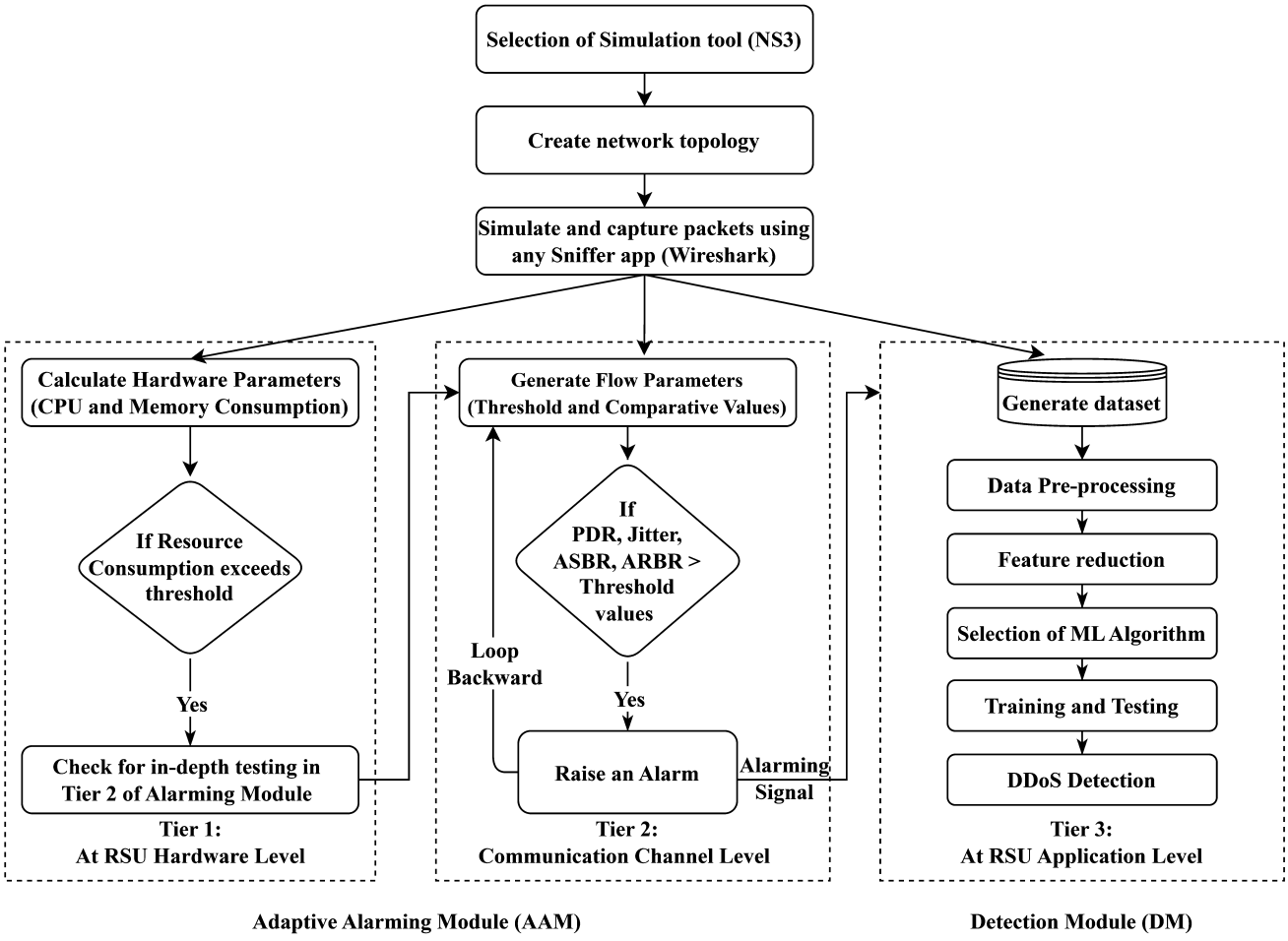



**Fig. 2.** Overall functionality of proposed system.

Regarding the potential impact on performance, we acknowledge that virtualization can introduce overhead. However, in our specific experimental context, the performance impact was deemed acceptable for the following reasons.

- Experiment Isolation: We designed our experiments to minimize interference between the virtualized environment and the host system. This involved allocating sufficient resources to the virtual machine to ensure it could operate independently.
- Resource Allocation: The physical machine hosting the virtual environment was configured with an adequate amount of RAM, CPU, and other resources to support the simultaneous operation of the host OS and the virtual machine.
- Experiment Validation: We performed preliminary experiments to validate the performance of our proposed system in this virtualized setup. The results indicated that the impact on performance was within acceptable limits for our research objectives.

**Table 2**
Attributes of experimental environment.

| Parameter | Value |
|---|---|
| Simulation Platform | NS3.2.7 |
| No. of Vehicles | 12 |
| Attacker Nodes | 1 |
| Bot Nodes | 10 |
| Victim | 1 |
| No. of RSUs | 1 |
| Routing Protocol | UDP, TCP, ICMP |
| Visualization Tool | NetAnim |
| DDoS Rate | 20480 kbps |
| Normal data rate | 512 kbps |
| Maximum Bulk Bytes | 100000 kbps |
| Simulation Time | 40 Seconds |
| Data Transmission Rate | 100 Mbps |
| Communication Range | 100 m X 100 m |
| Mobility Model | Random mobility |

*3.1.2. Network topology*

In our simulation, we employ a distinct attack topology using 12 vehicles and one RSU for communication. We use the terms “nodes” and “vehicles” interchangeably throughout the paper. Among these 12 nodes, one serves as the attacker, one as the victim, and the remaining 10 as Bot nodes. Fig. 3 illustrates this vehicular topology. In Fig. 3, the red car represents the attacker node, the green car represents the victim node, and the orange cars represent the bot nodes. We employ routing protocols such as UDP, TCP, and ICMP in the simulation. We use a data rate of 512 kbps for benign traffic and a data rate of 20480 kbps for DDoS traffic. The maximum bulk data possible is 100,000 kbps. The simulation runs for a total of 40 seconds and the communication area covers 10,000 square meters. We apply a random mobility model for all nodes, which keeps the nodes moving within the specified region in unpredictable directions. The environmental parameters of our experiments are summarized in Table 2.

VAIDANSHH effectively manages scalability through various strategic approaches. By adopting a multi-layered architecture, VAIDANSHH strategically places multiple detection points i.e. Tier-1, 2, and 3 in the vehicular network, distributing the processing load and ensuring scalability as the network size increases. VAIDANSHH leverages parallel processing capabilities to analyze network traffic concurrently,

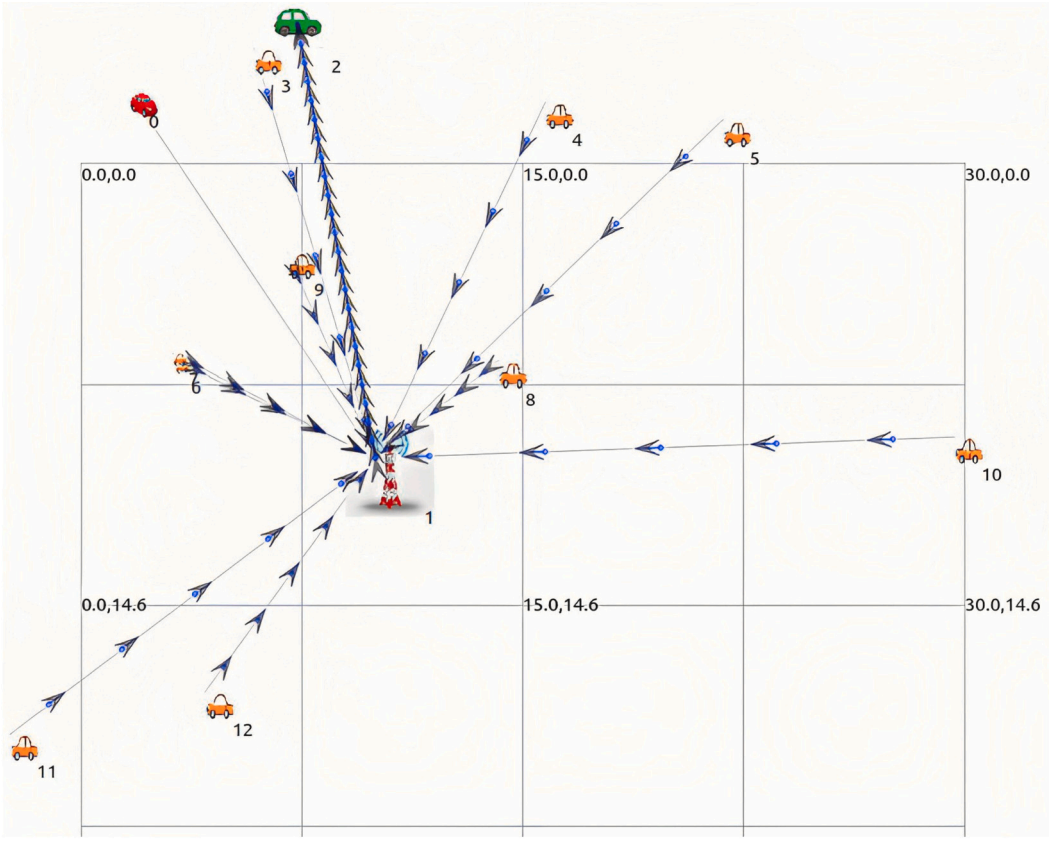



**Fig. 3.** Vehicle set up in the network.

breaking down analysis tasks into smaller units and processing them simultaneously at different layers for efficient scalability. The use of scalable hardware resources, such as powerful processors and additional memory, allows VAIDANSHH to accommodate growing demands for analyzing and processing larger amounts of vehicular network data effectively. Optimizing signature detection databases and algorithms contributes to scalability by reducing the processing overhead associated with matching patterns of known threats. Additionally, VAIDANSHH incorporates flow-based analysis, grouping related packets into flows for collective examination, reducing processing load while maintaining effective threat detection. The integration of machine learning for anomaly detection enables VAIDANSHH to adapt to changing vehicular network conditions, continuously learning and updating its understanding of normal network behavior for scalable and dynamic threat detection. This way VAIDANSHH handles the scalability of the network.

#### 3.1.3. Attack methodology

In the attack scenario, we use two types of attacks at the same time. Firstly, we use a vehicle-to-infrastructure (V2I) attack, where infrastructure is affected because RSU remains busy handling malicious messages. Second, we employ a vehicle-to-vehicle (V2V) attack in which one malicious bot node sends a large amount of traffic toward the victim vehicle, and the victim node's resources are depleted as a result of processing this massive traffic. The attacker uses the topology of small distances to make the attack more deadly. In this paper, we generate a flooding attack (UDP and TCP) using a botnet. We use peer-to-peer botnet architecture for our experiments. In this attack methodology, the attacker uses indirect communication methods and does not directly target the victim. The attacker injects some malicious code into vehicles and compromises these vehicles. Once these nodes are compromised, these nodes communicate with each other. These bot nodes, collectively forming a botnet, work together without knowledge of one another and are controlled by the attacker [41]. The attack is initiated by the attacker sending commands to the bot nodes, which subsequently send large amounts of data towards the target through the RSU. We simulate three different scenarios to incorporate packet features of all types. *1. Normal Scenario:* In this scenario, vehicles communicate with each other without being attacked. There is only benign traffic in this scenario. *2. Attack-only scenario:* In this scenario, we create only a flooding attack without any benign traffic. An attack-only scenario evaluates the system's capacity and performance under a heavy influx of incoming traffic. *3. Mix Scenario:* In this scenario, we include a DDoS attack with benign traffic. UDP and TCP floods are two types of flooding attacks in this scenario.

As discussed in the previous section 1.3, one unique feature of our model is that it can operate with different types of vehicles that have different communication standards and protocols. In real-world traffic, electric or fuel-powered vehicles may participate in communication, and these vehicles can use unique components. These components may create compatibility issues with elements of other automobiles. Our proposed solution works very well with all these vehicles and accommodates heterogeneous vehicles.

### 3.2. Adaptive alarming module

#### 3.2.1. Preliminary understanding

Some terms need prior explanation before understanding the workings of AAM. These terms are threshold values, comparative values, and adaptivity which are as follows:

*Threshold values:* Threshold values define the maximum values that the flow parameter should hold. Values beyond this threshold indicate something suspicious is happening in the network. The AAM calculates threshold values when there is only legitimate traffic in the network and there is no malicious traffic. When we deploy the model, the AAM calculates threshold values for the first time. Later, these threshold values may adapt dynamically after a fixed time interval, which may be an

hour, a day, a week, or a month. By recalculating the threshold values at regular intervals, the AAM model identifies malicious traffic as soon as it enters the network.

*Comparative values:* These are the values of the flow parameters of current traffic for which the system finds abnormalities. The AAM calculates these values for comparison against threshold values. These comparative values are computed at predetermined time intervals, but these intervals are shorter than threshold values. If a comparative value is above the threshold, the system will alert users to an anomaly.

*Adaptivity:* One unique trait of this module is that it is adaptive and keeps changing threshold values according to traffic. Traffic at different periods is not always the same; it requires changes in threshold traffic parameters based on input traffic conditions. From a DDoS point of view, increased rates always have the likelihood of attacks. Threshold values change dynamically depending on the circumstances. During flash-crowd, there are spikes in legitimate traffic. So, during flash-crowd, we calculate these values again.

Adaptive thresholds in VANET traffic involve dynamically adjusting threshold values based on current traffic conditions and network characteristics. The process begins by monitoring and collecting various traffic parameters such as vehicle speed, acceleration, density, and inter-vehicle distance. Statistical analysis is then performed on the collected data to determine the normal range of values for each parameter under typical traffic conditions, utilizing measures like mean, standard deviation, and percentiles.

These calculations establish threshold values that define the upper and lower bounds of normal parameter values. The thresholds are dynamically adjusted in real-time to adapt to changing conditions. Factors such as time of day, location, weather conditions, road type, and historical data influence the adjustment process. The adaptive thresholds account for varying traffic patterns and environmental influences by incorporating these factors. Once the thresholds are established and dynamically adjusted, they serve as reference points to detect abnormal traffic patterns. When the current parameter values exceed or fall below the adaptive thresholds, it indicates the presence of suspicious behavior.

Abnormal traffic patterns encompass various scenarios, including sudden changes in vehicle speed, abrupt accelerations or deceleration, excessively close following distances, increased traffic, or any behavior significantly deviating from the established normal range. Upon detecting abnormal patterns, the system generates warnings or alerts and communicates to nearby vehicles or the centralized traffic management system. These warnings allow for timely responses to potential hazards or abnormal situations on the road.

*Heterogeneity:* Heterogeneity in VANET refers to the diversity and variation in vehicle characteristics, behavior, and communication capabilities within the network. In a VANET environment, vehicles vary in size, type, speed, communication protocols, and driving behavior, among other factors.

In vehicular environments, heterogeneous hosts refer to the presence of diverse computing devices, systems, or nodes that participate in the communication and networking infrastructure. These hosts differ in capabilities, computing power, communication protocols, and functionalities. Here are some examples of heterogeneous hosts in vehicular environments: Vehicles: The primary heterogeneous hosts in vehicular environments are the vehicles themselves. Vehicles from different manufacturers or models may have varying communication capabilities, processing power, storage capacities, and onboard systems. These differences can arise due to factors such as vehicle type (e.g., cars, trucks, motorcycles), age of the vehicle, and the level of technology integration. Roadside Infrastructure: Vehicular environments often involve deploying roadside infrastructure, such as roadside units (RSUs) or intelligent transportation systems (ITS). These infrastructure components act as hosts facilitating communication and providing services to vehicles. RSUs vary in capabilities, ranging from primary communication relays to more advanced units capable of performing data processing or traffic management functions. Cloud Servers or Data Centers: In some cases, vehicular environments may leverage cloud servers or data centers to support computing and storage requirements. These cloud-based hosts can provide additional processing power, storage capacity, and services to the vehicular network. Cloud servers' locations, service providers, and resource availability can be heterogeneous. Portable Devices: Portable devices carried by drivers or passengers, such as smartphones, tablets, or laptops, can also act as heterogeneous hosts in vehicular environments. These devices can connect to the vehicular network and interact with other hosts, enabling communication, data sharing, and accessing services. Dealing with heterogeneous hosts in vehicular environments requires considering interoperability, communication protocols, and resource allocation.

VAIDANSHH in addressing the challenges associated with heterogeneous traffic in VANETs. While it is true that the existence of heterogeneous traffic has been acknowledged in the literature, VAIDANSHH introduces several innovative features that set it apart from existing methods.

- Dynamic Protocol Translation: VAIDANSHH incorporates a dynamic protocol translation module, which allows it to seamlessly integrate vehicles using different communication protocols. Unlike traditional methods that may rely on predefined static mappings, our approach dynamically adapts to the communication standards of each vehicle in real time. This ensures that vehicles with diverse communication protocols can effectively communicate within the VANET, overcoming a common limitation in existing solutions.
- Vendor-Agnostic Adaptability: One distinctive aspect of VAIDANSHH is its vendor-agnostic adaptability. We introduce a vehicle abstraction layer that decouples the communication logic from the underlying hardware and vendor-specific implementations. This enables VAIDANSHH to integrate vehicles from different manufacturers without requiring custom adaptations for each vendor. In contrast, many existing methods struggle with vendor-specific nuances, making them less flexible in accommodating a wide range of vehicles.
- Context-Aware Traffic Management: VAIDANSHH goes beyond acknowledging heterogeneity and introduces a context-aware traffic management system. By considering the specific characteristics and capabilities of each vehicle in the network, our approach optimizes the allocation of resources and communication strategies dynamically. This level of adaptability ensures efficient communication even in highly diverse traffic scenarios, a feature not commonly found in existing methods that may employ generic traffic management strategies.
- Experimental Validation in Diverse Scenarios: To substantiate our claims, we conducted extensive experiments in diverse scenarios, including simulations of urban environments with a mix of autonomous and non-autonomous vehicles. The results demonstrate that VAIDANSHH outperforms existing methods in terms of communication reliability, resource utilization, and overall network efficiency in the presence of heterogeneous traffic.

### 3.2.2. Functional modules

The AAM is the first module of VAIDANSHH that checks hardware resources and flow parameters to generate an alert. We deploy the AAM at an RSU or a central TA. The efficiency of the AAM increases when we deploy it in RSU or TA. In-vehicle components are not capable of handling heavy traffic and can not do heavy computations. For this reason, we do not install it inside the vehicle. The AAM has two layers of security defined in terms of Tier-1 and Tier-2. Tier 1 uses RAM and CPU utilization as the main parameters for finding an abnormality in the network. Tier 2 uses flow parameters like the average sent bit rate, received bit rate, average packet loss ratio, average mean delay, jitter, and flow IDs to find an abnormality. In the case of a DDoS attack, resource consumption of hardware increases significantly. Flow parameters like packet loss ratio, delay time, and the sent bit rate and received

bit rate also increase. AAM maintains some threshold values and compares these values with current traffic values to find abnormalities in the network. If hardware resource consumption and current traffic values exceed threshold values then this pattern indicates some abnormality in the network, which may be a DDoS attack. The AAM immediately raises an alarm by sending a message to DM. Detailed description of its two tiers is as follows:

*a) Tier-1: Hardware level:* The first tier of our proposed security system operates on the physical level, specifically at the hardware level. It utilizes CPU and memory utilization as key indicators to detect abnormal network behavior. During normal network traffic, the consumption of hardware resources remains relatively low. However, during an attack, traffic increases and causes an abnormal increase in resource consumption. Excess resource consumption leads to packet drop and waiting in queues for resource availability, resulting in a Denial of Service (DoS) attack. Our experiments with VAIDANSHH have shown that the CPU load increases to its maximum limit of 100%, and memory consumption exceeds 90%.

*b) Tier-2: Channel level:* In our proposed model, the communication channel serves as the second tier. The second tier calculates the flow parameters in real-time to find anomalies in the network. As data moves from the source to the destination, it generates various patterns that help in understanding traffic behavior. The second tier observes these patterns and actively calculates flow parameters such as jitter, delay, and packet drop ratio. By setting threshold values for these flow parameters, the second tier actively compares them to the current traffic and detects any abnormalities. If it finds comparative values higher than the threshold, it indicates an anomaly and takes immediate action by raising an alert. Alarms are triggered by the AAM when it detects unusual or abnormal activity that may indicate a DDoS attack. These alarms are typically generated in real-time, and they can be configured to trigger based on a variety of conditions like unusual patterns in network traffic, or a high number of requests from a specific IP address or group of IP addresses. The AAM generates an alarm when the comparative values cross the threshold values. After generating the alarm, the function of the detection system starts.

In our proposed architecture, tier 2 indeed generates an alarming signal based on its analysis of network behavior and anomalies. This alarming signal serves as a trigger for the Detection Module in tier 3. The Detection Module, upon receiving the alarming signal, initiates a series of machine-learning tasks to assess the nature of the incoming packets. The Adaptive Alarming Module i.e. tier 2 continuously monitors network traffic and conducts anomaly detection. When the module identifies potentially malicious activity or anomalies surpassing a predefined threshold, it generates an alarming signal. The Detection Module, situated in tier 3, acts as a recipient of the alarming signal from tier 2. Upon receiving this signal, the Detection Module engages in a comprehensive set of machine learning tasks, including but not limited to feature extraction, pattern recognition, and classification. This reinforces the hierarchical flow of information from tier 2 to tier 3, ensuring that the Detection Module undertakes the necessary analysis when alerted by the alarming signal.

### 3.3. Detection module the Tier-3

The DM is the second main module of our proposed model that uses the BayesNet ML algorithm for DDoS attack detection. The DM works at the application level of RSU/TA because here we use ML applications as an interface. We call this application level, the tier-3 of VAIDANSHH. Tier-3 of the DM starts immediately after receiving alerts from tier-2 of AAM. The DM uses a Waikato Environment for Knowledge Analysis for packet observation, validation, and classification. WEKA can analyze network traffic patterns and identify malicious and benign data packets. The DM uses a synthesized dataset generated from various simulation scenarios in NS3. The DM refines this dataset and converts it to a proper format using data pre-processing and feature reduction. Only relevant features are kept and useless features are ignored. After feature reduction, we apply the BayesNet ML algorithm to check which kinds of packets are malicious and which are benign. A model is trained to find the attack packets. After proper training, the DM tests the dataset and finds DDoS attack packets. We explain the full functionality of DM in the following subsections. Further sections provide details about the dataset generation, feature extraction and reduction, training, and testing.

**Table 3**
Selected attributes of synthetic dataset.

| Attribute | Description |
|---|---|
| Time | Indicates the time of packet sent |
| Source | IP address of the sender. |
| Destination | IP address of the receiver. |
| Protocol | Protocol used for communication. |
| Stream Index | It maps the source and destination port of two IP addresses. |
| Time to Live | Time for which a packet should remain in the network. |
| Time Since the Previous Frame | Time since the previous frame was transmitted in the network. |
| Arrival Time | Time when the packet arrived at its destination |
| Epoch Time | Time in seconds used by Wireshark for pcap files. |
| Info | Information about packets sent. |
| Class | Attribute for labeling the packet type. |

#### 3.3.1. Dataset generation

We either use publicly available datasets or create our dataset by collecting traffic data from the VANET. The choice of dataset for ML depends on the specific problem that we are trying to solve and the type of model that we are building. Some important factors to consider when choosing a dataset include relevance, size, quality, annotation (labeled dataset), etc. In this experimentation, we use our synthesized dataset based on the attack scenario discussed in subsection 3.1.3 i.e., attack methodology. It contains an adequate number of features suitable for attack detection. In NS3, we trace packets and monitor their movement using a flow monitor. Out of 12 nodes, we observe the target node by enabling its “pcap” option. The “pcap” option allows us to generate all packet-related information. We open this “pcap” file in Wireshark, select the relevant packet attributes, and save it as a .csv file. There are 28 attributes and 6,97,792 records in our synthetic dataset.

#### 3.3.2. Feature extraction and dimension reduction

The DM uses a supervised learning method called the CFS Subset Evaluator in combination with the best first search method in WEKA to select features. CFS evaluates the feature subset by calculating the correlation coefficient between each feature and the class attribute [42]. The evaluator identifies the most important attributes for classification. It may choose fewer attributes than what is specified in the evaluator’s settings. It then combines the selected features in a subset and calculates the correlation coefficient between the subset and the class attribute. The feature subset with the highest correlation coefficient is considered the best subset. Selected attributes are visible in Table 3.

Table 3 displays all the selected attributes that belong to the generated dataset of the proposed system. We do the feature reduction using nine different methods or algorithms. Among these nine methods, the first five are ranker methods, the following three are greedy step-wise search methods, and the last is the best first search method [43]. All nine approaches propose different attributes for use in the model, but we select only the top 10 for use in the model. It makes selected features more reliable in the classification process. With the help of this feature set, we obtain high accuracy, and other parameters also show better efficiency.

#### 3.3.3. Training and testing

We train and test our model using different data splits or proportions and select the split that yields the highest classification accuracy. We use a synthetic dataset that contains 6,97,792 instances. We divide

**Table 4**
Training and testing results with different data splits.

| BayesNet | 30% Training 70% Testing | 40% Training 60% Testing | 50% Training 50% Testing | 60% Training 40% Testing | 70% Training 30% Testing |
|---|---|---|---|---|---|
| Time to Build Model in Seconds | 3.44 | 2.84 | 2.83 | 2.66 | 2.94 |
| Time to Test Model in Seconds | 1.53 | 1.00 | 0.87 | 0.77 | 0.55 |
| Instances Tested | 488454 | 418675 | 348896 | 279117 | 209338 |
| Correctly Classified | 488355 | 418585 | 348828 | 279065 | 209298 |
| Correctly Classified %age | 99.9797 | 99.9785 | 99.9805 | 99.9814 | 99.9809 |
| In Correctly Classified | 99 | 90 | 68 | 52 | 40 |
| In Correctly Classified %age | 0.02 | 0.02 | 0.01 | 0.01 | 0.01 |
| TP Rate | 1 | 1 | 1 | 1 | 1 |
| FP Rate | 0 | 0 | 0 | 0 | 0 |
| Precision | 1 | 1 | 1 | 1 | 1 |
| Recall | 1 | 1 | 1 | 1 | 1 |
| F-Measure | 1 | 1 | 1 | 1 | 1 |

the dataset into different splits to see which data split is best. We evaluate our model based on the time taken, accuracy, TPR, FPR, precision, recall, and f-measure. After comparing the results, we find it suitable to choose the second case in which the 40% split is for training, and the remaining 60% is ideal for testing the model. Results show that a data split of $40\% - 60\%$ ratio gives high accuracy with high testing speed. Data split of $40\% - 60\%$ yields an accuracy rate of 99.9785%. It is an excellent accuracy rate obtained with a small error rate. The total time to train the model is 2.84 seconds and the time to test this model is 1 seconds. Results obtained from the training and testing are visible in Table 4.

### 3.3.4. DDoS detection

The BayesNet algorithm operates in two stages: structure learning and parameter learning. The first stage, structure learning, focuses on determining the dependencies between variables. The process begins with an empty network and repeatedly adds edges, evaluating the score of each new structure and selecting the one with the highest score. We continue this process until no further improvements can be made. The final network structure is the one with the highest score. The second stage, parameter learning, focuses on determining the probability distributions of each variable given its parents. Methods such as maximum likelihood estimation or Bayesian estimation can accomplish this. After learning the network structure and parameters, the network can make predictions about the probability of each variable given the values of the other variables.

It also uses the K2 searching method for ML detection. The K2 search method applies. K2 assigns scores to different network structures based on the probability of the data given the structure. It also assigns scores to a set of input variables in a random sequence. The order of variables greatly impacts the algorithm's performance. Incorrect ordering of variables can lead to incorrect learning of network topology. After determining the network structure, the DM selects a class in the estimate package to decide how to learn the probability tables. In this model, we are using the Simple Estimator Class (SEC) which uses the equation (1) to directly estimate conditional probabilities.

$$P(x_i = k | pa(x_i) = j) = \frac{N_{ijk} + N'_{ijk}}{N_{ij} + N'_{ij}}. \quad (1)$$

In Equation (1), $N'_{ijk}$, i.e., the alpha parameter is set at the default value of 0.5. It is set to the default value of 0.5 to balance the trade-off between over-fitting and under-fitting. The default value of 0.5 provides a good starting point for most applications and can be fine-tuned later. The detection model applies filters like replace-missing-values, string-to-nominal, and remove-percentage-split. We use ten characteristics and a total of 418675 instances. VAIDANSHH takes 2.84 seconds to build the training set and 1 seconds to test the detection functionality. Once the DM checks all the instances and finds malicious packets, it discards all those malicious packets. Only benign packets are processed. Our model VAIDANSHH efficiently detects the DDoS attack. The complete working of the VAIDANSHH model is defined in the following Algorithm 1.

**Algorithm 1** Algorithm of VAIDANSHH.

**Input:** Hardware resource consumption parameters, Network flow metrics, and dataset.
**Output:** DDoS Attack Classification
**procedure** ALARM()
  $interval1 \leftarrow time1$;
  $interval2 \leftarrow time2$;
  $stTimeThr \leftarrow time3$;
  $stTimeComp \leftarrow time4$;
  **procedure** TIMER(interval2, stTimeThr)
    $thr = calculateThreshold()$;
    $return(thr)$;
  **endprocedure**
  **procedure** CALCULATETHRESHOLD()
    *PythonScript();*
  **endprocedure**
  **procedure** TIMER(interval, stTimeComp)
    $comp = calculateParameter()$;
    $return(comp)$;
  **endprocedure**
  **procedure** CALCULATEPARAMETER()
    $PythonScript()$;
  **endprocedure**
  **if** $comp \geq thr$ **then**
    raise alarm;
    DETECT();
  **else**
    *process packets;*
    *continue();*
  **end if**
**endprocedure**
**procedure** DETECT()
  *pre-process data;*
  *select features;*
  *select ML algorithm;*
  *train the ML model;*
  *use dataset;*
  **if** $Packet = Malicious$ **then**
    *sinkhole malicious packets;*
    *update log;*
  **else**
    *process packet;*
  **end if**
**endprocedure**

Our model divides the complete process into procedures of $alarm()$ and $detect()$ that relate to our work. Section 3.2.2 and Section 3.2.2 define the working of the AAM. In the algorithm, procedure alarm() defines the functionality of the AAM. Procedure alarm() uses two variables to define the time interval when the system calculates the threshold and

comparative values and two variables to define the starting time of the procedures. A timer procedure starts at the specific start time and repeats after the predefined time interval. The timer procedure calls the CalculateThreshold() and CalculateParameter() procedures which run a Python script to calculate and return threshold and parameter values. The system then compares threshold and parameter values. If the comparative value becomes higher than the threshold value, the alarm() procedure sends a message to the RSU controllers or network security administrators. Network security administrators can go for immediate action or further diagnosis.

**Real-life implications of VAIDANSHH**

- Enhanced Security: VAIDANSHH's multi-tiered security approach enhances security in diverse vehicular network environments.
- Detection of Diverse Threats: The system's capability to detect various security threats, including DDoS attacks like UDP flooding, is crucial for safeguarding communication.
- Applicability Across Environments: VAIDANSHH can be deployed in different vehicular settings, ensuring adaptability to the dynamic nature of real-world scenarios.
- Reduced False Positives: The proactive monitoring of the Adaptive Alarming Module contributes to a reduction in false positives, minimizing unnecessary disruptions.
- Scalability and Performance: Designed for scalability, VAIDANSHH is suitable for deployment in large-scale vehicular networks while maintaining optimal performance.
- Integration with Infrastructure: VAIDANSHH integrates seamlessly with existing RSUs and TMCs, ensuring compatibility with current vehicular infrastructure.

## 4. Results and discussion

In this section, we present the results of our proposed VAIDANSHH model for DDoS attack detection. They applied the proposed solution to the synthesized dataset and obtained the following results. This section has two subsections. The subsection 4.1 defines the metrics based on which we have evaluated the model. The second subsection 4.2 shows the results of all three tiers presented in tables and figures.

### 4.1. Performance metrics

There are several performance metrics that we use to evaluate the effectiveness of DDoS attack detection systems. Some of the common metrics include detection rate, false positive rate, false negative rate, throughput, latency, accuracy, and false alarm rate. We use separate metrics in the AAM and the DM.

#### 4.1.1. AAM metrics

We evaluate six metrics and test them through the AAM. These metrics are average packet loss ratio, average mean delay, number of flows, average received and sent bitrate, and jitter.

**Average packet loss ratio:** In network communication, all the packets do not reach their destination. Due to congestion and errors in the network, the network drops or loses many of these packets. While determining the performance of the network packet loss should be minimal. Packet Loss Ratio (PLR) is the ratio of lost packets to the sum of packets sent and appears in percentage. The following Equation (2) calculates the PLR for each flow.

$$PLR = (\frac{n^r - n^s}{n^s}) * 100, \tag{2}$$

where PLR stands for packet loss ratio, $n^r$ for the total number of packets received, $n^s$ for the total number of packets sent. The system calculates APLR, i.e., the average of all flows from the Equation (3).

$$APLR = \frac{(PLR_1 + PLR_2 + PLR_3 + ... + PLR_n)}{n}. \tag{3}$$

In the above Equation (3), PLR1 stands for the Packet Loss Ratio of flow 1, PLR2 for the Packet Loss Ratio of flow 2, PLRn stands for the Packet Loss Ratio of the last flow, and n is the number of the last flow. The packet loss ratio is also known as the packet drop ratio (PDR). Networks with higher APLR or APDR will have low performance and vice-versa.

**Average mean delay:** Delay is also a network performance parameter that defines the time taken by a bit travel from source to destination or from one endpoint to another endpoint. Higher delay indicates low performance and vice-versa. It is also known as transmission delay. Subtract the receiving time from the sending time to calculate the delay. Then combine all delays obtained from all the packets to get the total delay. Calculate the mean delay by dividing the total delay by the total packet count. Similarly, calculate Average Mean Delay (AMD) using the following Equation (4).

$$AMD = \frac{(MD_1 + MD_2 + ... + MD_n)}{n}. \tag{4}$$

In the above Equation (4), AMD stands for the average mean delay, MD1 stands for the mean delay of flow 1, MD2 for mean delay of flow 2, MDn stands for the mean delay of the last flow, and n is the number of the last flow.

**Number of flows:** Flow has a different meaning in different computing fields. Here flow represents a group of bits or streams generated through a session and moving between two nodes. When there is any DDoS attack in any network using Botnets, there is always more flow of data streams than usual. In the case of computer networks, there may be some cases in which the number of flows increases, but there is no attack. These scenarios represent flash crowd, a temporary load on a particular server that increases for some reason. But in the case of vehicular networks, increased flows of suspicious activity need further investigation.

**Average received and sent bitrate:** The bitrate refers to the amount of data that passes through a communication channel in a particular time unit. It is also known as the Data Transfer Rate. Its measurement unit is generally bits per second, but it may be in bytes or kilobytes per second. In case of a DDoS attack victim node receives large volumes of data and sends limited or no data. The node remains busy processing incoming packets and cannot send packets. In this model, we have used the average received bit rate and average sent bit rate to see their impact. The model calculates the average received bitrate using the following Equation (5).

$$ARB = \frac{(RBR_1 + RBR_2 + ... + RBR_n)}{n}. \tag{5}$$

In the above Equation (5) ARB stands for average received bitrate, RBR1 stands for the received bitrate of flow 1, RBR2 for the received bitrate of flow 2, and RBRn stands for the received bitrate of the last flow, and n is the number of the last flow.

$$RBR = \frac{(RBytes * 8)}{RDur}. \tag{6}$$

$$RDur = timeLastRxPkt - timeFirstRxPkt. \tag{7}$$

In a similar way average sent bitrate is calculated using the following Equation (8)

$$ASBR = \frac{(SBR_1 + SBR_2 + ... + SBRn)}{n}. \tag{8}$$

In the above formula, ASBR stands for the average sent bitrate, SBR1 stands for the sent bitrate of flow 1, SBR2 for the sent bitrate of flow 2 and SBRn stands for the sent bitrate of the last flow and $n$ is a number of the last flow.

$$SBR = \frac{(SBytes * 8)}{SDur} \tag{9}$$

and

$$SDur = timeLastTxPkt - timeFirstTxPkt, \tag{10}$$

where $SDur$ stands for the duration of sent packets which is obtained by deducting the time of the first packet from the time of the last packet.

**Jitter:** Packet jitter is also known as packet delay variation. When data packets move from one node to another, they take time to reach their destination. If all sent packets take the same time to reach their goal node, there is no jitter. If the time delay is different for packets, then this variance in time delay is jitter. It should be minimal from the performance point of view. Jitter is the average time difference between two consecutive packets. We calculate the time difference using the following Equation (11).

$$TDif_1 = TTFP - RTFP. \tag{11}$$

Where TTFP denotes the Transmit time of the first packet of a particular flow and RTFP Receiving time of the first packet of a particular flow.

$$Jitter = \frac{(TDif_1 + TDif_2 + ... + TDif_n)}{n}. \tag{12}$$

In our proposed system, we have used the average jitter generated by different flows. Therefore, we calculate it by combining all the jitter values and dividing them by the number of flows.

#### 4.1.2. DM metrics

We use accuracy, precision, recall, and F1 as metrics of the DM.

**Accuracy:** It is a value that defines the correctness of our model in predicting the correct group (Labels or classes) of packets. Datasets and algorithms both have equal importance while finding accuracy or correctness. Normalized datasets produce higher accuracy in comparison to non-normalized datasets. Similarly, various ML algorithms have different learning, training, and testing methods, so each produces a different result. In ML, we calculate accuracy using the following Equation (13):

$$Accuracy = \frac{TP + TN}{N}, \tag{13}$$

where the $TP$ is True Positive, $TN$ is True Negative and $N$ is Total number of samples.

**Precision:** Precision is the ratio of True Positives and Total predictions of Positives. While testing, there may be multiple classes or categories for which the system classifies instances, and each class ML algorithm produces one precision value. The Precision metric is useful when there is a need for correct prediction. It makes sure that our model is working fine while predicting any label.

$$Precision = \frac{TP}{TP + FP}. \tag{14}$$

**Recall:** Recall may be defined as the ratio of correct predictions and actual labels. It can be defined as follows:

$$Recall = \frac{TP}{TP + FN}. \tag{15}$$

In Equation (14) and Equation (15), TP, FP, and FN stand for true positive, false positive, and false negative, respectively.

**F1 Score:** If we want our model to have a balanced precision and recall score, we average them to get a single metric. Here comes the F1 score, the harmonic mean of recall and precision.

### 4.2. Evaluations

We obtain results after the implementation of this simulation and the proposed model. We can categorize the results into three subsections. In the subsection 4.2.1, we calculate intermediate results for tier-1 which is the hardware level. In this part, we consider CPU utilization and memory consumption as parameters. The subsection 4.2.2 shows the results of tier-2, i.e., the result of the communication channel. It includes parameters like average packet loss ratio, mean delay, sent bitrate, the average received bitrate, number of flows, and jitter. The subsection 4.2.3, shows the results in the context of attack detection.

**Table 5**
Results of alarming module from flow parameters.

| Flow parameter | Normal Traffic (Thresholds) | Normal Traffic with Attack | UDP Flood Only |
|---|---|---|---|
| Average Packet Loss Ratio (APLR) | 0.00 | 30.87 | 46.09 |
| Average Sent bitrate (ASB) in kbit/s | 0.31 | 12001.65 | 18006.16 |
| Average Received bitrate (ARB) in kbit/s | 0.31 | 12001.65 | 18006.16 |
| Average Mean delay (AMD) in milli seconds | 1.02 | 27.04 | 35.58 |
| Jitter in milli seconds | 0.00 | 541.26 | 415.62 |
| Number of Flow IDs generated | 5.00 | 12.00 | 18.00 |

These results are from the tier-3 i.e. application level which uses accuracy, precision, recall, and F1-score as main parameters.

#### 4.2.1. Results of tier-1

While analyzing the results, at first, we check the hardware resources. Tier 1 deals with hardware resources like memory consumption and CPU load. During the attack, the CPU load reaches its maximum of 100%. In real situations like the simulated attack, the system crashes, and communication stops. We observe an increased memory consumption. Memory consumption rises to 1.77 GB during the attack. It is the maximum available memory that gets exhausted during the attack.

#### 4.2.2. Results of tier-2

As discussed earlier, VAIDANSHH not only uses packet information but also uses channel parameters for attack identification. With the help of a Python script, it extracts these parameters. These flow parameters are the average packet loss ratio, average sent and received bitrate, mean delay, and flow IDs. These parameters appear in Table 5. The first column shows the result of benign traffic and indicates threshold values. A description of the results is available in Table 5.

From the results of the AAM, we infer the following points. The number of flow IDs has been increased in the attack scenarios. PLR increases significantly from 0% to 46%, which indicates abnormal behavior in the traffic. The average mean delay increases from $1ms$ to $35ms$, which is an indication of abnormal behavior in the traffic. The average sent bitrate increases significantly from almost $0.5kbit/s$ to $18007kbits/s$. Jitter is increased from 0 to 541.26 in the mixed scenario and 415.62 in the UDP flood attack. Detection accuracy is 99.9785%, which is very good, and the detection process works within a short time of 1 second. The true positive rate is 1, and the false positive rate is 0, which means malicious packets are perfectly identified by the model and dropped. Precision, recall, and F-Measure values of 1 indicate that the VAIDANSHH model is efficient in terms of these parameters.

#### 4.2.3. Results of tier-3

Various parameters used in this module are accuracy, precision, recall, and f1-score. Out of these factors, a significant concern is to accurately detect attacks. So, accuracy is the main parameter that is kept on top.

We show the accuracy results in Table 6 and show the confusion matrix of the model as per Table 7. It indicates that out of 276942 instances of the Mix Flood class, the model correctly classifies 276894 records, and misclassifies 48 instances. In the same way, out of a total of 2854 instances of a legitimate category, all are correctly classified, and no instances are incorrectly classified. In the case of the UDP flood, out of 138879 instances of the UDP Flood class, 138837 are correctly classified, and only 60 instances are incorrectly classified as instances belonging to the legitimate class.

**Table 6**
Detection results.

| Parameter | Value | Percentage |
|---|---|---|
| Correctly Classified Instances | 418585 | 99.9785 |
| Incorrectly Classified Instances | 90 | 0.0215 |
| Kappa statistic | 0.9995 | - |
| Mean absolute error | 0.0001 | - |
| Root mean squared error | 0.0119 | - |
| Relative absolute error | - | 0.0474 |
| Root relative squared error | - | 3.0688 |
| Total Number of Instances | 418675 | - |

**Table 7**
Confusion matrix.

| A | B | C | Classified as |
|---|---|---|---|
| 276894 | 48 | 0 | A = Mix Flood |
| 0 | 2854 | 0 | B = Normal |
| 0 | 42 | 138837 | C = UDP Flood |

Other results are defined in further sections and summarized in Table 8. We use three different classes in Table 8 to show the important parameters. In the first row, results belong to a mixed class where True Positive Rate is 1 i.e. 100%, False-Positive Rate is 0 i.e. 0%, and recall is 1 i.e. 100%. The Precision and F-Measure values are also 1. Similarly, parameters of the normal class and flooding class are also shown in the second and third rows. Finally, the table shows the average results in the last row. Results are very good in all three scenarios and there is not too much variation in results. It indicates that VAIDANSHH is a good model for attack detection.

## 5. Comparative analysis

In this section, we compare the results of VAIDANSHH with other ML algorithms applied to the Synthesized dataset and a public dataset. We compare the final results based on the parameters described in the previous section.

### 5.1. Naive Bayes family

The naive Bayes family includes multiple algorithm variations of its techniques. We apply these to the same dataset with different splits. The proposed model uses 40% of training and 60% of testing data with the BayesNet algorithm, and other algorithms consider the same partition. The result parameters in which BayesNet has performed better are correctly classified instances, percentage of correctly classified cases, and incorrectly classified instances. In addition to that percentage of incorrectly classified samples, mean absolute error, root mean squared error, relative absolute error, true positive rate, false-positive rate, precision, recall, f-measure, and MCC are also proved better. We show the results in Table 9 and Fig. 4.

From Fig. 4 it is clear that the BayesNet algorithm is providing the best results when compared to other algorithms in the NaiveBayes family.

### 5.2. Other classification algorithms

This part of the results shows the output of some other algorithms from the non-Bayes family. After testing the algorithms from the Bayes family, we tested some additional ML Classifiers. It includes J48, AdaBoostM1, Ensemble Selection, OneR, and Grading algorithm. Some of these algorithms have performed very well, and some have not. Although J48 and Ad-BoostM1 algorithms give a high accuracy rate, they take much time for training and do not yield all result values properly. Ensemble Selection takes a very long time for training and testing. Even after this long processing time, the results produced are not good in terms of accuracy. OneR and Grading algorithms are fast in training and testing, but their effects are not fruitful. These results are shown in Table 10 and Fig. 5.

Again from Fig. 5 it is clear that BayesNet is superior among other algorithms in terms of TP Rate, FP Rate, Precision, Recall, and F1-Measure.

### 5.3. Comparison with existing solutions

In this part of the results, we have compared our model VAIDANSHH with some novel solutions in a similar domain. We have compared these solutions based on different parameters. From the results point of view, we consider accuracy as the main parameter. Zang et al. (2021) propose an ML-based IDS that uses streaming engines to analyze, handle, and visualize massive data in VANET [31]. The accuracy of the solution is 95%. Another approach for VANET security is a hybrid data-driven model to identify known network breaches [32]. The accuracy of the solution is 96%. Goncalves et al. (2021) proposed an ML solution to provide an intelligent hierarchical security framework for VANET [33]. The accuracy of the solution is 96%. Soni et al. (2022) introduce a system to secure V-RSU communication from blackhole and wormhole attacks in VANETs [34]. The accuracy of the solution is 89%. Malik et al. (2022) suggest another method for the mitigation (detection and prevention) of blackhole attacks in VANET [35]. The accuracy of the solution is 95%. At last, we show the results of VAIDANSHH which has an accuracy of 99.9%, see Table 11.

## 6. Limitations and implementation challenges

While the suggested approach proves effective in identifying and mitigating DDoS attacks within VANETs, there exist certain constraints that necessitate attention in future research endeavors. Notably, the current solution confines its scope to DDoS attacks, overlooking other potential threat vectors, and lacks the integration of supplementary security measures aimed at bolstering VANET security. Moreover, the simulation outcomes are contingent on specific assumptions and scenarios, warranting further scrutiny through diverse testing environments for comprehensive validation.

An area for enhancement pertains to the standardization of datasets essential for evaluating and comparing detection and prevention systems. Establishing standardized datasets would facilitate academics in constructing and evaluating models and algorithms on an equitable basis, thereby enabling more accurate comparisons of system efficacy.

Another aspect warranting refinement is the dynamic nature of DDoS attacks, posing a challenge in crafting adaptive systems that can counteract novel attack tactics. The ever-changing strategies employed by attackers necessitate continuous efforts to discern and thwart distinctive attack patterns effectively.

Furthermore, the substantial network traffic volumes encountered in real-world scenarios impede the identification of attacks while endeavoring to maintain low false positive rates. False positives contribute to disruptions and unwarranted alarms by erroneously categorizing legitimate traffic as malicious.

Lastly, the intricate nature of communication network infrastructure complicates the development and deployment of efficient detection and prevention systems. Resource constraints may impede detection and prevention capabilities in certain circumstances, and the system must demonstrate compatibility with diverse network topologies.

In addition to these limitations, some practical implementation challenges need redressing before implementation.

Complexity: Managing a three-tier architecture can be complex, especially concerning the coordination and communication between different tiers. Integration and synchronization of components across tiers might be challenging, leading to complexities in system design and maintenance.

**Table 8**
Detailed accuracy by class.

| TPR | FPR | Precision | Recall | F-Measure | MCC | ROC Area | PRC Area | Class |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | Mixed Flood |
| 1 | 0 | 0.969 | 1 | 0.984 | 0.984 | 1 | 1 | Normal Only |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | Flooding Only |
| **Average results** | | | | | | | | |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | - |

**Table 9**
Results obtained from techniques of Naive Bayes family.

| N B Family at 40%-60% | Bayes Net | Naive Bayes | Multinomial | Simple | Updatable |
|---|---|---|---|---|---|
| Time to Build Model in Seconds | 2.84 | 0.97 | 0.08 | 0.16 | 0.68 |
| Time to Test Model in Seconds | 1.00 | 2.29 | 0.36 | 0.76 | 1.86 |
| Instances Tested | 418675 | 418675 | 418675 | 418675 | 418675 |
| Correctly Classified | 418585 | 414927 | 276942 | 405851 | 414927 |
| Correctly Classified %age | 99.97 | 99.10 | 66.14 | 96.93 | 99.10 |
| In Correctly Classified | 90 | 3748 | 141733 | 12824 | 3748 |
| In Correctly Classified %age | 0.02 | 0.89 | 33.85 | 3.06 | 0.89 |
| Mean Absolute Error | 0.00 | 0.01 | 0.30 | 0.02 | 0.01 |
| Root Mean Squared Error | 0.01 | 0.06 | 0.38 | 0.11 | 0.06 |
| Relative Absolute Error | 0.04 | 4.11 | 100 | 7.72 | 4.11 |
| MCC | 1 | 0.98 | 0.99 | 0.93 | 0.98 |
| ROC Area | 1 | 1 | 0.5 | 1 | 1 |
| PRC Area | 1 | 1 | 0.548 | 1 | 1 |

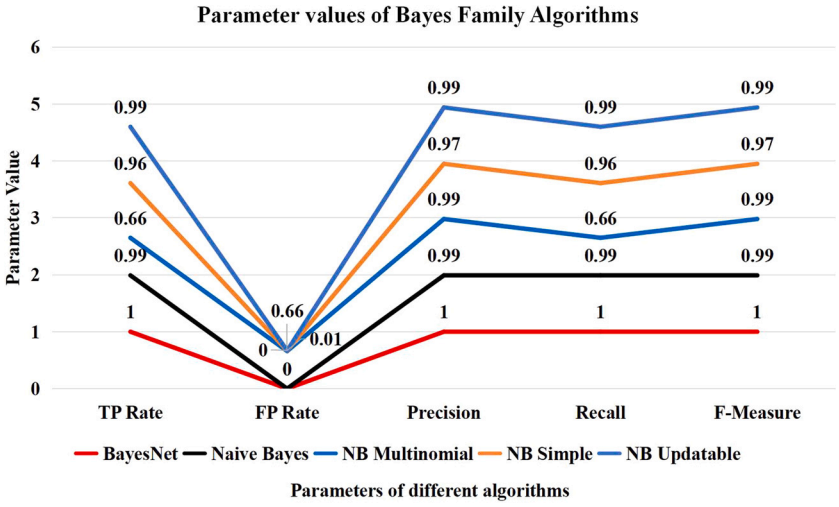



**Fig. 4.** Results obtained from group of Bayes algorithms.

**Table 10**
Results obtained from other algorithms.

| BayesNet and other algorithms | BayesNet | J48 | AdaBoost | Ensemble | OneR | Grading |
|---|---|---|---|---|---|---|
| Time to Build Model in Seconds | 2.84 | 6.23 | 16.30 | 294.20 | 1.14 | 5.66 |
| Time to Test Model in Seconds | 1.00 | 0.49 | 0.50 | 206.52 | 0.47 | 0.77 |
| Instances Tested | 418675 | 418675 | 418675 | 418675 | 418675 | 418675 |
| Correctly Classified | 418585 | 485112 | 415817 | 276807 | 277740 | 276942 |
| Correctly Classified %age | 99.97 | 99.31 | 99.31 | 66.11 | 66.33 | 66.14 |
| In Correctly Classified | 90 | 3342 | 2858 | 141868 | 140935 | 141733 |
| In Correctly Classified %age | 0.02 | 0.68 | 0.68 | 33.88 | 33.66 | 33.85 |
| Mean Absolute Error | 0.00 | 0.00 | 0.00 | 0.30 | 0.22 | 0.22 |
| Root Mean Squared Error | 0.01 | 0.03 | 0.03 | 0.38 | 0.47 | 0.47 |
| Relative Absolute Error | 0.04 | 0.94 | 0.94 | 100.13 | 74.58 | 74.00 |
| MCC | 1 | 1 | 0.99 | 0.99 | 0.04 | 0.99 |
| ROC Area | 1 | 1 | 1 | 0.50 | 0.50 | 0.50 |
| PRC Area | 1 | 1 | 1 | 0.54 | 0.55 | 0.54 |

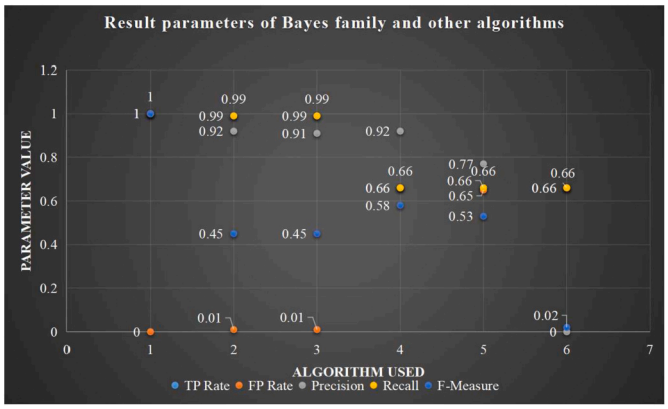



**Fig. 5.** Results obtained from other algorithms.

**Table 11**
Comparison of VAIDANSHH with existing models.

| Reference | Methodology | Detection type | Accuracy claims |
|---|---|---|---|
| Zang et al. (2021) [31] | Random Forest | Anomaly-based | 95% |
| Bangui et al. (2021) [32] | Random Forest and K-means | Anomaly & Signature-based | 96% |
| Goncalves et al. (2021) [33] | RF, J48, and MLP | Anomaly-based | 96% |
| Soni et al. (2022) [34] | Particle Swarm Optimization Approach | Behavior-based | 89% |
| Malik et al. (2022) [35] | Detection of Black-hole attack | Anomaly-based | 95% |
| Proposed VAIDANSHH | BayesNet | Anomaly-based | 99.9% |

Resource Intensiveness: Implementing and managing three tiers simultaneously can be resource-intensive in terms of computational power, memory, and network bandwidth. This could be a concern, especially in resource-constrained environments, where allocating resources to multiple tiers might affect the overall system performance.

Increased Latency: The processing involved in multiple tiers can introduce latency in the system. As data passes through each tier for analysis and security checks, the response time might increase. In time-sensitive applications like VANETs, increased latency can be a significant drawback.

Maintenance Challenges: Maintenance and updates can be more challenging in a multi-tier system. Ensuring that security protocols, thresholds, and algorithms are consistently updated across all tiers can be a cumbersome task, especially when dealing with evolving security threats and technologies.

Increased Development Time: Designing, developing, and testing a multi-tier system can be time-consuming. Coordinating the efforts of different teams or developers working on each tier, ensuring compatibility, and conducting comprehensive testing at each level can significantly increase the development time.

## 7. Conclusion and future work

In this proposed model, DDoS attack detection is based on the selection of parameters. The AAM can indicate suspicious activity in the network. Various parameters like jitter, delay, and packet loss ratio are good indicators of suspicious activities. The DM also efficiently detects the attack with an average accuracy of 99.9785 with precision, recall, and F1-Score of 1 each. BayesNet algorithm performs very well. The weighted TPR is 1, and the weighted false positive rate is 0, which means all packets are ideally classified, and legitimate users get all the services. There are the following observations about the proposed model. VAIDANSHH model is suitable for attack detection with a high accuracy of almost 100%. TP Rate, FP rate, Precision, and recall values of 1 and 0 indicate that the proposed model is also suitable in terms of other parameters. The synthesized dataset used in the model has the appropriate number of parameters and records. The dataset presents a real representation of the vehicular networks. Other researchers working on intrusion detection in VANETs may also use this solution for understanding DDoS attack nature. Concentration is required for attack prevention to avoid extra processing overheads. Flow parameters can identify suspicious activities and provide good results.

In future research, we will concentrate on developing a system that can detect and respond to attacks in an early stage. Our focus will be on identifying the parameters that are effective in preventing attacks. Implementing preventative measures can significantly reduce resource consumption and enhance the overall performance of the intrusion detection system. In the future, we will apply this model in a realistic environment to test its efficiency in a real vehicular environment.

In addition to this, there will be a focus on developing NIDS that dynamically adapts to evolving adversarial tactics, ensuring continuous learning and updating to counter new intrusion techniques effectively. Secondly, the integration of Explainable AI (XAI) techniques in NIDS will be explored to enhance transparency and interpretability, enabling cybersecurity analysts to understand and trust the decision-making processes. Another avenue for future research involves investigating mechanisms for NIDS to facilitate cross-domain collaboration and information sharing, improving threat detection through enhanced inter-organizational communication. Lastly, there is a call for the development of quantum-resistant NIDS, anticipating the cryptographic challenges posed by quantum computing advancements and ensuring the continued security of networked systems. These future directions collectively contribute to the ongoing evolution and innovation in the field of NIDS.

## CRediT authorship contribution statement

**Amandeep Verma:** Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing. **Rahul Saha:** Conceptualization, Investigation, Methodology, Software, Supervision, Validation, Writing – original draft, Writing – review & editing. **Gulshan Kumar:** Conceptualization, Software, Supervision, Writing – original draft, Writing – review & editing. **Mauro Conti:** Conceptualization, Data curation, Methodology, Supervision, Writing – review & editing. **Joel J.P.C. Rodrigues:** Conceptualization, Supervision, Validation.

## Declaration of competing interest

On behalf of all the authors, I declare that we do not have any 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.

## Acknowledgement

This work is partially funded by Brazilian National Council for Scientific and Technological Development - CNPq, via Grant No. 306607/2023-9.

## References

[1] Rahul Saha, Gulshan Kumar, Mritunjay Kumar Rai, Reji Thomas, Se-Jung Lim, Privacy ensured *e*-healthcare for fog-enhanced IoT-based applications, IEEE Access 7 (2019) 44536–44543.

[2] Ziyang Chen, Panlong Yang, Jie Xiong, Yuanhao Feng, Xiang-Yang Li, Tagray: contactless sensing and tracking of mobile objects using cots RFID devices, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, IEEE, 2020, pp. 307–316.
[3] A. Verma, R. Saha, G. Kumar, T.H. Kim, The security perspectives of vehicular networks: a taxonomical analysis of attacks and solutions, Appl. Sci. 11 (10) (2021) 4682.
[4] Gulshan Kumar, Rahul Saha, Mritunjay Kumar Rai, Tai-Hoon Kim, Multidimensional security provision for secure communication in vehicular ad hoc networks using hierarchical structure and end-to-end authentication, IEEE Access 6 (2018) 46558–46567.
[5] G. Kumar, R. Saha, M.K. Rai, W.J. Buchanan, R. Thomas, G. Geetha, J.J. Rodrigues, A privacy-preserving secure framework for electric vehicles in IoT using matching market and signcryption, IEEE Trans. Veh. Technol. 69 (7) (2020) 7707–7722.
[6] Ashish Kumar, Rahul Saha, Mamoun Alazab, Gulshan Kumar, A lightweight signcryption method for perception layer in Internet-of-Things, J. Inf. Secur. Appl. 55 (2020) 102662.
[7] Guangyu Li, Qiang Sun, Lila Boukhatem, Jinsong Wu, Jian Yang, Intelligent vehicle-to-vehicle charging navigation for mobile electric vehicles via VANET-based communication, IEEE Access 7 (2019) 170888–170906.
[8] Youssef Inedjaren, Mohamed Maachaoui, Besma Zeddini, Jean-Pierre Barbot, Blockchain-based distributed management system for trust in VANET, Veh. Commun. 30 (2021) 100350.
[9] Bashar Ahmad Khalaf, Salama A. Mostafa, Aida Mustapha, Mazin Abed Mohammed, Moamin A. Mahmoud, Bander Ali Saleh Al-Rimy, Shukor Abd Razak, Mohamed Elhoseny, Adam Marks, An adaptive protection of flooding attacks model for complex network environments, Secur. Commun. Netw. (2021) 2021.
[10] Deepak Kshirsagar, Sandeep Kumar, A feature reduction based reflected and exploited DDoS attacks detection system, J. Ambient Intell. Humaniz. Comput. 13 (1) (2022) 393–405.
[11] Ila Naqvi, Alka Chaudhary, Ajay Rana, Intrusion detection in VANETs, in: 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), IEEE, 2021, pp. 1–5.
[12] H. Hasrouny, A.E. Samhat, C. Bassil, A. Laouiti, VANET security challenges and solutions: a survey, Veh. Commun. 7 (2017) 7–20.
[13] Wajdy Othman, Miao Fuyou, Kaiping Xue, Ammar Hawbani, Physically secure lightweight and privacy-preserving message authentication protocol for VANET in smart city, IEEE Trans. Veh. Technol. 70 (12) (2021) 12902–12917.
[14] Ganesh Reddy, A delay sensitive multi-path selection to prevent the rushing attack in VANET, in: 2021 5th International Conference on Information Systems and Computer Networks (ISCON), IEEE, 2021, pp. 1–7.
[15] Fatih Sakiz, Sevil Sen, A survey of attacks and detection mechanisms on intelligent transportation systems: VANETs and IoV, Ad Hoc Netw. 61 (2017) 33–50.
[16] Jyoti Grover, Nitesh Kumar Prajapati, Vijay Laxmi, Manoj Singh Gaur, Machine learning approach for multiple misbehavior detection in VANET, in: International Conference on Advances in Computing and Communications, Springer, Berlin, Heidelberg, 2011, pp. 644–653.
[17] Wenjia Li, Anupam Joshi, Tim Finin, SVM-CASE: an SVM-based context aware security framework for vehicular ad-hoc networks, in: 2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall), IEEE, 2015, pp. 1–5.
[18] Fuad A. Ghaleb, Anazida Zainal, Murad A. Rassam, Fathey Mohammed, An effective misbehavior detection model using artificial neural network for vehicular ad hoc network applications, in: 2017 IEEE Conference on Application, Information and Network Security (AINS), IEEE, 2017, pp. 13–18.
[19] Myeongsu Kim, Insun Jang, Sukjin Choo, Jungwoo Koo, Sangheon Pack, Collaborative security attack detection in software-defined vehicular networks, in: 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), IEEE, 2017, pp. 19–24.
[20] Yao Yu, Lei Guo, Ye Liu, Jian Zheng, Y.U.E. Zong, An efficient SDN-based DDoS attack detection and rapid response platform in vehicular networks, IEEE Access 6 (2018) 44570–44579.
[21] Dimitrios Karagiannis, Antonios Argyriou, Jamming attack detection in a pair of RF communicating vehicles using unsupervised machine learning, Veh. Commun. 13 (2018) 56–63.
[22] Junwei Liang, Jianyong Chen, Yingying Zhu, Richard Yu, A novel intrusion detection system for vehicular ad hoc networks (VANETs) based on differences of traffic flow and position, Appl. Soft Comput. 75 (2019) 712–727.
[23] Dimitrios Kosmanos, Apostolos Pappas, Leandros Maglaras, Sotiris Moschoyiannis, Francisco J. Aparicio-Navarro, Antonios Argyriou, Helge Janicke, A novel intrusion detection system against spoofing attacks in connected electric vehicles, Array 5 (2020) 100013.
[24] Jasleen Kaur, Tejpreet Singh, Kamlesh Lakhwani, An enhanced approach for attack detection in vanets using adaptive neuro-fuzzy system, in: 2019 International Conference on Automation, Computational and Technology Management (ICACTM), IEEE, 2019, pp. 191–197.
[25] Moayad Aloqaily, Safa Otoum, Ismaeel Al Ridhawi, Yaser Jararweh, An intrusion detection system for connected vehicles in smart cities, Ad Hoc Netw. 90 (2019) 101842.
[26] Raenu Kolandaisamy, Rafidah Md Noor, Muhammad Reza Zaba, Ismail Ahmedy, Indraah Kolandaisamy, Markov chain based ant colony approach for mitigating DDoS attacks using integrated vehicle mode analysis in VANET, in: 2019 IEEE 1st International Conference on Energy, Systems and Information Processing (ICESIP), IEEE, 2019, pp. 1–5.
[27] Praveensankar Manimaran, NDNIDS: an intrusion detection system for NDN based VANET, in: 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), IEEE, 2020, pp. 1–5.
[28] David A. Schmidt, Mohammad S. Khan, Brian T. Bennett, Spline-based intrusion detection for VANET utilizing knot flow classification, Internet Technol. Lett. 3 (3) (2020), e155.
[29] Kaushik Adhikary, Shashi Bhushan, Sunil Kumar, Kamlesh Dutta, Hybrid algorithm to detect DDoS attacks in VANETs, Wirel. Pers. Commun. 114 (4) (2020) 3613–3634.
[30] Nivedita Kadam, Raja Sekhar Krovi, Machine learning approach of hybrid KSVN algorithm to detect DDoS attack in VANET, Int. J. Adv. Comput. Sci. Appl. 12 (7) (2021).
[31] Mingyuan Zang, Ying Yan, Machine learning-based intrusion detection system for big data analytics in VANET, in: 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), IEEE, 2021, pp. 1–5.
[32] Hind Bangui, Mouzhi Ge, Barbora Buhnova, A hybrid data-driven model for intrusion detection in VANET, Proc. Comput. Sci. 184 (2021) 516–523.
[33] Fábio Gonçalves, Joaquim Macedo, Alexandre Santos, An intelligent hierarchical security framework for VANETs, Information 12 (11) (2021) 455.
[34] Gaurav Soni, Kamlesh Chandravanshi, Mahendra Ku Jhariya, Arjun Rajput, An IPS approach to secure V-RSU communication from blackhole and wormhole attacks in VANET, in: Contemporary Issues in Communication, Cloud and Big Data Analytics, Springer, Singapore, 2022, pp. 57–65.
[35] Abdul Malik, Muhammad Zahid Khan, Mohammad Faisal, Faheem Khan, Jung-Taek Seo, An efficient dynamic solution for the detection and prevention of black hole attack in VANETs, Sensors 22 (5) (2022) 1897.
[36] Muammer Türkoğlu, Hüseyin Polat, Cemal Koçak, Onur Polat, Recognition of DDoS attacks on SD-VANET based on combination of hyperparameter optimization and feature selection, Expert Syst. Appl. (2022) 117500.
[37] Laura García-García, Jose M. Jiménez, Miran Taha Abdullah Abdullah, Jaime Lloret, Wireless technologies for IoT in smart cities, Netw. Protoc. Algorithms 10 (1) (2018) 23–64.
[38] Christos Gkountis, Miran Taha, Jaime Lloret, Georgios Kambourakis, Lightweight algorithm for protecting SDN controller against DDoS attacks, in: 2017 10th IFIP Wireless and Mobile Networking Conference (WMNC), IEEE, 2017, pp. 1–6.
[39] Miran Taha, Ali Aree, Smart algorithm in wireless networks for video streaming based on adaptive quantization, Concurr. Comput., Pract. Exp. 35 (9) (2023) e7633.
[40] Miran Taha, Ali Aree, Redirection and protocol mechanisms in content delivery network-edge servers for adaptive video streaming, Appl. Sci. 13 (9) (2023) 5386.
[41] Wong Yan Shen, Selvakumar Manickam, Mahmood A. Al-Shareeda, Review of advanced monitoring mechanisms in Peer-to-Peer (P2P) botnets, arXiv preprint, arXiv:2207.12936, 2022.
[42] A.R.A. Yusof, N.I. Udzir, A. Selamat, H. Hamdan, M.T. Abdullah, Adaptive feature selection for denial of services (DoS) attack, in: 2017 IEEE Conference on Application, Information and Network Security (AINS), IEEE, 2017, November, pp. 81–84.
[43] Megha Aggarwal Amrita, Performance analysis of different feature selection methods in intrusion detection, Int. J. Sci. Technol. Res. 2 (6) (2013).