APPLIED ARTIFICIAL INTELLIGENCE
2025, VOL. 39, NO. 1, e2538453 (27 pages)
https://doi.org/10.1080/08839514.2025.2538453

Context-Aware Intrusion Detection in Vehicular Communication Networks: Enhanced Attack Modeling and Dataset

Muhammad Danish Khan [a], Vinh-Thong Ta [b], Husnain Rafiq [a], and Nonso Nnamoko [a]

[a]Department of Computer Science Edge Hill University, Ormskirk, UK; [b]Centre for Defence and Security Management and Informatics, Cranfield University, Defence Academy of the UK

ABSTRACT

Vehicular Communication Networks (VCNs) are essential for autonomous vehicles and Intelligent Transportation Systems but face challenges in security vulnerabilities and data sparsity. Traditional attack models inadequately represent VCN dynamics, weakening threat detection, while existing datasets lack real-world mobility and spatiotemporal details. This study addresses these gaps by developing a comprehensive attack simulation framework, enhancing critical network attacks i.e. position spoofing, Sybil, and wormhole through realistic mobility patterns, positional dynamics, and temporal interactions. The resulting dataset contains legitimate and malicious instances: Spoofing (45,975 legitimate, 589 malicious), Wormhole (52,237 legitimate, 5,219 malicious), and Sybil (14,829 legitimate, 1,753 malicious). It includes essential vehicular-specific features such as mobility dynamics, inter-vehicle distances, and end-to-end communication patterns. For validation, machine learning algorithms, including Random Forest, K-Nearest Neighbors, and Logistic Regression were employed. Detection performance was evaluated using accuracy, precision, recall, and two F1-score variants (standard and macro). Results indicate high detection efficacy, with Random Forest achieving accuracy between 93.6% and 99.8% and F1-macro scores from 88.5% to 97.7%. Compared to previous studies lacking spatiotemporal considerations, our dataset's enhanced realism demonstrates significant potential in advancing data-driven anomaly detection and real-world threat mitigation in dynamic vehicular environments.

ARTICLE HISTORY
Received 15 May 2025
Revised 30 June 2025
Accepted 18 July 2025

Introduction

The rapid advancement of Intelligent Transportation Systems (ITS) has been transforming modern transportation by improving road safety, reducing congestion, and enhancing traffic efficiency. ITS are based on vehicular communication network (VCN), which facilitates real-time information exchange between vehicles and infrastructure, supporting coordinated vehicle operation, optimized routing, and effective traffic management (Contreras-Castillo, Zeadally, and Guerrero-Ibañez 2017). However, the increased complexity and interconnectivity of VCNs expose these networks to unique and sophisticated security threats.

Unlike traditional networks, where node positions remain static and connections are relatively stable, VCNs operate in highly dynamic environments characterized by continuous topology changes, varying vehicle mobility, and decentralized communication. These inherent characteristics pose significant challenges to existing security mechanisms. Furthermore, the operational context of VCNs introduces unique malicious behaviors that are more intricate due to the spatiotemporal relationships inherent in vehicular environments (Parno and Perrig 2005).

Despite the growing threat landscape, existing attack models for VCNs predominantly reflect the security paradigms of static or less-mobile network architectures (Verma et al. 2021), failing to capture the fundamentally different semantics of attacks in VCNs. The evolving nature of these threats necessitates a fundamental reevaluation of traditional attack models to ensure they accurately represent the dynamic

CONTACT Nonso Nnamoko Nnamokon@edgehill.ac.uk Department of Computer Science Edge Hill University, St. Helens Road, Ormskirk, United Kingdom, L39 4QP

nature of vehicular communication. Addressing these challenges requires innovative tools and methodologies capable of adapting to the highly mobile and complex threat landscape of VCNs.

Artificial Intelligence (AI) has demonstrated potential in data-driven anomaly detection through advanced pattern recognition, complex data analysis, and real-time adaptability (Rajapaksha et al. 2023). However, existing Machine Learning-based Intrusion Detection Systems (IDS) for VCNs (Alsarhan et al. 2021; Gad, Nashat, and Barkat 2021; Hanif et al. 2022) reveal critical limitations. One of the primary challenges lies in the datasets used for IDS training. Popular datasets such as Elkan (2000); Tavallaee et al. (2009); Sharafaldin, Lashkari, and Ghorbani et al. (2018); Van Der Heijden, Lukaseder, and Kargl (2018); Moustafa (2021) were originally designed for stationary or less-mobile network environments, such as the Internet of Things (IoT). The absence of vehicular-specific features in these datasets makes them inadequate for training ML models in a VCN context. Consequently, models trained solely on these datasets often lack the sophistication required to detect nuanced, context-driven threats in VCNs effectively.

To overcome these challenges, there is an urgent need for approaches to attack detection that incorporate the spatiotemporal and contextual complexities of vehicular environments, and comprehensive datasets that accurately reflect the unique dynamics of VCNs. Addressing these gaps is crucial to developing effective security solutions capable of meeting the evolving threat landscape in VCNs. To this end, our work makes the following contributions to VCN security:

(1) We redefine classical network attacks – position spoofing, Sybil, and wormhole – to reflect the dynamic nature of VCNs, introducing a dataset enriched with vehicular-specific features like speed, inter-vehicle distance, mobility dynamics, and end-to-end communication patterns. Specific details are outlined below:
   - **Sybil** modeling leverages positional anomalies using three spatiotemporal indicators: multi-position reporting, convergent position reporting, and temporal repetition anomaly.
   - **Position spoofing** modeling employs mobility-aware thresholds to validate position claims against realistic speed and movement constraints.
   - **Wormhole** modeling adapts packet leashing to VCNs through dynamic statistical baselines, ensuring anomaly detection aligns with multi-hop and high-mobility conditions.
(2) We set a new benchmark performance for vehicular network attack detection with machine learning (ML) by leveraging enriched features that are absent in prior work. Existing research is not directly comparable due to the lack of these features, making this contribution the new reference point for future studies.

Figure 1 illustrates the block diagram of our study design. The remainder of this paper is organized as follows: "Related Works" looks into related work, exploring existing datasets and methodologies in VCN security. "Attacks Modelling" thoroughly presents our approach to modeling crucial network attacks within the VCN context, "Data Generation" introduces the proposed dataset, detailing its comprehensive features and the labeling. In "Evaluation", we describe the evaluations including the simulation tools, feature extractions, and evaluation metrics. "Results and Discussion" discusses the results, evaluating the effectiveness of our approach through a series of machine learning experiments. Finally, "Conclusion and Future Work" concludes the paper with a summary and potential avenues for future research in the field of VCN security.

Related Works

AI-based intrusion detection has garnered significant attention in recent years, particularly in the context of enhancing network security. However, much of this research relies on existing benchmark datasets, many of which have been adapted for VCNs despite their limitations. This section begins with an examination of the widely used datasets in the field, followed by an exploration of the intrusion detection techniques applied specifically to VCNs.

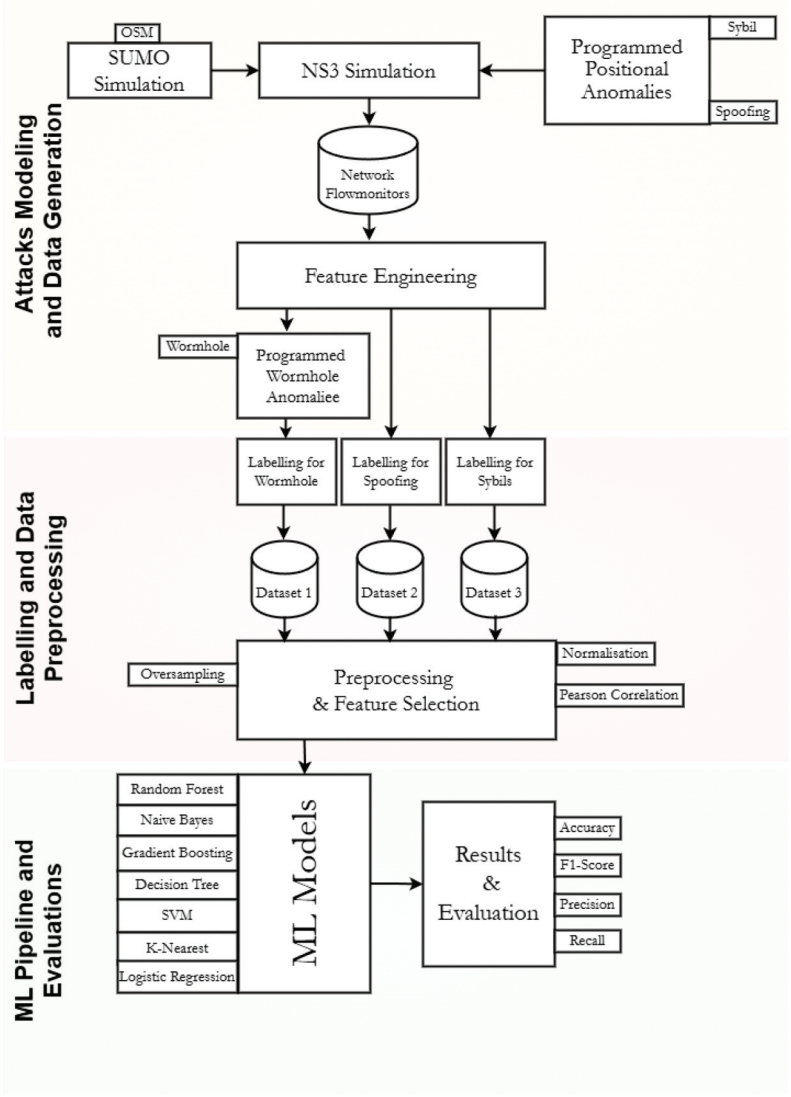

Figure 1. Study Design.

Existing Datasets

CICIDS-2017 dataset proposed by Sharafaldin, Lashkari, and Ghorbani et al. (2018) is a comprehensive resource for intrusion detection studies, offering an extensive compilation of benign and common attack network flows. It includes a wide range of attack scenarios, such as DoS, DDoS, brute force, SQL injection, infiltration, port scan, and botnet. A number of ML-based studies like Bangui, Ge, and Buhnova (2022); Karthiga et al. (2022); Korium et al. (2024); Rani and Sharma (2023); Yang, Moubayed, and Shami (2021); Yang and Shami (2022) have used CICIDS-2017 for intrusion detection in vehicular networks. Despite its widespread adoption, the CICIDS-2017 dataset primarily mirrors the characteristics and threat scenarios pertinent to traditional computer networks more than the dynamic vehicular networks, underscoring a significant gap in the dataset's applicability to vehicular contexts.

Van Der Heijden, Lukaseder, and Kargl (2018) presented a simulated dataset, VeReMi, making a substantial contribution to the field of VCN security. The dataset contains five different positional anomalies behaviors along with benign ones, setting a baseline dataset that reflects the complex nature of vehicular networks. However, while the VeReMi dataset is robust, it may not align perfectly with all aspects of intrusion detection requirements in VCNs. Its initial focus on position falsification might limit its direct applicability for more complex or varied types of vehicular network attacks not initially included.

ToN-IoT, dataset by Moustafa (2021), outlines a testbed architecture that aimed at dynamically generating and managing network communications across edge, fog, and cloud layers for IoT devices. The dataset includes a diverse range of data sources: telemetry datasets from IoT services and datasets from both Windows- and Linux-based systems, along with network traffic data. Capturing real-world normal and malicious scenarios. ToN-IoT has been adopted by different studies like Gad, Nashat, and Barkat (2021) for AI-based security in vehicular networks. Despite its extensive applications for IoT and network security, the dataset still lack to meet comprehensive requirement of VCNs.

Kamel et al. (2020) extended the scope of the VeReMI by including a broader range of attacks like Sybil, DoS, and data replay. The extension also incorporates a realistic sensor error model, which adds errors to the data fields like position, velocity, acceleration, and heading. Despite the addition of new attacks and error adjustments, attacks, while varied, are based on patterns that do not adequately mimic the adaptive strategies of sophisticated attackers.

The NSL-KDD dataset (Tavallaee et al. 2009) is an enhanced version of the widely used KDD Cup 99 dataset. With about 4.9 million single connection vectors and 41 different features, this dataset has been a primary benchmark for evaluating anomaly detection systems in network security. Despite its broad use in numerous IDS studies, including those in vehicular networks (Amaouche et al. 2024, 2023; Kasongo 2023), it faces several limitations when applied to the specific context of VCN security because the dataset's static network behavior model does not reflect the transient and variable nature of vehicular communication. Additionally, the dataset's categorized and labeled attack types are based on older and more generic network threats. Table 1 provides an overview of these datasets, highlighting their strengths, weaknesses, and reasons for unsuitability for VCNs.

Intrusion Detection Techniques

While significant progress has been made in the applications for intrusion detection, existing techniques often oversee the impact of spatiotemporal and mobility-related complexities of VCNs. This section reviews the key intrusion detection approaches applied in vehicular networks, focusing on their methodologies, strengths, and limitations in handling VCN-specific challenges. Belenko, Krundyshev, and Kalinin (2018) proposed a methodology for generating synthetic datasets tailored to intrusion detection in VANETs, addressing common network attacks such as DoS, DDoS, Sybil, and Wormhole attacks, among others. While their work provides a valuable foundation by incorporating a range of anomalies into the datasets, it primarily relies on traditional attack models, lacking considerations for the dynamic and evolving characteristics unique to VCNs.

Table 1. Comparison of Existing Datasets used for Vehicular Network Security.

Dataset	Strengths	Weaknesses	Reasons for unsuitability in VCNs
KDD-CUP 99 (Elkan 2000)	Widely used and benchmark dataset; includes various attack types.	High redundancy in data; biased toward frequent records; outdated.	Outdated nature and lack of vehicular-specific features like speeds and positions.
NSL-KDD (Tavallaee et al. 2009)	Improved over KDD-CUP 99; no duplicate records; balanced dataset.	Still inherits some unrealistic traffic characteristics.	Limited suitability due to focus on generic threats like DoS, Probe, U2R, and R2l; most of which are unrelated to VCNs.
CIC-IDS2017 (Sharafaldin et al. 2018)	Includes modern attack types like DDoS, Botnets; labeled dataset with diverse features.	Missing vehicle kinematics like mobility modeling	Not designed for vehicular environments, focused on static networks assuming stable network topology.
VeReMi (Van Der Heijden, Lukaseder, and Kargl 2018)	Tailored for misbehavior detection in VANETs; includes GPS and position data.	Limited attack diversity	The dataset focuses primarily on position falsification attacks, lacking holistic coverage of VCN aspects.
ToN-IoT (Moustafa 2021)	Comprehensive IoT dataset; heterogeneous data from telemetry and network traffic.	Not specific to vehicular environments;lacking vehicular communication standards like IEEE 802.11p (DSRC)	feature set focuses on system logs and telemetry rather than mobility-driven parameters like speeds, relative distance, and spatiotemporal consistency
VeReMi Extension (Kamel et al. 2020)	Improved attack diversity and sensor error modeling; realistic scenarios.	Non-Adaptive Attack Models	While vehicular attacks often exploit location/time correlations, the dataset still missing spatiotemporal context.

Several studies have addressed spoofing detection in vehicular networks, with varying approaches and considerations. Oligeri et al. (2022) utilized an experimental dataset with crowd-sourced GPS data to identify positional inconsistencies. Their technique relies on overlapping location reports from multiple vehicles to verify the authenticity of position claims. However, the absence of speed and mobility over time considerations significantly limits the applicability of their method, particularly in dense and high-mobility vehicular scenarios. Zhou and Han (2023) proposed a channel-spatial-temporal attention-based autoencoder network for detecting sensor spoofing attacks on autonomous vehicles. Their approach utilizes a memory-augmented spatial-attention block and PSE-Res2Net block-based encoder and decoder to capture multi-dimensional features from sensor data. While this method incorporates some spatial and temporal aspects, it primarily focuses on individual sensor data rather than the broader vehicular network context.

The study by Stepien and Poniszewska-Maranda (2021) introduces two algorithms aimed at detecting and mitigating Sybil attacks in vehicular networks by analyzing vehicle behaviors and interactions. Leveraging a footprint mechanism, these algorithms improve detection accuracy for false messages and Sybil identities, significantly reducing detection time. However, their approach remains limited by an absence of a comprehensive network-wide perspective, which is critical for addressing the broader dynamics of vehicular communication networks. Laouiti et al. (2022) utilized the VeReMi dataset for Sybil detection by employing the Adaboost classifier. Their method incorporated position plausibility checks, focusing on variations in acceleration between consecutive messages from sender nodes. However, their approach was limited to sender-specific data and lacked a comprehensive consideration of interaction histories for the transmitting nodes, potentially overlooking critical behavioral patterns. Zaidi et al. (2015) proposed a host-based Intrusion Detection System (IDS) for VANETs, leveraging statistical techniques to detect anomalies through congestion to identify rogue nodes involved in Sybil and false information attacks. Designed for distributed deployment at each host node within the vehicular network, the system demonstrated high detection rates for rogue nodes in both normal and accident scenarios, as evidenced by their simulation results.

The seminal work by Hu, Perrig, and Johnson (2003) introduces the concept of packet leashes as a defense mechanism against wormhole attacks in wireless ad hoc networks. The authors define two types of leashes: geographic and temporal, designed to restrict the transmission distance of packets and thereby detect illicit tunnels created by wormhole attackers. This approach highlights the vulnerabilities of ad hoc routing protocols, such as AODV and DSR, to wormholes and provides a structured framework to mitigate these threats. Kuma, Srilakshmi, and Reddy (2024) proposed a two-level fast and efficient distance-based external wormhole detection and prevention system. This method considered vehicle mobility, geographical location, and distance parameters to identify and isolate external wormhole attacker nodes. While it incorporated some contextual information, the approach did not fully leverage the complex interactions and patterns within the vehicular network.

A variety of ML techniques have been employed for Sybil, position spoofing, and wormhole attacks detection in vehicular networks. Elsadig et al. (2025) applied a lightweight machine learning framework on the VeReMi, NSL-KDD, and ToN-IoT datasets for Sybil attack detection in vehicular networks. Using Random Forest and Gradient Boosting, their model achieved 97.4% accuracy in detecting Sybil attacks and 96.2% for Wormhole attacks, indicating strong predictive capabilities for real-time vehicular intrusion detection systems. Suman et al. (2024) developed a deep learning approach validated on CICIDS and ToN-IoT datasets. Their model achieved an Attack Detection Rate (ADR) of 98.3% for Sybil and 95.7% for Position Spoofing attacks. They also emphasized using Decision Tree and SVM classifiers as baselines, which scored slightly lower but reinforced model interpretability. Azam et al. (2022) proposed a collaborative ensemble framework to detect Sybil attacks using K–Nearest Neighbors (KNN), Naïve Bayes, Decision Tree, SVM, and Logistic Regression combined via majority voting, achieving 95% accuracy on a simulated dataset.

Sharma and Jaekel (2021) used KNN, Decision Tree, Random Forest, and Naïve Bayes on the VeReMi dataset to detect position spoofing, with KNN delivering 99.2% precision and 98.8% recall. Another study by Ercan, Ayaida, and Messai (2021) focused on position spoofing attacks, applying a combination of ensemble ML classifiers in a supervised setting. Their system demonstrated high efficacy, recording an F1-score of 91% and accuracy around 92.5% on simulation datasets. Boualouache and Engel (2023) surveyed multiple ML-based misbehavior detection frameworks for connected vehicular systems. They reported that

classifiers like Gradient Boosting and Logistic Regression could successfully detect position falsification attempts with performance ranging between 88% and 93%, depending on data sparsity.

Wormhole detection has also benefited from ML intervention. A study by Singh et al. (2019) leveraged k-NN and Random Forest on SUMO+NS-3-generated flow-monitor logs. Their model achieved a detection accuracy of 94.2%, with a false positive rate below 3%, making it effective in fast-paced vehicular communication scenarios. Ben Rabah and Idoudi (2022) proposed an ML-based intrusion detection framework that included wormhole and GPS spoofing threats. Random Forest was identified as the top-performing model with 95.3% detection accuracy and robustness under varying vehicular densities.

Collectively, these studies establish that classic ML models such as Random Forest, SVM, Decision Tree, and KNN maintain strong performance profiles for detecting spatial misbehaviors in vehicular, particularly under Sybil, position spoofing, and wormhole conditions.

Attacks Modeling

As our first contribution, we redefine classical network attacks i.e., Sybil, Position Spoofing, and wormhole specifically to the VCNs. We present a multi-perspective analysis of positional anomalies, enabling the detection of both position spoofing and Sybil attacks. Furthermore, contextual considerations have been incorporated into wormhole detection mechanisms to reflect the variable conditions of vehicular operations.

Sybil

Unlike static networks, where Sybil attacks can be easily detected through multiple identities, detection in VCNs is significantly challenging due to the lack of persistent identity verification mechanisms. Sybil attack in vehicular contexts may involve the manipulation of position data, where attacker simulates presence at multiple locations to disrupt routing efficiency and spatial awareness, hence posing significant threats to systems relying on geographical routing (Sakiz and Sen 2017).

Conventional Sybil detection methodologies, which predominantly concentrate on identity verification, prove inadequate for VCNs due to their decentralized nature. This has been exemplified in protocols such as Wireless Access in Vehicular Environments (WAVE) (Jiang and Delgrossi 2008). Addressing this gap, we redefine Sybil through positional anomalies rather than simple cases of identity fraud. This perspective is operationalized through the introduction of three specific spatiotemporal behaviors, adapting it to the inherently dynamic nature of vehicular movements.

Simultaneous Multiposition Reporting (SMR): This behavior represents attack scenarios where a node falsely claims multiple geographic locations simultaneously.

Convergent Position Reports (CPR): Here, multiple nodes collude to report the same geographic location falsely, aiming to mislead the system about their actual distribution or density.

Temporal Position Frequency (TPF): This behavior focuses on the frequency of position reports over a short time interval, identifying cases where positions are reported with unnaturally high frequency.

Figure 2 illustrates a Sybil situation where $vehicle_3$ exhibits SMR, while $vehicle_1, vehicle_2, vehicle_4$, and $vehicle_5$ are exhibiting CPR. As detailed in Algorithm 1, we model both SMR and CPR, capturing the complex dynamics of these Sybil behaviors in vehicular networks. Furthermore, Temporal Repetition Anomaly is addressed separately, as outlined in Algorithm 2, providing a method to detect artificially repeated position reporting within a short time-frame.

Let T represent set of timestamps and N denote the set of nodes in the network. For any node $n \in N$, its position at a specific time $t \in T$ is given by:

$$\mathbf{p}_{n,t} = (x_{n,t}, y_{n,t})$$

where $x_{n,t}$ and $y_{n,t}$ are the x- and y-coordinates of node n at time t.

Simultaneous Multiposition Reports

To portray behavior-I, we model the node reporting multiple positions simultaneously as:

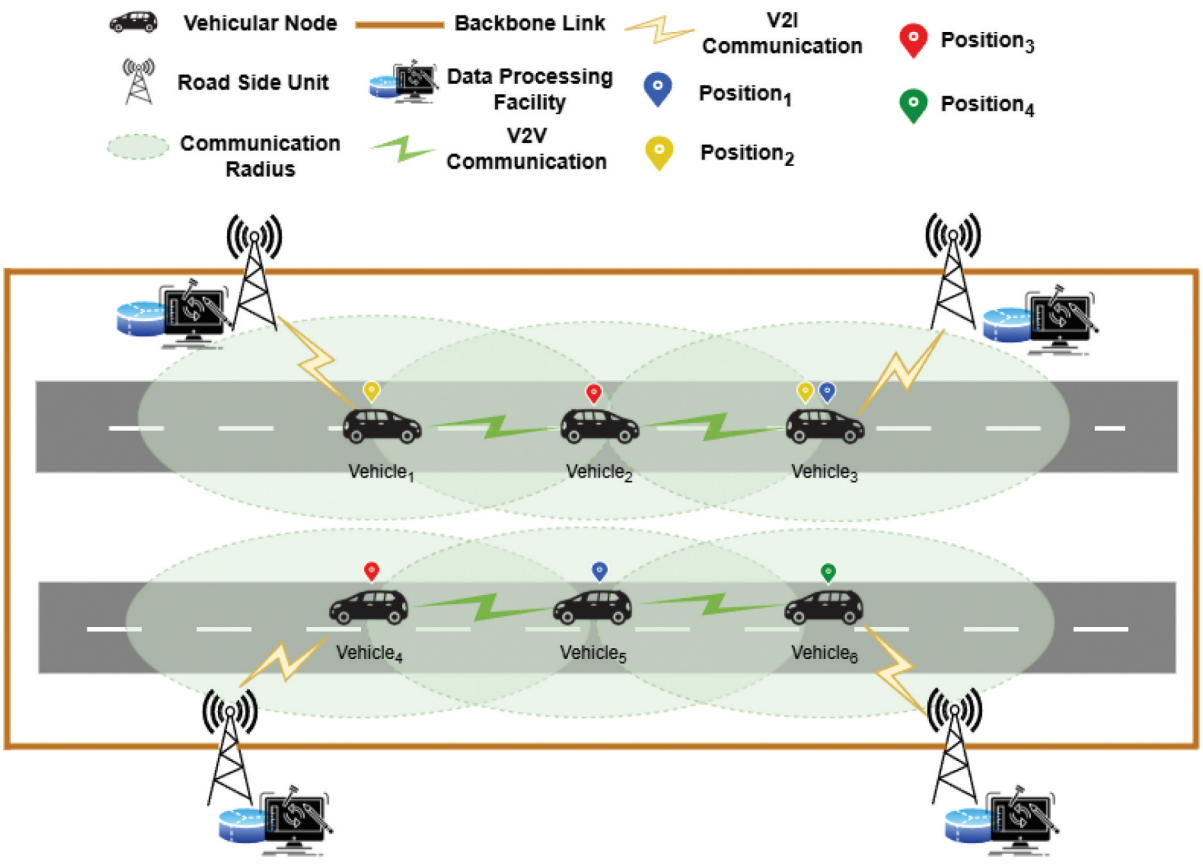

Figure 2. Positional Anomalies toward Sybil Behaviors.

$$\mathbf{p}_i(t) = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \quad \text{for the same } t$$

In this scenario, the node reports different locations across multiple communication flows simultaneously, effectively mimicking multiple identities within the network. This spatiotemporal inconsistency is characteristic of Sybil behavior, where a node claims more than one position at the same timestamp, as represented by:

$$P_{multiPos}(n,t) = \begin{cases} 1, & \text{if} |\{(x_i, y_i) : (x_i, y_i) \in \mathbf{P}(n,t), \\ & \quad (x_i, y_i) \neq (x_j, y_j), \forall i \neq j\}| > 1, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Where:

- n: a unique node
- t: a specific timestamp
- $\mathbf{P}(n,t)$: Set of all positions reported by node v at time t

An anomalous $P_{multiPos}$ highlights a potential marker of Sybil behavior.

In addition to multi-position anomalies, we modeled *displacement anomalies*, focusing on the displacement from the previously reported position (DFP). This metric calculates the distance between a node's initial reported position and its subsequent positions within the same timestamp:

$$\text{DFP}_{n,t} = \sqrt{(x_{n,t} - x'_{n,t})^2 + (y_{n,t} - y'_{n,t})^2} \tag{2}$$

Where:

- t: A specific timestamp
- $x_{n,t}$ and $y_{n,t}$: Coordinates of node n at time t
- $x'_{n,t}$ and $y'_{n,t}$: Coordinates of node n at the previously reported position at t

A significant non-zero $DFP_{n,t}$ within the same timestamp is a strong marker of anomalous behavior of a vehicular node, hence, this indicates a Sybil.

Convergent Position Reporting

This behavior exhibits an artificially high concentration of nodes at a single location, which refers to multiple node IDs simultaneously reporting identical or near-identical spatial coordinates, an unlikely event under normal mobility. This anomaly is quantified using the Position Repetition Count (PRC), which tracks the frequency of position reports within a short temporal window. For example, when a malicious entity duplicates positions across cloned identities, PRC spikes. This metric serves as an anomaly indicator in our detection framework. Formally, PRC at a position p and timestamp t is computed as:

$$\mathrm{PRC}_{\mathbf{p},t} = \sum_{n \in \mathcal{N}_t} \delta(n, \mathbf{p}, t), \tag{3}$$

where:

- $\mathbf{p}$: The position of interest.
- t: The timestamp being analyzed.
- $\mathcal{N}_t$: The set of nodes present at timestamp t.
- $\delta(n, \mathbf{p}, t)$: A binary indicator function defined as:

$$\delta(n, \mathbf{p}, t) = \begin{cases} 1, & \text{if } \mathbf{p}_{n,t} = \mathbf{p}, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

- $\mathbf{p}_{n,t}$: The position reported by the node n at timestamp t

In practice, $\mathrm{PRC}_{\mathbf{p},t} > 1$ suggests that more than one node has reported the same position $\mathbf{p}$ at time t, which can indicate potential Sybil behavior. This is based on the assumption that in legitimate vehicular mobility, no two distinct vehicles are expected to occupy the *exact* same coordinates at the *exact* same time, due to spatial constraints and motion dynamics.

To further clarify this anomaly, we analyze PRC patterns over the entire dataset *at every timestamp*. This is because our framework operates over discrete communication flows, where multiple transmissions occur concurrently. For each timestamp, we identify all unique positions $\{\mathbf{p}_1, \mathbf{p}_2, \ldots\}$ and evaluate their corresponding PRC scores. An unusually high PRC score at any position is flagged as suspicious.

This approach helps isolate Sybil attacks that manifest through cloned identities reporting the same position to manipulate neighboring nodes' perception. As illustrated in Figure 2, this behavior results in multiple nodes such as vehicle 1 and 3 reporting identical positions (e.g., position 2), creating a false impression of traffic density.

Temporal Position Frequency (TPF)

To replicate the behavior where a position is falsely reported repeatedly over a very short time interval, we modeled the *time-based position frequency* (TPF). This time-based aggregation provides insight into whether a particular position is artificially "over-represented" by multiple nodes, a potential indicator of Sybil activity. We made it sure to ensure fairness by not inflating the temporal frequency due to repeated reports by the same node. The TPF metric is defined as:

$$\mathrm{TPF}_{\mathbf{p},t} = \left| \{ n \in \mathcal{N}_{\mathbf{p},t} | \exists t' \in \mathcal{T}_t^{\varepsilon}, \delta(n, \mathbf{p}, t') = 1 \} \right|, \tag{5}$$

where:

- $\mathbf{p}$: The position of interest.
- t: The current timestamp being analyzed.
- $\mathcal{T}_t^{\varepsilon}$: The temporal window, defined as $\mathcal{T}_t^{\varepsilon} = [t - \varepsilon, t]$, where ε is the window size.
- $\mathcal{N}_{\mathbf{p},t}$: The set of nodes that report position $\mathbf{p}$ within $\mathcal{T}_t^{\varepsilon}$
- $\delta(n, \mathbf{p}, t')$: A binary indicator function defined as:

$$\delta(n, \mathbf{p}, t') = \begin{cases} 1, & \text{if } n \text{ reports } \mathbf{p} \text{ at } t', \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The temporal window $\mathcal{T}_t^{\varepsilon} = [t - \varepsilon, t]$ specifies the range of past timestamps used to compute $\text{TPF}_{\mathbf{p},t}$. Each node $n \in \mathcal{N}_{\mathbf{p},t}$ contributes at most once to the frequency count, ensuring that repeated reports of $\mathbf{p}$ by the same node within the window do not inflate the metric. If $\text{TPF}_{\mathbf{p},t} > 1$, all instances i with $t_i > t$ and $\mathbf{p}_i = \mathbf{p}$ are flagged as Sybil.

Position Spoofing

Position spoofing is a security threat in which a node deliberately reports a false location. In traditional networks, sudden or unexplained changes in position can typically be detected and flagged as potential anomalies (Khan, Mohsin, and Iqbal 2021). However in VCN, frequent legitimate updates in node positions complicate the detection of such malicious activities. The challenge intensifies in scenarios involving high-speed vehicular movement, where legitimate changes could mimic spoofing behaviors typically indicative of attacks in more static settings. To effectively tackle this, our approach redefines the detection mechanisms by integrating an understanding of vehicular dynamics. Position spoofing attacks have been modeled to incorporate critical factors such as speed, location, and time. By considering these elements, the detection framework is capable of assessing the plausibility of reported positions against the backdrop of each vehicle's trajectory and speed over time.

To detect position falsification, we leverage the historical movement data of each vehicle, including its previously recorded speeds and positions. This approach allows us to assess the physical plausibility of reported positions over time by examining if the displacement between two successive reports aligns with the vehicle's known mobility profile. This behavior is illustrated in Figure 3, where a node's reported position jumps implausibly far between two timestamps beyond what is physically possible.

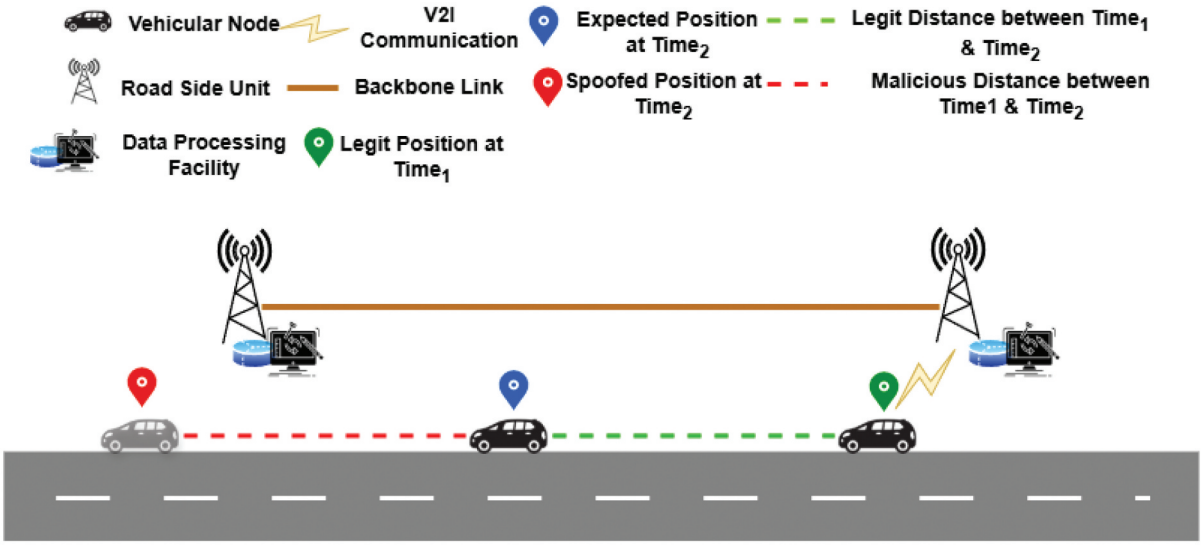

Figure 3. Position Spoofing.

For each node n, we maintain this history in a sliding window and use this to derive $s_{\max,n}$, the maximum observed speed over the entire known trajectory. This value reflects the highest speed a node could plausibly travel under normal conditions. We then compare the spatial distance between two consecutive reported positions $\mathbf{p}_{1,n}$ and $\mathbf{p}_{2,n}$ with the distance that would be physically possible within the time elapsed. The distance between these positions is calculated as the Euclidean distance:

$$\mathbf{d}(p_{1,n}, p_{2,n}) = \sqrt{(x_{2,n} - x_{1,n})^2 + (y_{2,n} - y_{1,n})^2} \tag{7}$$

The detection mechanism is formally defined as:

$$\text{isSpoofed} = \begin{cases} 1 & \text{if } \mathbf{d}\,(p_{1,n}, p_{2,n}) > s_{\max,n} \cdot \Delta t_n + \epsilon, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Where:

- $\mathbf{d}(p_{1,n}, p_{2,n})$: Euclidean distance between successive positions reported by node n.
- $s_{\max,n}$: The maximum speed observed for node n.
- Δt_n: Time interval between the two reported positions.
- ϵ: A small error tolerance threshold to accommodate random elements.

To ensure realistic spoofing detection, an error tolerance ε was incorporated in the displacement check to account for potential simulation noise and timing variability. Specifically, a tolerance factor of 10% was added to the maximum permissible distance ($d_{allowed}$) a node can travel between two known appearances. This modifies the detection threshold to:

$$d_{\text{allowed}} = s_{\max} \cdot \Delta t + \epsilon, \quad \text{where} \varepsilon = 0.1 \cdot s_{\max} \cdot \Delta t \tag{9}$$

This decision was based on empirical observations during our data exploration stage, where minor fluctuations in vehicle positioning (especially at low speeds or short Δ t intervals) introduced unavoidable spatial jitter up to 10% of allowed displacements. These anomalies were typically due to quantization effects in simulation coordinates and instantaneous transmission irregularities in high-mobility scenarios. The selected threshold therefore acts as a calibrated buffer to mitigate false positives, ensuring only meaningful mobility violations are flagged.

Sliding Window Maintenance: For a node i, let W_i be the window:

$$W_i = \{(x_{t-n}, y_{t-n}), (x_{t-n+1}, y_{t-n+1}), \ldots, (x_{t-1}, y_{t-1})\} \tag{10}$$

Where $n = W_{size} - 1$ (i.e., the window size minus one).

At each new instance t, the window is updated as:

$$W_i = W_i \cup \{(x_t, y_t)\} \tag{11}$$

If the window exceeds its size $|W_i| > W_{size}$ then the oldest position is removed from W_i. Table 6 summarizes the description of symbols used in Algorithm 3.

Wormhole

Wormhole attacks, wherein two malicious nodes create an illicit tunnel to expedite packet relay across a network, present distinct challenges in VCNs compared to more static environments. Wormholes can be identified by tracking routes that exhibit exceptionally high transmission speeds or significantly low hop counts (Hanif et al. 2022; Lai 2016), an approach feasible in static networks due to the stable physical distances between nodes. VCNs operate in a multi-hop communication manner. the varying distances and frequent topological changes introduce substantial fluctuations in routing metrics like hop counts and packet delivery times, making traditional detection methods based on static assumptions ineffective.

One promising technique for combating wormholes in wireless networks is packet leashing, which leverages geographic or temporal constraints to ensure that packets traverse realistic routes within expected boundaries (Hu, Perrig, and Johnson 2003). Geographic leashes use physical distance information to verify

transmission feasibility, while temporal leashes apply strict timing constraints to detect artificially shortened paths.

To address the complexities of wormhole detection in VCN, our approach introduces an enhanced mechanism that align with and extend the principles of packet leashing. By employing contextual spatio-temporal clustering, we dynamically group nodes based on average distance and rate of distance change, establishing adaptive baselines that reflect realistic vehicular mobility. These baselines facilitate the comparison of communication paths, hop counts, and packet speeds against realistic vehicular expectations, enabling the effective identification of deviations indicative of wormhole. Dynamic statistical analysis is employed to continuously update key metrics, such as effective packet speed and hop counts, capturing the real-time state of the network. The effective packet speed metric, in particular, offers a sophisticated means of evaluating the plausibility of packet transmission speeds relative to physical distances and expected travel times.

Figure 4 presents an abstract illustration of a wormhole tunneling scenario, where packets are relayed through a high-speed path, creating the illusion of faster transmission. To model wormhole attacks realistically within the VCN context, we computed the *effective packet speed* (**EPS**) for each instance in the dataset. **EPS** represents the rate at which packets traverse the network relative to the physical distance between the source and destination nodes. It is derived as a function of spatial, temporal, and communication characteristics and is calculated as:

$$EPS = \frac{avgDstnc}{\frac{flowDuration}{rxPkts}} \tag{12}$$

Where:

- *avgDstnc*: Average physical distance between source and destination nodes during the packet flow.
- *flowDuration*: Total duration of the flow.

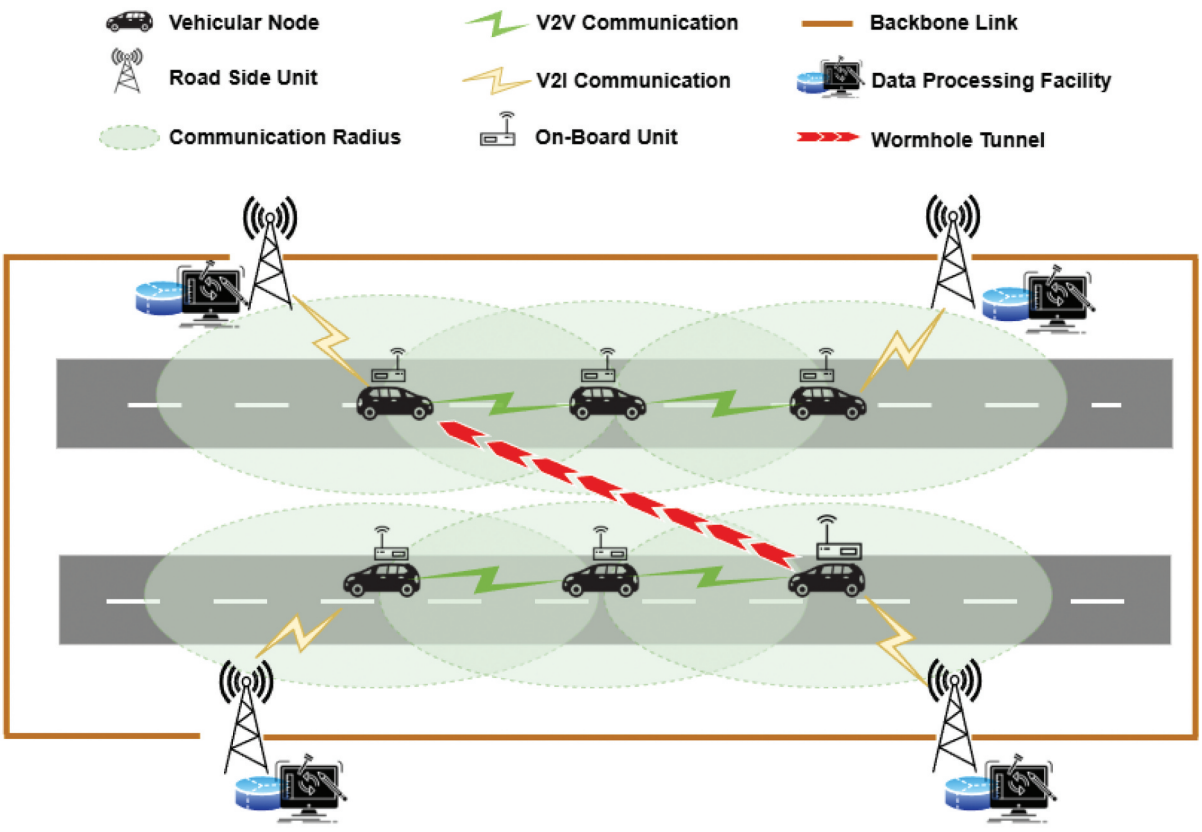

Figure 4. Wormhole Tunneling.

- *rxPkts*: Number of packets successfully received during the flow.

Wormhole behavior is characterized by an abnormally high effective packet speed, reflecting artificially shortened paths caused by malicious tunneling.

Clustering for Contextual Representation

We first employed clustering to establish context-sensitive baselines for network characteristics. Clustering was performed using two key features that capture mobility behaviors and spatial attributes:

(1) *Average Distance (*avgDstnc): Average distance between source and destination nodes.

(2) *Rate of Distance Change (*rateDstncChng): Rate of change of distance between communicating nodes over time.

Using K-means clustering (Ahmad and Dey 2007), flows were grouped into n clusters:

$$\mathcal{C}_i = \{x_j | \text{Cluster}(x_j) = i, \forall j\} \tag{13}$$

where $\mathcal{C}_i$ represents the i-th cluster, and x_j is a flow instance. Clustering preserves the mobility patterns and spatial relationships unique to vehicular environments, providing a context-aware mechanism for wormhole detection.

Analysis for Dynamic Thresholding. Within each cluster j, we computed key statistics to define *dynamic thresholds* for wormhole detection:

- Mean Effective Packet Speed (μ_{EPS_j}): Represents the average packet transmission speed for flows in the cluster calculated as:

$$\mu_{EPS_j} = \frac{1}{|\mathcal{C}_j|} \sum_{i \in \mathcal{C}_j} s_i \tag{14}$$

- Standard Deviation (σ_{EPS_j}): Captures the natural variation in packet speeds within the cluster as:

$$\sigma_{EPS_j} = \sqrt{\frac{1}{|\mathcal{C}_j|} \sum_{i \in \mathcal{C}_j} (EPS_i - \mu_{EPS_j})^2} \tag{15}$$

- Minimum hop count:

$$h_{j,\min} = \min_{i \in \mathcal{C}_j} (h_i) \tag{16}$$

employing these, for a cluster j the threshold τ for identifying anomalous effective packet speeds is defined as:

$$\tau_j = (\mu_{EPS_j} + \sigma_{EPS_j} + \mathcal{K} \cdot \sigma_{EPS_j}) \tag{17}$$

where:

- $\mathcal{K}$: Error factor to account for natural variations.

A flow f, that is an instance, within a cluster $\mathcal{C}_i$ is flagged as a wormhole instance if it satisfies all of the following conditions.

$$EPS_f > \tau_j, \quad h_f \leq h_{j,\min}, \tag{18}$$

Data Generation

As part of our next contribution, we generate data based on the redefined attack models in the previous section. To comprehensively model real-world vehicular communication scenarios, we utilized a dual-simulation approach combining traffic and network-level simulations. The Simulation of Urban MObility (SUMO) platform (Lopez et al. 2018) was employed for simulating on-road traffic dynamics. By integrating with OpenStreetMaps (OSM) (Haklay and Weber 2008).

To simulate communication among nodes, we used the Network Simulator (NS-3) (Riley and Henderson 2010), a widely adopted tool for evaluating network behaviors. Post-simulation, flow monitors from NS-3 were used to collect detailed communication flows, with each flow encapsulating statistics such as transmission metrics, node identities, and positional data between source and destination nodes. From these flows, we obtained a set of primary features and engineered additional derived features to provide a more nuanced analysis of communication behaviors. These features encompass spatiotemporal characteristics, mobility patterns, and communication statistics, forming the foundation for an in-depth investigation of normal and anomalous behaviors within VCNs.

Primary Features: The primary features in the dataset are directly derived from the data produced by our simulation framework. These features serve as the foundation for subsequent feature engineering. Each feature reflects key attribute of vehicular communication, such as transmission timing, node identities, positions, speeds, and packet statistics. These features are instrumental in understanding the core behavior of VCNs under both normal and anomalous conditions. Table 2 summarizes the description of these features.

Engineered Features: These features are derived from the base features in Table 2 to provide deeper insights into the behavior of nodes under varying conditions.

Flow Duration: Measures the total duration of a communication session, essential for analyzing longer or shorter than expected durations than can indicate disruptions or manipulations in communication patterns.

$$\text{Flow Duration} = \text{timeLastRx} - \text{txStrtTime} \tag{19}$$

Starting Distance: Provides the initial spatial distance between vehicles at the start of a communication session. It is one of the main factors contributing to spatiotemporal understanding in deployed VCN. It is calculated as:

$$\text{Starting Distance} = \sqrt{\begin{matrix}(srcStrt(x) - destStrt(x))^2 + \\ (srcStrt(y) - destStrt(y))^2\end{matrix}} \tag{20}$$

Final Distance: Measures the distance between vehicles at the end of a communication flow, important for understanding the distance dynamics over time. It is calculated as:

Table 2. Primary Features in the Dataset.

Feature	Description
txStrtTime	Start time of the transmission flow
srcID	Unique identifier of the source node
srcStrt(x)	Initial x-coordinate of the source node
srcStrt(y)	Initial y-coordinate of the source node
srcEnd(x)	Final x-coordinate of the source node
srcEnd(y)	Final y-coordinate of the source node
destID	Unique identifier of the destination node
destStrt(x)	Initial x-coordinate of the destination node
destStrt(y)	Initial y-coordinate of the destination node
destEnd(x)	Final x-coordinate of the destination node
destEnd(y)	Final y-coordinate of the destination node
avgSrcSpd	Average speed of the source node over the flow duration
avgDestSpd	Average speed of the destination node over the flow duration
timeLastRx	Timestamp when the last packet of the flow was received
txPkts	Total number of packets transmitted during the flow
rxPkts	Total number of packets received at the destination
rxBytes	Total number of bytes received at the destination
avgDelay	Average end-to-end delay per packet within the flow
hopCount	Number of intermediate nodes (hops) that packets traverse from source to destination

$$\text{Final Distance} = \sqrt{\begin{matrix}(srcEnd(x) - destEnd(x))^2 + \\ (srcEnd(y) - destEnd(y))^2\end{matrix}} \tag{21}$$

Throughput: Throughput is a direct measure of network performance, highlighting how effectively data is transmitted across VCNs, it is critical for aiding on further to the detection

$$\text{Throughput} = \frac{\text{rxBytes} \times 8}{\text{Flow Duration}} \tag{22}$$

Delivery Ratio: Shows the percentage of packets successfully received relative to those sent, offering insights into anomalies. Low delivery ratios can be indicative of packet loss or interception, common in attacks where packets are captured and resent through fabricated routes. It is calculated as:

$$\text{Delivery Ratio} = \frac{\text{rxPkts}}{\text{txPkts}} \times 100 \tag{23}$$

Average Distance: Averages the distance between vehicles throughout a communication session. This measure is crucial for summative representation varying distances between source and destination pairs.

$$\text{Average Distance} = \frac{\text{strtDstncTxRx} + \text{endDstncTxRx}}{2} \tag{24}$$

Rate of Distance Change: Measures how does the distance between communicating vehicles changes, providing a dynamic measure of how quickly network conditions vary. This measure is critical for establishing a contextual representation of dynamic network conditions in real-time.

$$\text{Rate of Distance Change} = \frac{endDstncTxRx - strtDstncTxRx}{FlowDuration} \tag{25}$$

V_{max}: V $_{max}$ for a node (source/destination) is the maximum speed observed historically up to the current record *i*. It is updated iteratively by comparing the node's current speed with its previous maximum. This feature was added for a historical representation of nodes speeds in spoofing detection.

$$V_{\max}^{(i)} = \max\left(V_{\max}^{(i-1)}\right) \tag{26}$$

Speed Deviation from V $_{max}$: This feature represents the deviation between a node's max speed and its current speed. This feature was engineered for spoofing detection.

$$\text{Deviation}^{(i)} = \begin{cases} \frac{\text{Distance}^{(i)}}{\Delta T^{(i)}} - V_{\max}^{(i)} / V_{\max}^{(i)}, & \text{if} \Delta T^{(i)} > 0 \\ \quad \text{and } V_{\max}^{(i)} > 0 & \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

Where *distance* is the Euclidean distance between the node's current and previous positions, and Δ T is the time difference between node's current and previous appearance.

Unlike VeReMi (Van Der Heijden, Lukaseder, and Kargl 2018) and other benchmark datasets which either ignore vehicular mobility features or simulate attacks using abstract packet manipulations, our dataset emphasizes spatiotemporal realism, node identity patterns, and context-aware transitions. As such, applying the same models across incompatible feature spaces would result in misleading interpretations. This highlights the critical gap we addressed: the need for accurate VCN-specific datasets to enable fair and realistic IDS benchmarking. Table 3 provides a structured comparison between our dataset and widely used IDS datasets, highlighting key differences in domain relevance, feature design, and contextual fidelity.

Data Labeling

Following attacks modeling and data generation, the next crucial step involved labeling the dataset for use in machine learning models for anomaly detection. To accurately label positional anomalies, i.e., Sybil, we

Table 3. Comparative analysis of the proposed dataset with existing IDS datasets.

	Proposed Dataset	Van Der Heijden et al. (2018)	Tavallaee et al. (2009)	Sharafaldin et al. (2018)	Moustafa (2021)
Domain	VCNs	VCNs	Traditional Networks	Enterprise/IoT Networks	IoT
Attack Types	Sybil, Position Spoofing, Wormhole	DoS, RSU Spoofing	Probe, DoS, U2R, R2L	DoS, Brute Force, Botnet	DDoS, Data Exfiltration
Spatiotemporal Features	✓	–	–	–	–
Mobility Realism	✓	✓	–	–	–
Custom Feature Engineering	✓	–	✓	✓	✓
Context-Aware Detection Feasibility	High	Limited	–	–	Moderate

Table 4. Symbols and Descriptions for Algorithm 1.

Symbol	Description
$\mathcal{D}$	input dataset
$\mathcal{D}_{labeled}$	Output labeled dataset
T	set of timestamps
t	individual timestamp
$\mathcal{N}$	Set of all nodes
$\mathcal{N}_t$	Set of all nodes at t
n	an individual node
$\mathcal{P}$	Set of all positions
t	A single time stamp
I_t	Subset of $\mathcal{D}$ consisting of all instances recorded at t.
$\mathcal{N}_t$	Set of nodes that have appeared at t.
$\mathbf{P}_{n,t}$	Set of positions reported by node n at t.
$\mathbf{p}$	a unique position
$\mathbf{p}_{n,t}$	Current position of node n at t
$\mathbf{p}'_{n,t}$	Previous position of node n at t
DFP	Distance from previous position
PRC	Position Repeat Count

implemented algorithm 1 and algorithm 2; for position spoofing, we implemented algorithm 3. We labeled the wormhole instances by implementing algorithm 4.

The detection algorithms developed in this work, targeting Sybil, position spoofing, and wormhole behaviors were designed as data labeling mechanisms, where they operate on low-level, timestamped simulation logs to identify attack behavior with high contextual fidelity, using VCN-specific spatiotemporal metrics, mobility thresholds, and communication inconsistencies. As such, these algorithms are closely tied to the engineered features in our dataset and were purpose-built to produce ground truth labels for training and evaluating machine learning models.

Labeling for Sybil

To ensure accurate labeling of Sybil instances in the dataset, two algorithms were designed to comprehensively capture positional and temporal anomalies. Table 4 describes the symbols used in Algorithm 1. The details of these algorithms are as follows:

Using SMR and CPR for Labeling Sybil Instances. Algorithm 1 focuses on identifying and labeling Sybil instances by analyzing positional anomalies. It evaluates both multi-position reporting and position repeatability within the same timestamp. *Lines 1–2*: The algorithm iterates over each timestamp t in the dataset T, extracting instances (I_t) corresponding to t. This isolates the data relevant to the current timestamp for focused analysis. *Lines 3–5*: For each node n active at timestamp t, the algorithm extracts the set of reported positions ($\mathbf{P}_{n,t}$) by the node. If a node reports more than one position ($|\mathbf{P}_{n,t}| > 1$), these positions are subjected to further evaluation. *Lines 6–15*: For every pair of reported positions by the node, the Distance From Previous (DFP) metric is calculated using Equation 2. If the DFP exceeds a defined threshold (> 0), the corresponding instances are flagged as Sybil by assigning $label_i \leftarrow 1$. *Lines 16–18*: The algorithm extracts the position ($\mathbf{p}_i$) from each instance at timestamp t. This step sets the stage for evaluating

Algorithm 1 Labelling Sybil Instances using SMR and CPR

Input: $\mathcal{D} = \{(T, \mathcal{N}, \mathcal{P})\}$ **Output:** $\mathcal{D}_{labeled}$: Dataset with Sybil labels

for each timestamp $t \in T$ **do**
  Extract $I_t \subset \mathcal{D}$ corresponding to t
  Extract $\mathcal{N}_t$
  for each node $n \in \mathcal{N}_t$ **do**
    Extract positions $\mathbf{P}_{n,t}$
    if $|\mathbf{P}_{n,t}| > 1$ **then**
      for each pair of positions $(\mathbf{p}_{n,t}, \mathbf{p'}_{n,t})$ **do**
        Calculate $\mathrm{DFP}_{n,t}$ using Equation 2
        if $\mathrm{DFP}_{n,t} > 0$ **then**
          For each instance $i \in I_t$ with $\mathbf{p}_i \in \mathbf{P}_{n,t}$:
            $label_i \leftarrow 1$
        end if
      end for
    end if
  end for
  for each instance $i \in I_t$ **do**
    Extract the position $\mathbf{p}_i$ from instance i
  end for
  for each position $\mathbf{p} \in \{\mathbf{p}_i \mid i \in I_t\}$ **do**
    Compute $\mathrm{PRC}_{\mathbf{p},t}$ using Equation 3
    if $\mathrm{PRC}_{\mathbf{p},t} > 1$ **then**
      For each instance $i \in I_t$ where $\mathbf{p}_i = \mathbf{p}$:
        $label_i \leftarrow 1$
    end if
  end for
end for

second behavior of position-based anomalies. *Lines 19–24*: For each position (**p**) reported in timestamp t, the Position Repeat Count (PRC) is computed using Equation 3. If the PRC value exceeds a threshold (>1), all instances reporting the position (**p**) are flagged as Sybil.

Sybil Labeling through TPF

Algorithm 2 extends the analysis by incorporating temporal anomalies. It evaluates the temporal repetition of positions across timestamps to detect Sybil behaviors. Table 5 describes the symbols used in Algorithm 2. *Lines 1–2*: The algorithm identifies all unique positions (**p**) in the dataset $\mathcal{D}$, laying the foundation for position-based temporal frequency analysis. *Lines 3–5*: For each unique position (**p**), the algorithm extracts the corresponding instances ($I_{\mathbf{p}}$) and their associated timestamps ($T_{\mathbf{p}}$). *Lines 6–9*: The Temporal Position Frequency ($TPF_{\mathbf{p},t}$) is computed for each timestamp t where the position (**p**) is reported. If $TPF_{\mathbf{p},t}$ exceeds the defined threshold (τ), the instances associated with the timestamp are labeled as Sybil ($label(i) \leftarrow 1$).

Algorithm 2 Sybil Labelling through TPF

Input: Dataset $\mathcal{D} = \{(T, \mathcal{N}, \mathcal{P})\}$
Output: $\mathcal{D}_{labeled}$

Step: Detect Temporal Repetition Frequency (TPF)
Capture all unique positions $\mathbf{p} \in \{\mathbf{p}_i \mid i \in \mathcal{D}\}$
for each position $\mathbf{p} \in \{\mathbf{p}_i \mid i \in \mathcal{D}\}$ **do**
  Extract all instances $I_{\mathbf{p}} \subset \mathcal{D}$ where $\mathbf{p}_i = \mathbf{p}$
  Capture timestamps $T_{\mathbf{p}} = \{t_i \mid i \in I_{\mathbf{p}}\}$
  for each timestamp $t \in T_{\mathbf{p}}$ **do**
    Compute $\mathrm{TPF}_{\mathbf{p},t}$ using equation 5:
    if $\mathrm{TPF}_{\mathbf{p},t} > 1$ **then**
      Label the current instance $i \in I_{\mathbf{p}}$ where $t_i = t$ as Sybil: $\mathrm{label}(i) \leftarrow 1$
    end if
  end for
end for

Table 5. Symbols and Descriptions for Algorithm 2.

Symbol	Description
T	Set of all timestamps in the dataset
$\mathcal{N}$	Set of all nodes in the dataset
$\mathcal{P}$	Set of all positions reported by nodes in the dataset
$\mathcal{D}$	Initial dataset consisting of nodes, their timestamps, and positions
$\mathcal{D}_{labeled}$	Labeled Dataset
$\mathbf{p}$	A particular position in the dataset
$I_{\mathbf{p}}$	Subset of $\mathcal{D}$ where position $\mathbf{p}$ is reported.
$T_{\mathbf{p}}$	Set of timestamps associated with the position $\mathbf{p}$.
TPF	Temporal Position Frequency

Labeling for Position Spoofing

Algorithm 3 identifies and labels position spoofing anomalies by leveraging positional and speed information of nodes over time.Table 6 describes the symbols used in Algorithm 3. *Line 1*: The algorithm initializes an empty sliding window W_n for each node to store recent positional and speed data. Additionally, $s^n_{\max}$ is initialized to track the maximum speed observed for each node. This setup allows the algorithm to maintain historical context for subsequent calculations. *Lines 2–3*: The algorithm processes each instance in the dataset, extracting information about the timestamp t_i, set of nodes $\mathcal{N}_i$, their positions $\mathcal{P}_i$, and speeds $\mathcal{S}_i$. *Lines 4–10 (Node-Level Analysis)*: For each node n_i in the current instance: if this node has no prior data in the sliding window (W_{n_i} is empty), it initializes the window with the current position and speed. The node is marked as legitimate (not spoofed), and the loop moves to the next iteration. *Lines 11–15 (Retrieving*

Algorithm 3 Position Spoofing Labelling

Input: Dataset $\mathcal{D} = \{(t_i, \mathcal{N}_i, \mathcal{P}_i, \mathcal{S}_i)\}_{i=1}^{N}$ **Parameters:** W_{size}, ϵ
Output: $\mathcal{D}_{labeled} = \{(t_i, n_i, \mathbf{p}_{n_i}, s_{n_i}, \text{isSpoofed}_{n_i})\}_{i=1}^{N}$

1: **Initialisation:** $W_n \leftarrow \emptyset, \quad s^n_{\max} \leftarrow 0 \quad \forall n \in \mathcal{N}$
2: **for** each instance $i \in \mathcal{D}$ **do**
3: Extract $(t_i, \mathcal{N}_i, \mathcal{P}_i, \mathcal{S}_i)$ from i
4: **for** each $n_i \in \mathcal{N}_i$ **do**
5: **if** $W_{n_i} = \emptyset$ **then**
6: $W_{n_i} \leftarrow \{(t_i, \mathbf{p}_{n_i}, s_{n_i})\}$
7: $s^{n_i}_{\max} \leftarrow s_{n_i}$ ▷ Set initial maximum speed
8: $\text{isSpoofed}_i \leftarrow 0$
9: **Continue**
10: **end if**
11: Retrieve $(t_{last}, \mathbf{p}_{last}, s_{last}) \in W_{n_i}$ such that $t_{last} < t_i$
12: **if** t_{last} does not exist **then**
13: $\text{isSpoofed}_i \leftarrow 0$
14: **Continue**
15: **end if**
16: Compute $\Delta t_{n_i} = t_i - t_{last}$
17: Compute d_{actual} using equation 7
18: Compute $d_{\text{allowed}} = s^{n_i}_{\max} \cdot \Delta t_{n_i} + \epsilon$
19: **if** $d_{\text{actual}} > d_{\text{allowed}}$ **then**
20: $\text{isSpoofed}_i \leftarrow 1$
21: **else**
22: $\text{isSpoofed}_i \leftarrow 0$
23: Update W_{n_i} using equation 11
24: **if** $|W_{n_i}| > W_{size}$ **then**
25: Remove the oldest entry from W_{n_i}
26: **end if**
27: Recompute $s^{n_i}_{\max}$:

$$s^{n_i}_{\max} \leftarrow \max(s \mid (t, \mathbf{p}, s) \in W_{n_i})$$

28: **end if**
29: **end for**
30: **end for**
31: **Output:** $\mathcal{D}_{labeled}$

Table 6. Symbols and descriptions for algorithm 3.

Symbol	Description
t_i	timestamp of the instance *i*
$\mathcal{N}_i$	set of nodes in *i*
n_i	individual node in *i*
$\mathcal{P}_i$	set of positions of nodes in instance *i*
$\mathbf{p}_{n_i}$	position of individual node *n* in instance *i*
$\mathcal{S}_i$	set of speeds of nodes in instance *i*
s_{n_i}	speed of individual node *n* in *i*
W_{size}	Sliding window size
ε	Error margin
$\mathcal{D}_{labeled}$	Labeled dataset
isSpoofed_i	Spoofing detection label for *i*

Historical Data): If the sliding window contains prior data for the node, the algorithm retrieves the most recent record. If no such record exists, the node is marked as legitimate, and the loop continues. *Lines 16–22 (Position Verification through speed)*: The algorithm computes:

- *Time Difference* (Δt_{n_i}): The time elapsed since the last recorded position of node n_i.
- *Actual Distance* (d_{actual}): The Euclidean distance between node's current and last positions.
- *Allowed Distance* (d_{allowed}): The maximum plausible distance a node can travel, calculated as the product of its historical maximum speed ($s_{\max}^{n_i}$) and Δt_{n_i}, with an additional error margin ε.

If d_{actual}exceedsd_{allowed}, the instance is flagged as spoofed, otherwise legitimate.

Lines 23–28 (Updating the Sliding Window): If no anomaly is detected, the current position and speed data are added to the sliding window, if the window exceeds its defined size (W_{size}), the oldest entry is removed to maintain the window's constraints. The historical maximum speed ($s_{\max}^{n_i}$) is recalculated based on the updated window, ensuring it reflects the most recent data.

Labeling for Wormholes

Algorithm 4, the wormhole labeling algorithm leverages metrics such as effective packet speed (EPS), clustering features, and cluster-specific statistics, the algorithm captures the dynamic mobility of VCNs and ensures accurate detection of wormhole anomalies. Table 7 describes the symbols used in Algorithm 4. *Lines*

Algorithm 4 Wormhole Labelling

Input: $\mathcal{D} = \{(t_i, \mathcal{N}_i, \mathcal{P}_i, duration_i, \mathcal{S}_i, d_i^{avg}, \delta d_i, r_i, h_i)\}_{i=1}^{N}$ **Output:** $\mathcal{D}_{labelled} = \{(t_i, \mathcal{N}_i, \mathcal{P}_i, duration_i, \mathcal{S}_i, d_i^{avg}, \delta d_i, r_i, \text{label}_i)\}_{i=1}^{N}$
Parameter: $\mathcal{K}$

for each $i \in \mathcal{D}$ **do**
  Calculate EPS_i using equation 12:
end for
Use clustering features d_i^{avg} and δd_i for all instances
Apply K-means clustering to group the instance into n_c clusters using equation 13,
Assign cluster label g to each i; $g_i = j$ where $i \in \mathcal{C}_j$
for each cluster $j \in \mathcal{C}$ **do**
  Compute cluster stats:
    i. μ_{EPS_j} using equation 14
    ii. σ_{EPS_j} using equation 15
    iii. τ_j using equation 17
end for
for each $i \in \mathcal{D}$ **do**
  Retrieve the cluster stats for i (based on g_i)
  Check if i meets conditions in equation 18:
  if all conditions are satisfied **then**
    $\text{label}_i \leftarrow 1$
  else
    $\text{label}_i \leftarrow 0$
  end if
end for
return $\mathcal{D}_{labelled}$

Table 7. Symbols and Descriptions for Algorithm 4.

Symbol	Description
$\mathcal{D}$	Dataset
N	Number of instances in $\mathcal{D}$
$\mathcal{C}$	Set of Clusters
n_c	Number of total Clusters
i	a single instance $\in \mathcal{D}$
t_i	timestamp of i
$\mathcal{N}_i$	Set of nodes in i
n_i	individual node in i
$\mathcal{P}_i$	Set of positions of nodes in instance i
$\mathbf{p}_{n_i}$	Position of node n in i
$duration_i$	Time-duration of the flow in i
$\mathcal{S}_i$	Set of speeds of nodes in i
s_{n_i}	speed of node n in i
d_i^{avg}	average distance between source and destination in i
δd_i	Rate of distance change between source and destination nodes in i
r_i	Number of received packets in i
h_i	Packets Hop count from source to destination in i
g_i	Cluster label of i
μ_{EPS_j}	Mean effective packet speed of cluster j
σ_{EPS_j}	Standard deviation of effective packet speeds for cluster j
τ_j	Wormhole speed threshold for cluster j
$\mathcal{K}$	Error factor
label_i	Wormhole label for i

1–3 (EPS Calculation): For each flow i in the dataset, *Effective Packet Speed (EPS)* is calculated at which packets travel between source and destination nodes. *Lines 4–6 (Contextual Clustering)*: The flows are grouped into clusters $\mathcal{C}$ using *K-means clustering* based on average distance between the source and destination nodes, and the rate of distance change between source and destination nodes, and each instance is assigned a cluster label (g_i) *Lines 6–9 (Cluster-Specific Statistics Computation)*: For each cluster j, the algorithm computes: average EPS of all flows within the cluster, the deviation of EPS values, and wormhole speed threshold that distinguishes legitimate flows from potential wormhole. *Lines 10–18 (Wormhole Detection and Labeling)*: For each instance flow i in the dataset, its corresponding cluster's statistics are obtained based on the instance's cluster label (g_i). The flow is evaluated against the wormhole detection conditions: i. if EPS_i exceeds the cluster threshold (τ_j); ii. if hop count of i is the cluster's minimum hop count ($h_{j,\min}$). If all conditions are satisfied, the flow is labeled as a wormhole ($label_i = 1$); otherwise, it is labeled as legitimate ($label_i = 0$). *Output*: The algorithm outputs a labeled dataset ($\mathcal{D}_{labeled}$), where each flow is annotated with a wormhole detected label ($label_i$).

Data Preprocessing

Data preprocessing was carried out in a structured manner to enhance the dataset's quality for machine learning-based anomaly detection. The preprocessing pipeline included noise injection, oversampling, and feature scaling, each serving a crucial role in improving the richness of the detection models. To introduce variability and simulate real-world uncertainties in vehicular communication, Gaussian noise (Grandvalet, Canu, and Boucheron 1997) was added to the numerical features. This step ensures that the models generalize well to minor fluctuations in data, preventing them from over-fitting to fixed patterns. Noise was applied selectively to the training data, maintaining the integrity of the test set for fair evaluation. Given the natural imbalance in attack instances, oversampling was performed to prevent bias toward majority classes. The Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al. 2002) was employed to generate synthetic data points for the under-represented class rather than merely duplicating existing samples. This method enhances the model's ability to detect rare attack instances, ensuring a balanced learning process. Feature scaling was then applied to normalize the numerical attributes, ensuring that models do not disproportionately favor features with larger numerical ranges. Min-Max scaling (Patro and Sahu 2015) was used to transform all numerical features to a 0–1 range while preserving the relative differences between data points. The scaler was fitted on the training data and applied consistently across both training and test sets to maintain uniformity in model evaluation. To ensure that subsequent analyses

focus on the attributes most indicative of malicious activity, we performed targeted feature selection, guided by Pearson correlation analysis (Freedman, Pisani, and Purves 2020). By evaluating the correlation between features, we identified and retained only those that exhibited a strong relationship with the attack patterns, ensuring that redundant or weakly contributing attributes were excluded. This process enhanced the model's ability to differentiate between normal and anomalous communication behaviors while reducing computational complexity.

Machine Learning Pipeline

To evaluate the effectiveness of our anomaly representations and the resulting dataset, we employed various machine learning algorithms and conducted a series of experimental ML evaluations. Common supervised learning algorithms, i.e., Random Forest (Ho 1995), K-Nearest Neighbors (Altman 1992), Logistic Regression (Hosmer Jr et al. 2013), Decision Tree (Wu et al. 2008), Naive Bayes (Rish et al. 2001), Gradient Boosting (Friedman 2001), and Support Vector Machines (Cortes and Vapnik 1995) were implemented to explore how effectively the dataset could support anomaly detection under various settings. By adjusting parameters and comparing outcomes, we gained insights into the discriminative power of the engineered features. For training and testing, we partitioned the dataset into two-thirds for training and one-third for testing. This widely used split ratio ensures sufficient data for model training while providing a test set large enough to reliably assess generalization performance and avoid overfitting.

Evaluation

To evaluate our proposed approach and resulting dataset for intrusion detection in VCNs, we designed and implemented a comprehensive experimental setup that integrates simulation, feature extraction, preprocessing, and model evaluation. This setup reflects real-world vehicular environments while ensuring controlled and repeatable testing conditions. As an experimental validation, the primary objective is to demonstrate the usability and relevance of our dataset in capturing VCN-specific attack behaviors. More detailed, application-specific evaluations and real-world deployments are left open for the research community to explore in future studies.

Simulation Environment

The dataset was generated using a combination of traffic and network simulation tools. *SUMO* was employed for vehicular traffic modeling, leveraging real-world road maps imported from *OSM*. SUMO's microscopic traffic control capabilities allowed us to model realistic mobility patterns, including variable vehicle speeds, lane-changing behaviors, and stop-and-go movements. This provided a dynamic vehicular environment essential for accurately simulating attack scenarios.

For network communication simulation, we utilized *NS-3*, which facilitated the modeling of vehicular communication protocols under dynamic topology conditions. NS-3 simulated the packet transmission between vehicles and infrastructure, enabling us to capture network-level behaviors such as delays, routing metrics, and node interactions. The communication data generated from NS-3 was subsequently processed to extract critical network parameters relevant to attack detection. NS-3 simulations incorporated the physical (PHY) and media access control (MAC) layers specific to vehicular communications (IEEE 802.11p) (Jiang and Delgrossi 2008). Additionally, the Ad hoc On-Demand Distance Vector (AODV) (Perkins, Belding-Royer, and Das 2003) routing protocol was utilized to manage the dynamic and mobile nature of vehicular networks.

To simulate the VCN environment, we modeled a dynamic network consisting of vehicular nodes with On-board Units (OBU), roadside units (RSUs), and a backbone communication infrastructure, as illustrated in Figure 5. In this setup, vehicles communicate using vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) channels, with the RSUs serving as central points for data aggregation and processing. Each RSU is equipped with data processing facilities, enabling the implementation of machine learning models to evaluate network behavior and detect anomalies in real-time. The RSUs not only facilitate local processing but also communicate over the backbone link to ensure that contextual and mobility-related data

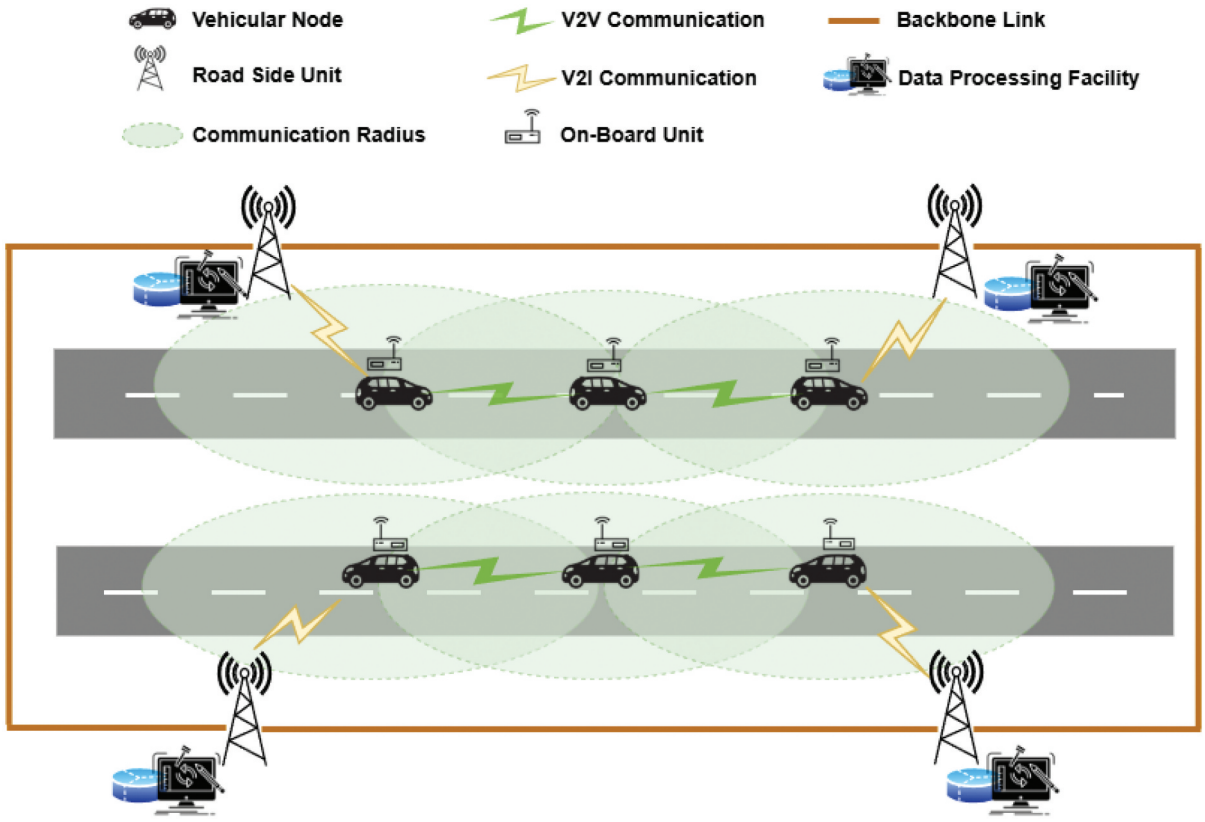

Figure 5. VCN scenario in our experimental setup.

from different segments of the network are considered. This setup aligns with our experimental assumptions, where machine learning-based evaluations are performed directly at the RSUs, ensuring low latency and efficient processing.

A critical aspect of our simulation involved the introduction of malicious nodes. We simulated anomalies, ranging from 10% to 30% of the total nodes. These entities were programmed to execute attack behaviors dynamically as described in Sybil, Position Spoofing, and Wormhole above. Simulations were conducted on Ubuntu 22, with run times ranging from 600 to 3600 seconds to capture both transient and long-term behaviors. Table 8 summarizes the key simulation parameters and configurations used.

Feature Extraction and Dataset Construction

Following the simulation phase, raw network and mobility data were collected to construct the dataset. The extracted features comprehensively encapsulated three critical aspects of VCNs: *spatial*, *temporal*, and *network communication behaviors*. These base features included node positions, velocities, inter-vehicle distances, hop counts, packet delivery statistics, and communication delays. Additionally, a set of *engineered*

Table 8. Simulation Parameters.

Parameter	Value
Operating System	Ubuntu 22
Mobility Simulator	SUMO (with OSM integration)
Network Simulator	NS-3
Simulation Duration	Upto 1500 seconds
Scenarios	Urban/Metropolitan/Highway (Mixed)
PHY/MAC Layer	IEEE 802.11p
Routing Protocol	AODV
Transport Protocol	UDP
Total Vehicles	Varies with density
Malicious Vehicles	10% – 30% of total vehicles

features was derived to capture more nuanced patterns indicative of attacks. The dataset was then labeled using the proposed algorithms in Data Labelling, which systematically identified attack instances by analyzing mobility and communication inconsistencies.

Evaluation Metrics

We employed several metrics to evaluate the performance of machine learning models in detecting anomalies, including *Accuracy, Precision, Recall*, and *F1-Score* (Binary, Macro, and Micro). These metrics provide both a general overview of the model's performance and a class-specific assessment to ensure fairness across the imbalance in attack classes. Below, we describe each metric and its mathematical formulation:

Accuracy

Accuracy measures the proportion of correctly classified instances over the total instances. It is calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{28}$$

where TP denotes true positives, TN true negatives, FP false positives, and FN false negatives.

Precision

Precision quantifies the proportion of true positive predictions out of all positive predictions made by the model. It is calculated as:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{29}$$

Recall

Recall measures the proportion of actual positive instances correctly identified by the model. It is calculated as:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{30}$$

F1-Score

The F1-Score is the harmonic mean of Precision and Recall, balancing the trade-off between false positives and false negatives:

$$\text{F1} - \text{Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{31}$$

Due to class imbalance considerations, the F1-Score is extended as:

$$\text{MacroF1} = \frac{1}{C} \sum_{c=1}^{C} \text{F1}_c \tag{32}$$

where C is the number of classes.

Computational Setup

The simulations and experiments were conducted on a *system with an Intel Core i7 processor, 32GB RAM, and an NVIDIA RTX GPU* for accelerated computations. Python-based implementations using *Scikit-learn* (Pedregosa et al. 2011), *Pandas* (McKinney et al. 2010), *Matplotlib* (Hunter 2007), *Seaborn* (Waskom 2021), and *Imbalanced-learn* (LemaA˜ Z˘tre et al. 2017) were employed for model training and analysis.

Results and Discussion

The evaluation of intrusion detection across three vehicular network attack scenarios reveals distinct performance patterns, with tree-based ensembles consistently demonstrating superior accuracy and F1-macro scores (Tables 9–11).

For Sybil detection, Random Forest achieved the highest accuracy (0.9326) and F1-macro (0.880), outperforming Gradient Boosting (F1-macro: 0.781) and Decision Trees (F1-macro: 0.782). Notably, K-Nearest Neighbors (KNN) achieved moderate results (F1-macro: 0.726), while SVM and Logistic Regression lagged in both accuracy and F1 metrics. The low performance of Naïve Bayes suggests that simplistic probabilistic approaches fail to capture the intricate spatiotemporal relationships essential for Sybil detection.

In spoofing detection, models excelled overall, with Random Forest (accuracy: 0.999, F1-macro: 0.981) and Gradient Boosting (accuracy: 0.998, F1-macro: 0.973) demonstrated near-flawless performances, reinforcing the advantage of ensemble-based learning techniques. Decision Tree closely matched these results with an F1-macro 0.977, while SVM and Logistic Regression showed stark precision-recall imbalances (SVM precision: 0.306 vs. recall: 0.990; Logistic Regression precision: 0.152 vs Recall: 0.900), reducing their F1-macro scores to 0.726 and 0.613, respectively.

Wormhole detection saw the strongest overall performance, with Random Forest (accuracy: 0.994, F1-macro: 0.983) and Decision Tree (F1-macro: 0.971) achieving near-perfect metrics demonstrating their effectiveness in adapting packet leashing concepts to a dynamic multi-hop VCN. Gradient Boosting followed closely with an F1-macro score of 0.965 while also performing well in recall (0.974), Naive Bayes unexpectedly outperformed SVM and Logistic Regression (F1-macro: 0.812 vs. 0.707/0.695), suggesting its conditional utility in specific attack contexts.

Table 9. Performance Metrics of Sybil Detection Models.

Model	Accuracy	Precision	Recall	F1-Score	F1 (Macro)
Random Forest	0.932600	0.803778	0.798749	0.801255	0.880337
Gradient Boosting	0.855800	0.552518	0.800834	0.653895	0.781412
SVM	0.654487	0.281870	0.666319	0.396156	0.577084
K-Nearest Neighbors	0.804895	0.457607	0.793535	0.580473	0.726682
Logistic Regression	0.661227	0.284549	0.654849	0.396715	0.580602
Decision Tree	0.861830	0.570978	0.754953	0.650202	0.782058
Naive Bayes	0.175417	0.170064	0.991658	0.290337	0.153211

Table 10. Performance Metrics of Spoofing Detection Models.

Model	Accuracy	Precision	Recall	F1 (Binary)	F1 (Macro)
Random Forest	0.999053	0.951456	0.975124	0.963145	0.981333
Gradient Boosting	0.998610	0.901345	1.000000	0.948113	0.973704
SVM	0.971450	0.306626	0.990050	0.468235	0.726783
K-Nearest Neighbors	0.945364	0.068831	0.263682	0.109166	0.540492
Logistic Regression	0.935195	0.152485	0.900498	0.260807	0.613459
Decision Tree	0.998863	0.929577	0.985075	0.956522	0.977973
Naive Bayes	0.672183	0.033651	0.895522	0.064865	0.433060

Table 11. Performance Metrics of Wormhole Detection Models.

Model	Accuracy	Precision	Recall	F1-Score	F1 (Macro)
Random Forest	0.994318	0.955616	0.983089	0.969158	0.983014
Gradient Boosting	0.988021	0.901460	0.974634	0.936620	0.965003
SVM	0.850269	0.359873	0.833145	0.502636	0.707252
K-Nearest Neighbors	0.903916	0.484100	0.883878	0.625574	0.785230
Logistic Regression	0.838239	0.341564	0.842165	0.486012	0.695014
Decision Tree	0.990427	0.930081	0.967306	0.948328	0.971527
Naive Bayes	0.913642	0.512738	0.987035	0.674889	0.812549

Comparative Analysis and Insights

Across all attack scenarios, Random Forest consistently demonstrated superior performance, achieving the highest accuracy and F1 (Macro) across all three detection tasks. The results underscore the need for adaptable, and context-aware intrusion detection mechanisms specifically for VCNs. Three key insights emerge from these evaluations:

(1) Tree-Based Learning Dominates: Random Forest and Decision Tree consistently achieved the highest F1-macro scores across all attack types, confirming their ability to adapt to VCN-specific mobility dynamics and evolving attack strategies. Their low false positive rates make them suitable for deployment where frequent mis-classifications can lead to unnecessary security consequences.

(2) F1-Macro as a Key Metric: While accuracy remained high across all models, F1-macro scores better captured detection rate across diverse attack scenarios. For instance, K-Nearest Neighbors achieved 0.938 accuracy in spoofing detection but had an F1-macro score of only 0.552, revealing its inefficacy in classification. This highlights the importance of prioritizing F1-macro over accuracy in VCN security assessments.

(3) Precision-Recall Trade-offs Matter: Certain models, particularly SVM and KNN, exhibited notable precision-recall disparities, making them unreliable for vehicular threat detection where high recall alone does not guarantee effective security. Their inability to distinguish legitimate high-speed vehicular transitions from malicious anomalies renders them less suitable for deployment in highly dynamic networks.

Overall, the findings highlight the need for attack-specific detection strategies that go beyond traditional IDS designed for static networks. The consistent success of ensemble-based models in capturing dynamic mobility-driven attack behaviors suggests that future VCN security frameworks should prioritize adaptive, data-driven anomaly detection techniques that leverage both spatial and temporal contextualization.

Conclusion and Future Work

In this paper, we have presented the modeling and detection of crucial VCN security threats, such as Sybil, wormhole, and position spoofing attacks. By acknowledging and addressing the unique dynamics of VCNs, our work significantly deviates from traditional static network security models. The realistic simulation and comprehensive feature engineering employed have helped in constructing a dataset that not only mimics real-world vehicular scenarios but also enriches the potential for accurate anomaly detection.

Our approach emphasizes the integration of spatial and temporal data, reflecting the mobility and speed of vehicles, which are crucial for identifying inconsistencies that may indicate security breaches. By redefining how attacks are modeled within vehicular networks, we address a critical gap in existing datasets that often overlook the impact of high mobility and dynamic topologies. The performance evaluation of multiple machine learning models further highlights the effectiveness of our dataset and feature design in facilitating AI-driven anomaly detection in VCNs.

While our dataset and models provide tools for understanding and mitigating threats, they also lay a foundation for future research to refine these mechanisms for real-time deployment in diverse vehicular environments. One of the key areas for expansion could be the detection of *malicious flooding*, which exploit the high communication frequency in VCNs to overwhelm nodes, leading to denial-of-service conditions. Additionally, further optimizations in our algorithms will focus on enhancing detection efficiency, ensuring adaptability to rural settings, and improving real-time implementation feasibility.

Author Contributions

CRediT: **Muhammad Danish Khan:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Visualization, Writing – original draft; **Vinh-Thong Ta:** Conceptualization, Investigation, Methodology, Resources, Supervision, Writing – review & editing; **Husnain Rafiq:** Conceptualization, Investigation, Resources, Supervision, Writing – review & editing; **Nonso Nnamoko:** Conceptualization, Investigation, Methodology, Resources, Supervision, Writing – review & editing.

Disclosure Statement

All authors declare that they have no affiliations with, or involvement in, any organisation or entity with any financial or non-financial interest in the subject matter or materials presented in this manuscript. One of the authors, Nonso Nnamoko, acknowledges his role as an Associate Editor for the Taylor & Francis Journal of Applied Artificial Intelligence and requests to be excluded from all editorial processes related to this manuscript.

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

Funding

This study received no external funding. The work was undertaken as part of a PhD project supported by Edge Hill University through the Graduate Teaching Assistant PhD studentship scheme. Sincere thanks are extended to the Department of Computer Science for their support, time, and resources that made this research possible.

ORCID

Muhammad Danish Khan http://orcid.org/0000-0002-0527-9464
Vinh-Thong Ta http://orcid.org/0000-0003-0399-9633
Husnain Rafiq http://orcid.org/0000-0002-3178-4026
Nonso Nnamoko http://orcid.org/0000-0002-5064-2621

Data Availability Statement

The data underpinning the findings of this study are openly available in the Mendeley Data repository under the title Comprehensive Vehicular Communication Network Attack Dataset, licensed under CC BY 4.0. The dataset contains the raw data that were used for experiments leading to the results presented in the manuscript and can be accessed via the DOI: https://doi.org/10.17632/x7tdmsf27x.1 This dataset is essential for replicating all results reported in the article.

Author Contribution Statement

Formal analysis, Investigation & Writing – original draft preparation: [Muhammad Danish Khan]; Conceptualisation & Methodology: [Muhammad Danish Khan, Vinh-Thong Ta, Husnain Rafiq, Nonso Nnamoko]; Resources, Supervision, Writing – review and editing: [Nonso Nnamoko, Vinh-Thong Ta, Husnain Rafiq]. All authors have read and approved the final version of the manuscript. During the preparation of this work, the authors used OpenAI's ChatGPT to refine the phrasing of certain sections, including providing suggestions for expanding explanations and enhancing readability with caution. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

Ahmad, A., and L. Dey. 2007. A k-mean clustering algorithm for mixed numeric and categorical data. *Data and Knowledge Engineering* 63 (2):503–27. doi:10.1016/j.datak.2007.03.016.

Alsarhan, A., A.-R. Al-Ghuwairi, I. T. Almalkawi, M. Alauthman, and A. Al-Dubai. 2021. Machine learning-driven optimization for intrusion detection in smart vehicular networks. *Wireless Personal Communications* 117 (4):3129–52. doi:10.1007/s11277-020-07797-y.

Altman, N. S. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46 (3):175–85. doi:10.1080/00031305.1992.10475879.

Amaouche, S., A. Guezzaz, S. Benkirane, and M. Azrour. 2024. A robust model for predicting abnormal behavior in vehicular networks using adaBoost and chi-square. *Wireless Personal Communications* 138 (4):2583–611. doi:10.1007/s11277-024-11615-0.

Amaouche, S., A. Guezzaz, S. Benkirane, M. Azrour, S. B. A. Khattak, H. Farman, and M. M. Nasralla. 2023. Fscb-ids: Feature selection and minority class balancing for attacks detection in VANETS. *Applied Sciences* 13 (13):7488. doi:10.3390/app13137488.

Azam, S., M. Bibi, R. Riaz, S. S. Rizvi, and S. J. Kwon. 2022. Collaborative learning based sybil attack detection in vehicular ad-hoc networks (VANETS). *Sensors* 22 (18):6934. doi:10.3390/s22186934.

Bangui, H., M. Ge, and B. Buhnova. 2022. A hybrid machine learning model for intrusion detection in VANET. *Computing* 104 (3):503–31. doi:10.1007/s00607-021-01001-0.

Belenko, V., V. Krundyshev, and M. Kalinin (2018). Synthetic datasets generation for intrusion detection in VANET. In *Proceedings of the 11th international conference on security of information and networks*, 1–6.

Ben Rabah, N., and H. Idoudi. 2022. A machine learning framework for intrusion detection in vanet communications. In *Emerging trends in cybersecurity applications*, 209–27. Springer.

Boualouache, A., and T. Engel. 2023. A survey on machine learning-based misbehavior detection systems for 5g and beyond vehicular networks. *IEEE Communications Surveys and Tutorials* 25 (2):1128–72. doi:10.1109/COMST.2023.3236448.

Chawla, N. V., K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* 16:321–57. doi:10.1613/jair.953.

Contreras-Castillo, J., S. Zeadally, and J. A. Guerrero-Ibañez. 2017. Internet of vehicles: Architecture, protocols, and security. *IEEE Internet of Things Journal* 5 (5):3701–09. doi:10.1109/JIOT.2017.2690902.

Cortes, C., and V. Vapnik. 1995. Support-vector networks. *Machine Learning* 20 (3):273–97. doi:10.1023/A:1022627411411.

Elkan, C. 2000. Results of the kdd'99 classifier learning. *Acm Sigkdd Explorations Newsletter* 1 (2):63–64. doi:10.1145/846183.846199.

Elsadig, M. A., A. Altigani, Y. Mohamed, A. H. Mohamed, A. Kannan, M. Bashir, and M. A. Adiel. 2025. Connected vehicles security: A lightweight machine learning model to detect VANET attacks. *World Electric Vehicle Journal* 16 (6):324. doi:10.3390/wevj16060324.

Ercan, S., M. Ayaida, and N. Messai. 2021. Misbehavior detection for position falsification attacks in VANETS using machine learning. *IEEE Access* 10:1893–904. doi:10.1109/ACCESS.2021.3136706.

Freedman, D., R. Pisani, and R. Purves. 2020. Statistics: Fourth international student edition. *WW Nort Co Httpswww Amaz ComStatistics-Fourth-Int-Stud-Free Accessed* 22.

Friedman, J. H. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 1189–232.

Gad, A. R., A. A. Nashat, and T. M. Barkat. 2021. Intrusion detection system using machine learning for vehicular ad hoc networks based on ton-iot dataset. *IEEE Access* 9:142206–17. doi:10.1109/ACCESS.2021.3120626.

Grandvalet, Y., S. Canu, and S. Boucheron. 1997. Noise injection: Theoretical prospects. *Neural Computation* 9 (5):1093–108. doi:10.1162/neco.1997.9.5.1093.

Haklay, M., and P. Weber. 2008. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing* 7 (4):12–18. doi:10.1109/MPRV.2008.80.

Hanif, M., H. Ashraf, Z. Jalil, N. Z. Jhanjhi, M. Humayun, S. Saeed, and A. M. Almuhaideb. 2022. Ai-based wormhole attack detection techniques in wireless sensor networks. *Electronics* 11 (15):2324. doi:10.3390/electronics11152324.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, 278–82. IEEE.

Hosmer, D. W., Jr, S. Lemeshow, and R. X. Sturdivant. 2013. *Applied logistic regression*. John Wiley & Sons.

Hu, Y.-C., A. Perrig, and D. B. Johnson (2003). Packet leashes: A defense against wormhole attacks in wireless networks. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, volume 3, 1976–86. IEEE.

Hunter, J. D. 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering* 9 (3):90–95. doi:10.1109/MCSE.2007.55.

Jiang, D., and L. Delgrossi (2008). Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments. In *VTC Spring 2008-IEEE vehicular technology conference*, 2036–40. IEEE.

Kamel, J., M. Wolf, R. W. Van Der Hei, A. Kaiser, P. Urien, and F. Kargl (2020). Veremi extension: A dataset for comparable evaluation of misbehavior detection in VANETS. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, 1–6. IEEE.

Karthiga, B., D. Durairaj, N. Nawaz, T. K. Venkatasamy, G. Ramasamy, and A. Hariharasudan. 2022. Intelligent intrusion detection system for VANET using machine learning and deep learning approaches. *Wireless Communications and Mobile Computing* 2022 (1):5069104. doi:10.1155/2022/5069104.

Kasongo, S. M. 2023. A deep learning technique for intrusion detection system using a recurrent neural networks based framework. *Computer Communications* 199:113–25. doi:10.1016/j.comcom.2022.12.010.

Khan, S. Z., M. Mohsin, and W. Iqbal. 2021. On gps spoofing of aerial platforms: A review of threats, challenges, methodologies, and future research directions. *PeerJ Computer Science* 7:e507. doi:10.7717/peerj-cs.507.

Korium, M. S., M. Saber, A. Beattie, A. Narayanan, S. Sahoo, and P. H. Nardelli. 2024. Intrusion detection system for cyberattacks in the internet of vehicles environment. *Ad Hoc Networks* 153:103330. doi:10.1016/j.adhoc.2023.103330.

Kuma, R. P., U. Srilakshmi, and K. G. Reddy. 2024. Fast and efficient distance based external wormhole detection and prevention system (fedewdps). *Majlesi Journal of Electrical Engineering* 18 (1).

Lai, G.-H. 2016. Detection of wormhole attacks on ipv6 mobility-based wireless sensor network. *EURASIP Journal on Wireless Communications and Networking* 2016 (1):274. doi:10.1186/s13638-016-0776-0.

Laouiti, D. E., M. Ayaida, N. Messai, S. Najeh, L. Najjar, and F. Chaabane (2022). Sybil attack detection in VANETs using an AdaBoost classifier. In *2022 International Wireless Communications and Mobile Computing (IWCMC)*, 217–22. IEEE.

Lemaâžtre, G., F. Nogueira, and C. K. Aridas. 2017. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research* 18 (17):1–5.

Lopez, P. A., M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner (2018). Microscopic traffic simulation using sumo. In *2018 21st international conference on intelligent transportation systems (ITSC)*, 2575–82. Ieee.

McKinney, W., et al. 2010. Data structures for statistical computing in python. *scipy* 445 (1):51–56.

Moustafa, N. 2021. A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets. *Sustainable Cities and Society* 72:102994.

Oligeri, G., S. Sciancalepore, O. A. Ibrahim, and R. Di Pietro. 2022. Gps spoofing detection via crowd-sourced information for connected vehicles. *Computer Networks* 216:109230.

Parno, B., and A. Perrig (2005). Challenges in securing vehicular networks. In *Workshop on hot topics in networks (HotNets-IV)*, 1–6. Maryland, USA.

Patro, S., and K. K. Sahu (2015). Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of Machine Learning Research* 12:2825–30.

Perkins, C., E. Belding-Royer, and S. Das (2003). Ad hoc on-demand distance vector (AODV) routing. Technical report.

Rajapaksha, S., H. Kalutarage, M. O. Al-Kadri, A. Petrovski, G. Madzudzo, and M. Cheah. 2023. Ai-based intrusion detection systems for in-vehicle networks: A survey. *ACM Computing Surveys* 55 (11):1–40.

Rani, P., and R. Sharma. 2023. Intelligent transportation system for internet of vehicles based vehicular networks for smart cities. *Computers and Electrical Engineering* 105:108543.

Riley, G. F., and T. R. Henderson. 2010. The ns-3 network simulator. In *Modeling and tools for network simulation*, 15–34. Springer.

Rish, I., et al. (2001). An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, 41–46. Seattle, USA.

Sakiz, F., and S. Sen. 2017. A survey of attacks and detection mechanisms on intelligent transportation systems: Vanets and iov. *Ad Hoc Networks* 61:33–50.

Sharafaldin, I., A. H. Lashkari, A. A. Ghorbani, et al. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* 1 (2018):108–16.

Sharma, A., and A. Jaekel (2021). Machine learning approach for detecting location spoofing in VANET. In *2021 International conference on computer communications and networks (ICCCN)*, 1–6. IEEE.

Singh, P. K., R. R. Gupta, S. K. Nandi, and S. Nandi (2019). Machine learning based approach to detect wormhole attack in VANETs. In *Workshops of the international conference on advanced information networking and applications*, 651–61. Springer.

Stepien, K., and A. Poniszewska-Maranda. 2021. Security measures with enhanced behavior processing and footprint algorithm against sybil and bogus attacks in vehicular ad hoc network. *Sensors* 21 (10):3538.

Suman, P., S. Padhy, N. Kumar, A. Suman, A. Singh, K. K. Singh, Á. K. Castilla, and T. S. S. AL-Zahrani (2024). An improved deep learning-based intrusion detection for reliable communication in VANET. *IEEE Transactions on Consumer Electronics*.

Tavallaee, M., E. Bagheri, W. Lu, and A. A. Ghorbani (2009). A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, 1–6. Ieee.

Van Der Heijden, R. W., T. Lukaseder, and F. Kargl (2018). Veremi: A dataset for comparable evaluation of misbehavior detection in VANETs. In *International conference on security and privacy in communication systems*, 318–37. Springer.

Verma, A., R. Saha, G. Kumar, and T.-H. Kim. 2021. The security perspectives of vehicular networks: A taxonomical analysis of attacks and solutions. *Applied Sciences* 11 (10):4682.

Waskom, M. L. 2021. Seaborn: Statistical data visualization. *Journal of Open Source Software* 6 (60):3021.

Wu, X., V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, et al. 2008. Top 10 algorithms in data mining. *Knowledge and Information Systems* 14 (1):1–37.

Yang, L., A. Moubayed, and A. Shami. 2021. Mth-ids: A multitiered hybrid intrusion detection system for internet of vehicles. *IEEE Internet of Things Journal* 9 (1):616–32.

Yang, L., and A. Shami (2022). A transfer learning and optimized cnn based intrusion detection system for internet of vehicles. In *ICC 2022–IEEE International Conference on Communications*, 2774–79. IEEE.

Zaidi, K., M. B. Milojevic, V. Rakocevic, A. Nallanathan, and M. Rajarajan. 2015. Host-based intrusion detection for VANETs: A statistical approach to rogue node detection. *IEEE Transactions on Vehicular Technology* 65 (8):6703–14.

Zhou, M., and L. Han. 2023. Sensor spoofing detection on autonomous vehicle using channel-spatial-temporal attention based autoencoder network. *Mobile Networks and Applications* 1–14.

Context-aware intrusion detection in vehicular communication networks: enhanced attack modeling and dataset

Khan, Muhammad Danish

2025

Khan MD, Ta V-T, Rafiq H, Nnamoko N. (2025) Context-aware intrusion detection in vehicular communication networks: enhanced attack modeling and dataset. Applied Artificial Intelligence, Volume 39, September 2025, Article number 2538453

https://doi.org/10.1080/08839514.2025.2538453