5-16-2024

Traffic Congestion Sybil Attack Detection in VANETs using Machine Learning Techniques

Sarthak Khanduja
University of Windsor

TRAFFIC CONGESTION SYBIL ATTACK DETECTION IN VANETS USING MACHINE LEARNING TECHNIQUES

By

SARTHAK KHANDUJA

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2024

TRAFFIC CONGESTION SYBIL ATTACK DETECTION IN VANETS USING MACHINE LEARNING TECHNIQUES

by

SARTHAK KHANDUJA

APPROVED BY:

J. Pathak
Odette School of Business

D. Wu
School of Computer Science

A. Jaekel, Advisor
School of Computer Science

April 26, 2024

DECLARATION OF CO-AUTHORSHIP

I. Co-Authorship

I hereby declare that this thesis incorporates material that is the result of research conducted under the supervision of Dr. Arunita Jaekel. In all cases, the key ideas, primary contribution, experimental designs, data analysis, and interpretation were performed by the author, and the contribution of the co-author was primarily through providing feedback and the proofreading of the published manuscripts.

I am aware of the University of Windsor Senate Policy on Authorship, and I certify that I have properly acknowledged the contribution of other researchers to my thesis and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis. I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my work.

II. General

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication. I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution. I understand that my thesis may be made electronically available to the public.

ABSTRACT

Integrating Vehicular Ad-Hoc Networks (VANETs) into modern Intelligent Transportation Systems (ITS) introduces critical security concerns. This research addresses the emerging threat of Traffic Congestion Sybil Attacks, where malicious entities inject spurious data to fabricate artificial traffic congestion. The methodology involves a detailed examination of the VeReMi dataset, a benchmark for VANET research, coupled with state-of-the-art classification machine learning algorithms. The analysis includes training and evaluating these models to identify patterns indicative of Grid Sybil attacks. Preliminary results obtained through meticulous testing demonstrate a substantial enhancement in various classification metrics, showcasing promising improvements, especially in enhancing the recall value for accurately identifying maliciously induced traffic congestions. These initial findings underscore the potential for further refinements and heightened classification metrics in subsequent phases of the research. This thesis emphasizes the urgency of securing VANETs against Traffic Congestion Sybil Attacks, presenting an innovative solution through the fusion of machine learning techniques and the VeReMi dataset. The outcomes contribute to theoretical understanding and hold practical implications for enhancing the security of vehicular communication networks.

DEDICATION

To my parents, whose unwavering support and encouragement have been the foundation of my journey. Without your love and guidance, I would not be at this point, writing and working on this thesis.

And, to my sister, who set forth the foundation of the path I followed toward achieving this great milestone in my academic career.

ACKNOWLEDGEMENTS

I want to express my gratitude to my supervisor, Dr. Arunita Jaekel, whose expertise, guidance, and passion for the subject matter have been invaluable. Your mentorship has shaped my understanding and made this endeavor possible.

And to Muhammad Anwar Shahid for his guidance and support on the subject matter.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

VANET	Vehicular ad-hoc network
RSU	Road-side Unit
OBU	On-board Unit
DSRC	Dedicated short range communication
C-V2X	Cellular Vehicle to everything
PKI	Public key infrastructure
BSM	Basic safety message
ITS	Intelligent Transportation system
WAVE	Wireless Access in vehicular Environment
VeReMi	Vehicular Reference Misbehaviour Dataset

CHAPTER 1

Introduction

1.1 Vehicular ad-hoc networks

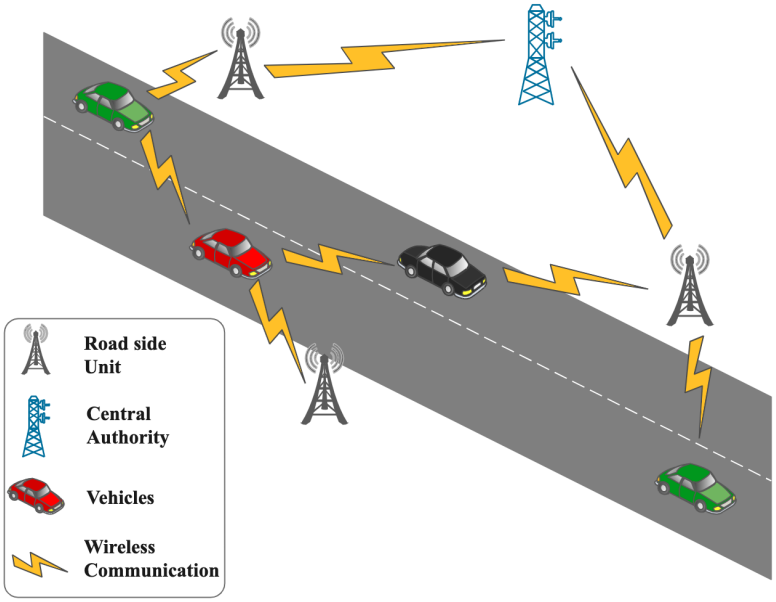

Figure 1.1: An example of Vehicular ad-hoc network [1]

Imagine a busy intersection with many vehicles, traffic lights, and streetlights. In this scenario, a smart network forms by connecting sensors on vehicles, streetlights, and traffic lights. This type of network is called Vehicular Ad-Hoc Networks (VANETs), a crucial part of Intelligent Transportation Systems (ITS).

VANETs form an interconnection of sensors on cars, streetlights, traffic signals, and other nodes that talk to each other. This communication helps improve transportation by sharing real-time information, as shown in Fig. 1.1 [1], making the system safer and

more efficient.

In recent years, VANETs have seen a lot of research and development. This growth is largely due to the popularity and cost reduction of IEEE 802.11 technologies, vehicle makers' use of information technology to address safety and environmental issues, and governments' allocation of wireless spectrum for vehicle communication [2].

Intelligent Transportation System integrates information, communications, computers, and other technologies and applies them in the field of transportation to build an integrated system of people, roads, and vehicles by utilizing advanced data communication technologies [3].

VANETs are a specific area within ITS and are a subset of Mobile Ad-Hoc Networks (MANETs), which focus on communication among moving nodes. VANETs adapt these principles to deal with the unique challenges of vehicles in motion.

1.1.1 Components of a VANET

Primary Components of a VANET are:

- **On-Board Unit (OBU):** The On-Board Unit is a key component installed in each vehicle participating in the VANET. It facilitates communication among vehicles and with roadside infrastructure. The OBU is equipped with communication modules and sensors, enabling it to exchange information with other OBUs and roadside units (RSUs).
- **Road-Side Unit (RSU):** The Road-Side Unit (RSU) is a fixed infrastructure component installed along roadways. RSUs provide a communication interface between vehicles and the broader transportation infrastructure. They may be strategically placed at traffic signals, intersections, or other locations to enhance communication range and reliability.
- **Global Positioning System (GPS):** GPS technology is often integrated into OBUs and RSUs to provide accurate location information. This is essential for various VANET applications, including traffic management, navigation, and collision avoidance.

- **Sensors:** Vehicles equipped with sensors contribute to the VANET's data pool. These sensors can include cameras, RADAR, LIDAR, and other environmental sensors. The data collected by these sensors are shared within the VANET, aiding in real-time traffic monitoring, hazard detection, and other safety applications.

- **Communication Protocols:** VANETs rely on specific communication protocols to facilitate effective information exchange. Commonly used protocols include IEEE 802.11p for wireless communication between vehicles and infrastructure.

1.1.2 Communication in VANETs

Communication within Vehicular Ad-Hoc Networks (VANETs) unfolds in the wireless spectrum, presenting an array of communication types, each with its own set of vulnerabilities. While offering flexibility and adaptability, the wireless nature of these communications also introduces the susceptibility to various security threats and attacks.

Vehicle to Vehicle (V2V):

V2V communication involves direct interaction between vehicles on the road. It enhances road safety by allowing vehicles to exchange real-time information, such as location, speed, and potential hazards.

However, this form of communication is not without its challenges. The wireless nature of V2V communication makes it susceptible to eavesdropping, where unauthorized entities may intercept and gather sensitive data. Message tampering is also risky, where the integrity of the information may be compromised.

In Vehicular Ad Hoc Networks (VANETs), the seamless exchange of vital information among Vehicle-to-Vehicle (V2V) compatible nodes is paramount. This communication is facilitated through the periodic dissemination of a fundamental safety message (BSM), which serves as a conduit for relaying real-time data of the transmitting vehicle. The BSM encapsulates a spectrum of crucial information, including the vehicle's current position, velocity, heading, brake status, steering wheel angle, and other pertinent details essential for ensuring road safety and facilitating intelligent vehicular operations [4]. This continuous exchange of BSMs among vehicles forms the cornerstone of VANETs, enabling

collaborative and proactive decision-making processes to enhance traffic efficiency, preempt potential hazards, and foster a secure and resilient vehicular ecosystem.

Vehicle to Infrastructure (V2I):

V2I communication connects vehicles and fixed infrastructure elements like traffic lights or roadside units. It is crucial for optimizing traffic flow, coordinating traffic signals, and providing timely information to drivers.

The reliance on wireless communication exposes V2I links to potential denial-of-service attacks. Malicious actors may attempt to disrupt or overload communication channels, hindering the exchange of critical information. Unauthorized access to infrastructure points is also a concern.

Vehicle to All (V2X):

V2X communication encompasses vehicle interactions and a broader spectrum of entities, including other vehicles, infrastructure, and pedestrians. It is the most comprehensive form, enabling a holistic exchange of information for enhanced safety and efficiency.

Due to its expansive nature, V2X faces many potential threats. Privacy concerns arise from exchanging sensitive information, and there's a risk of impersonation, where malicious entities may pose as legitimate participants. The challenges in securing V2X communications are diverse and complex.

1.1.3 Attacks on VANETs

Due to the wireless nature of communication between elements within a VANET, attackers with malicious intent can always try and disrupt the network.

Safety-related messages, traffic management, and navigation messages are disseminated periodically in vehicular network systems. These messages are vulnerable to several attacks, from passive eavesdropping to active interference [5].

Passive Attacks:

- **Eavesdropping:** Unauthorized entities secretly intercept and monitor communication, gathering sensitive information without actively altering the transmitted data.

Active Attacks:

- **Jamming:** Malicious entities intentionally disrupt communication channels, causing interference and potentially rendering the network inoperable.
- **Spoofing:** Attackers create fake identities or manipulate their identity to deceive the network about their legitimacy.
- **Replay Attacks:** Malicious entities capture and re-transmit valid messages to deceive the network, repeating outdated or irrelevant information.
- **Selective Forwarding:** Malicious nodes selectively forward or block specific messages, affecting the overall reliability of the network.

Integrity Attacks:

- **Message Tampering:** Attackers alter the content of messages, injecting false or misleading information into the system.
- **Data Injection:** Malicious entities inject unauthorized data into the network, potentially causing confusion and disrupting normal operations.

Privacy Attacks:

- **Location Tracking:** Malicious entities attempt to track the location of vehicles, compromising the privacy of individuals within the network.
- **Profile Inference:** Attackers analyze communication patterns to infer sensitive information about the behavior and preferences of network participants.

Sybil Attacks:

- **Sybil Attacks:** Malicious nodes create multiple fake identities to undermine the authenticity and trustworthiness of communication within the network.

Denial-of-Service (DoS) Attacks:

- **Jamming:** While mentioned earlier, jamming attacks can also be considered a form of denial-of-service, where communication channels are deliberately disrupted to prevent the normal functioning of the network.

- **Resource Exhaustion:** Attackers overwhelm network resources, making it challenging for legitimate participants to access and utilize the communication infrastructure.

Physical Attacks:

- **Vandalism:** Physical damage to communication infrastructure components, such as RSUs or OBUs, can disrupt the overall functionality of the network.

Misbehavior Attacks:

- **Selfish Behavior:** Nodes may act selfishly by selectively participating in communication or refraining from forwarding messages, undermining the collaborative nature of VANETs.

This is not an exhaustive list but a broad categorization of different attacks seen in VANETs.

For the purposes of this thesis, we will be focusing our attention on Sybil Attacks.

1.2 Motivation

In selecting my research focus on mitigating Traffic Congestion Sybil Attacks in Vehicular Ad-Hoc Networks (VANETs), my motivation stems from the profound impact these attacks can have on our transportation systems' efficiency, trustworthiness, and safety. Beyond the evident disruptions to traffic flow and navigation, these attacks introduce resource wastage whose repercussions extend beyond the immediate inconveniences. Resource wastage becomes a palpable concern as vehicles are redirected inefficiently, leading to increased fuel consumption and environmental impact. Misleading behavior induced among drivers in the network can result in unexpected maneuvers, increasing the likelihood of accidents and compromising road safety.

Recognizing the imperative to address these challenges, my research aims to develop precise and effective methods for classifying vehicles and distinguishing between legitimate and malicious entities within the VANET. This work is driven by a commitment to fortifying vehicular communication's integrity, contributing to the resilience and reliability of future connected transportation networks.

1.3 Problem Statement

In the realm of Vehicular Ad-Hoc Networks (VANETs), the emergence of Traffic Congestion Sybil Attacks poses a significant challenge. This threat involves malicious vehicles fabricating deceptive identities, compromising network message integrity. The potential ramifications extend beyond mere disruption, encompassing the risk of fatal accidents and widespread traffic disturbances. Existing detection approaches, primarily relying on Plausibility and Integrity checks, have demonstrated limitations regarding precision and susceptibility to mimicking legitimate behavior.

This research aims to address these shortcomings by leveraging robust Machine Learning models. The objective is to enhance the classification precision and recall of vehicles engaged in Traffic Congestion Sybil Attacks. The endeavor seeks to fortify the security and integrity of vehicular communication within VANETs, thereby fostering safer and more resilient transportation networks.

1.4 Solution Outline

The envisioned solution to address the challenge of detecting Traffic Congestion Sybil Attacks in Vehicular Ad-Hoc Networks (VANETs) entails the application of a generalized machine learning model. This model is trained on historical data to discern attacking vehicles from normal ones with a heightened degree of precision and recall. Given the nature of VANETs, where vehicles continually transmit messages throughout their journeys, the proposed solution leverages the observation that malicious vehicles adopting multiple identities tend to generate more messages than their legitimate counterparts.

Furthermore, using pseudonyms by attacker vehicles becomes a distinguishing factor, as duplicated identities are denied digital certificates by the Public Key Infrastructure (PKI). The training dataset is derived from the VeReMi dataset, originally generated through LuST [6] and VEINS [7] simulations, and encompasses varying traffic densities to ensure adaptability to diverse traffic scenarios.

The solution framework unfolds across the following key stages:

Dataset Preparation:

- Collation and consolidation of the original JSON data, distributed across sender vehicle folders, into a unified dataset.

Data Pre-processing:

- Removal of non-contributory features to assess a vehicle's attacker status.
- In-depth analysis and extraction of valuable insights from the dataset distribution to identify features significantly impacting classification.

Feature Engineering:

- Introduction of a supplementary set of features derived from the insights gathered during the pre-processing stage, augmenting the dataset for enhanced processing.

Splitting the Dataset:

- Adoption of a bespoke dataset split methodology to preserve the temporal order of time-series data, mitigating overfitting concerns experienced with random splits.

Classification using ML Models:

- Employing the trained machine learning model to classify vehicles, utilizing a test set for performance evaluation. The outcome shows notable improvement over previous attempts, demonstrating heightened efficacy in detecting Traffic Congestion Sybil Attacks within VANETs.

This meticulously structured solution aims to refine the detection capabilities in the context of VANETs and contribute to the broader landscape of securing vehicular communication systems.

1.4.1 Contributions

The contribution of this research is summarized as follows:

- Applied machine learning to precisely identify Traffic Congestion Sybil attacks in VANETs.
- Introduced a novel dataset splitting method to preserve temporal features in time-series data using classical machine learning algorithms.
- Elevated Recall and F-1 score metrics to enhance the detection accuracy of Traffic Congestion Sybil Attacks.
- Introduced additional features to bolster the detection capabilities of Sybil Attacks in VANETs.

1.5 Thesis Organization

The remaining outline of this thesis is as follows: Chapter 2 includes an overview of fundamental concepts of VANET and Traffic Congestion Sybil Attacks, along with a literature review of related work in Misbehavior Detection using Machine Learning approaches. Chapter 3 covers an overview of the suggested methodology and briefly describes the VeReMi dataset, followed by Chapter 4, which includes the experimental setup and the results. Finally, Chapter 5 portrays a conclusion of our work, followed by possible future work on the proposed methodology.

CHAPTER 2

Background Review

2.1 Overview of VANET

Vehicular Ad-Hoc Networks (VANETs) represent a specialized subset within the broader domain of Mobile Ad-Hoc Networks (MANETs), garnering escalating attention from researchers and practitioners alike due to a plethora of compelling factors. Although cellular networks enable convenient voice communication and simple infotainment services to drivers and passengers, they are not well-suited for certain direct vehicle-to-vehicle or vehicle-to-infrastructure communications. However, vehicular ad hoc networks (VANETs), which offer direct communication between vehicles and to and from roadside units (RSUs) can send and receive hazard warnings or information on the current traffic situation with minimal latency [2].

With the surge in urbanization and the exponential growth of vehicular traffic, the need for efficient and secure communication infrastructures has become paramount.

However, the very essence of VANET communication, occurring entirely wirelessly as elaborated in the preceding section, renders it inherently vulnerable to various forms of cyber-attacks. These vulnerabilities necessitate robust security measures and innovative approaches to ensure the integrity, confidentiality, and reliability of communication within VANETs.

Indeed, recognizing the susceptibility of VANETs to malicious activities due to their wireless communication nature, the emergence of Misbehavior Detection Systems has become paramount. These systems serve the critical function of identifying and flagging malicious nodes, often called "attackers," within the network. By employing sophisticated

algorithms and security mechanisms, Misbehavior Detection Systems play a pivotal role in safeguarding the integrity and reliability of VANET communication, mitigating potential threats, and ensuring the smooth functioning of vehicular ad-hoc networks.

2.1.1 Security and Privacy Threats in VANET

Continuing the discussion of security and privacy threats in VANETs due to the wireless nature of communication within it, from the previous section, we can broadly classify the types of threats into five categories [8].

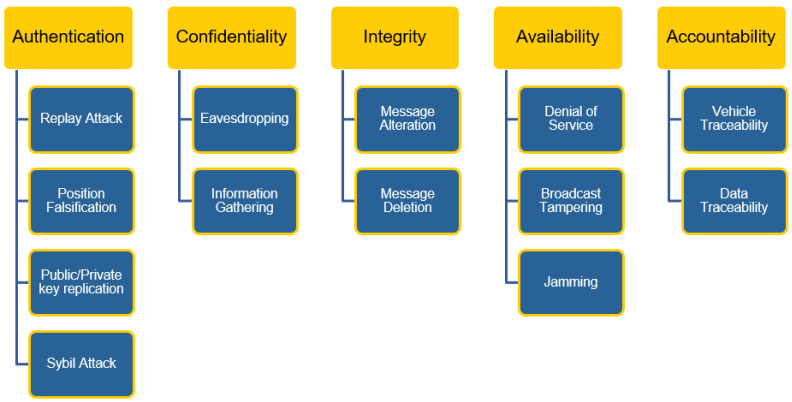

Figure 2.1: Security and Privacy Threats in VANETs [8]

Referring to the above Fig. 2.1, the attacks are introduced below:

- **Authentication:**

  VANETs are susceptible to various authentication attacks due to their dynamic and decentralized nature. These attacks exploit vulnerabilities in the authentication mechanisms designed to verify the identity and integrity of communicating entities within the network.

  There are several types of Authentication Attacks, some of them are:

  - **Replay Attacks:**

    Replay attacks involve re-transmitting previously recorded messages to deceive legitimate nodes. Attackers exploit authentication weaknesses to impersonate valid nodes or inject false data.

- **Position Falsification Attacks:**

  Whenever a vehicle travels, it sends out Basic Safety Messages (BSM) to its neighboring vehicles to ensure that every vehicle has a broad understanding of how the network moves. The attack caused by sending false positional information within the BSM into the network is called the Position Falsification Attack [9].

- **Identity and Certificate Replication:**

  Attackers create multiple nodes with the same identity by replicating management and RF certificates, aiming to confuse authorities and evade detection.

- **Sybil Attacks:**

  A single malicious entity generates multiple fake identities to gain unfair advantages or disrupt network operations, which may lead to confusion or denial of service for legitimate participants. Besides being one of the most dangerous forms of attack, Sybil attack is also among the most difficult to detect [10].

- **Confidentiality:**

  In VANETs (Vehicular Ad Hoc Networks), confidentiality refers to the protection of sensitive information exchanged between vehicles (nodes) and between vehicles and roadside units (RSUs) from unauthorized access or interception by malicious entities.

  Eavesdropping involves a malicious entity (attacker) passively monitoring the communication between vehicles or between vehicles and RSUs [11]. Once the attacker successfully intercepts the messages exchanged between vehicles or between a vehicle and an RSU, they gain access to the contents of these messages. With access to intercepted messages, the attacker can exploit the information for various malicious purposes, such as Tracking, Theft, Sabotage, Identity Theft, Traffic Manipulation, etc.

  In addition to passive eavesdropping, attackers may engage in active attacks by impersonating legitimate nodes or RSUs. Attackers may set up rogue RSUs strategically to intercept communication from passing vehicles. These bogus RSUs can

appear legitimate, leading vehicles to establish connections and exchange sensitive data unwittingly.

- **Integrity:**

  Attacks on integrity in VANETs, such as timing attacks, aim to manipulate the timing or sequence of messages without altering their content. Timing attacks can seriously affect vehicular communication networks' reliability and safety.

  In a timing attack, the attacker deliberately introduces delays or alterations in the timing of messages exchanged between vehicles or between vehicles and RSUs [8]. This manipulation can disrupt the normal flow of communication and lead to various undesirable outcomes.

  For example, delaying the dissemination of congestion warnings to nearby vehicles can lead to increased congestion or accidents as drivers are not adequately informed about road conditions, or delaying emergency messages, such as collision warnings or obstacle alerts, can result in delayed reactions from drivers or autonomous vehicles, increasing the likelihood of accidents or collisions.

- **Availability:** Denial of service (DoS) attack is the most common intrusive attack against the availability [8]. DoS attacks aim to disrupt or degrade the availability and performance of VANET services by overwhelming network resources or exploiting vulnerabilities in communication protocols.DOS can be classified into three levels of attacks, which are (1)Basic level, (2)Extended level, and (3)Distributed Denial of service attack (DDoS) [12].

  Other types of attacks on Availability include Broadcast Tampering, Jamming, Black Hole Attack, etc. [8].

- **Accountability:**

  In this attack, malicious entities aim to compromise the privacy and security of target nodes by exploiting vulnerabilities in the network's communication protocols and security mechanisms [8]. The attacker monitors the communication activities of specific target nodes within the VANET and sends malicious packets or "viruses" to nearby vehicles near the target node. When nearby vehicles receive the virus packets,

their systems may become overwhelmed or compromised. When that happens, they may inadvertently reveal the identity and location information of the target node [13]. The attacker can exploit this information for nefarious purposes, such as tracking the target's movements, impersonating the target, or launching attacks against the target or other network participants.

Our research focuses specifically on accurately detecting ***Sybil*** attacks, which are discussed below.

2.1.2 Sybil Attacks

Vehicular Ad-hoc Networks (VANETs) offer promising advancements in road safety and traffic management. However, their reliance on open wireless communication channels exposes them to security threats, including Sybil attacks.

In a Sybil attack, a malicious entity assumes multiple fake identities (Sybils) within the network, gaining undue influence and disrupting its normal operation. Attackers leverage these fake identities to manipulate data, spread misinformation, and launch further attacks.

For example, a malicious node may misleadingly report a traffic jam or accident to cause a nearby vehicle driver to reroute, potentially causing a real traffic jam on another road. Alternatively, the malicious node may exploit the fake virtual nodes to use a large amount of bandwidth, disrupting communication [14].

Sybil attacks in VANETs can be grouped into three main categories [15] based on how they communicate, their identity, and their involvement in the network:

- **Communication Category:**

  In this type of attack, a malicious vehicle listens in when an honest vehicle sends a message over the radio. Similarly, messages sent from Sybil (malicious) vehicles appear to be sent from another malicious device. Communication with Sybil nodes can be direct, where they communicate directly with legitimate vehicles, or indirect, where they communicate through another malicious vehicle.

- **Identity Category:**

  Attackers create new identities for their malicious vehicles. These identities can either be completely random (fabricated) or mimic the legitimate identity of a nearby vehicle (stolen identity).

- **Participation Category:**

  In this category, multiple Sybil identities created by malicious vehicles can simultaneously take part in an attack, or the attacker can use them simultaneously. Sometimes, a single identity may repeatedly join and leave the network. The number of identities the attacker uses is usually equal to or fewer than the number of physical vehicles. Attacks involving multiple Sybil nodes can seriously disrupt the normal functioning of the network.

As seen in Fig. 2.2, the vehicle with the malicious intent of disrupting the network is shown in yellow color. This attacker vehicle creates multiple "fake" identities of itself within the network (shown using red-colored vehicles). It intends to send out fake BSMs using these fake identities to disrupt the network. The malicious node (attacker) creates multiple false identities or pseudonyms to manipulate network operations, deceive legitimate nodes, and disrupt communication.

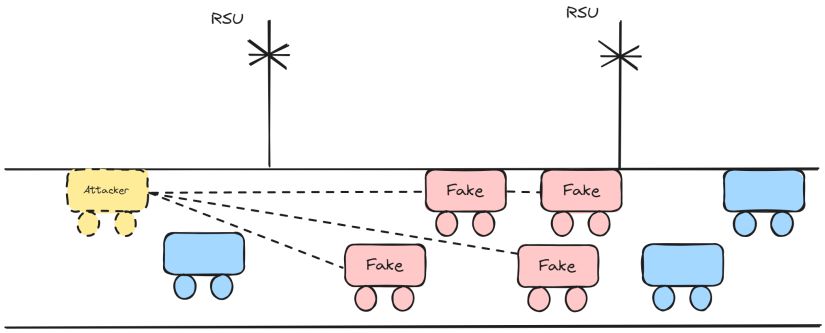

Figure 2.2: Sybil Attack in VANETs

Traffic Congestion Sybil Attack

Traffic Congestion Sybil Attack was first introduced in 2020. It is an attack aimed at creating fake traffic congestion. The attacker generates a grid of fake vehicles in a chosen

position by maintaining a new identity and a correct message frequency for each fake vehicle [16].

2.2 VeReMi Dataset

The VeReMi dataset, also known as the Vehicular Reference Misbehaviour Dataset, is a valuable resource in Vehicular Ad-Hoc Networks (VANETs), specifically designed for evaluating misbehavior detection techniques in this domain. VeReMi separates itself from its predecessors by being the first publicly available and expandable dataset, providing researchers and practitioners with a vital tool for developing state-of-the-art VANET security [17].

VeReMi, which consists of message logs from onboard units (OBUs), is methodically designed to cover a wide range of events and obstacles. It has three separate density levels to accommodate diverse traffic circumstances, three malfunctions, and six different attacks [16]. This diversity allows for thorough testing and validation of misbehavior detection algorithms under various conditions, assuring robustness and dependability in real-world deployment settings.

Table 2.1: Attacker Distribution by Traffic Density in VeReMi

	Dataset ID	**Time Density**	**Attacker Vehicles**	**Attacker Messages**	**Genuine Vehicles**	**Genuine Messages**
[16]	Attack_0709	37.03V/km2	1,220	924,251	2,846	2,221,825
	Attack_1415	16.36 V/km2	505	249,612	1,179	569,723
	MixAll_0024	23.29 V/km2	7,399	7,505,418	17,264	11,951,210

The dataset is developed using a simulated environment called Framework for Misbehavior Detection (F2MD), provided using the LuST scenario [6] and the VEINS (Vehicles

in Network Simulation) framework [7], which is based on the OMNeT++ and SUMO platforms. By modeling actual vehicle interactions and communication protocols, VeReMi accurately depicts the complexities of VANET systems, providing a high-fidelity simulation of potential security risks and vulnerabilities.

VeReMi is built around Basic Safety Messages (BSMs) received by cars in the simulated environment, which are scrupulously maintained with a labeled ground truth file that depicts attacker behavior. This degree of detail enables researchers to precisely analyze and evaluate misbehavior detection techniques, allowing them to successfully discover and address security flaws.

2.3 Overview of Machine Learning

Machine learning is the Artificial Intelligence branch that facilitates machines to perform specific jobs faster and skillfully using statistical learning [18]. It is a branch of artificial intelligence (AI) that focuses on developing algorithms and statistical models that enable computers to learn from and make predictions or decisions based on data without being explicitly programmed [19]. It encompasses various techniques and approaches to enable systems to learn and improve from experience automatically.

One fundamental distinction in machine learning is between supervised and unsupervised learning.

- **Supervised Learning**

  In supervised learning, the algorithm learns from labeled data, where each training example is associated with a corresponding target or outcome. The goal is to learn a mapping from input features to the correct output labels, allowing the algorithm to make predictions on unseen data.

- **Unsupervised Learning**

  On the other hand, unsupervised learning involves learning from unlabeled data, where the algorithm tries to find patterns or intrinsic structures within the data without explicit guidance. This type of learning is often used for tasks such as

clustering, dimensionality reduction, and anomaly detection.

2.3.1 Basic Machine Learning Concepts and Terminologies

Basic terminologies and processes of machine learning [**20**] used in this thesis are defined below:

1. **Data Preprocessing:** Refers to the cleaning, transforming, and normalizing raw data to prepare it for machine learning algorithms, often involving tasks such as handling missing values, encoding categorical variables, and scaling numerical features.

2. **Feature Engineering:** The process of creating new features or modifying existing ones from the raw data to improve the performance of machine learning models, typically involving techniques such as feature selection, dimensionality reduction, and creating interaction terms.

3. **Hyperparameter Tuning:** The process of optimizing the hyperparameters of machine learning algorithms to improve their performance on unseen data is often done using techniques such as grid search, random search, or Bayesian optimization.

4. **Exploratory Data Analysis (EDA):** An initial step in data analysis aimed at exploring and understanding the structure, patterns, and relationships within the data using statistical graphics, visualization techniques, and summary statistics.

5. **Train-Test Split:** The dataset division into separate training and testing subsets, where the training set is used to train the machine learning model, and the testing set is used to evaluate its performance on unseen data.

6. **Train Set:** The portion of the dataset used to train the machine learning model by feeding input data along with corresponding labels or target variables to learn patterns and relationships.

7. **Test Set:** The portion of the dataset reserved for evaluating the performance of the trained machine learning model on unseen data is typically used to assess its generalization ability and predictive accuracy.

8. **Cross-Validation:** A resampling technique used to assess the performance of machine learning models by splitting the data into multiple subsets, training the model on different subsets, and averaging the results to obtain a more reliable performance estimate.

9. **Overfitting and Underfitting:** Common issues in machine learning where the model either learns the training data too well (overfitting), leading to poor generalization on unseen data or fails to capture the underlying patterns in the data (underfitting), resulting in low predictive performance.

2.3.2 Classification Models in Machine Learning

Within supervised learning, one common task is classification, where the goal is to predict the categorical class labels of new instances based on past observations. Classification algorithms aim to learn a mapping from input features to discrete output classes, allowing the algorithm to classify new data points into predefined categories. These algorithms are trained on labeled data, where each instance is associated with a class label, and they learn to distinguish between different classes based on the features provided. Some popular classification algorithms include **K-Nearest Neighbors [21], Decision Trees [22], Random Forests [23], support vector machines (SVM), and Logistic Regression.**

Each algorithm has its strengths and weaknesses, and the choice of algorithm often depends on factors such as the data's nature, the dataset's size, and the desired level of interpretability or complexity. In detecting attacks in vehicular ad hoc networks (VANETs), classification algorithms can be applied to learn normal behavior patterns and distinguish between legitimate and malicious activities based on features extracted from network traffic or sensor data. These algorithms play a crucial role in developing effective intrusion detection systems (IDS) for ensuring the security and integrity of VANETs.

Given the nature of the dataset and the specific problem of detecting attacks in vehicular ad hoc networks (VANETs), three classification models have shown promising performance: K-Nearest Neighbors (KNN), Decision Trees, and Random Forest.

- **K-Nearest Neighbours**

  KNN is a simple yet powerful algorithm for classification and regression tasks. It operates on the principle of similarity, where a new data point is classified based on the majority class of its nearest neighbors in the feature space. It tries to classify an unknown sample based on the known classification of its neighbors [21]. In the context of VANET security, KNN can be effective because it doesn't assume any underlying probability distribution of the data and can capture complex decision boundaries. However, its performance may degrade with high-dimensional data or imbalanced class distributions.

  In the context of the K-Nearest Neighbors (KNN) algorithm, the process described in Fig. 2.3 represents the fundamental principle of how KNN determines the classification of a target point. In KNN, the algorithm classifies a data point by considering the class labels of its nearest neighbors in the feature space. This approach allows KNN to classify data points based on the consensus of their nearest neighbors in the feature space, making it a simple yet effective algorithm for classification tasks. However, the choice of 'k' and the distance metric used can significantly impact the performance of KNN, and careful parameter tuning is often necessary to achieve optimal results.

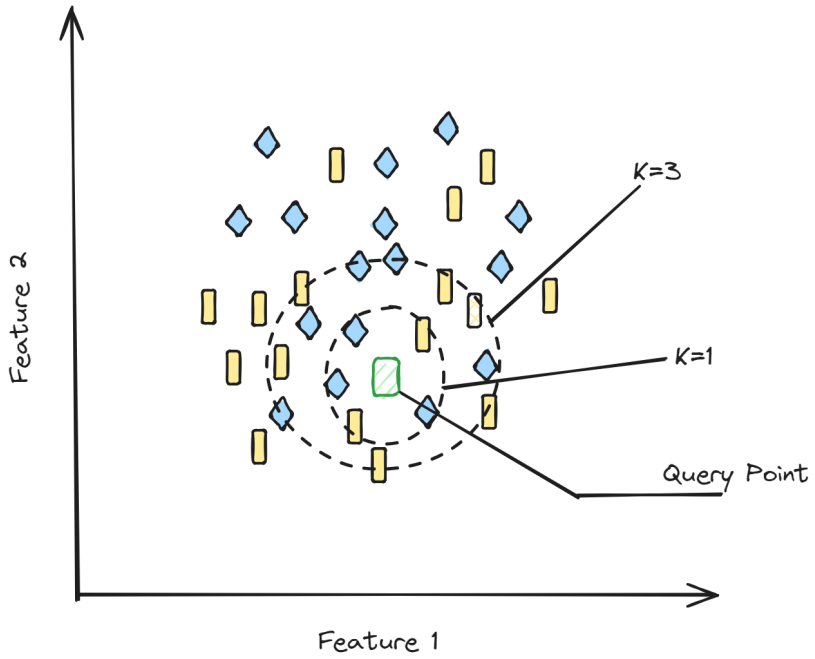

Figure 2.3: K-Nearest Neighbours

- **Decision Trees**

  Decision trees are non-parametric supervised learning models used for classification and regression tasks. They partition the feature space into disjoint regions based on the values of input features, and each region is associated with a specific class label [24]. Decision trees are interpretable and easy to understand, making them suitable for explaining the logic behind classification decisions. However, they are prone to overfitting, especially when the tree depth is not properly controlled.

  In the context of decision trees, Fig. 2.4 likely represents the hierarchical structure of decision-making within the algorithm. They recursively partition the feature space into smaller regions, making decisions at each node based on the features' values. At each node of the decision tree, a decision is made based on a particular feature and a threshold value. The decision made at each node creates new branches of possible decisions to be made. Each branch corresponds to a specific range or category of the evaluated feature. As the data points traverse the decision tree branches, they eventually reach a leaf node. A leaf node makes no further decisions but represents a final classification or regression decision. This hierarchical decision-

making structure allows decision trees to partition the feature space effectively and make predictions based on simple rules at each node. However, the decision tree's complexity can vary depending on factors such as the depth of the tree, the number of features, and the splitting criteria used.

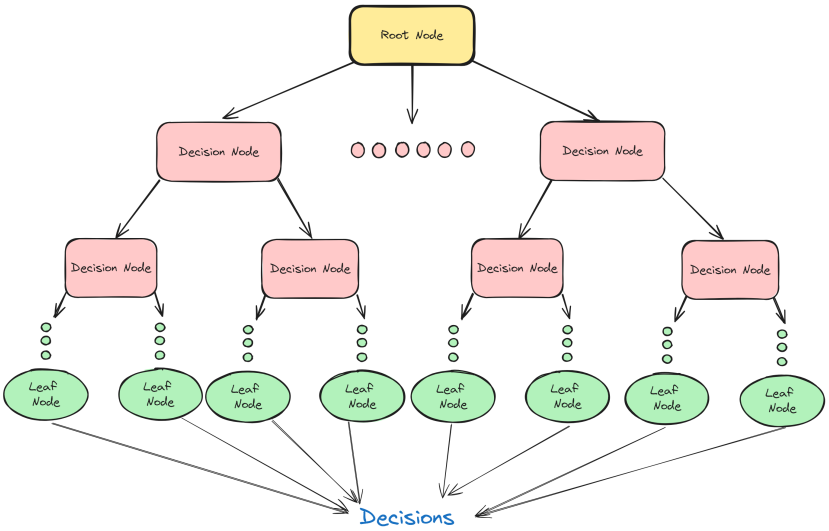

Figure 2.4: Decision Trees

- **Random Forest**

  Random Forest is an ensemble supervised learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or the individual trees' mean prediction (regression). It mitigates the overfitting problem of decision trees by averaging the predictions of multiple weak learners. Random Forest is robust to noise and outliers and can handle high-dimensional data effectively. Moreover, it provides an estimate of feature importance, which can be valuable for understanding the underlying factors contributing to classification decisions. However, Random Forest models may be computationally expensive and require hyperparameter tuning to optimize performance.

  The number of trees necessary for good performance grows with the number of predictors. The best way to determine how many trees are necessary is to compare predictions made by a forest to predictions made by a subset of the forest. You have enough trees when the subsets work, and the full forest [23].

As depicted in Fig. 2.5, multiple decision trees are created, and during classification, each tree independently evaluates the input data. The final classification decision is then determined through an aggregation process, where the mode of the classes predicted by individual trees is taken as the overall prediction.

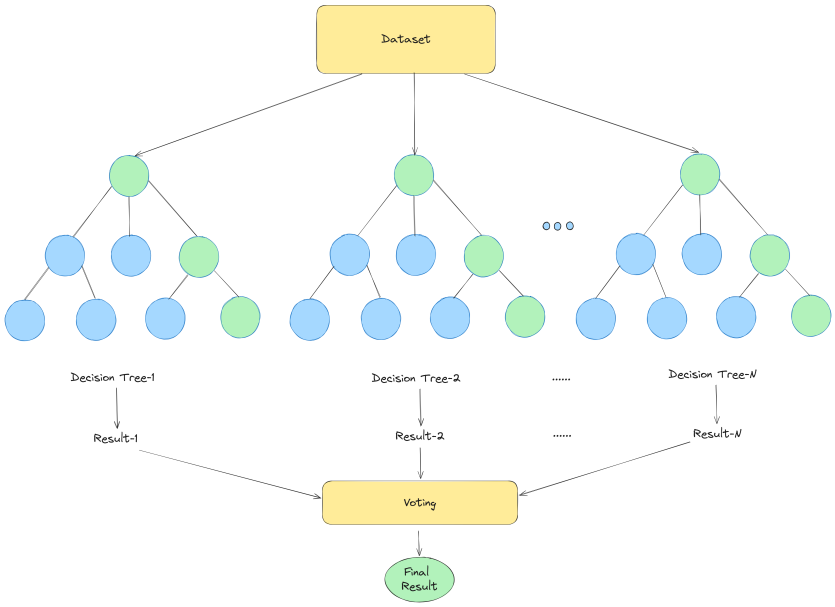

Figure 2.5: Random Forest

2.4 Literature Review

Detecting misbehavior (such as transmission of misleading information) in vehicular ad hoc networks (VANETs) is a problem with numerous ramifications., including safety-related and congestion-avoidance applications. Most MDS are concerned with the detection of malicious nodes. In most situations, vehicles send wrong information for selfish reasons by their owners, e.g., to gain access to a particular lane. It is, therefore, more important to detect false information than to identify misbehaving nodes [25].

In vehicle communication networks, detecting misbehavior is crucial, but it's no easy task. Many types of attacks can happen, and each one needs its way of being spotted. We must understand these attacks well before we can find effective solutions. That's why we must first figure out all the different kinds of attacks that could happen. Only then

can we start to create smart ways to catch them.

2.4.1 VANET and Attacks in VANET

Codeca, Lara, et al. [6] addressed creating a realistic traffic scenario suitable for vehicular networking research. Recognizing the importance of accurate simulations in evaluating network protocols and applications, they devised the Luxembourg SUMO Traffic (LuST) Scenario. Leveraging real-world data from the City of Luxembourg, including OpenStreetMap and government statistics, they meticulously constructed a comprehensive scenario encompassing road topology, demographics, mobility, and traffic patterns. Their work culminated in the development of LuST, providing the research community with a freely available framework for studying vehicular networking and facilitating the comparison of protocols and applications.

Since this was still a simulation and not a concrete set of data that could be produced using which we could analyze patterns and differentiate attacks, van der Heijden, Rens W., et al. [17] recognized the need for a standardized dataset to enable fair and comparable evaluation of different misbehavior detection approaches. To address this, they introduced the first-ever publicly extensible dataset allowing anyone to reproduce the generation process, contribute attacks, and use the data to compare new detection mechanisms against existing ones, called VeReMi - Vehicular Reference Misbehaviour Dataset. It comprises realistic vehicular communication traces enriched with various malicious activities. VeReMi [17] leveraged LuST [6]scenario to generate realistic vehicular communication traces enriched with various types of misbehavior, enabling researchers to evaluate misbehavior detection techniques in a representative urban traffic environment.

Hezam et al. [8] categorized different types of security and privacy threats in VANETs and mentioned that to maintain secure vehicular communication and networks, the VANET security system should satisfy the requirements mentioned as the categories of different threats in section 2.1.1. However, this is still a broad categorization of threats.

While introducing the VeReMi dataset, van der Heijden, Rens W., et al. [17] also introduced a set of 5 different types of *position falsification* attacks in VANETs which could be reproduced in a simulation by anyone utilizing the VeReMi dataset.

An extension of the VeReMi Dataset was introduced in 2020, where van der Heijden, Rens W., et al. [16] and mention that while message authenticity is ensured through digital certificates, the correctness of the information exchanged remains uncertain. Existing studies lack a common reference dataset, making comparing and validating results challenging. The authors extend the original VeReMi dataset to address its limitations. They introduce a realistic sensor error model, enhance the dataset with more data points, and incorporate a new set of attacks. These enhancements aim to improve the dataset's realism and usefulness for researchers in the field of misbehavior detection in VANETs. The paper introduces several new types of attacks to the VeReMi dataset:

- **Denial-of-Service (DoS) Attacks:** Vehicles send messages at a frequency higher than the standard limit.
- **DoS Random:** DoS attacks with all message fields set to random values.
- **Data Replay:** Sending previously received information from a specific target neighbor, signed with the attacker's certificate.
- **Disruptive Attacks:** Information replay of previously received data from random neighbors.
- **Sybil Attacks:** Vehicles pretend to be multiple entities on the road simultaneously.
- **Traffic Congestion Sybil:** Creates fake traffic congestion by generating a grid of fake vehicles.

Consequently, this paper is the first-ever time the "Traffic Congestion Sybil Attack" was formally introduced in VANETs.

2.4.2 Misbehavior Detection in VANET

Ghosh et al.[26] propose a misbehavior detection scheme tailored for Post Crash Notification (PCN) applications within vehicular ad hoc networks (VANETs), aiming to address the problem of detecting and evicting incorrect messages, which can lead to false crash alerts. The scheme relies on observing driver behavior post-alert reception, distinguishing genuine crash alerts from false ones by comparing the actual vehicle trajectory with the expected trajectory following a crash. If the deviation exceeds a threshold, indicating a false alert recognized by the driver, it's classified as false; otherwise, it's deemed true. Limitations include reliance on accurate position information in false alerts, potential variability in effectiveness across mobility dynamics and applications, and assumption of a Markovian mobility model, potentially restricting applicability in diverse scenarios.

In their work on misbehavior detection in VANETs, Ruj et al.[25] adopt a novel approach centered on observing vehicle alert messages and scrutinizing subsequent actions to discern inconsistencies indicative of false alerts. Unlike conventional schemes focusing on node classification or revocation, their method takes a data-centric stance, distinguishing between accurate and erroneous information rather than categorizing nodes as trustworthy or untrustworthy. By eschewing the revocation of misbehaving nodes or their credentials, Ruj et al.'s approach conserve communication bandwidth and computational resources while safeguarding location privacy through pseudonyms. However, the efficacy of their approach presupposes that misbehavior primarily arises from self-serving motives, potentially overlooking malicious intent. Moreover, the reliance on precise location data poses a vulnerability, as nodes could disseminate false location information alongside false alerts, undermining the system's accuracy.

van der Heijden et al. [17] propose misbehavior detection techniques tailored for VANETs, leveraging vehicular reputation systems to assess participating vehicles' trustworthiness based on past behavior and interactions. Reputation-based approaches, including direct and indirect trust evaluation mechanisms, enable vehicles to exchange rep-

utation reports and make informed decisions about neighboring nodes' trustworthiness. Additionally, a collaborative detection scheme aggregates reputation information from multiple sources to enhance detection reliability and accuracy, mitigating malicious activities and improving overall VANET security.

However, drawbacks persist despite the promising aspects of the proposed techniques. Firstly, reliance on reputation systems introduces vulnerabilities related to reputation information accuracy and integrity, as malicious nodes may manipulate scores to disguise their misbehavior or tarnish legitimate vehicles' reputations. Scalability issues and communication overhead, especially in large-scale VANETs with high mobility, pose additional challenges. Secondly, the collaborative nature of detection mechanisms raises concerns about privacy and data confidentiality, as vehicles must share sensitive information, potentially compromising privacy and leading to data leakage. Moreover, the dynamic nature of vehicular environments, including network congestion and environmental conditions, can impact detection algorithm accuracy, further complicating misbehavior detection effectiveness.

In their newly proposed extension of the VeReMi dataset [16], van der Heijden et al. present an extension to the VeReMi dataset for misbehavior detection in VANETs, aiming to address the lack of a common reference dataset in the field. The approach involves employing a misbehavior detection system that combines local plausibility detectors with a fusion detection mechanism. This system conducts plausibility and consistency checks on received messages, analyzes them using threshold mechanisms, and classifies vehicles as misbehaving based on predefined criteria. Compared to other approaches, the extension provides a comprehensive and standardized dataset for evaluating misbehavior detection techniques in VANETs. However, limitations include the dependency on predefined thresholds and the simplicity of the detection algorithm, which may not capture all forms of misbehavior accurately, especially in complex real-world scenarios.

2.4.2.1 Misbehavior Detection using Machine Learning in VANET

Alzahrani et al. [27] propose an enhanced misbehavior detection scheme for Vehicular Ad Hoc Networks (VANETs) using a combination of Kalman filtering and Artificial Neural

Networks (ANNs). The approach involves data acquisition, feature extraction, and misbehavior detection. Kalman filtering is employed to integrate signal properties like angle of arrival (AoA) and received signal strength indicator (RSSI) with Cooperative Awareness Messages (CAMs) for feature extraction, focusing on the innovation errors as indicative of misbehavior. These features are then input into an ANN classifier trained on a labeled dataset generated through simulation. Compared to other approaches, this method replaces static detection thresholds with a dynamic ANN-based classifier, providing better adaptability to changing network conditions and potentially improving accuracy. However, limitations include relying on supervised learning in a dynamic environment, excluding certain types of attacks like tampering with transmission range, and the need for further evaluation in scenarios with higher mobility.

So et al. [28] propose a novel approach for misbehavior detection in Vehicular Ad Hoc Networks (VANETs) by integrating plausibility checks with machine learning techniques. The method involves two main phases: plausibility checks and machine learning-based detection. In the plausibility checks phase, the system examines the received messages for consistency with expected behavior based on predefined rules and thresholds, aiming to filter out invalid messages. Then, in the machine learning-based detection phase, the remaining messages are analyzed using machine learning algorithms to identify subtle or complex misbehaviors. The system can effectively detect misbehavior types by combining these two approaches while minimizing false positives. However, the approach relies on the availability of accurate models for expected behavior, and the effectiveness may depend on the quality of the training data and the chosen machine learning algorithms. Additionally, the system may face challenges distinguishing between genuine anomalies and intentional misbehavior designed to evade detection.

Kamel et al. [29] propose an enhanced misbehavior detection mechanism, named CaTch, for Vehicular Ad Hoc Networks (VANETs), focusing on embedded misbehavior detection mechanisms in C-ITS. The approach incorporates physical measurement uncertainty into plausibility detectors, allowing for a more accurate assessment of the credibility of received messages. By simulating scenarios involving simple offset faults and more complex Sybil attacks, the study demonstrates that CaTch outperforms legacy detectors,

particularly in detecting subtle misbehavior where attackers attempt to remain within plausible ranges. CaTch provides an uncertainty factor for each detector, aiding intelligent detection applications in making more informed decisions. However, the approach requires additional computational power and consideration for potential manipulation of the uncertainty factor by attackers. It is noted that manipulating CaTch is more complex than manipulating legacy detectors. Thus, while CaTch offers improved detection capabilities, its effectiveness hinges on developing intelligent decision-making algorithms and carefully considering computational resources and potential attack vectors.

Grover et al. [30] propose an ensemble-based machine learning approach for detecting misbehaviors in Vehicular Ad Hoc Networks (VANETs), a critical aspect given the potential safety implications of inaccurate messages transmitted by illegitimate vehicles. Leveraging features extracted from experiments conducted in the NCTUns-5.0 simulator, the approach utilizes classifiers such as Naive Bayes, Instance-Based Learner (IBK), Random Forest, Decision Tree (J-48), and AdaBoost to classify nodes as legitimate or malicious. By employing a majority voting scheme, the ensemble method combines the strengths of individual classifiers, thereby enhancing detection accuracy. The paper highlights the importance of accurately classifying various misbehaviors, including identity spoofing, position forging, packet suppression, replay, and detention attacks, to mitigate potential safety hazards in VANETs. Compared to traditional methods, the ensemble-based approach demonstrates superior classification accuracies, particularly in improved true positive and negative rates. However, the paper acknowledges limitations such as the need for extensive training data and the challenge of adapting the framework to different types of misbehavior.

Sharma et al. [9] propose an approach for detecting misbehavior in VANETs that utilizes a Machine Learning (ML)-based framework that focuses specifically on position falsification attacks, where vehicles transmit incorrect position information in Basic Safety Messages (BSMs). Unlike traditional cryptographic techniques, which may not effectively detect such attacks, the proposed "2BSM approach" combines information from consecutive BSMs sent by a vehicle to enhance accuracy. The key innovation lies in deploying the misbehavior detection framework at Road Side Units (RSUs) instead of individual vehi-

cle On-Board Units (OBUs), leveraging the RSUs' superior computational resources and enabling network-wide information sharing. The approach effectively classifies vehicles as legitimate or malicious by analyzing features extracted from BSMs and training ML models, achieving high precision and recall rates across various attack types. Moreover, the approach addresses the challenge of imbalanced datasets and incorporates hyperparameter tuning and cross-validation techniques for model optimization. However, limitations include reliance on RSU infrastructure, making it more suitable for urban areas, and the assumption of real-time access to a shared database, which may not always be feasible. Additionally, while the approach demonstrates robust performance against known attack types, its effectiveness against emerging threats and other types of misbehavior beyond position falsification requires further exploration, potentially by developing more sophisticated deep learning-based techniques.

Gyawali et al. [31] propose a machine learning-based misbehavior detection system for Vehicular Ad-hoc Networks (VANETs) to address attacks such as false alert generation and position falsification. Unlike traditional cryptographic methods, which are less effective against insider attacks, the proposed system leverages machine learning techniques trained on realistic VANET simulation datasets. The VeReMi dataset is utilized to derive labeled datasets for training and testing the position verification scheme, which aims to detect various types of position falsification attacks in VANETs. The derived datasets are then used to compare the performance of the proposed scheme with the detectors used in VeReMi for different position attacker types. The system can accurately identify malicious behavior by utilizing features extracted from vehicle-to-vehicle communication and applying algorithms like decision trees and random forests. The machine learning approach offers the advantage of detecting known and unknown attacks compared to signature or specification-based systems. However, the proposed system's effectiveness relies heavily on the quality and representativeness of the training data, and it may face challenges in scenarios with rapidly changing network topologies and resource-constrained environments.

Khot et al.[32] proposed a machine-learning framework to predict the next position of the vehicle in the network. The authors used beacon messages from neighboring vehicles

and created features such as distance between sender and receiver. Machine learning algorithms were utilized to train and test them. The authors compared the predicted value with the actual value of the vehicle comparison. If the position is not equal to the prediction, it is classified as an attack vehicle. The authors claimed that Random Forest performs best among other algorithms.

2.4.3 Detecting Sybil Attacks in VANET

Ayaida et al. [33] propose an approach to detect Sybil attacks in Cooperative Intelligent Transport Systems (C-ITS) by leveraging traffic flow theory. Unlike other approaches, which may rely on resource testing or Public Key Infrastructure (PKI), this method utilizes the inherent characteristics of traffic flow models to estimate vehicle speeds and detect anomalies indicative of Sybil attacks. The approach can identify discrepancies that signal potential attacks by comparing real vehicle speeds with those estimated using V2V communications and traffic flow models. The algorithm is distributed, efficient, and does not require additional hardware or complex infrastructure, making it suitable for implementation in real-world vehicular networks. However, it does have limitations, such as the dependence on accurate traffic flow models and the need for a sufficient number of neighboring vehicles for reliable detection, especially in congested traffic scenarios. Additionally, it may struggle to detect attacks when the number of attackers is low or when the speed threshold for detection is set too high, leading to false negatives.

Baza et al. [34] proposed an approach for detecting Sybil attacks in VANETs involving leveraging proofs of work (PoW) and location information obtained from multiple Roadside Units (RSUs) encountered by vehicles. Unlike traditional methods that rely on signatures from individual RSUs, this approach requires contributions from at least t ('t' represents the minimum number of Roadside Units (RSUs) required to contribute their location information) RSUs using a threshold signature scheme, thereby mitigating the risk of RSU compromise attacks. A PoW algorithm prevents malicious vehicles from creating multiple Sybil trajectories simultaneously. Experimental results and a mathematical model validate the approach's effectiveness, demonstrating high detection rates and low false negative rates with short detection times. Compared to existing methods,

the proposed approach offers better resilience against attacks and provides acceptable computation and communication overheads. However, it still has limitations, such as the reliance on dense RSU deployment for effectiveness and the need for careful parameter tuning to balance detection accuracy and computational overhead.

Hammi et al. [35] present a novel approach for detecting Sybil attacks in Cooperative Intelligent Transportation Systems (C-ITS) using machine learning classifiers within a VANET context. The approach involves a three-step process: data monitoring and collection, data aggregation, and classification of stations' activity based on aggregated data matrices. The detection system operates at the Road Side Unit (RSU) level, leveraging the RSUs' computational capabilities to alleviate the burden on vehicles. The approach's effectiveness is demonstrated through extensive simulations using real-world datasets, showing high accuracy in detecting Sybil attacks, particularly those involving random or static values. However, the approach's performance diminishes when faced with more complex attack scenarios, such as replayed values. The paper acknowledges the need for a decentralized approach to scale with the distributed nature of C-ITS environments, suggesting future work in this direction.

Laouiti et al.[36] present a focus on detecting misbehavior in VANETs, specifically targeting the Sybil attack, a significant threat in vehicular networks. It introduces a method that leverages machine learning algorithms and features derived from vehicle behavior data to identify Sybil nodes. By preprocessing the data into Driving Pattern Matrices (DPM) and computing their eigenvalues, the proposed method enhances the accuracy and efficiency of detection compared to previous approaches. Including features such as predicted position and acceleration variations and introducing a plausibility factor significantly improves detection performance. The approach outperforms other methods regarding F1-score and accuracy, achieving notable results of 89% and 87%, respectively. However, the main limitation lies in the time-consuming nature of preprocessing, which can be mitigated with more powerful computing resources.

2.4.3.1 Detecting Traffic Congestion Sybil Attack

Mahmoudi et al.[37] proposed a comprehensive work on a misbehavior detection framework tailored to C-ITS. Their primary objective was to bolster system security using ML techniques at the Misbehavior Authority (MA) level. Through a multi-step process encompassing local misbehavior detection, reporting to the MA, global detection, and subsequent reaction, the framework aimed to classify reported ITS-S behavior as misbehaving, faulty, or genuine, thereby enhancing the overall security posture of C-ITS. To evaluate their approach, the authors utilized the F2MD framework within the VEINS extension for simulations, leveraging OMNeT++ for network simulation and SUMO for road traffic simulation. While the VeReMi dataset was not employed, training data was sourced from the Luxembourg SUMO Traffic (LuST) scenario, with testing conducted on a test bench featuring random vehicle trace data extracted from Paris-Scalay. In assessing the efficacy of their framework, various ML algorithms, including XGBoost, LightGBM, and neural networks, were employed, selected for their robustness and ability to model time-dependent data effectively. However, algorithms like Random Forests and Gradient Boosted trees, offering insights into feature relevance, were sparingly used due to limitations in modeling time-dependent data in a purely supervised fashion. Despite these challenges, the study reported promising results, with the proposed LSTM-based solution achieved an exceptional overall detection accuracy of 97% for various types of misbehavior in C-ITS. However, the study acknowledged certain limitations, including reliance on synthetic data for evaluation and addressing challenges associated with real-world deployment scenarios.

Zhao et al.[38] proposed a FedMix approach for detecting Sybil attacks within VANETs. Their method ingeniously combines federated learning with cross-layer information fusion to bolster privacy protection. By adeptly training detection models without transmitting sensitive data such as BSM, Zhao et al. leverage physical and application layers data. Their experimental results showcase FedMix's efficacy, surpassing centralized training methods in detection accuracy, precision, recall, and F1-score, while notably reduc-

ing communication costs. Furthermore, Zhao et al. introduce the SFedAvg aggregation algorithm, which outperforms the baseline FedAvg in terms of efficiency. Leveraging both VeReMi and BSMList datasets (simulated using F2MD simulator), Zhao et al. enhance FedMix's robustness, having partially extended the VeReMi dataset by adding two fields: the location and the received signal strength when the vehicle receives the message. These enhancements, aggregated with the BSMList dataset, contribute to FedMix's performance. Notably, Zhao et al. achieved recall and F1-score values of 85% and 91%, respectively, showcasing FedMix's exceptional effectiveness in Sybil attack detection. Despite these advancements, potential limitations such as reliance on communication infrastructure and vulnerabilities in federated learning setups warrant further consideration.

Alladi et al.[39] propose a Deep Learning-based Anomaly Detection Framework (DeepADV) tailored specifically for Vehicular Ad-Hoc Networks (VANETs), aiming to detect various anomalies, including faults and attacks, within VANET data sequences. Their approach involves training six different Deep Neural Network (DNN) architectures exclusively on genuine data sequences, utilizing Mean Absolute Error (MAE) as the loss function for backpropagation. The trained models are then employed as sequence reconstructors in fault-only, attack-only, and combined anomaly scenarios, with evaluation metrics such as Precision, Recall, Accuracy, and F1-score used to assess performance. The CNN-LSTM model (M-1) consistently outperforms others across all scenarios. The results showcase high accuracy, precision, recall, and F1 scores for detecting genuine and anomalous classes, highlighting the framework's effectiveness in detecting anomalies, including attacks, within VANET data sequences. However, the requirement for fixed-length message subsequences is a limitation of their approach.

Liu et al.[40] proposed the SVMDformer, a data-centric framework for vehicular misbehavior detection in VANETs, leveraging semi-supervised learning and Transformer architecture. Their approach involves transforming vehicular message sequences into misbehavior scores and classifying misbehavior using a threshold. Through comprehensive experiments, they compared three supervision methods and found semi-supervised learn-

ing to be the most effective. They analyzed and discussed individual misbehaviors by visualizing scores, and identifying the optimal training volume for difficult-to-detect types. Compared to baseline models like CNN and Bi-LSTM, their SVMDformer demonstrated superior performance in metrics such as AUC, accuracy, precision, and F1 score, notably reducing false detection rates. However, the SVMDformer can only determine whether a vehicle exhibits misbehavior without identifying the exact type, leaving room for future improvements in misbehavior type identification and efficient deployment in IoV simulation systems.

Reviewing the literature on misbehavior detection in Vehicular Ad-Hoc Networks (VANETs), we note that there is a wide range of attacks that have been considered. While some studies employ non-machine learning (ML) methods for this purpose, their effectiveness diminishes as the scale of VANETs increases. Additionally, certain research works rely on privately simulated datasets, rendering their findings less reproducible and applicable in real-world scenarios. Furthermore, most existing approaches target attacks other than Traffic Congestion Sybil Attacks, leaving this specific threat largely unaddressed. To the best of our knowledge, the only works done in Traffic Congestion Sybil Attack has been mentioned in this section. While some works utilize privately simulated datasets, others use public datasets like VeReMi coupled with detection algorithms using Deep Learning since they claim that algorithms like Random Forests have limitations in modeling time-dependent data in a purely supervised fashion.

Consequently, my thesis identifies this gap in the literature on detecting Traffic Congestion Sybil Attacks, particularly using the VeReMi dataset and classical ML approaches. This research aims to fill this crucial gap by proposing innovative classical ML-based techniques tailored to detect and mitigate Traffic Congestion Sybil Attacks in VANETs while utilizing the VeReMi dataset. This decision was driven by the need to ensure that the findings and methodologies are easily reproducible for further validation and refinement.

Table 2.2 compares various research papers in the literature review related to detecting Sybil attacks in VANETs. The table evaluates different aspects of each paper, including whether machine learning techniques were utilized, whether the VeReMi dataset was

employed, the type of attack addressed in the paper, and the approach proposed to detect attacks.

Table 2.2: Comparison table of Literature Review

No.	**Paper**	**Machine Learning Used?**	**VeReMi Dataset Used?**	**Attack**	**Approach**
1	Heijden et al.	No	Yes	Position Falsification; DoS; Sybil	Trust based approach
2	Kamel et al.	Yes	No	Sybil; Constant Offset	Embedded MBD; Integrates data inaccuracy
3	Baza et al.	No	No	Sybil	Proofs of Work
4	Hammi et al.	Yes	No	Sybil	ML based classifiers
5	Laouiti et al.	Yes	Yes	Sybil	Adaboost classifiers
6	Mahmoudi et al.	Yes	No	Traffic Congestion Sybil	Deep Learning
7	Zhao et al.	Yes	No	All	Federated Machine Learning
8	Alladi et al.	Yes	No	All incl. Traffic Congestion Sybil	Deep Learning
9	Liu et al.	Yes	Yes	All incl. Traffic Congestion Sybil	Semi-supervised, transformer-based Deep learning
10	**Proposed Method**	Yes	Yes	Traffic Congestion Sybil	Custom split using classical ML

CHAPTER 3

Machine Learning Based Classification of Attacker Vehicles

3.1 Introduction

In Vehicular Ad-Hoc Networks (VANETs), ensuring the integrity and security of communication systems is paramount, particularly in the face of emerging threats such as Traffic Congestion Sybil Attacks. These attacks, a sub-type of Sybil Attacks, involve malicious entities creating multiple false identities to simulate traffic congestion artificially, leading to disruption and potentially dangerous situations on the road. This research addresses this critical challenge by proposing a novel methodology for detecting Traffic Congestion Sybil Attacks in VANETs utilizing machine learning (ML) approaches specifically applied to the VeReMi dataset. VANETs represent a dynamic and complex network environment where vehicles communicate with each other and with infrastructure components to enhance transportation efficiency and safety. However, the proliferation of digital threats, including Sybil Attacks, poses significant risks to the reliability and safety of VANETs. Despite the increasing attention to security issues in VANETs, the specific threat of Traffic Congestion Sybil Attacks has been relatively overlooked in the existing literature. Through this research, we aim to bridge this gap by introducing a comprehensive detection methodology explicitly tailored to address Traffic Congestion Sybil Attacks. This approach is motivated by the observed limitations in previous works, where either no direct focus on this attack vector was given or methodological constraints hindered evidence, reproducibility, and comparison with other detection techniques. Central to

our methodology is the application of classical machine learning algorithms, which, while proven effective, are augmented by innovative techniques for dataset interpretation and feature engineering. The proposed methodology aims to:

- Provide a framework to detect traffic congestion Sybil attacks with better Recall and F-1 score but with less computational overhead
- Introduce a custom data split method to preserve temporal features of the dataset during the training phase while using classical machine learning algorithms
- Incorporate new features (extended from the existing ones) into the dataset to improve the generalizability and effectiveness of ML models.
- Detect vehicles that create false identities of themselves with a malicious intent to disrupt the VANET, with high recall and F-1 score.

3.2 Outline of Proposed Approach

The proposed methodology has four main stages:

- Data Extraction
- Exploratory Data Analysis and Data Pre-processing
- Feature Engineering
- Classification using Machine Learning models

A detailed discussion of these four stages is as follows:

3.2.1 Data Extraction

The VeReMi dataset has several simulations with different traffic scenarios for each type of attack. Each simulation consists of two types of files: Ground truth file and log files. There is only one ground truth file in a simulation that includes the vehicle's actual behavior in the network. Ground truth file also comprise an attack type, differentiating

legitimate vehicles from misbehaving vehicles. On the other hand, the number of log files in a simulation is equal to the number of vehicles in the network. Each sender vehicle creates a log file that includes all the BSMs sent to other vehicles.

To extract the dataset from the publicly available VeReMi dataset's repository[41], we need to do the following:

- For a single simulation, the number of log files equals the number of senders; hence, the first step is to combine the separate log files into a single file.
- Each log file consists of BSMs in the JSON format. Hence, a consolidated log file must first be generated with all the BSMs.
- Once we have the JSON, we need to convert this consolidated JSON file into a CSV format. The conversion of data from JSON to CSV format enhances compatibility with machine learning frameworks, facilitating seamless integration and computational processing.

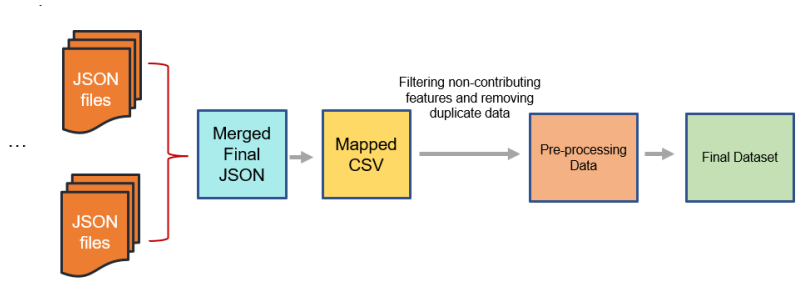

Figure 3.1: Dataset Preparation Flow

The resultant consolidated dataset serves as a robust foundation for exploratory data analysis that can be used to discern patterns and anomalies critical for informing detection mechanisms within VANETs.

3.2.2 Exploratory Data Analysis (EDA) and Data pre-processing

Before we begin the pre-processing steps, let's understand the consolidated dataset first.

- There are **1,501,368 rows** and **26 columns** in the consolidated dataset.

- **4 columns were dropped** since their x, y, and z coordinate splits were available in the dataset
- On checking for the uniqueness of values across the dataset, a **further 6 columns were dropped** since their value was constant or similar to another column throughout the 1.5 million rows of the dataset.
- There are **more BSMs related to attacker vehicles than non-attacker instances**, as shown in the Fig. 3.2. This is coherent with the idea that since multiple attacker vehicles create fake identities of themselves, the number of BSMs transmitted by those would be higher than that of non-attacker vehicles.

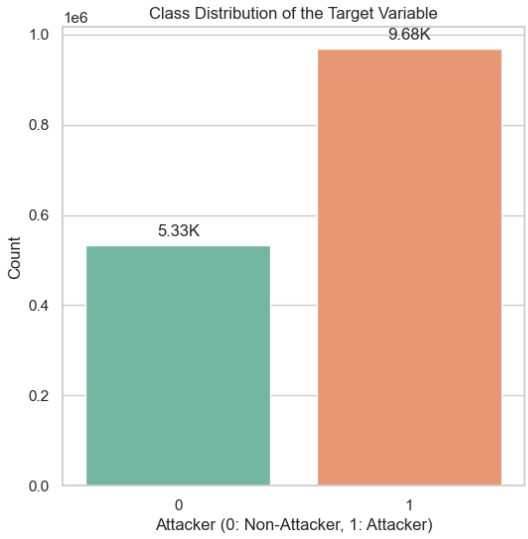

Figure 3.2: Distribution of target variable

- Without feature engineering included, we find that the **senderPseudo** features have the highest correlation with the target variable. This could imply that attackers use certain pseudonyms more frequently or change pseudonyms in a specific pattern. It can also be verified from the correlation matrix shown in Fig. 3.3:

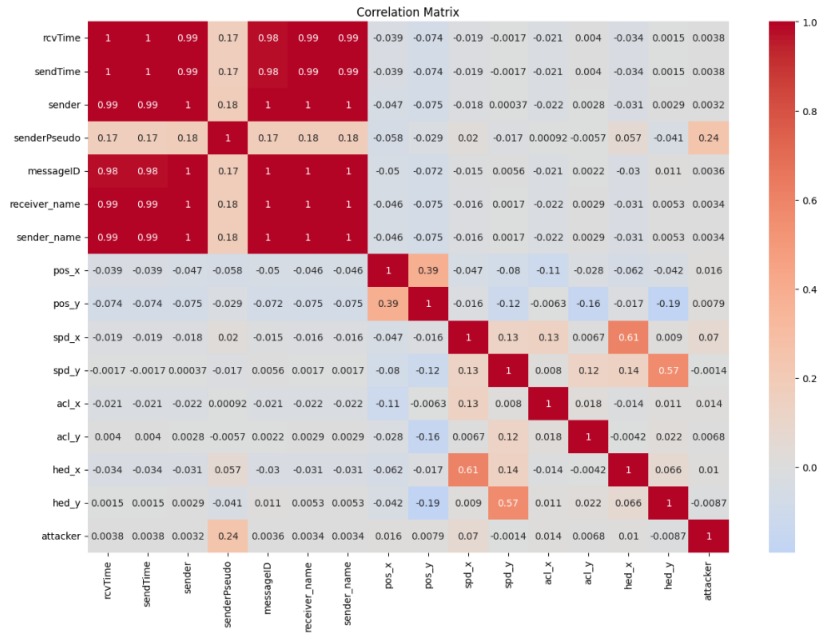

Figure 3.3: Correlation matrix

In interpreting the correlation matrix presented in Fig. 3.3, one can analyze the relationships between each feature in the dataset by examining the correlation values within each feature's row. This allows for identifying strong positive and negative correlations between different features, providing insights into the interdependencies within the data.

- There are **no null values or categorical variables** in the dataset.
- There are more unique **senderPseudo** column entries than actual senders, which indicates that some vehicles might change or have multiple pseudonyms over time. Again, this is a typical behavior of an attacker in our case.
- Sender with **pseudoID 1** is significantly more active than any other pseudonym, with a staggering **179,375** messages

3.2.3 Feature Engineering

Based on the features we examined earlier, we noticed a big problem: the model didn't do a good job even after we prepared the data. It struggled to understand the difference

between normal and attacker behavior. So, we looked back at what we learned from the Literature Review section. We realized we needed to give the model more information to work with. By adding extra details to the dataset, we hoped to make it easier for the model to tell the difference between normal driving and the tricky maneuvers of attackers. This process of tweaking the data is what we call feature engineering, and it's key to improving our results.

The following features were added to the dataset:

- **Message Frequency**

  This feature represents the number of messages sent out by each sender up to the current point in time. By tracking the message count over time, we can analyze the communication behavior of individual senders and detect any unusual spikes or drops in message transmission rates.

  This was calculated by grouping the data frame by the sender and then applying the **transform** function to count the number of messages sent by each sender. This operation ensures that the message count is calculated for each row in the data frame, considering all messages sent by the corresponding sender up to that point.

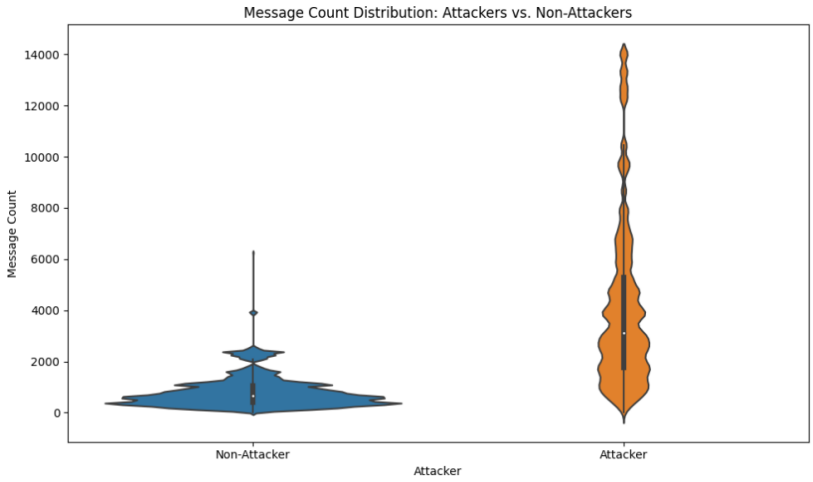

Figure 3.4: Message Count Distribution: Attackers vs. Non-Attackers

From Fig. 3.4, we can derive that the median of the message frequency (The white dot inside each violin represents the median message count for the corresponding group) for attackers and non-attackers is very different. The width of the violin at

each point represents the density of message counts at that value. This aspect of the plot provides a detailed view of the distribution, allowing us to observe peaks, valleys, and changes in density more effectively than a traditional histogram. The attackers have a low density and high variability of message count. In contrast, the message count for non-attackers is more dense within a given range and doesn't show high variability.

- **Distance**

  This feature calculates the distance traveled by each sender based on its speed and the time elapsed between consecutive messages. It provides information about the spatial movement patterns of vehicles in the VANET network.

  The distance between vehicles is calculated using the Euclidean distance formula, which measures the straight-line distance between two points in a two-dimensional space.

  This formula computes the distance between consecutive positions (divided in pos_x and pos_y spaces) of vehicles in the dataset, assuming that these positions represent coordinates in a two-dimensional space.

$$\text{Distance} = \sqrt{(x - x_{\text{shift}})^2 + (y - y_{\text{shift}})^2}$$

  Here, x and y refer to the original position of the vehicle, and x-shift and y-shift refer to the vehicle's new "shifted" position.

- **Statistical values of speed and acceleration**

  The calculated statistical columns were **mean, median, and standard deviation** of each of the x and y coordinates of **'spd' and 'acl'** features. These statistical measures were chosen because they provide valuable insights into the behavior and characteristics of the vehicles in the dataset.

  - **Mean:** The mean provides a measure of the central tendency of the data. It gives us an average value, which can be useful for understanding typical or average behavior.

- **Median:** The median represents a dataset's middle value when sorted in ascending order. Unlike the mean, extreme values or outliers do not influence the median. Therefore, it provides a robust measure of central tendency, particularly in the presence of skewed or non-normally distributed data.
- **Standard Deviation** The standard deviation quantifies the dispersion or variability of the data points around the mean. A higher standard deviation indicates greater variability, while a lower standard deviation suggests that the data points are closer to the mean.

These statistical measures were chosen based on their ability to capture different aspects of the distribution and variability of the data. By calculating these measures over time (i.e., using a running average approach), we can observe how the average, median, and variability of speed and acceleration values evolve as new data points are collected.

We used the **expanding()** function provided by the 'pandas' library [**42**] in Python to implement the running average aspect. This function allows us to calculate cumulative statistics over a rolling window of increasing size, effectively computing the running average for each data point as we iterate through the dataset.

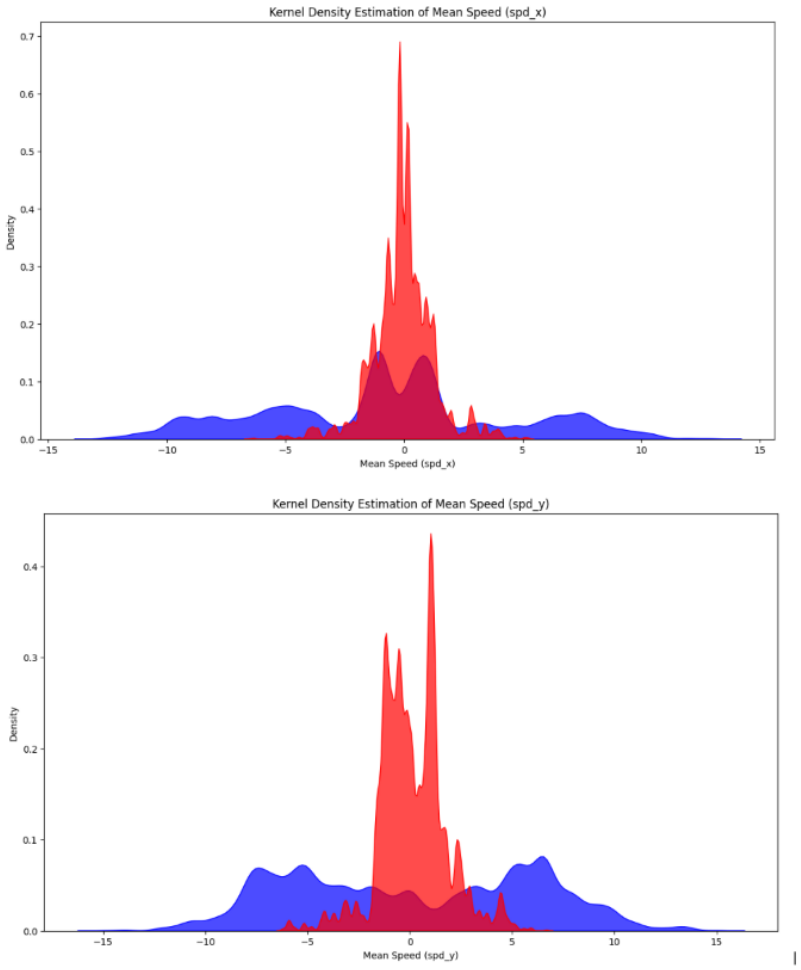

Figure 3.5: Statistical Summary of 'spd' feature

As seen in Fig. 3.5, attackers exhibit a more focused and less varied behavior in terms of mean speed. At the same time, non-attackers show a broader and more diverse range. The figure depicts the statistical summary of the 'spd' feature in the dataset for attackers and non-attackers.

3.2.4 Classification using Machine Learning models

Finally, we can use the pre-processed dataset to classify the attacker vehicles from the non-attacker vehicles. In a typical Machine Learning flow, we go through the following steps:

1. Splitting the dataset into training and testing sets

2. Scaling the features (if required)
3. Model training
4. Evaluation
5. Optimizing the model performance

We have customized the above steps to address the specific needs of VANET misbehavior detection systems, as detailed below -

1. **Splitting the dataset**

   The dataset is randomly split into training and testing sets in a supervised machine-learning pipeline. A random split refers to the process of dividing a dataset into two subsets: a training set and a testing (or validation) set. This division is performed randomly, meaning that the samples in the dataset are shuffled, and a specified percentage (e.g., 80% for training and 20% for testing) of the shuffled samples are allocated to each subset. In this split process, the dataset undergoes shuffling initially, effectively randomizing the order of samples to mitigate any inherent biases in their arrangement. Following shuffling, a predetermined percentage of samples is then allocated to the training set, with the remaining samples designated for the testing set, ensuring both subsets represent the overall dataset.

   This random split is usually recommended in the community for several reasons:

   - **Generalization Performance:**

     It ensures that the model is trained on diverse data samples, allowing it to learn generalizable patterns rather than memorizing specific instances in the dataset.

   - **Unbiased Evaluation:**

     We can use random splits to evaluate the model's performance on unseen data, which helps assess its ability to generalize to new samples. This unbiased evaluation is crucial for estimating the model's real-world performance.

- **Efficiency:**

  Random splits are simple to implement and computationally efficient, making them widely used in practice.

While random splits are effective in many cases, ***they can lead to overfitting in scenarios where the data exhibits temporal dependencies or sender-specific patterns***, ***as we experienced in our research***.

Overfitting occurs when the model learns to capture noise or idiosyncrasies in the training data, resulting in poor generalization to unseen data. We were very unsure of why this was happening, and hence, we employed the industrial best practices to address overfitting an understanding of causes:

- **Cross-Validation:**

  Implemented k-fold cross-validation to assess the model's performance on multiple train-test splits. This helps obtain more reliable estimates of model performance and identify potential sources of overfitting.

- **Feature Importance Analysis:**

  Conducted feature importance analysis to identify which features contribute the most to the model's predictions. This helps understand which aspects of the data are driving overfitting and can guide feature selection.

- **Learning Curves:**

  Plotted learning curves to visualize the model's performance on training and validation data as a function of training set size. This helps diagnose issues related to underfitting or overfitting and determine whether collecting more data would benefit.

At the end of a bunch of the above techniques ensembled together, we understood that the dataset contains some sender-specific patterns that the model cannot generalize upon. Moreover, since BSMs are time-sensitive, the dataset primarily represents a time-series dataset, which, as pointed out by the work done by Mahmoudi et al. [37], is difficult to generalize when working in a purely supervised fashion.

However, we devised and implemented a custom split strategy that can be more appropriate in scenarios where the dataset contains temporal features or sender-specific patterns.

Custom Split:

The custom split, as opposed to a random split, is tailored to preserve the temporal sequence of data in a dataset, which is particularly crucial in scenarios where the order of events matters, such as in time series data or sequential data like in vehicular networks, as in our case. Here, the dataset is partitioned based on a specific criterion, often involving time stamps or event sequences. In the context of vehicular networks, the custom split ensures that data from the same sender vehicle is grouped together and then split into training and testing sets, thereby maintaining the chronological order of events for each vehicle. This approach allows machine learning models to learn from past events in the training phase and generalize to future events during testing, mirroring the real-world scenario more accurately. By preserving the temporal aspect of the data, the custom split enhances the model's ability to capture temporal dependencies and make predictions based on the sequential nature of the data.

In our case, the custom split involves:

- **Sender-based split:**

  Grouping the dataset by the sender and splitting the senders into distinct training and testing sets ensures that data from each sender is exclusively present in either set. This preserves temporal dependencies and sender-specific patterns, leading to more accurate model evaluation.

- **Temporal Preservation:**

  Sorting the data by timestamp before splitting ensures that the temporal order of the data is maintained. This is crucial in VANET applications where the timing of events is significant, and models need to learn from sequential data.

- **Real-World Relevance:**

  The custom split reflects real-world deployment scenarios, where models encounter new senders or time-varying patterns not seen during training. This promotes better generalization and ensures the model's performance evaluation aligns with real-world usage.

This was achieved by employing the following steps:

(a) **Sender Identification:**

Identify unique sender vehicles in the dataset.

(b) **Temporal Sorting:**

Sort the dataset based on temporal attributes such as received time or event sequence.

(c) **Partitioning:**

Split the sorted dataset into training and testing sets while ensuring that data from the same sender remains intact within each set.

(d) **Preservation of Order:**

Maintain the chronological order of events for each sender vehicle in both the training and testing sets.

(e) **Temporal Dependency:**

Enable machine learning models to learn from past events during training and generalize to future events during testing, thereby capturing temporal dependencies effectively.

By employing the custom split strategy, we can address the challenges posed by our dataset's temporal dependencies and sender-specific patterns, leading to more robust and accurate machine learning models for VANET applications.

2. **Scaling features (if required)**

Often, in the case of classical machine learning problems, we must scale our features before they're given for training. This is for the models that often rely on distance-based calculations, gradient descent optimization, or regularization techniques where

scaling can help improve their performance. However, it's important to note that not all classification algorithms require feature scaling, and the necessity depends on the specific algorithm and the nature of the data.

Scaling features is a crucial preprocessing step in machine learning that offers numerous benefits. Firstly, it aids in converging optimization algorithms like gradient descent by preventing features with larger magnitudes from dominating the process, thereby facilitating faster convergence. Secondly, scaling ensures that all features contribute equally to the learning process, preventing biases towards features with larger scales. This equal treatment of features enhances the model's ability to generalize well to new data. Additionally, scaling improves the conditioning of the optimization problem, making it easier to find the optimal solution, especially when features vary widely in scale. Moreover, it enhances the model's robustness to outliers by reducing their disproportionate influence, particularly in distance-based algorithms. Lastly, scaling can aid in the interpretability of the model by making coefficients or weights associated with features more comparable, facilitating easier assessment of feature importance.

Many types of Scaling algorithms and methodologies are available, like MinMaxScaler, RobustScaler, StandardScaler, Normalization, etc.
In our case, we use K-Nearest Neighbors as our base classifier, and hence, we utilize the **StandardScaler** for our pre-processing in the case of KNNs. It scales the features to have a mean of 0 and a standard deviation of 1. This is achieved by subtracting the mean and dividing by the standard deviation of each feature.

3. **Model Training:**

In our model training phase, we adopted a pragmatic approach by leveraging supervised machine learning algorithms such as KNN, Decision Trees, and Random Forests to classify attacker vehicles from non-attacker vehicles in vehicular ad hoc networks (VANETs). We made this decision based on careful consideration of the computational overhead associated with more complex algorithms. We aimed to balance model accuracy and computational efficiency by opting for these algorithms,

ensuring that our classification process remains tractable even with large datasets. Moreover, we employed a custom split technique to preprocess the data, dividing it into training and testing sets. This partitioning allowed us to train our models on a subset of the data while reserving another portion for evaluation, enabling us to assess the model's performance on unseen data. Through this approach, we aimed to develop robust classification models capable of accurately distinguishing between attacker and non-attacker vehicles in real-world VANET scenarios.

Furthermore, we implemented several standard practices during the model training phase to optimize our classifiers' performance and generalization ability. This included feature scaling to ensure that all features contributed equally to the learning process and regularization techniques to prevent overfitting. Additionally, we conducted hyperparameter tuning to fine-tune the parameters of our algorithms and optimize their performance on the training data. We also implemented cross-validation to assess how the results of our statistical analysis will be generalized to an independent data set. This helps to prevent problems like overfitting, ensuring that the model performs well not just on the training data but also on new, unseen data. Through iterative experimentation and validation, we iteratively refined our models, adjusting various parameters and configurations to achieve the best possible classification metrics. By following these best practices in model training, we aimed to develop reliable and effective classifiers capable of accurately identifying potential attackers in VANETs while minimizing computational overhead and maintaining scalability.

4. **Evaluation:**

   In the evaluation phase, we used classification metrics like Accuracy, Precision, Recall, and F-1 score to determine the performance of the models. The definitions of these metrics and the results of our approach are outlined in Chapter 4 in detail.

5. **Hyper-parameter tuning:**

   In our pursuit of creating highly accurate and reliable machine learning models for the detection of attacker vehicles, we embarked on an extensive hyperparameter

tuning exercise, focusing on key parameters such as the number of neighbors in K-Nearest Neighbors, maximum depth, and minimum samples per leaf in Decision Trees and Random Forest models. This meticulous process, grounded in a 5-fold cross-validation framework that honors the temporal progression of our dataset, enabled us to discern the most effective configurations that balance model complexity with predictive power. By carefully adjusting these select hyperparameters, we aimed to fine-tune our models' ability to generalize well to unseen data, enhancing their precision in identifying threats without succumbing to overfitting, thus laying a solid foundation for optimizing the overall performance of our predictive modeling efforts.

Similarly, we employed cross-validation to evaluate different parameter combinations. We chose the settings **(mentioned in chapter 4)** that yielded the highest recall and F-1 scores, considering the importance of accurately identifying attacker vehicles. Additionally, we ensured the robustness of our hyperparameter tuning strategy by employing stratified k-fold cross-validation to prevent bias in the evaluation process and shuffling the training data to mitigate any sequence-related biases. We found that the value of k=5 for the cross-validation yielded the best results. This meticulous approach to hyperparameter tuning allowed us to optimize the performance of our machine-learning models and enhance their ability to classify attacker vehicles effectively.

Our aim throughout this process was to model the real-world scenario as closely as possible, which is why our Custom Split makes more sense when we look at it compared to a general random split of the data. Additionally, the additional features engineered into the dataset were deliberately extended from the existing ones since this offers a very minimal computational overhead compared to the ones that might require extra external plausibility checks. If this architecture were to be deployed in real-world vehicles, each of the OBUs would be able to compute these values for each BSM they receive, as they receive, and hence would facilitate a real-time misbehavior detection system.

3.3 Assumptions

This proposed approach has a few assumptions for the network on which this methodology will perform to its full potential. The assumptions are as follows:

- **Data Quality:**
  We assume that the data collected from VANET sensors is accurate and reliable, free from significant noise or errors that could affect the performance of the detection algorithms.

- **Data Privacy and Security:**
  Since Misbehavior Detection is all about data integrity, we assume that measures are in place to protect the privacy and security of sensitive data collected from VANETs, ensuring that only authorized entities have access to the information and that it is used solely for legitimate purposes such as attack detection and prevention.

3.4 How the Proposed methodology differs from existing approaches

1. **Use of publicly available benchmark dataset**
   VeReMi [16] was the first publicly extensible dataset we are using to allow future works to reproduce ours easily using a standard dataset. Many studies mentioned in the literature survey use customized and private datasets, making it difficult to reproduce their work.

2. **Classical Machine Learning and Training Time** Most studies [37] [38] [39] [40] involved in detecting Traffic Congestion Sybil Attacks, even Sybil Attacks for that matter, make use of Deep Learning techniques since they claim the dataset to contain complex patterns and the fact that supervised machine learning algorithms are unable to preserve the temporal features of the data. However, we think otherwise. By utilizing classical machine learning models, we can detect Traffic Congestion

Sybil Attack and significantly reduce training times and the computational resources required compared to deep learning-based techniques.

3. **Custom Split**

   Many studies claim that classical machine learning algorithms cannot preserve the temporal features of time-series data like that in VeReMi. Our work shows otherwise since we use classical machine learning, ensuring that the temporal features of the data are preserved throughout.

   Moreover, we have taken special care to ensure that even while implementing cross-validation and hyper-parameter tuning to enhance our models' performance, we ensure that each data fold is split using our custom split and not randomly.

4. **Feature Engineering**

   Studies using VeReMi have either augmented the dataset with external features (calculated through plausibility and integrity checks) or not. In our case, we have extended the features already present in the dataset by adding more context so the model can pick up on the patterns and generalize them more easily.

CHAPTER 4

Results

Due to safety concerns, high infrastructure costs, facilities, and resource requirements, conducting experiments to test the efficiency of a detection system in a real-world scenario is hazardous and difficult. Simulated data is crucial in experimental research, providing a controlled environment for testing algorithms and models. Its cost-effectiveness, scalability, and ability to manipulate parameters enable comprehensive system performance evaluations. Moreover, simulated data often comes with known ground truth labels, enhancing the reliability of results. Additionally, it helps mitigate ethical concerns associated with real-world data. However, validating findings with real-world data remains essential to ensure the applicability of research findings in practical scenarios. In this chapter, section 4.1 reviews the setup discussion regarding simulation tools and parameters used in the VeReMi dataset, experimental setup toolkits, classification parameters, and evaluation metrics for measuring the proposed classification model's performance. Section 4.2 discusses the results obtained, followed by a comparison with existing approaches in section 4.3.

4.1 Setup Discussion

4.1.1 Simulation setup of VeReMi Dataset

In this research, we use the VeReMi dataset [41], and this dataset was extracted through simulation tools. As mentioned by van der Heijden et al. [16], to generate the dataset, they utilized the Framework For Misbehavior Detection (F2MD) , an extension of VEINS

[7] designed to recreate and detect various instances of Misbehavior Detection (MBD). VEINS, an open-source simulator for Inter-Vehicular Communication, operates on top of OMNeT++ [43] and SUMO [44]. OMNeT++ is a C++ simulation library for building network simulators, while SUMO is a widely used open-source suite for simulating road traffic.

Table 4.1: Simulation parameters used in VeReMi dataset [17]

Parameters	**Value**	**Description**
Mobility	SUMO LuST	Luxembourg SUMO traffic
Simulation Area	2300, 5400–6300, 6300	Various road types
Simulation Duration	100s	
Attacker Probability	(0.1, 0.2, 0.3)	Attacker probability in the network
Simulation Start	(3, 5, 7)h	Control Density
Signal interference model	Two-ray interference	VEINS Default
Obstacle shadowing	Simple	VEINS Default
Shadowing	Log-normal	VEINS Default
MAC implementation	802.11p	VEINS Default
Thermal power	-110dBm	VEINS Default
Bit-rate	6Mbps	VEINS Default
Sensitivity	-89dBm	VEINS Default
Antenna model	Monopole on roof	VEINS Default
Beaconing rate	1Hz	VEINS Default

Moreover, the researchers employed vehicle traces from the Luxembourg SUMO Traffic (LuST) scenario [6], which has been validated with real data by the VehicularLab at the University of Luxembourg. The LuST scenario provides synthetic traffic patterns for research purposes. The researchers focused on a subsection of this network spanning 1.61

square kilometers, with a peak density of 67.4 vehicles per square kilometer. This combination of tools and datasets facilitated the creation of a realistic environment for studying vehicular communication and misbehavior detection, contributing to the advancement of research in this domain.

4.1.2 Dataset Analysis and Classification parameters

The VeReMi dataset encompasses three distinct traffic density profiles: **"0709"** denotes the period between 0700 and 0900 hours, corresponding to peak traffic hours; **"1416"** represents the interval from 1400 to 1600 hours, characterized by lower traffic volumes; and **"MixAll_24"** aggregates data from the entire day, presenting the most comprehensive dataset. This study employed the **"GridSybil_1416"** dataset, focusing on low traffic density scenarios pertinent to the Grid Sybil, or Traffic Congestion Sybil Attack. This selection was driven by the objective of enhancing model resilience with minimal data, rendering it a judicious choice among the available options.

Table 4.2: Traffic Densities in VeReMi dataset

dataset_id	**Time density**	**Attacker Vehicles**	**Attacker Messages**	**Genuine Vehicles**	**Genuine Messages**
Attack_0709	37.03 V/km^2	1,220	924,251	2,846	2,221,825
Attack_1416	16.36 V/km^2	505	249,612	1,179	569,723
MixAll_0024	23.29 V/km^2	7,399	7,505,418	17,264	11,951,210

Within a designated folder structure, individual subfolders house distributed JSON files corresponding to each vehicle in the network, documenting the Basic Safety Messages (BSMs) transmitted and received by each entity. Accompanying these files is a ground truth document. As delineated in chapter 3 of this thesis, these JSON files were consolidated and transformed into a unified CSV format to facilitate subsequent processing and model training endeavors. This data preprocessing step streamlines analysis and

lays the groundwork for applying machine learning techniques to investigate vehicular communication integrity and misbehavior detection strategies.

Model Selection

A model is selected to perform classification. There are different algorithms for classification, as discussed in chapter 2. This research uses three classifiers: K-Nearest Neighbour, Decision trees, and Random Forest.

Hyperparameter tuning

As mentioned in chapter 3, hyperparameter tuning is the process of optimizing the hyperparameters of a machine learning model to enhance its performance by systematically searching through a predefined hyperparameter space. This optimization aims to find the best set of hyperparameters that maximize the model's predictive accuracy or other performance metrics.

For K-Nearest Neighbors (KNN), our exploration of a range of values for the number of neighbors (k) was aimed at identifying the optimal 'k' value. Uniquely, we tested 'k' values from 1 to 101 to avoid ties and ensure decisive classification. Through a 5-fold cross-validation process adapted to respect the data's temporal sequence, we computed the mean accuracy for each 'k' value. Notably, our findings revealed that the recall metric began to plateau after **k=81**, signaling a diminishing return on model sensitivity with further increases in 'k'.

For Decision Trees (DT), tuning parameters such as the maximum depth of the trees, the minimum number of samples required to split an internal node, and the minimum number of samples required at a leaf node led to significant insights. The optimal parameters were identified as **'criterion': 'gini', 'max_depth': 15, 'min_samples_leaf': 8, 'min_samples_split': 10**. These settings suggest a balanced approach between model complexity and generalization ability. Specifically, the choice of Gini as the criterion indicates a preference for the Gini impurity measure in tree-splitting decisions. A max_depth

of 15 prevents the model from becoming overly complex and overfitting. At the same time, the min_samples_leaf and min_samples_split parameters set at moderately conservative values ensure that each split contributes meaningfully to model accuracy, thereby enhancing the model's ability to recall attacker vehicles effectively.

In tuning our Random Forest model, we identified optimal parameters as **'max_depth': 15, 'min_samples_leaf': 10, 'min_samples_split': 15, 'n_estimators': 50**, reflecting a strategic balance aimed at enhancing recall while preventing overfitting. The choice of a max_depth of 15 and higher thresholds for min_samples_leaf and min_samples_split suggests a model that prioritizes learning from more significant, representative data patterns, thereby improving its generalizability and accuracy in identifying attacker vehicles. Opting for 50 trees (n_estimators) strikes a careful balance between the ensemble's robustness and computational efficiency, ensuring the model captures data complexities effectively without excessive resource use.

Cross Validation

Cross-validation is a technique used to assess the performance of a machine learning model by dividing the dataset into multiple subsets or folds. The model is trained on a subset of the data and validated on the remaining folds. This process is repeated multiple times, with each fold serving as the validation set exactly once. Cross-validation helps evaluate the model's ability to generalize to unseen data and provides a more reliable estimate of its performance than a single train-test split.

We tried to split the dataset into 5 folds, keeping the randomization off since it's a time-series dataset and keeping the scoring metric as recall since that's our metric of main concern in this research. we also ensured that the data was split using our custom split in each fold, not randomly, to ensure our approach's integrity and results.

4.1.3 Evaluation Metrics

The VeReMi Dataset encompasses records of both legitimate and malicious vehicles. However, it's important to note that this dataset exhibits an inherent class imbalance, meaning

there's a significant disparity in the distribution between normal and malicious vehicle instances. Consequently, relying solely on accuracy as an evaluation metric would be inadequate for assessing model performance. Instead, precision, recall, and the F1-score are employed to gauge the effectiveness of the proposed model. These metrics offer a more nuanced understanding of how well the model distinguishes between normal and malicious vehicles by considering factors such as the proportion of correctly identified malicious instances (recall), the accuracy of identified malicious instances among all instances classified as malicious (precision), and a harmonic mean of both precision and recall (F1-score). A confusion matrix summarizes the model's performance by tabulating the correct and incorrect predictions, as shown in Table 4.3.

Table 4.3: Confusion Matrix

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

- **True Positive (TP):** As shown in 4.3, a true positive occurs when the model correctly predicts a positive instance (belonging to the positive class).
- **True Negative (TN):** As shown in 4.3, a true negative occurs when the model correctly predicts a negative instance (belonging to the negative class).
- **False Positive (FP):** As shown in 4.3, a false positive occurs when the model incorrectly predicts a positive instance (classifies a negative instance as positive).
- **False Negative (FN):** As shown in 4.3, a false negative occurs when the model incorrectly predicts a negative instance (classifies a positive instance as negative).
- **Accuracy:** Accuracy measures the proportion of correctly classified instances (both true positives and true negatives) out of the total number of instances in the dataset. Accuracy provides an overall assessment of the model's predictive performance but

may not be suitable for imbalanced datasets.

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

- **Precision:** Precision measures the proportion of true positive predictions (correctly predicted positives) out of all positive predictions made by the model. Precision indicates the model's ability to avoid false positive predictions.

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

- **Recall (Sensitivity):** Recall, also known as sensitivity or true positive rate (TPR), measures the proportion of true positive predictions out of all actual positive instances in the dataset. Recall indicates the model's ability to capture all positive instances.

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

- **F1-Score:** The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall, taking into account both false positives and false negatives. It is often used as a single metric to evaluate a model's overall performance in binary classification tasks.

$$\text{F1-Score} = \frac{2 \times (\text{Recall} \times \text{Precision})}{\text{Recall} + \text{Precision}}$$

4.1.4 Implementation Environment and Toolkit

All the experiments in this research were conducted in the following environment and configuration:

- **Operating system**: Windows 11 Home
- **Processor**: Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz
- **Memory**: 8.00 GB

Tools and libraries used for the implementation of this research are:

- **Programming language**: Python 3.7
- **Scripting language**: Shell script
- **Integrated Development Environment**: Jupyter Notebook
- **Libraries**: Scikit-learn, matplotlib, NumPy, pandas, seaborn

4.2 Classification Results

We applied three supervised classification machine learning models to our processed and feature-engineered dataset, as mentioned in chapter 3. A summarized view of our best results throughout extensive experiments can be found in the table 4.4, and a detailed discussion about the results can be found below.

We encountered a significant challenge with overfitting, particularly when employing random splits for dividing our dataset. While commonly used, this method failed to maintain the critical temporal structure inherent to our data. More importantly, we observed that the model was inadvertently learning to recognize specific sender details, treating them as predictive features. As a result, whenever the model encountered similar sender information in the test set, it could easily classify it, albeit for the wrong reasons. This over-reliance on sender details and disregard of the dataset's temporal sequence due to random splitting compounded the overfitting issue.
This is why we moved to work on a custom split. After a series of experiments, we came up with this split, which would ensure that the data's temporal features would always be preserved, and we call this dataset our "Base Custom Split", as mentioned in Table 4.4. This dataset gave us the base set of results, which we then worked towards improving upon.

After analyzing the results and the dependency factors of the model's classification report, we understood that we needed to create more features from the existing ones so that the model could pick up those patterns better and faster. A detailed discussion on

Table 4.4: Results

Experiment Stage	**Algorithm**	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
Random Split	KNN	0.97	0.97	0.97	0.97
Random Split	Decision Trees	0.99	0.99	0.99	0.99
Random Split	Random Forest	1.00	1.00	1.00	1.00
Base Custom Split	KNN	0.75	0.74	0.73	0.73
Base Custom Split	Decision Trees	0.73	0.72	0.71	0.71
Base Custom Split	Random Forest	0.76	0.75	0.75	0.75
Feature Engineered Custom Split	KNN	0.94	0.94	0.94	0.94
Feature Engineered Custom Split	Decision Trees	0.95	0.95	0.95	0.95
Feature Engineered Custom Split	Random Forest	0.98	0.98	0.98	0.98
Hyper-parameter tuned; Feature Engineered Custom Split	KNN	0.94	0.96	0.95	0.95
Hyper-parameter tuned; Feature Engineered Custom Split	Decision Trees	0.96	0.96	0.96	0.96
Hyper-parameter tuned; Feature Engineered Custom Split	Random Forest	**0.98**	**0.98**	**0.98**	**0.98**

which features were added and how they contribute to a better performance of the model is mentioned in chapter 3. This dataset has new additional features and is used with the custom split to get new results as "Feature Engineered Custom Split" in the table 4.4.

Finally, to enhance the model performance, we also performed hyper-parameter tuning. The exact hyperparameters that were tuned have been mentioned in section 4.1.2 for each model. In this scenario, the feature-engineered data is first used with the custom split; then, the models are applied with tweaked hyper-parameters to get the most optimal results. These results are mentioned within the "Hyper-parameter tuned; Feature Engineered Custom Split" row of table 4.4.

From table 4.4, we can see that Random Forests is our best-performing model, and hence, we will use this to compare previous work mentioned in chapter 2 and our proposed approach.

4.3 Comparison with Existing Approaches

Table 4.5 compares the proposed approach with existing techniques to detect Traffic Congestion Sybil Attack, mentioned in chapter 2.

As mentioned in chapter 2, below is also a summarization of each of the works and their techniques used, and how it is different from our proposed approach:

- van der Heijden [16] uses VeReMi, but they use a trust-based approach.
- Mahmoudi et al. [37] used a privately simulated dataset using an F2MD simulator with VEINS, and they employed an LSTM-based approach to get their best results.
- Zhao et al. [38] use FedMix, which means federated machine learning approach, and they are using VeReMi in conjunction with the BSMList dataset, which has been privately simulated.
- Alladi et al. [39] use deep learning and cannot work with variable-length messages.
- Liu et al. [40] use a Transformers-based approach and employ the VeReMi dataset to assess the performance.

Table 4.5: Comparison of results

Paper	**Dataset**	**Approach**	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
[16]	VeReMi	Trust-based	0.8001	0.9972	0.5842	0.7367
[37]	Privately simulated	LSTM	x	1.00	0.92	0.96
[38]	VeReMi + BSMList	Federated ML	x	0.985	0.851	0.913
[39]	VeReMi	CNN-LSTM	0.987	0.998	0.964	0.981
[40]	VeReMi	Transformer based	0.9943	0.9886	1.00	0.9943
Proposed	VeReMi	Classical ML	**0.98**	**0.98**	**0.98**	**0.98**

Looking at the difference between approaches by Alladi et al.[39] and Liu et al. [40] versus the approach proposed by this thesis, we can see that both the previous works include the use of Deep Learning. This introduces a significant computational overhead, leading to extensive computational resources and increased training time for these models. Realistically, this makes them unsuitable for real-time detection of attacks as they come. Moreover, these approaches utilize the VeReMi dataset but do not mention preserving its temporal features. We could very well argue that the results they have received might be overfitting since if they used a Random Split, the temporal features of the VeReMi dataset would not be preserved. Hence, there is a lack of clarity on their data-splitting approach.

However, our thesis adopts a method that emphasizes the importance of this temporal aspect by using the custom split while utilizing classical machine learning algorithms that are less demanding computationally and quicker to train. This approach ensures our models are efficient and capable of real-time attack detection, offering a practical solution for vehicular networks where time is of the essence.

CHAPTER 5

Conclusion and Future Work

5.1 Conclusion

In conclusion, this thesis presents a novel approach for detecting Traffic Congestion Sybil Attacks in VANETs using classical machine learning algorithms while preserving the temporal features of the dataset. Our contributions include the development of a custom split technique to ensure the integrity of the temporal data, preventing leakage between training and testing sets. By sorting the data according to the received time, we maintain the time-series nature of the dataset, which is crucial for understanding the dynamics of traffic congestion and Sybil attacks.

Additionally, we introduced three types of features to enhance the understanding of patterns within the data. Statistical distributions of vehicle speed and acceleration values, frequencies of messages, and distances between vehicles were incorporated into the dataset, providing valuable insights for classification. Leveraging the publicly available VeReMi dataset further strengthened the reproducibility and extensibility of our work, allowing for comparisons with existing research in the field.

Our use of classical machine learning algorithms addresses scalability and computational overhead challenges prevalent in previous works. While deep learning techniques may offer complex architectures, they often require substantial computational resources and time for training. In contrast, our approach demonstrates that with their simpler architectures, classical machine learning models can effectively detect Traffic Congestion Sybil Attacks without compromising performance.

By overcoming the perceived limitations of classical machine learning algorithms in

preserving temporal features, our work opens new avenues for research in VANET security. The successful application of classical machine learning models underscores the importance of methodological innovation and dataset extensibility in advancing research in intelligent transportation systems and VANET security.

5.2 Future Work

Although our work removes the computational overhead and training time issue, it would be interesting to see how reinforcement learning could make this a real-time detection technique. Our work could also be extended to see how this approach with a custom split and classical machine learning algorithms can perform a multi-class classification of Sybil Attacks in VANETs. Also, since we have applied our methods to the "GridSybil_1416" traffic density only, future work could include applying these techniques to the "GridSybil_0709" dataset.

REFERENCES

[1] Aekta Sharma and Arunita Jaekel. "Machine Learning Based Misbehaviour Detection in VANET Using Consecutive BSM Approach". In: *IEEE Open Journal of Vehicular Technology* 3 (2022), pp. 1–14. DOI: 10.1109/OJVT.2021.3138354.

[2] H. Hartenstein and L.P. Laberteaux. "A tutorial survey on vehicular ad hoc networks". In: *IEEE Communications Magazine* 46.6 (2008), pp. 164–171. DOI: 10.1109/MCOM.2008.4539481.

[3] Sheng-hai An, Byung-Hyug Lee, and Dong-Ryeol Shin. "A survey of intelligent transportation systems". In: *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks*. IEEE. 2011, pp. 332–337.

[4] Hamza Ijaz Abbasi et al. "Cooperative BSM architecture to improve transportation safety in VANETs". In: *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. 2017, pp. 1016–1022. DOI: 10.1109/IWCMC.2017.7986425.

[5] Sohan Gyawali and Yi Qian. "Misbehavior Detection using Machine Learning in Vehicular Communication Networks". In: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. 2019, pp. 1–6. DOI: 10.1109/ICC.2019.8761300.

[6] Lara Codecá et al. "Luxembourg sumo traffic (lust) scenario: Traffic demand evaluation". In: *IEEE Intelligent Transportation Systems Magazine* 9.2 (2017), pp. 52–63.

[7] Christoph Sommer et al. "Veins: The open source vehicular network simulation framework". In: *Recent Advances in Network Simulation*. Springer, 2019, pp. 215–252.

[8] Mohammed Ali Hezam et al. "Classification of security attacks in VANET: A review of requirements and perspectives". In: (2018).

[9] Aekta Sharma and Arunita Jaekel. "Machine Learning Based Misbehaviour Detection in VANET Using Consecutive BSM Approach". In: *IEEE Open Journal of Vehicular Technology* 3 (2022), pp. 1–14. DOI: 10.1109/OJVT.2021.3138354.

[10] Irshad Sumra et al. "Classes of attacks in VANETs". In: May 2011, pp. 1–5. DOI: 10.1109/SIECPC.2011.5876939.

[11] José María De Fuentes, Ana González-Tablas Ferreres, and Arturo Ribagorda. "Overview of Security Issues in Vehicular Ad-hoc Networks". In: *Handbook of Research on Mobility and Computing: Evolving Technologies and Ubiquitous Impacts* (Jan. 2011). DOI: 10.4018/978-1-60960-042-6.ch056.

[12] Irshad Sumra. "Denial of Service (DOS) Attack and Its Possible Solutions in VANET". In: Apr. 2011.

[13] Chaitanya Karn and C.P. Gupta. "A survey on VANETs security attacks and Sybil Attack detection". In: *International Journal of Sensors Wireless Communications and Control* 6 (Feb. 2016), pp. 45–62. DOI: 10.2174/2210327905999151103170103.

[14] Sofia Azam et al. "Collaborative Learning Based Sybil Attack Detection in Vehicular AD-HOC Networks (VANETS)". In: *Sensors* 22.18 (2022). ISSN: 1424-8220. DOI: 10.3390/s22186934. URL: https://www.mdpi.com/1424-8220/22/18/6934.

[15] James Newsome et al. "The sybil attack in sensor networks: analysis defenses." In: Jan. 2004, pp. 259–268. DOI: 10.1109/IPSN.2004.1307346.

[16] Joseph Kamel et al. "VeReMi Extension: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs". In: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. 2020, pp. 1–6. DOI: 10.1109/ICC40277.2020.9149132.

[17] Rens W van der Heijden, Thomas Lukaseder, and Frank Kargl. "Veremi: A dataset for comparable evaluation of misbehavior detection in vanets". In: *International*

Conference on Security and Privacy in Communication Systems. Springer. 2018, pp. 318–337.

[18] Mohssen Mohammed, Muhammad Badruddin Khan, and Eihab Bashier Mohammed Bashier. *Machine learning: algorithms and applications*. Crc Press, 2016.

[19] *AI Fundamentals*. https://djere.com/ai-fundamentals.html.

[20] *MLCheatsheet*. https://ml-cheatsheet.readthedocs.io/en/latest/glossary.html.

[21] Antonio Mucherino, Petraq J Papajorgji, and Panos M Pardalos. "K-nearest neighbor classification". In: *Data mining in agriculture*. Springer, 2009, pp. 83–106.

[22] Anuja Priyam et al. "Comparative analysis of decision tree classification algorithms". In: *International Journal of current engineering and technology* 3.2 (2013), pp. 334–337.

[23] Andy Liaw, Matthew Wiener, et al. "Classification and regression by randomForest". In: *R news* 2.3 (2002), pp. 18–22.

[24] Dristi Datta et al. "Comparative Analysis of Machine and Deep Learning Models for Soil Properties Prediction from Hyperspectral Visual Band". In: *Environments* 10.5 (2023). ISSN: 2076-3298. DOI: 10.3390/environments10050077. URL: https://www.mdpi.com/2076-3298/10/5/77.

[25] Sushmita Ruj et al. "On Data-Centric Misbehavior Detection in VANETs". In: *2011 IEEE Vehicular Technology Conference (VTC Fall)*. 2011, pp. 1–5. DOI: 10.1109/VETECF.2011.6093096.

[26] Mainak Ghosh et al. "Distributed Misbehavior Detection in VANETs". In: May 2009, pp. 1–6. DOI: 10.1109/WCNC.2009.4917675.

[27] Mohammed Alzahrani et al. "An Improved Robust Misbehavior Detection Scheme for Vehicular Ad Hoc Network". In: *IEEE Access* 10 (2022), pp. 111241–111253. DOI: 10.1109/ACCESS.2022.3214838.

[28] Steven So, Prinkle Sharma, and Jonathan Petit. "Integrating plausibility checks and machine learning for misbehavior detection in VANET". In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2018, pp. 564–571.

[29] Joseph Kamel et al. "CaTch: A Confidence Range Tolerant Misbehavior Detection Approach". In: *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. 2019, pp. 1–8. DOI: 10.1109/WCNC.2019.8885740.

[30] Jyoti Grover, Vijay Laxmi, and Manoj Singh Gaur. "Misbehavior detection based on ensemble learning in vanet". In: *International Conference on Advanced Computing, Networking and Security*. Springer. 2011, pp. 602–611.

[31] Sohan Gyawali and Yi Qian. "Misbehavior detection using machine learning in vehicular communication networks". In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE. 2019, pp. 1–6.

[32] Ankita Khot and Mayank Dave. "Position Falsification Misbehavior Detection in VANETs". In: *Mobile Radio Communications and 5G Networks*. Springer, 2020, pp. 487–499.

[33] Marwane Ayaida et al. "A Novel Sybil Attack Detection Mechanism for C-ITS". In: *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*. 2019, pp. 913–918. DOI: 10.1109/IWCMC.2019.8766572.

[34] Mohamed Baza et al. "Detecting Sybil Attacks Using Proofs of Work and Location in VANETs". In: *IEEE Transactions on Dependable and Secure Computing* 19.1 (2022), pp. 39–53. DOI: 10.1109/TDSC.2020.2993769.

[35] Badis Hammi, Mohamed Yacine Idir, and Rida Khatoun. "A machine learning based approach for the detection of sybil attacks in C-ITS". In: *2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS)*. 2022, pp. 1–4. DOI: 10.23919/APNOMS56106.2022.9919991.

[36] Dhia Eddine Laouiti et al. "Sybil Attack Detection in VANETs using an AdaBoost Classifier". In: *2022 International Wireless Communications and Mobile Computing (IWCMC)*. 2022, pp. 217–222. DOI: 10.1109/IWCMC55113.2022.9824974.

[37] Issam Mahmoudi et al. "Towards a Reliable Machine Learning-Based Global Misbehavior Detection in C–ITS: Model Evaluation Approach". In: *Vehicular Ad-hoc Networks for Smart Cities*. Ed. by Anis Laouiti, Amir Qayyum, and Mohamad Naufal Mohamad Saad. Singapore: Springer Singapore, 2020, pp. 73–86. ISBN: 978-981-15-3750-9.

[38] Jing Zhao and Ruwu Wang. "FedMix: A Sybil Attack Detection System Considering Cross-layer Information Fusion and Privacy Protection". In: *2022 19th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. 2022, pp. 199–207. DOI: 10.1109/SECON55815.2022.9918586.

[39] Tejasvi Alladi et al. "DeepADV: A Deep Neural Network Framework for Anomaly Detection in VANETs". In: *IEEE Transactions on Vehicular Technology* 70.11 (2021), pp. 12013–12023. DOI: 10.1109/TVT.2021.3113807.

[40] Zhikang Liu et al. "SVMDformer: A Semi-Supervised Vehicular Misbehavior Detection Framework Based on Transformer in IoV". In: *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. 2023, pp. 887–897. DOI: 10.1109/ICDCS57875.2023.00035.

[41] *VeReMi dataset — VeReMi-dataset.github.io*. https://veremi-dataset.github.io/. (Accessed on 02/28/2021). 2018.

[42] *Pandas Documentation*. https://pandas.pydata.org/docs/.

[43] Andras Varga. "OMNeT++". In: *Modeling and tools for network simulation*. Springer, 2010, pp. 35–59.

[44] Daniel Krajzewicz et al. "Recent Development and Applications of SUMO - Simulation of Urban MObility". In: *International Journal On Advances in Systems and Measurements* 34 (Dec. 2012).

VITA AUCTORIS

NAME:	SARTHAK KHANDUJA
PLACE OF BIRTH:	Bhilai, Chhattisgarh, India
EDUCATION:	B.Tech in Computer Science, SRM Institute of Science and Technology, Chennai, Tamil Nadu, India, (2021)
	M.Sc. Computer Science, University of Windsor, Windsor, Ontario, Canada, (2024)