Multi layered Misbehavior Detection for a connected and autonomous vehicle

Mohammed Bouchouia

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAT018

Thèse de doctorat

Multi layered Misbehavior Detection for a connected and autonomous vehicle

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°626 Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Palaiseau, le 2 Juin 2023, par

BOUCHOUIA MOHAMMED LAMINE

Composition du Jury :

Hassine Moungla Professeur, Université de Paris	Président / Rapporteur
Cristina Nita-Rotaru Professeure, Khoury College of Computer Sciences Northeastern University	Rapporteur
Jonathan Petit Principal Engineer, Qualcomm	Examinateur
Franscesca Bassi Professeure, IRT SystemX	Examinateur
Isabelle Chrisment Professeure, Télécom Nancy, Université de Lorraine	Examinateur
Rida Khatoun Professeur, Télécom Paris	Directeur de thèse
Ons Jelassi Docteure, Télécom Paris	Co-directrice de thèse

Multi layered Misbehavior Detection for a connected and autonomous vehicle

Acknowledgement

I would like to thank my supervisors Prof Rida Khatoun and Prof Ons Jelassi for their invaluable guidance and support throughout the writing of this dissertation and the defense of my thesis. I also want to extend my heartfelt thanks to my colleagues at Telecom Paris and Telecom SudParis. Your camaraderie, encouragement, and friendship have made the last couple of years easier and more enjoyable. Thank you to Sara, Abdess, Ayoub, June Ho, Fadl, Ali, Hamza, Aina, Nickson, Nathaniel, Houda, Arthur, and Adam for your unwavering support and positivity.

I am deeply grateful to my family and friends for their unwavering support throughout my work. My parents, brothers, and sisters have always been there for me, providing the love and encouragement I needed to push through even the toughest moments.

My dear friend Kamelia deserves a special mention for her constant positivity and unwavering support. She always knows just what to say to lift my spirits and put me back on track when I'm feeling discouraged and disappear. And then there's Wissam, her belief in me has been a constant source of motivation. Her unwavering support has been a blessing throughout this journey, and I know it will continue to be for the rest of my life.

I also want to express my sincere gratitude to the people I worked with, Jonathan Petit and Wafa ben Jaballah. Your expertise and collaboration have been invaluable, and I am honored to have had the opportunity to work alongside you.

Last but not least, I want to extend my heartfelt thanks to Prof Houda Labiod and Dr. Jean-Philippe Monteuuis, who have been my mentors and guides throughout my work. Your exceptional support, valuable advice, and encouragement have made me a better researcher, and I am forever grateful for your contributions to my success.

In closing, I want to reiterate my gratitude to all those who have helped me along the way. Without your support, this thesis would not have been possible. Thank you.

Résumé substantiel

De nos jours, les véhicules autonomes et les villes intelligentes visent à moderniser notre vie urbaine en créant un système interconnecté où l'information est échangée entre humains, infrastructures et véhicules. Pour cela, l'intégration de la communication V2X (Vehicle-to-Everything) dans les véhicules connectés est nécessaire pour permettre le partage d'informations, pour renforcer la perception des véhicules et améliore la sécurité routière. Cependant, une dépendance à la communication V2X expose le système à des attaques de sécurité, compromettant le bon fonctionnement du système. De plus, les véhicules autonomes doivent faire face à des défis liés aux défaillances des capteurs et à des menaces, telles que des attaques de manipulation de capteurs. La recherche dans ce domaine donc s'efforce de relever ces défis en concevant des méthodes de détection d'anomalies en temps réel des données pour garantir la sécurité des véhicules autonomes et des usagers de la route. Des études, telles que celle menée par Rens W. van der Heijden et d'autres, ont élaboré une classification des anomalies et des comportements anormaux, ainsi qu'une catégorisation des méthodes classiques de détection d'anomalies avec des contre-mesures. Cependant, ces définitions sont souvent génériques et ne prennent pas en compte les détails des composants du véhicule affectés, ce qui pourrait aider à la détection. Plusieurs recherches ont montré qu'il est possible de détecter divers comportements anormaux en utilisant des méthodes conventionnelles, telles que des contremesures internes dans les véhicules et des modèles d'apprentissage automatique appliqués aux données capturées. Toutefois, ces travaux n'intègrent pas toujours le contexte en temps réel des capteurs, des communications et des composants internes, ni n'évaluent la détection de nouveaux comportements anormaux. Par conséquent, cette recherche vise à concevoir une architecture multicouche pour la détection en temps réel des comportements anormaux dans le système C-ITS (Communications for Intelligent Transportation Systems) en utilisant l'apprentissage automatique, pour sécuriser les communications des véhicules connectés et autonomes, tout en prenant en compte l'émergence de nouveaux comportements anormaux. Cette approche comprend une définition des com-

portements anormaux, des prétraitements de données relatives au véhicule, un modèle basé sur l'apprentissage par renforcement pour la détection et la réaction en temps réel aux comportements anormaux. Une simulation est également utilisée pour tester cette architecture et ces modèles, en incluant des données V2X et des données de capteurs, et permet l'injection et la détection d'attaques en temps réel.

Les comportements anormaux

Nous proposons une nouvelle définition du comportement anormal basée sur une analyse complète des travaux existants et une classification des solutions de pointe pour la détection de ce comportement, en se concentrant sur les algorithmes d'apprentissage automatique. Dans le contexte de C-ITS, le choix entre des méthodes d'apprentissage supervisé, non supervisé ou par renforcement est essentiel pour détecter les comportements anormaux. Chacune de ces méthodes résout différents problèmes. Une classification des méthodes de détection, qu'elles soient supervisées, non supervisées ou par renforcement, est proposée pour aider le lecteur à choisir l'algorithme approprié pour la tâche en cours et donner un aperçu général des méthodes utilisées pour la détection de comportements anormaux ainsi qu'une sélection des outils disponibles pour appliquer ces méthodes.

Architecture CAV-MBDA

Notre architecture CAV-MBDA, conçue pour la détection de comportements malveillants dans les systèmes (C-ITS) a pour but de sécuriser ces systèmes du point de vue des véhicules, mettant en avant la nécessité de surveiller les données entrantes pour détecter d'éventuelles anomalies et garantir leurs bons fonctionnements. L'architecture comprend trois modules interconnectés : le prétraitement, la détection et la décision. Le prétraitement est responsable du nettoyage des données, de la création de nouvelles caractéristiques, du formatage des données d'entrée et de leur classification. Le module de détection est au cœur de l'architecture et traite tous les types de données, appliquant des méthodes de détection à différents niveaux et couches, telles que le niveau des caractéristiques, le niveau des données sources, le niveau contextuel et le niveau système. Le module de décision final prend les résultats de la détection et de la classification pour actionner le véhicule à réagir aux comportements malveillants détectés, et de générer des rapports des comportements malveillants. Cette architecture est conçue pour être générique,

capable de traiter divers types de données en entrée et de fournir des réponses flexibles en fonction des comportements malveillants détectés.

Modèle d'apprentissage par renforcement

L'environnement du véhicule connecté autonome dans le système C-ITS est un environnement dynamique et interactif qui est partiellement observable, similaire aux caractéristiques des processus décisionnels markoviens (MDP) associés à l'apprentissage par renforcement (RL). Pour résoudre ce problème de détection de comportements malveillants dans un environnement pareil, nous proposons donc un modèle RL basé sur l'architecture Q-learning. Ce modèle est conçu pour permettre à un véhicule de détecter et de réagir aux comportements malveillants d'autres véhicules dans le système C-ITS.

Caractéristiques du modèle d'apprentissage par renforcement

Le modèle est conçu comme suite : État et données : L'état est défini à l'aide d'une fonction d'approximation qui prend en compte les observations reçues par le véhicule. Ces observations sont basées sur les messages de communication V2X (Vehicle-to-Everything) et servent de données pour former le modèle d'apprentissage par renforcement (RL).

Actions et récompenses : L'agent, représenté par un véhicule dans la simulation, peut prendre des actions telles que : accepter un message ou ajouter un véhicule à une liste noire s'il est suspecté de comportement anormal. Les récompenses sont basées sur la perte de temps et les pénalités associées à ces actions.

Architecture d'apprentissage par renforcement : Le modèle RL utilise une architecture de deep q-learning associée à une fonction de mise à jour pour représenter l'état actuel du véhicule. L'agent apprend à la fois à choisir la meilleure action et à construire la meilleure représentation d'état en utilisant les données de reçues.

Limitations du monde réel : l'efficacité des données, le temps de formation, les exigences en temps réel et l'équilibre entre l'exploration et l'exploitation doivent être pris en compte pour garantir l'efficacité et la sécurité dans des environnements de conduite autonomes réels.

Simulation

Dans le but de créer un environnement réaliste intégrant différents types de données et comportements inopportuns pour la détection et le développement d'algorithmes d'apprentissage automatique. Nous avons fusionné et synchronisé les simulateurs CARLA, SUMO et ARTERY, élaboré un module de "simulation de perception coopérative" combinant ces simulateurs pour permettre une synchronisation réaliste des véhicules et l'acquisition de données sensorielles en 3D. En intégrant également Artery pour les données V2X. Nous avons mis en place des modules de détection et d'injection d'attaque pour créer un système complet. Cette simulation nous a permis d'évaluer notre système de détection en temps réel.

Résultats et conclusion

Nous avons proposé dans ce travail une architecture multicouche pour détecter les comportements anormaux et sécuriser les communications, les capteurs et les composants internes des véhicules. L'architecture est polyvalente, permettant l'utilisation de divers algorithmes, y compris des approches basées sur des règles, l'apprentissage automatique et l'apprentissage en profondeur, à différents niveaux de communication. De plus, l'architecture intègre un nouveau modèle neuronal basé sur l'apprentissage par renforcement pour détecter les comportements incorrects, surpassant les algorithmes actuels de pointe dans les simulations. Nous mettons l'accent sur l'importance de définir et de comprendre les comportements incorrects pour couvrir les menaces de sécurité et les défaillances. Nous avons développé un environnement de simulation pour générer des données V2X et de capteurs, ainsi que pour évaluer différentes méthodes de détection. Les contributions comprennent aussi le développement d'une stratégie de validation croisée spatiale et temporelle pour la détection des comportements incorrects, un résumé détaillé sur la détection des comportements incorrects dans les CAV.

Contents

List of Figures

List of Tables

Table 1: List of acronyms

Acronym	Explanation
ANN	Artificial Neural Networks
ASIL	Automotive Safety Integrity Level
AV	Autonomous Vehicle
BIST	Built-in self-test
BSM	Basic Safety Messages
CAM	Cooperative Awareness Message
CAV	Connected and Autonomous Vehicles
C-ITS	Cooperative Intelligent Transport Systems
CPM	Cooperative Perception Message
CRC	Cyclic Redundancy Check
DAE	Deep-Autoencoders
DENM	Decentralized Environmental Notification Messages
DoS	Denial of service
DSRC	Dedicated Short-Range Communications
ECU	Electronic Controller Units
EDC/ECC	Error detection and correction codes
EoP	Elevation of privilege
ETSI	European Telecommunications Standards Institute
FFNN	Feed Forward Neural Network
ID	Information Disclosure
ITSS	Intelligent Transport Systems Station
K-NN	K-Nearest Neighbors
LDA	Linear Discriminant Analysis
LOF	Local Outlier Factor
LSTM	Long short-term memory
MFM	Modified Fading Memory
ML	Machine learning
MLP	Multi-Layer Perceptron

Continued on next page

Table 1: List of acronyms (Continued)

Acronym	Explanation
NGSIM	Next Generation Simulation
OCSVM	One Class Support Vector Machines
PDR	Packet delivery rates
PKI	Public key infrastructure
POMDP	partially observable Markov decision process
POS	Proportional Overlapping Scores
QDA	Quadratic Discriminant Analysis
RF	Random Forest
RL	Reinforcement learning
RSS	Received signal strength
RSSI	Received Signal Strength Information
RSUs	Road Side Units
SIEM	Security information and event management system
SPMD	Safety Pilot Model Deployment
SUMO	Simulation of Urban Mobility
SVM	Support Vector Machines
V2X	Vehicle-to-everything
VeReMi	Vehicular Reference Misbehavior
VANET	Vehicular ad hoc network
VRS	Variation of relative speed
WCVP	Wyoming Connected Vehicle Pilot
WEKA	Waikato Environment for Knowledge Analysis

Abstract

In recent years, the vehicular field has undergone significant advancements with the development of autonomous vehicles and smart cities. These advancements have brought about a modernization of human life, where everything is interconnected - from individuals through smartphones to infrastructure, cars, and motorcycles. In such a system, information is exchanged and processed and used to ensure the proper functioning of all entities. However, the increased reliance on V2X communication also makes it a target for security attacks, which could lead to the dissemination of false or manipulated information from malicious sources. This could pose a threat to the proper functioning of the system and can potentially result in accidents. To address this problem, it is crucial to validate and verify the communication to ensure its accuracy and prevent malicious attacks. We aim to formulate misbehavior and misbehavior detection for connected and autonomous vehicles of level 4/5 automation. In our thesis, we propose a multi-layered architecture for the detection of abnormal behaviors with automatic learning to secure the connected and autonomous vehicles' communications, sensors, and internal components. The architecture allows us to propose a novel reinforcement learning based neural architecture for the detection of misbehaviors where we showed in a simulated environment, through evaluation, that the model is capable of detecting novel misbehaviors and performs better than current state-of-the-art algorithms. Furthermore, we tackle data leakage in V2X data and propose a cross-validation method to avoid said leakage in machine learning applications. We also developed a simulation for vehicular environments capable of injecting and detecting misbehaviors for the evaluation of our thesis results. The ideas developed in this research have resulted in several publications and have the potential to significantly enhance the security and reliability of vehicular systems.

Chapter 1

Introduction

In this chapter, we introduce the thesis's context and the investigated domain and give a general overview of the problem and our contributions to this domain.

1.1 Context and problematic

Nowadays, the vehicular field including autonomous vehicles and smart cities is being developed to modernize human life in a city where everything is connected; humans through a smartphone, infrastructure, cars, and motorcycles. In such a system, information is exchanged, processed, and used for the proper functioning of any entity in the system (Figure 1.1). Moreover, the integration of vehicular communication (V2X) into the perception systems of connected vehicles is a major development in the vehicular industry aimed at modernizing our lives. The goal of this communication exchange is to enhance road safety and improve the functioning of the system by providing additional information to the perception systems. For example, a vehicle can communicate information about road objects unseen by other vehicles to enhance their perception. However, the increased reliance on V2X communication also makes it a target for security attacks, which could lead to the dissemination of false or manipulated information from malicious sources. This could pose a threat to the proper functioning of the system and potentially result in accidents. To address this, it is crucial to validate and verify the communication to ensure its accuracy and prevent malicious attacks. In addition to security threats, autonomous driving also faces challenges related to perception failures. For instance, components of the vehicle may become faulty, leading to an inability to detect the road environment. The road environment itself may also pose a threat, such as solar lights blinding

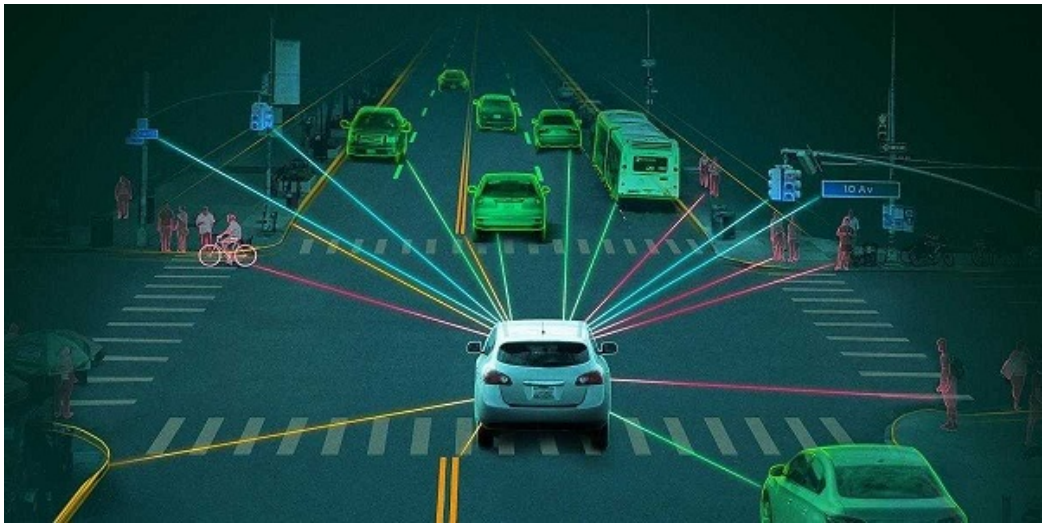


Figure 1.1: Connected and autonomous vehicle's context [1]

the vehicle's camera. For example, Duke University researchers successfully manipulated industry-standard autonomous vehicle sensors, causing them to misinterpret the proximity of nearby objects, making them appear either closer or farther than they actually are [10]. The attack method involves directing a laser beam toward a car's LIDAR sensor to introduce false information into the vehicle's perception system.

In recent years, manufacturers have been integrating an increasing number of advanced driving assistance systems into their vehicles. These systems often control critical functions of the car, thereby significantly influencing user safety which opens up novel access points [11], consequently exposing the vehicles to potential remote attacks [12, 13]. These attacks have the potential to compromise both the vehicle's structural integrity and the privacy of its occupants. Such malicious actions can be executed across three levels: exploiting physical access to the vehicle, proximity to the vehicle, or remote access via the internet. For instance, The study [14] shows that self-driving cars are vulnerable to adversarial evasion attacks at testing time, which can result in misclassification of images and significant degradation to the metrics used. The authors call for further research into the safety implications of these attacks in the self-driving car application domain. These challenges are currently being addressed to ensure the safe operation of autonomous vehicles and safety of the road users.

1.2 Motivation and Contributions

Cybersecurity risks motivate research in this field in order to secure and ensure the proper functioning of this system. In particular, Rens W. van der Heijden et al have made a study of this field and have analyzed the existing work. They developed a classification of the anomaly, the abnormal behavior, and a categorization of the classic methods of anomaly detection using countermeasures. This allowed them to formally redefine the basic elements in communications between entities in the system such as entity, attack, anomaly, and countermeasure. Moreover, It also allowed them to classify state-of-the-art works in relation to these elements. On the other hand, the definition developed to contain the attack and the anomaly is quite generic and does not include details on the vehicle components affected for example, while such information might help in detection. However, the classification and categories do not allow the identification of machine learning methods that have better detection accuracy [9]. Joseph et al [15] developed and evaluated anomaly detection models and compared the results obtained between machine learning models and rule-based methods. They worked on communication between vehicles exclusively. This allowed them to show that their solution is portable to other communications on the C-ITS system. Moreover, they showed that models based on supervised learning obtain a better detection percentage of an order of 10% for any anomaly. Jagielski et al [16] highlight the importance of addressing security concerns in connected cars and proposed the use of a combination of machine learning and anomaly detection techniques to identify abnormal behavior.

In summary, all of these studies show that it is possible to detect several types of abnormal behaviors using conventional methods such as internal countermeasures in-vehicle sensors (camera, radar, etc.) and ECUs. And also by using machine learning models on captured data (images, messages...). It is also possible to categorize abnormal behavior and learn the origin, causes, and consequences of these anomalies. However, the state-of-the-art works do not take into account, simultaneously, the information of the context, resulting from the various sensors, communications, and internal components in real time, and are not evaluated for the detection of new abnormal behaviors that can emerge in the future.

Thus, in our work, we aim to design a multi-layered architecture for the detection of abnormal behaviors of entities in the C-ITS system in real time with automatic learning to secure the connected and autonomous vehicles' communications, sensors, and internal components while taking into account the emergence of new abnormal behaviors in the evaluation of architecture. Firstly, we give a definition of misbehavior in C-ITS alongside its model and

classify misbehavior detection techniques in the C-ITS context [17]. Compared to the literature, these definitions cover attacks and failures of connected and autonomous vehicles (CAV) systems and help classify misbehaviors when detected [17]. Secondly, we define a preprocessing phase to prepare data, indeed, data in the C-ITS context use different dissemination processes such as broadcast and multi-hop and needs to be designed accordingly to ensure proper use of such knowledge especially in dealing with security issues [18]. Moreover, we model the C-ITS system as stream graphs and use it to prove that, without adequate preprocessing when using machine learning, it's easier to cause data leakage that invalidates the results of such techniques. Thirdly, we define an architecture composed of several detection layers depending on the C-ITS communication stack. Communicated messages at different layers hold different semantics, for example, about the physical radio signal, geographical positions, or network performance metrics, and thus can be secured independently or collectively. Each layer inhibits multiple levels of detection, from single features to multiple synchronized sources as opposed to securing only a single component or a message. This is motivated by the fact that single misbehavior can affect multiple levels and components and might not be easy to detect using single observations. Fourthly, we define an ML model to process data and detect security issues from one or many sources. The model used is reinforcement learning (RL) based on the deep Q-learning algorithm and allows for online detection and reaction to misbehaviors. Moreover, using our stream graphs C-ITS model, we formalize our reinforcement learning model using partially observed Markov decision processes (POMDP). The dynamicity and interactiveness of the vehicular environment give insights into choosing the detection method, and the fact that a misbehavior changes and improves over time suggests that the detection method should be capable of automatically doing the same. Finally, to build and test our architecture and models, we build a simulation environment [19] capable of feeding the architecture with V2X data and sensor data as opposed to only one of them, and also allowing for injecting and detecting attacks in a real-time manner.

1.3 Organization

The remainder of this document is organized as follows (overview in figure1.2).

In chapter 2, we present an overview of different technologies involved when tackling the problem of misbehavior detection for autonomous vehicles, survey classical and modern misbehavior detection mechanisms used

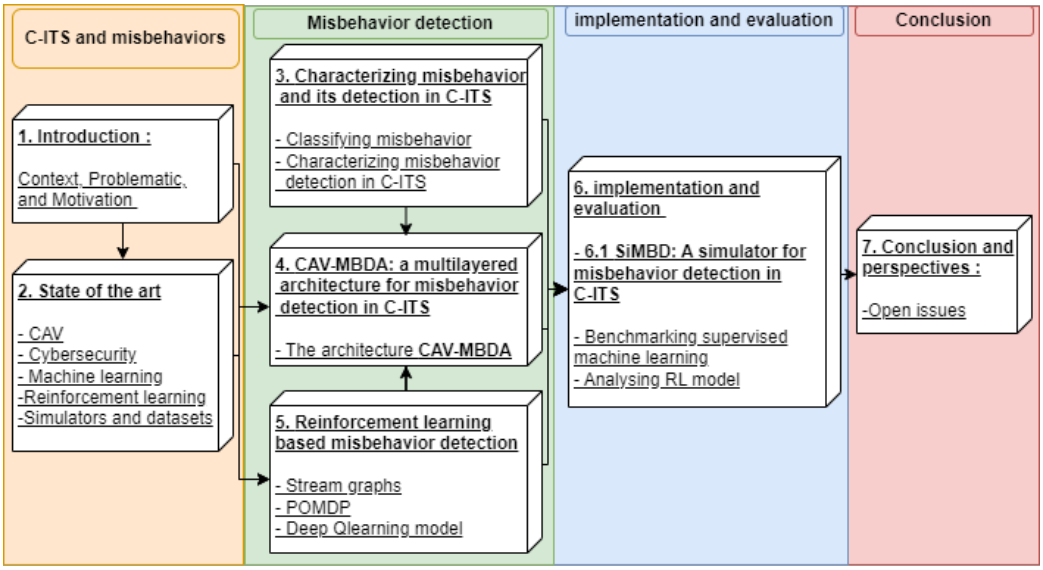


Figure 1.2: Overview of the thesis

in the literature, and present a set of tools that can be used to simulate and achieve realistic misbehavior detection for CAVs. Then, we discuss the limitations of the current literature on misbehavior classification, propose a detailed and general definition of misbehavior in CAVs covering faults and attacks and a hierarchical misbehavior classification to describe misbehaviors and constitutes a basis for the classification of research done in developing misbehavior detection mechanisms.

In chapter 4, we define our multi-layered architecture (CAV-MBDA) for the detection of such misbehaviors locally within the vehicle that takes into account all previously discussed kinds of data, and the different communication layers, and we define the whole pipeline from sensing knowledge to detecting the misbehavior.

In chapter 5, we categorize data and formalize interactions in a C-ITS system from a graph-oriented point of view, We formalize the misbehavior detection problem using partially observable markov decision processes and we propose a deep Q-learning based model architecture for the detection of misbehaviors.

In chapter 6, we develop our solution (SiMBD) to simulate V2X interactions and inject misbehaviors into them. We analyze a state-of-the-art dataset for misbehavior detection, identify data leakage in some applications of the ML to such data and propose a specific preprocessing as a solution to the problem. We benchmark supervised machine learning applications on VeReMi. And finally, we implement our MBD architecture including our RL

model in the simulation and present and analyze the results.

We conclude in chapter 7 and present a set of future works and open issues.

Chapter 2

State-of-the-art

In this chapter, we present an overview of different technologies involved when tackling the problem of misbehavior detection for connected and autonomous vehicles. Firstly, we describe connected and autonomous vehicles as independent entities and then as entities in the C-ITS system. Then we present state-of-the-art of cybersecurity issues related to CAVs. We survey classical and modern misbehavior detection mechanisms used in the literature and go in-depth with ML-based ones. Finally, we present a set of tools that can be used to simulate and achieve realistic misbehavior detection for CAVs.

2.1 Connected and autonomous vehicles

A CAV consists of several systems and components required to offer the requested functionalities defined in the $ISO26262$ standard [20]. Mainly, it is composed of Electronic Controller Units (ECU), sensors, and actuators connected through several field buses.

Figure 2.1 shows a state-of-the-art physical architecture of a CAV example, where several exteroceptive sensors (e.g., camera, LiDAR, RADAR) perceive the vehicle's environment. A set of actuators perform mechanical functionalities such as steering and accelerating. Together, these components offer functionalities to achieve full automation of the vehicle. There are six levels of automation, ranging from level 0, where the human driver has full control of the vehicle's functionalities, to level 5, where the vehicle is fully autonomous [21]. Levels 1 and 2 consider simple driver assistance and partial automation of some functionalities like steering and accelerating, but the vehicle cannot under any condition drive alone. Level 3 is conditional automation, where the vehicle can monitor the driving environment and ac-

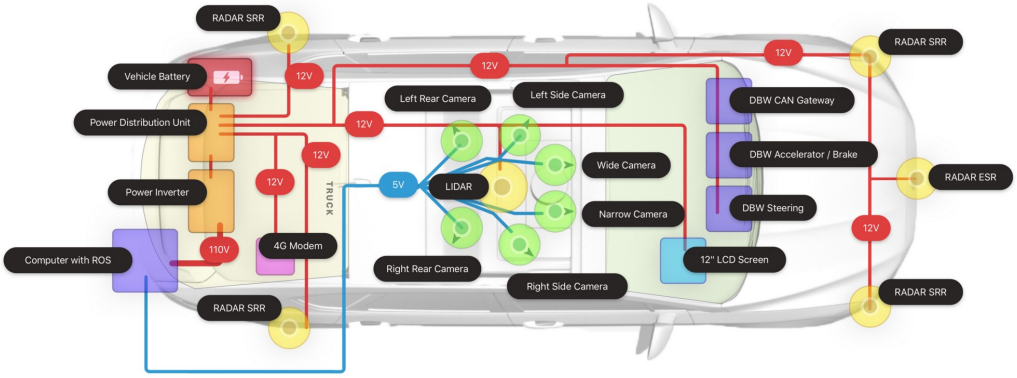


Figure 2.1: Physical architectures of a self-driving car [2]

complish simple driving tasks. The vehicle will fall back and give control to the human driver when the automation fails, which is called disengagement. Level 4 describes high automation where the vehicle can enter a safe state with no risk for the passenger or the vehicle itself without disengaging [21].

The CAV includes communication technologies such as ITS-G5/IEEE802.11p, C-V2X, and 5G. The CAV exchanges messages with other entities as part of the C-ITS. With the available information from all sensors and communications, an AI technology is anticipated to play a crucial role in controlling the dynamics and movements of vehicles, aiming to create the safest and most efficient driving experience possible.

In the next section, we describe the interactions that the CAV may have with the other entities of a global intelligent transportation system.

2.2 Cooperative Intelligent Transport Systems

Over the past decades, several technological discoveries have been introduced to intelligent transportation systems(ITS) and entities (ITS-station), and their applications are quite broad and varied. Advanced Driver Assistance Systems (ADAS), Adaptive Cruise Control (ACC), and Airbag that immediately inflate prior to the collision are some of the technologies that have been implemented in ITS [22]. While these technologies are placed within the ITS-station, Cooperative ITS (C-ITS) focuses on communications between those ITS-stations [23]. The European Telecommunications Standards Institute (ETSI) defines Cooperative ITS as *"an ITS in which the vehicles communicate with each other and/or with the infrastructure. C-ITS can greatly*

increase the quality and reliability of information available about the vehicles, their location, and the road environment. It improves existing services and will lead to new ones for the road users, which, in turn, will bring major social and economic benefits and lead to greater transport efficiency and increased safety" [24].

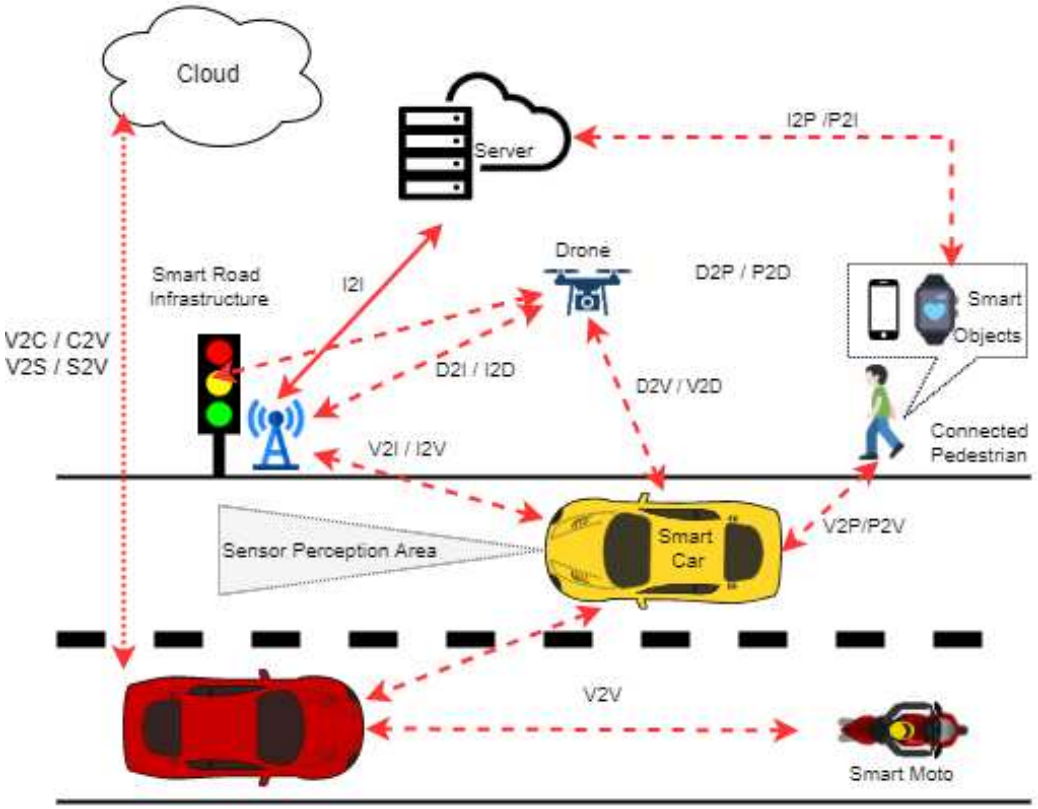


Figure 2.2: C-ITS System [3], Different communication modes.

2.2.1 Stations in C-ITS

Along with the CAVs, other stations, such as infrastructure stations and vulnerable road users (VRU), are connected through the C-ITS system. These stations are classified into four types [25], namely :

- **Central ITS-S**: An ITS-station that can provide centralized applications such as traffic and road operating. In particular, a cloud or a server (Fig 2.2) can be a central ITS-station providing services or contents.

- **Road side ITS-S**: Also called RSU, is an ITS-station that can provide TS applications from the roadside independently or cooperatively with other ITS-stations. Its functions include sending current traffic light information, sending warning messages, and evaluating traffic flow/speed/waiting times [26].

- **Vehicle ITS-S**: An ITS-station that provides ITS applications to vehicle drivers and passengers. It can access diverse in-vehicle data to provide such services and might communicate that knowledge with other ITS-stations (Fig 2.2).

- **Personal ITS-S**: An ITS-station that provides ITS applications to personal and nomadic devices (such as smart objects, and drones, Fig 2.2).

These stations exchange different standardized messages that may include kinematics information and knowledge about the station's surroundings, such as collisions, intersections, and road status, which are defined in the next section.

2.2.2 Standard communication in C-ITS

Communications in C-ITS include Vehicle to Vehicle (V2V), Infrastructure to Infrastructure (I2I), Vehicle to Infrastructure (V2I, I2V), Vehicle to Pedestrian (V2P, P2V), Vehicle to Cloud (V2C, C2V), Vehicle to Server (V2S, S2V), Vehicle to Device (V2D, D2V), Infrastructure to Pedestrian (I2P, P2I), Infrastructure to Device (I2D, D2I) (Fig 2.2). We refer to these different types of communications as Vehicles to everything (V2X).

Communications in C-ITS follow certain protocols (Fig 2.3). The communication stack defined by ETSI [27] is composed of several layers. An access layer defines the supported wireless access technology and utilizes the ITS-G5 of the IEEE 802.11 standard.

A network and transport layer supports the dissemination of messages from the source to the destination according to desired protocols (GeoNetworking and Basic Transport Protocol (BTP)).

The facility layer enables application functionality and describes the V2X messages and their structure, of which two messages stand out. The Cooperative awareness message (CAM) shares critical vehicle state information for safety and traffic efficiency applications to enhance receiving vehicles' perception of the road. The decentralized environmental notification message (DENM) disseminates event safety information in the geographical region around the vehicle or the RSU. There also exist other messages such as

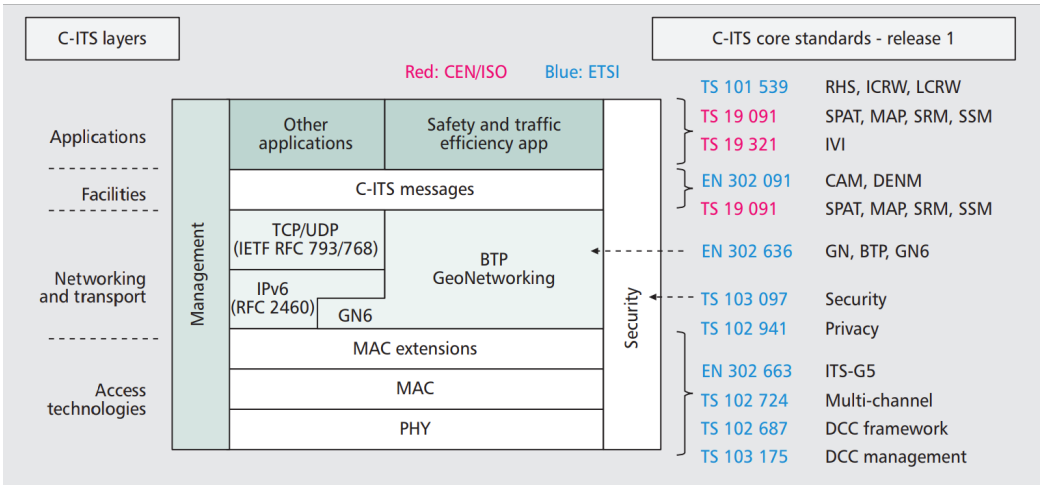


Figure 2.3: Protocols stack standards for C-ITS in Europe [4]

Intersection Collision Avoidance (ICA) messages, Emergency Vehicle Alert (EVA) messages, Position, Speed, and Time (PST) messages, Map Data messages, Signal Phase and Timing (SPaT) messages, Cooperative Perception message (CPM), Electric Vehicle Supply Equipment (EVSE) message, Weather and Road Surface Information (WRSI) message which are used for various applications such as positioning, navigation, localization, advanced driver assistance systems, autonomous driving, cooperative perception, electric vehicle charging, and weather and road surface information.

The application layer defines the desired C-ITS applications. A security layer ensures the security of all layers. For each layer, different types of information can be retrieved and used for security, and thus security mechanisms might vary depending on the layer.

To ensure privacy, each ITS-Station has several personal anonymous identifiers called pseudonyms used temporarily to identify the vehicle in its communications. Each station must acquire pseudonym certificates to sign the messages by contacting the public key infrastructure (PKI) [28]. With the apparent complexity of this whole system, we can already imagine the wide range of possible security and safety issues that can arise.

We consider in this thesis CAVs with automation levels 4 and 5, with automated driving that uses V2X as one of its data sources for building its perception. We also mainly consider V2X messages CAM and DENM. We consider that the vehicle incorporates a security gateway in its physical architecture that monitors all sources of data.

In the following, we provide an overview of the CAV security regarding

the security attack surfaces and possible system failures.

2.3 Cybersecurity and misbehavior detection

To achieve autonomous driving, a CAV is equipped with a set of sensors, actuators, and electronic control units (ECU). A local network connects these components inside the vehicle, allowing them to cooperate and complete the required functionalities to drive the vehicle. The presence of such components is mandatory i.e., the vehicle cannot move if it does not have the appropriate perception of its surroundings, such information is gathered using sensors e.g., cameras, Lidar, radars, and ultrasonic sensors, and cannot perform any action if not equipped with the required actuators and ECUs. Granting such power to these systems opens the vehicle to potentially dangerous consequences. Indeed, a faulty component, if not detected, can cause the whole system to fail and endangers the vehicle and its passengers. It is also possible for a malicious user to inflict severe damage on the vehicle and its passengers by controlling/tempering these components, to prevent this kind of event, several countermeasures must be implemented inside the vehicle (at the appropriate component) ensuring both security and safety. To create these measures, we must first identify the attack and fault vectors endangering the vehicle. A second major threat is a piece of misleading information. The CAV communicates periodically while on the road, diffusing information such as the position, speed, and state of the emergency brake system in the C-ITS network, other vehicles use this information to perceive their surroundings. A vehicle might diffuse wrong and misleading messages, for example, to occupy a parking slot by sending a fixed position or to slow other vehicles by communicating lower speed than the actual one observed on the road. This kind of event can be a consequence of wrong measurements from one of the vehicle components or an attack targeting inter-vehicle communications, in either case, different countermeasures should be made to ensure that the participants of the C-ITS network are behaving as expected. We present in the next section several efforts that have been made to define threats, faults, attacks, and their realizations.

2.3.1 Attacks, Faults, and Misbehavior

We recall in this section standard security-related definitions that apply to the context of C-ITS. What are attacks? faults and failures? threats?.

Definition 2.1 (attack)**.** *Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself. [29]*

Definition 2.2 (Fault and failure)**.** *Fault is an abnormal condition that can cause an element or an item to fail [20]. Failure refers to the consequence of a fault.*

Definition 2.3 (Threat)**.** *Threat is an event or condition that has the potential for causing asset loss and the undesirable consequences or impact from such loss [30]. Attacks and faults can be thought of as the materialization of a threat.*

An autonomous vehicle's function can be compromised due to internal and external circumstances affecting the functioning of a component or a service, whether due to malicious actions or unexpected failures. Several researchers already investigated a large set of possible attacks on connected and autonomous vehicles [31–42]. The authors show that these attacks affect the vehicle communication layers and the content of each message. To categorize these attacks, we use the extended STRIDELC [43] threat model based on the simpler STRIDE model [44]. STRIDELC defines eight different threats, namely, Spoofing, Tampering, Repudiation, Information Disclosure (ID), Denial of Service (DoS), Elevation of privilege (EoP), Linkability, and Confusion, that are appropriate for CAVs. STRIDELC is convenient for CAVs. Indeed, the additional categories defined in STRIDELC, namely, linkability and confusion, are important CAV-specific threats. Linkability refers to the ability to link pseudonyms to identify and track the owner of the CAV that breaches the privacy insurance for which pseudonyms are specifically created in this system. The other is "confusion" which refers to transmitting incorrect information in an authentic message without breaching integrity (e.g., the attacker uses his CAV to send authentic messages with false information while respecting the data structure).

Table 2.1 shows different attacks from the literature and their associated threats. This listing shows the diverse ways that can be used to endanger the CAV and motivates the urgent need for security over each layer of communication. A different attack vector is related to physical attacks on the different sensors, e.g., physically altering a component, and blinding the camera with light beams. For instance, the authors in [32] explained how to perform contact-less mechanisms to attack ultrasonic sensors, radars, and cameras using dedicated hardware. Outdoor tests on the Tesla Model S automobile caused the blindness and malfunction of the vehicle, which could eventually

Attack	**Threat**
Jamming	DoS
Flooding	DoS
Malware	DoS, Spoofing
Spamming	DoS
Denial of service	DoS
Wormhole	DoS
Masquerading	Tampering, ID, Spoofing, Repudiation, EoP
Data Replay	DoS, Spoofing, Confusion
Data alteration	DoS, Repudiation, EoP, Confusion
Data tampering	DoS, Repudiation, EoP, Confusion
Map database poisoning	Tampering Confusion
Eavesdropping	ID, Linkability
Data interception	ID, Spoofing
Man in the middle	ID, Spoofing
Hijacking	ID, Spoofing, EoP
Tracking	ID, Linkability
Gray hole	DoS
sinkhole	DoS
Black hole	DoS
Sybil	Spoofing Confusion
Impersonation	ID, Spoofing, Repudiation, EoP, Confusion
GPS spoofing	Spoofing, Confusion
Tunneling	DoS, Tampering, Spoofing
Timing	DoS, Confusion
Traffic analysis	ID, Linkability
ID disclosure	ID, Spoofing, Linkability
Key/Certificate replication	ID, Spoofing, EoP, Linkability
Illusion	Tampering, Spoofing, Confusion
Loss of traceability	Tampering, ID, Repudiation

Table 2.1: C-ITS attacks and Threats (STRIDELC)

result in a crash of the vehicle [32]. In the same way, the STRIDELC model can be used to categorize these attacks.

Table 2.2 shows possible physical attacks and their corresponding threat category.

Attack	Threat	reference
Signal relaying	DoS, Tampering, Confusion	[45], [32]
Signal spoofing	Spoofing, Confusion	[31], [45], [32]
Sensor blinding	DoS	[31], [45], [32]
MisCalibration	Tampering , DoS	[31]
Ground truth falsification	DoS, Tampering	[31], [46]
Acoustic Quieting	DoS	[32]

Table 2.2: Physical attacks and Threats (STRIDELC)

There also have been efforts to characterize attacks. The taxonomy presented in [47] gives a detailed description of the attack with the motivation of archiving it and building a database of the different events to facilitate threat and risk analysis. The taxonomy gives details about the tools used for the attack, the manufacturer of the CAV, the attack path, and an evaluation of the attack and its probability. This knowledge is acquired through post-analysis of historical attacks or simulation of predefined attacks, for which they provide the reference of the original work describing the attack.

In [42], the proposed attack taxonomy relies on the attacker, attack vector, target, motivation, and potential consequences, which are the main elements defining an attack. This taxonomy allowed the authors to discuss various methods for attacking and defending connected and autonomous vehicles, which should provide a better understanding of how targeted defenses should be developed for targeted attacks. The attack vector characterizes the attack, i.e., whether it requires physical access or not and if it is invasive. The target is the component of the vehicle undergoing the attack. Attackers, motivations, and potential consequences are self-explained. [48] propose an improvement over this taxonomy [42] by adding the communication stream as a possible target for an attack and adapting it to unmanned aerial vehicles.

The authors in [49] present a taxonomy of attacks on autonomous systems. It covers communication attacks, physical attacks, and software attacks and gives hierarchical paths to multiple known attacks on autonomous systems.

As mentioned earlier, component failure is one of the roots of CAVs failures. The authors in [8] studied disengagement reports of level 3 autonomous vehicles already deployed in the real world. Examining the data allowed them to identify the causes of disengagement. Consequently, they proposed several tags for the faults observed and built an ontology of failure categories on top of the tags.

Fault Tag	Fault Category	Description
Environment	ML/Design	Sudden change in external factors (e.g., construction zones, emergency vehicles, accidents)
Computer System	System	Computer-system-related problem (e.g., processor overload)
Recognition System	ML/Design	Failure to recognize outside environment correctly
Planner	ML/Design	Planner failed to anticipate the other driver's behavior
Sensor	System	Sensor failed to localize in time
Network	System	Data rate too high to be handled by the network
Design bug	ML/Design	av was not designed to handle an unforeseen situation
Software	System	Software-related problems such as hang or crash
AV Controller	System,	"System" when av controller does not respond to commands
AV Controller	ML/Design	"ML/Design" when av controller makes wrong decisions/predictions
Hang/Crash	System	Watchdog timer error

* ML: Machine learning

Table 2.3: Faults in an autonomous vehicle (AV) [8]

Table 2.3 highlights two main categories of failures. The first category, ML(machine learning)/Design failures, considers errors in the intelligent system ranging from perception to decision. The second category, called System failure, covers errors and faults in the vehicle's components and software/network. Among the tags, the "Environment" tag cover failures caused by adverse weather conditions or road conditions. Finally, the "Sensor" tag includes failures in different components of the system. This tag provides a high-level categorization of the vehicle's faults. To understand the causes of failures, we refer the reader to [35], where the authors describe the failure of some vehicle's components, such as over-voltage and short-circuit resulting in the Lidar failing, or foreign particles and rough vibration causing the failure of the camera, and also the probability of failure for each component.

Attempts at characterizing faults have been done in [35, 38]. Authors

in [35] studied the failure probability of CAV related to its components and the infrastructure, in the process, they build a hierarchical dependency from CAV failure to the failure of any element or functionality. For the failure related to the components, they identified five main parts: hardware, software, mechanical, communication, and interaction platform. Four main subjects are identified for the failure related to infrastructure: other road users, weather, construction zones, and road conditions. In [38], the authors simplified this model and introduced the failure of traffic signals and road signs in infrastructure-related failure.

These faults, attacks, and failures are the basis to define misbehavior for CAV. We recall state-of-the-art definitions of misbehavior.

Definition 2.4 (Misbehavior [50])**.** *Misbehaving ITS-stations are any ITS-stations that transmit erroneous data that they should not transmit when the hard and software are behaving as expected [50].*

Definition 2.5 (Misbehavior [51])**.** *Misbehavior is active and passive actions performed by communication endpoints that are not behaving according to predefined rules [51].*

A different definition that takes into account the unintentional kind of misbehavior is:

Definition 2.6 (Misbehavior [52])**.** *Misbehaving ITS-stations may send false information intentionally or due to unintentional faults in their operations. Misbehavior is a term used in ad hoc networks for any deviation from the expected behavior. In VANET, deviating from the normal operation can take many forms such as sending false information, concealing some information, tampering with messages content such as identity, alert type, event location, node position, and time, creating fake messages, or forcing another node to send a false message are considered misbehaviors in data and need to be detected each time a message received [52]*

Among other definitions, we find that these are the ones that cover the misbehavior scope for CAV. However, they show some shortcomings that will be discussed in the next section.

2.3.2 Contribution: Defining misbehavior

We recall definitions (2.4, 2.5, and 2.6) from the literature. These definitions cover most parts of the misbehaving scope separately. For instance, in definition 2.4 [50], they only consider the transmission of erroneous data when the system is working correctly, which we refer to as active intention misbehavior

(IMB). But, it does not cover other possibilities such as passive IMB that does not need to transmit any data. The second definition 2.5 covers both active and passive IMB [51], but both definitions do not consider unintentional misbehavior (UMB). The third definition 2.6 [52], covers both types of misbehavior but only considers faults in the operations of the CAV. In contrast, we can find unintentional faults caused by external factors. Aiming to cover several misbehavior types that are absent in previous definitions, we propose a new definition that groups previous generic ones covers IMB and UMB, and is adaptable to newly discovered attacks and failures.

We define a misbehavior with respect to an entity in C-ITS as follows:

Definition 2.7 (Misbehavior)**.** *A misbehaving entity manipulates and transmits data improperly or inappropriately, resulting in unexpected behavior. Such behavior could be either IMB or UMB.*

- *IMB is purposely behaving in unexpected ways, aiming to provoke harmful situations or gain unapproved benefits. It can be active (e.g., jamming/vandalism) or passive (e.g., eavesdropping).*
- *UMB is a result of faults and failures of a system or careless human-based behavior.*

The word "misbehavior" includes both attacks and faults from the previous chapter as IMB and UMB. Our definition first considers the object of the misbehavior, that is, an entity in the C-ITS system, such as CAVs, roadside units, or other VRUs that can have misbehaving conduct, for example, manipulating data flow intentionally (e.g., physically by altering speed limit signs or numerically using malware), or unintentionally acting on erroneously sensed data.

2.3.2.1 Intentional misbehavior

IMB actions can be either physical or non-physical. The physical ones require the attacker to interact with the entities and their components from the inside, such as manipulating the hardware of an ECU, a sensor, an actuator, an OBD (On-Board Diagnostics) port, or the physical link connecting them (e.g., altering the emergency brake system mechanically).

Physical actions can be performed outside by acting on surface components such as entity parts (camera, antenna) or the entity's environment such as infrastructures (roads and lights). On the other hand, non-physical actions require access to the entity or the network through an interface such as USB ports from the inside of the entity or the outside using Wireless connectivity

- **Label**: alphabetical name describing the misbehavior.
- **ID** : Numerical unique id.
- **Action**: Passive, Active.
- **Type** :
  - Physical: physically attack an ITS-S in the network.
  - Non-physical: attack on electronic objects in the ITS-S.
- **Origin** :
  - Inside: requires physical access to the ITS-S.
  - Outside: can be performed without access to the ITS-S.
- **Interface** : component of contact.
  - Hardware: a component hardware part.
  - Ports/Software: component software units.
  - Context ITS: weather, roads, RSU, infrastructure.
  - Surface component: external sensors such as camera, LiDar, and radar.
  - Wireless communication: connectivity.
- **Target**: ECU, Sensor, Actuator, Bus cables, Infrastructure, C-ITS layers.
- **Threat** : Spoofing, Tampering, Repudiation, ID, DoS, EoP, Linkability, Confusion [43].
- **Targeted property**: Authenticity, Integrity, Non-Repudiation, Confidentiality, Availability, Authorization, Unlinkability, Trustworthy [43].
- **Motivation**: Thrill, Political gain, Financial gain, Damage. [53].
- **Potential consequence** (Impact).
- Attacker model.

Table 2.4: Intentional misbehavior model

(WiFi, 4G/5G, etc.), both exposing the communication stack (C-ITS stack, CAN) and the software part of the entity's components (i.e., ECU, Sensors, Actuators). Based on our definition of IMB, we've derived an IMB model, presented in Table 2.4. This mode characterizes IMB by defining a set of features that give a snapshot of what kind of misbehavior it is and what are its causes and consequences when identified.

- **Label**: Alphabetical.
- **ID** : Numerical unique id.
- **Type** : Human based, non human based.
- **Context**:
  - Design: failure due to inaccurate design of the ITS-S.
  - Environment: failure due to environmental conditions (bad weather).
  - Element/Item: failure due to fault in an internal item.
  - Driver.
  - Other road users.
  - Safe Faults.
- **Interface**: Hardware, Software, Infrastructure, human, weather.
- **Root**: Lens, ECU, weather, cables, road signs, sensors, actuators ...
- **Fault class**: single point, multi-point, latent, residual, unclassified [20].
- **Affected property**: Authenticity, Integrity, Non-Repudiation, Confidentiality, Availability, Authorization, Unlinkability, Trustworthy. [43]
- **Impact**.

Table 2.5: Unintentional misbehavior model

2.3.2.2 Unintentional misbehavior

UMB can result from an item fault caused by a misconfiguration (i.e., wrong interconnections of components) or a component fault. The component fault can arise from faults in different parts. In the case of hardware, there are multiple fault types defined in [20], namely, residual faults, single-point faults, dual-point faults, multi-point faults, and latent faults. For software, faults are often due to bugs as well as calibration and perception errors. Another fault source is different transmission errors. Similarly, design errors can arise at the conception stages of both software and hardware.

Several scenarios of design faults are discussed in the EU Agency for Cybersecurity (ENISA) and Joint Research Centre (JRC) report [54] on the cybersecurity of artificial intelligence embedded in CAVs. Other possible types of UMB can arise from the external environment, that is, any aspect outside the ITS-S, for example, weather conditions (e.g., fog, natural disasters) or infrastructure faults such as Road Side Units (RSUs) and road infrastructures. Moreover, some are human-triggered, whether it be the driver, a passenger, or other vehicles and VRUs. Based on our definition, we've derived a UMB model, presented in Table 2.5. Similarly, this model gives a snapshot of the occurring UMB. We propose a global misbehavior model in Table 2.6 with respect to both the IMB and UMB models.

- **ID** : Numerical unique id.
- **Type** : Intentional, Unintentional.
- **Model** : Features describing the misbehavior (IMB & UMB models).
- **Attacker Model** [31]

Table 2.6: Misbehavior model

Along with its misbehavior model, the definition characterizes misbehavior with a detailed description including, for example, the root of the misbehavior, its impact, and the attacker model. We claim that it includes most of the misbehaving examples found in the literature on CAV and fairly defines misbehavior in this context.

Various countermeasures have been researched and developed to reliably detect these safety and security issues. In the next section, we provide an overview of vehicular security and detection methods incorporating such countermeasures in the literature.

2.3.3 Vehicle security

All faults and attacks can arise from within the vehicle or the wide range of connectivity implemented in the vehicle. For this reason, we separate security measures implemented in the vehicle into two categories: in-vehicle and inter-vehicle security.

2.3.3.1 In-vehicle Security

In-vehicle security refers to security measures implemented locally within the CAV components. Several safety mechanisms might be implemented locally within the CAV components to maintain the intended functionality of that component, as per requirement in the $ISO26262$ Automotive Safety Integrity Level (ASIL) [20]. These requirements depend on the functionality, the provided service, and the risk level caused by the malfunctioning component. The countermeasures are implemented at the component level, specifically tailored for some known failures and malicious activities, and effectively secure such threats and report malfunctions. For example, we can cite error detection and correction codes (EDC/ECC), built-in self-test (BIST), and end-to-end safety protocols (i.e., a combination of CRC, timestamp, and frame counter) [55]. The International Standard Organization (ISO) develops several vehicle-related standards. The beforementioned ISO 26262 [20] defines the functional safety of electrical and electronic devices for the automotive industry. The ISO/SAE 21434 [56] specifies cybersecurity standards for road vehicles. It provides vocabulary, objectives, requirements, and guidelines related to cybersecurity engineering and establishes minimum criteria to prevent cyberattacks on vehicles. Similarly, the US National Institute of Standards and Technology (NIST) Cybersecurity Framework (CSF) [57] helps to describe a system to manage cyber security risk and could be applied to design a system for connected and autonomous vehicle cybersecurity. For example, the US Department of Transportation (USDOT) sponsored the creation of a CSF Profile for Connected Vehicle Environments(CVE) [58]. It is an application of the NIST CSF based on existing standards, guidelines, and practices for organizations to better manage and reduce cybersecurity risk. It was designed to help foster risk and cybersecurity management communications. The profile is only a starting point and is extendable/adjustable to cover different scopes or objectives.

Nevertheless, the probability that the CAV fails due to one or multiple components without notice of safety mechanisms (that itself might be compromised due to tampering with the component) is still present and usually requires network and data level countermeasures to detect it, such as plausi-

bility checks, trust evaluation, and anomaly detection mechanisms.

2.3.3.2 Inter-vehicle Security

Inter-vehicle communications are also an essential part of the functioning of the CAV. Maintaining fluid communication, and ensuring the privacy and correctness of the messages are some of the required goals. State-of-the-art mechanisms such as time stamping, variable MAC, and IP address [36, 41] are examples of active countermeasures for multiple threats on the vehicular ad hoc network (VANET). However, they are not developed to discover new engineered attacks that consider the network dynamics and the CAV dependence on perceiving the surrounding environment. They also do not consider attacks from non-vehicle-connected objects such as pedestrians and infrastructures. Forging C-ITS messages can affect the performance of the network and all its entities. Indeed, transmitting the wrong speed and position may result in slow traffic or causes emergency brakes, which may result in collisions. Researchers are investigating different kinds of solutions to verify the validity of the received information by checking its consistency over time and its agreement with the behavior of the transmitting ITS-station (ITSS) [9, 59, 60]. Globally, standards developing organizations (SDO) are defining the functional requirements for misbehavior detection. Currently, the main standardization effort is driven by the ETSI which published a report TR 103 460 [61] and is working on a technical standard TS 103 759 [62]. This report provides a list of misbehavior attacks and detectors for several V2X messages, such as safety and awareness messages. Based on this report, this technical standard specifies the design and technical requirements of a misbehavior reporting service on field ITS-stations. **In this thesis, we focus mainly on inter-vehicular security, particularly, securing V2X communication using misbehavior detection techniques.**

In the next couple of sections, we provide an overview of different detection mechanisms.

2.3.4 Classical misbehavior detection

As endorsed in the previous section, misbehavior in the CAV context presents a high risk for the vehicle and its passengers. Thus, it needs special handling, first by detecting it and then by applying the favorable reaction to ensure the efficiency and security of the CAV.

This section surveys the detection part by first listing the different machine learning and rule-based methods classifications from the literature.

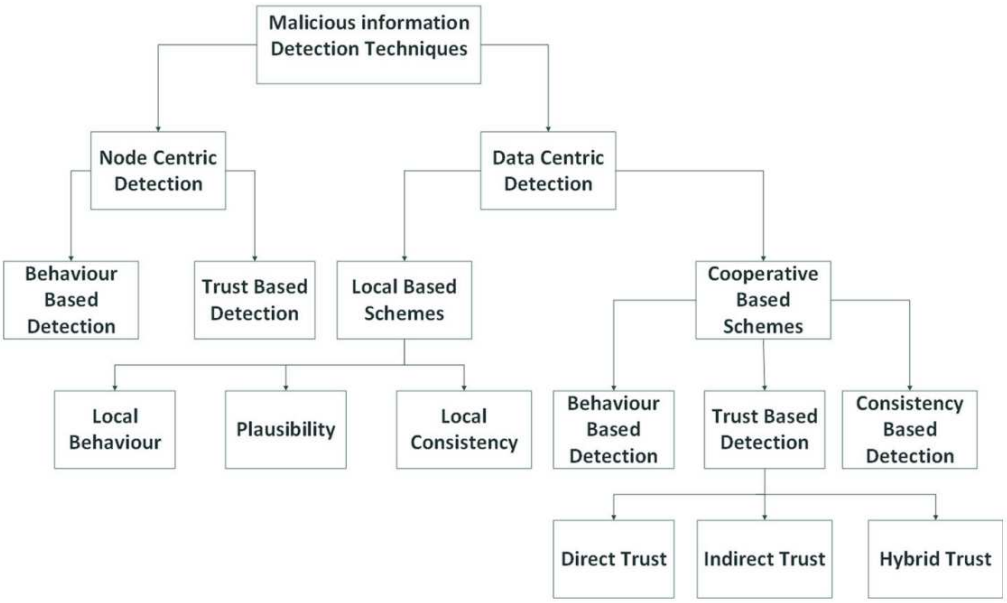


Figure 2.4: Literature misbehavior detection schemes [5]

Figure 2.4 shows the classical categorization of misbehavior detection schemes that are heavily used and improved in the literature [5, 50, 63–66]. In [50], the authors divide misbehavior detection into two main categories: node-centric and data-centric.

Node-centric *'mechanisms that are primarily concerned with the participants of the network. For example, they can verify the forwarding behavior of an ITS-station by analyzing packet frequencies, correctly formatted messages, and so on to decide on its trustworthiness' [50].* Two subcategories emerge from node-centric, namely, behavioral mechanisms exploiting patterns in the behavior of specific ITS-stations at the protocol level in an autonomous manner. Moreover, trust-based mechanisms exploit the fact that most ITS-stations in the network are honest and that infrastructures are available to remove malicious ITS-stations.

Data-centric mechanisms (introduced in [63]) are focused on the interaction between participants, verifying their trustworthiness based on the correctness of previous messages using the message content to determine its validity independently of who transmitted the message (i.e., no personal data about the sender vehicle is required). In addition, two subcategories emerged from this, namely, consistency-based detection and plausibility-based detection. Consistency-based detection uses the relations between packets, typically from multiple participants, to determine the trustworthiness of newly received data. Plausibility-based detection uses a specific underlying data

model to verify if the transmitted information is consistent with this model. For example, the plausibility of movement can be verified from two subsequent beacon messages by examining the distance traveled between them and comparing it to the speed in these messages. This definition is also found in other works such as [5, 64–66].

In [5], the authors extend the previous definition to consider local versus cooperative schemes of misbehavior detection. For cooperative detection, the ITS-stations also exchange ratings and other knowledge with nearby ITS-stations. This additional information is then used to establish a level of trust for each received message and/or vehicle.

Such misbehavior detection classification is mainly defined for communication-based misbehavior and does not cover physical attacks and faults on elements or broadcast data. It also does not cover misbehavior detection based on machine learning. Other works consider machine learning algorithms and provide a classification, such as [67–71]. In the next section, we survey machine learning based misbehavior detection methods used in the literature.

2.4 Machine learning based misbehavior detection

Classification of machine learning based methods for misbehavior detection in the context of C-ITS and IDS into several subcategories has been researched in the vehicular community to better understand and apply such methods for vehicular safety and security.

In contrast with the survey [50], where classical methods such as node-centric and data-centric methods are classified, the following focus on ML methods. In [67], the proposed IDS taxonomy considers audits based on knowledge, behavior, or a hybrid type of data using statistical, machine learning, or rule-based techniques. This taxonomy offers a high-level abstraction of the required components for making an IDS.

In [68], the goal is to provide a high-level classification of machine learning algorithms into supervised, unsupervised, semi-supervised, and reinforcement learning, and their subcategories without referring to actual methods for IDS.

In [69], the authors provide a similar framework to the one proposed in [67] and give additional details on machine learning algorithms used in IDS. Under anomaly-based IDS, the authors categorize the methods as statistical, knowledge, and machine learning based and give a listing of methods in each of the categories. Additionally, [70, 71] propose taxonomies of unsupervised

anomaly detection.

In [71], the authors categorize the techniques furthermore into distance-based, ensemble-based, statistical-based, domain-based, and reconstruction-based, and refer to famous methods in each category.

Table 2.7: Machine learning based Attacks Detection for a CAV

Year	Ref	Train.		ML algorithm	Threat	
		Simul	DataSet		DoS	Tamper
2007	[72]	✓	✗	K-means	✗	✓
2010	[73]	✓	✗	ANN	✓	✗
2011	[74]	✓	✗	RF, NB, IBK, J-48 A-Boost	✓	✓
2013	[75]	✓	✗	Fuzzy STS	✓	✓
2014	[76]	✓	✗	SVM	✓	✗
2015	[77]	✓	✗	OCSVM, K-means	✓	✗
	[78]	✓	✗	SVM	✓	✓
	[79]	✓	✗	ANN ,FFNN,\SVM	✓	✗
2016	[80]	✓	✗	K-NN	✓	✗
	[81]	✓	✗	SVM	✓	✗
2017	[82]	✗	6	SVM	✓	✗
	[83]	✗	3	ANN	✗	✓
	[84]	✓	✗	SVM	✗	✓
	[85]	✓	✗	LDA,\QDA	✓	✗
	[86]	✗	✗	SVM-MFM	✗	✓
2018	[59]	✗	5	MLP,\RF,\A-Boost	✗	✓
	[60]	✗	2	K-NN,\SVM	✗	✓
	[87]	✓	✗	Fuzzy C-means	✓	✗
	[88]	✓	✗	ANN	✓	✗
	[89]	✗	1	Pearson correlation analysis	✗	✓

Continued on next page

Table 2.7: Machine learning based Attacks Detection for a CAV (Continued)

Year	Ref	Train.		ML algorithm	Threat	
	[90]	✓	✗	MLP,\LSTM	✓	✗
	[91]	✗	✗	BNN	✗	✓
	[92]	✓	✗	SVM,\ANN	✓	✗
2019	[15]	✓	✗	MLP, XGB, SVM\C, LSTM	✗	✓
2019	[93]	✗	4	K-NN, SVM	✗	✓
	[94]	✗	4	LR, K-NN, DT, Bag, RF	✗	✓
	[95]	✗	4	LR,SVM	✗	✓
	[96]	✓	✗	DAE, SVM	✗	✓
	[97]	✗	1	CC, DBKM, FC, KM, FF	✗	✓
	[98]	✓	✗	2*CNN+LSTM	✓	✗
	[99]	✓	✗	K-NN,\RF	✓	✗
2020	[100]	✓	✗	RF, XGB, LGBM, NN, LSTM	✗	✓
2021	[101]	✗	4	K-NN, RF	✗	✓
	[102]	✗	4	CNN, LSTM, SVM	✗	✓
	[103]	✗	✗	GCN	✗	✓
	[104]	✗	✗	Graph Attention Network	✗	✓
2022	[105]	✗	4	boosting, DT	✗	✓
	[106]	✓	✗	fuzzy inference system	✗	✓

Some of these methods have been efficiently used for misbehavior detection in C-ITS. we carry out a survey on machine learning techniques used for misbehavior detection in the CAVs context, divided into two parts: intentional and unintentional MBD. For each category, we provide the major ML works for MBD.

2.4.1 Intentional Misbehavior detection

In Table 2.7, we review the main ML methods used for MBD. We grouped each work per year of publication and authors. Then, we highlight the ML model, the methodology to train and test their ML model, and the type of threat detected by the classifier.

In [72], the authors used entropy to represent the "abnormal" and "normal" behaviors of ITS-Ss, and k-means clustering to identify outliers which are the assumed attackers. This work supposes a majority of honest ITS-Ss. Thus, the suspected ITS-S eviction relies on the deviation between the attacking ITS-S and the majority of honest ITS-Ss. The scheme uses the position of each observed station to compute the entropy.

In the same context, in [73], the authors proposed a centralized IDS based on RSU for VANET. The network of connected buses, named BUSNet, individually eavesdrops and collects the data packets and routing control messages exchanged in VANET. BUSNet forwards the information to RSU to process and detect anomalies using a neural network.

To differentiate between legitimate and malicious ITS-Ss in VANET, the authors in [107] design a framework based on ML approaches. In particular, the study classifies multiple misbehaviors based on features. These features are speed deviation, distance, received signal strength (RSS), and the number of packets generated, delivered, dropped, and colluded. The authors measured the accuracy of two classifier types. The first one is a binary classifier whereas the second one is a multi-class classifier. Also, this study extracted the features of packets by performing experiments in the NCTUns-5.0 simulator [108] with various simulation scenarios and calculated by nearby ITS-Ss. Also, the authors used Waikato Environment for Knowledge Analysis (WEKA) [109] to classify misbehavior with several classifiers: Random Forest (RF), J-48, NB, Ada Boost1, and IBK. Experimental results show that RF and J-48 classifiers perform better compared to other classifiers. The RF and J-48 classifier gives better classification due to the boosting and bagging properties. Then, their extension used a majority voting scheme to improve the accuracy of their classifier [74]. The voting scheme is a plurality vote and decides based on the label which received the most votes among all voting classifiers. The proposed system shows a better result than any model used by itself.

In [75], the authors used a fuzzy time-series clustering for Sybil attack detection. Their scheme leverages the dispersion of vehicles by clustering their locations. Sybil ITS-Ss are those closely located and move together for a long period.

In [76,78], the authors proposed an intrusion detection framework, named AECFV, which monitors ITS-S mobility and frequent changes in a network topology. At its core, there is a clustering algorithm, where cluster-heads are selected based on the trust level of each vehicle and a boundary distance. Trust levels are evaluated based on majority voting and a reputation protocol and are broadcast periodically. The proposed framework uses two detection systems and a single decision system. The first system runs locally at each

cluster member and monitors the neighboring vehicles and the cluster-head. The second system runs globally at the cluster-head level and evaluates the trustworthiness of its members. The global decision system runs at the RSU level, computes, and classifies each vehicle based on the level of trust. Together, these systems constitute a network IDS as they make a decision based on the behaviors of ITS-Ss within their radio range. The two IDSs use rules and SVM to classify vehicle behavior.

In the context of a cloud-based mobile distributed IDS for VANET, the authors in [77] propose a framework that detects DoS attack leveraging OC-SVM with RBF kernel and score the anomalies using recursive K-means clustering.

Another study in [110] proposes a context-aware security framework for VANETs based on the SVM algorithm. The objective is to automatically differentiate malicious ITS-Ss from abnormal ITS-Ss due to contextual reasons such as movement speed, temperature, and transmission range. The proposed framework has three functional modules, starting with behavior data collection, then context sensing and processing, and finally MBD. The results demonstrate that the proposed framework achieves excellent accuracy, recall values, and an acceptable value of communication overhead.

In [79], the authors provide an IDS for VANETs based on Artificial Neural Networks (ANNs) to detect black hole attacks. The classifier uses spatial, temporal, and networking features as inputs for the training and testing phase. The proposed IDS exhibit a high error rate despite having a high accuracy score and a low false-positive alarm rate. Their first extension includes a method, named Proportional Overlapping Scores (POS), which reduces the number of extracted features. In black hole detection, the POS method ranked networking features above state features. Also, the extension uses the fuzzy set method, which improves the separation between label types. Moreover, the classier is a feed-forward neural network (FFNN). Overall, the classifier performances improved. However, the IDS requires more memory and computation resources than the previous work. In a second extension [111, 112], the IDS focuses on detecting the new type of DoS attacks known as the grey hole and rushing attacks. This extension used two classifiers (FFNN and SVM). Overall, FFNN has the best detection rate for grey holes and SVM has the best detection rate for rushing attacks. In a third extension [80], the authors include a new feature based on a hashed ICMetric number. Mathematical functions generate the "ICMetric" number based on sensors readings (magnetometer [80], gyroscope [113], and infrared sensors [114]). The hash function outputs a hash from the ICMetric number. The classifier is K-NN. Overall, the scheme shows a higher accuracy rate of detection with a low false alarm rate than their previous proposal [79]. In

their last extension, the authors evaluated and compared the performance of the Linear and Quadratic Discriminant Analysis (LDA and QDA) to detect DoS attacks [85]. The framework uses only V2X data without feature selection. Overall, the LDA classifier outperforms the QDA in both detection accuracy and computation time. In comparison with the previous work, their scheme outperforms previous proposals in terms of error and false alarm rates [115].

In [81], the authors propose a mechanism based on SVM to detect misbehaving ITS-S. Moreover, in [82], they propose a collaborative security attack detection mechanism in a software-defined vehicular cloud architecture. Each vehicle analyzes the received information and periodically transmits the results to the controller to train the SVM. After this, each vehicle classifies ITS-Ss.

An interesting study in [83] leverages the use of a model based on ANN using feedforward and backpropagation. The mechanism uses historical data to classify normal or malicious data, and the classifier uses the NGSIM dataset.

To detect Sybil attacks, the authors in [84, 116] use driving patterns and an ML model (Knn and SVM). In the work of [86], the authors propose the use of SVM with Modified Fading Memory (MFM) to classify legitimate and malicious ITS-Ss to reduce the computational overhead for the ML algorithm by only considering eligible ITS-Ss in the range of the VANET communication.

Another method was proposed in [60]. The authors use an ML-based mechanism to detect malicious V2X messages. The scheme uses the VeRemi dataset, and the classifiers are K-NN and SVM. In another work, they used an additional feature which is RSSI [93]. The authors in [94] used a similar approach by using the same dataset; however, they tested their classifier through the simulator VEINS.

In another study, [88], the authors combined several ideas for VANET IDS into a single multi-layered framework, which they show to be effective against a variety of different attacks. In all cases, detection compares audit features against thresholds. These include packet delivery rates (PDR) and Received Signal Strength Information (RSSI) for selective forwarding (gray hole) and blackhole attacks; duplicate packet rate and packet forwarding rate for DoS; RSSI and PDR for wormhole attacks; and the z-score of RSSI for Sybil attacks. Their evaluation, which is based on NS-3 simulation, has shown that this framework can achieve greater accuracy and lower overhead in terms of IDS-specific network traffic. The reduction of IDS traffic overhead is the result of adopting a game-theoretic approach in modeling the interaction between the IDS and the malicious vehicle as a two-player non-cooperative game and using the Nash Equilibrium to inform the choice of the monitoring

strategy.

In [89], the authors use Pearson Correlation to detect location forging attacks. The proposed solution works in real-time and requires at least four to seven seconds of history to be fully efficient. Experiments used the real datasets from Wyoming Connected Vehicle Pilot Deployment. [117].

The work [91] provides a Bayesian deep learning approach to detect VANET anomalies. However, the authors did not evaluate their contribution. In [92], the authors propose an ML-based IDS to detect intruders in VANET automatically. They implemented their approach using ANNs and SVMs.

In [15], the authors evaluate a local MBD [15]. This evaluation accounts for new features (e.g., Kalman Filter), different attack behaviors, and new attacks (e.g., DoS). In addition, the authors compare the performances between rule and ML-based methods. Overall, the evaluation shows ML algorithms detect better misbehaviors than rule-based methods. However, ML algorithms perform slower than threshold-based ones. This evaluation has been done with servers and a local machine instead of an ECU and hardware optimized for ML. Thus, the computation speed may not be relevant.

The authors in [118] studied how deep learning approaches can solve the safety problems regarding pedestrians. A Vehicle-Pedestrian Detection algorithm based on CNN is proposed. The results show that the method can recognize and identify safe interactions between connected autonomous vehicles and pedestrians with an accuracy of 81.98%.

In [100], the authors evaluate a global misbehavior detector based on ML models. Compared to a similar work [15], the evaluation includes additional models such as RF, Neural Networks, and Light GBM. The evaluation outcome implies that LSTM is the best classifier tested. In [96], the authors use unsupervised ML to detect VANET anomalies, where the classifier is a DAE. The goal is to identify the abnormal position in the V2X message based on the vehicle location and the RSSI. In [97], the authors use the WCVP dataset, and their classifier analyzes only Basic Safety Message (BSM) fields. Moreover, the authors created their attacks by modifying the dataset. The work in [99] leverages the use of supervised learning models (K-NN and RF). Features include the variation of relative speed (VRS); a radar measures the relative speed between the jammer and the receiver.

In [101], the authors propose machine learning mechanisms to detect position falsification attacks. They compared K-NN and Random forests using F1-score and accuracy. They finally combined both methods to achieve better performance. The resulting IDS achieved an accuracy of around 85% on average on different traffic densities.

In [119], the authors propose a method to detect misbehavior based on

changes in the positions' time series using anomaly scores. The method achieved an average of 0.5 f-measure on different attacks. The authors in [102] presented a deep learning-based method for misbehavior detection. They used both CNN and LSTM to reconstruct location information and an SVM to classify compromised vehicles. The results show that the proposed method detects 95.37% of compromised vehicles in the VeReMi dataset.

The authors in [105] propose a novel interpretable framework for detecting falsified data (BoostGuard). The framework uses a boosting decision tree ensemble to detect and classify attack types. The demonstrated results showed that BoostGuard outperformed all other methods and achieved an f-measure of 0.994 on the VeReMi dataset.

In [103], the authors propose novel credibility-enhanced Temporal GCN-based Sybil attack detection. GCNs are analogous to CNNs and perform similar operations where the model learns from neighboring nodes instead of pixels. The authors' TGCN-based classifier considers CAVs as the graph nodes and uses CAVs information to make real-time classifications of Sybil attacks. Simulation results show that their proposal effectively defends against different Sybil vehicles with low error rates.

In [104], a Graph Attention Network-based framework for malware detection in C-ITS is proposed. Graph Attention Network is similar to GCNs with the difference that connections to neighboring nodes are also weighted to give priority to some nodes over others depending on the quality of their information for the given task. Results show that the proposed attention network achieves an f-measure of 0.95 and over-performs GCN.

In [106], the authors propose a fuzzy-based context-aware approach for the detection of misbehaving vehicles. The approach consists of 4 phases. Acquisition of context through sensors by the vehicles followed by sharing the context with other vehicles in the communication range. Then the evaluation of the context in terms of the uncertainty of the observations and finally classification of misbehaving vehicles based on how they deviate from the context reference. The proposed model achieved 89.84% f-measure in optimal communication and then down to 79.63 % in worst communication scenarios (message loss).

2.4.2 Unintentional Misbehavior detection

In [120], the authors proposed an automatic engine for fault diagnosis using wavelet analysis. The fault features extracted by a discrete wavelet transform are used as the input for ANNs.

In [121], the authors propose a sensor fault detection system for the automotive pedestrian. This fault detection system relies on SVM models trained

and tested on the KITTI dataset. In the same context, the authors in [122] propose an automotive fault detection and diagnosis system. The fault detection system is an OC-SVM. The diagnosis system is a Kalman filter (KF) that computes the residuals between the predicted value and measured value, checks the normality of the residual's distribution, and validates whether the trajectory deviates in a checking period.

In [123], the authors use a combination of CNN, KF, and κ^2 to detect and identify automotive sensor anomalies. Similarly, in [124], the authors combine an adaptive extended KF, enhanced using a car-following motion model, and an OC-SVM to detect automotive sensor anomalies.

In [125], a DBN is used to predict a vehicle's malicious behavior. The result is then used to classify each vehicle as malicious, normal, or abnormal. Results show that their model overperforms all state-of-the-art methods and achieves a false alarm detection rate of 0.98.

The detection of UMB using ML has been poorly explored. Indeed, we find significantly less research done for this purpose. Acquiring data that includes UMB is quite cumbersome. There are very limited data sets that offer such an opportunity, which leads to constraints in the research on this subject.

2.4.3 Discussion

Throughout our study, we find that an ever-increasing number of research papers focus on ML for MBD and yield better performance than previously used rule-based methods such as plausibility checks and trust evaluation. The interest in using ML-based algorithms for intentional MBD keeps growing as the development of CAVs is ongoing. These algorithms understand and model IMB directly from the data. Using both the data and expertise as opposed to only using human expertise has greatly improved the performance in detecting misbehavior. Still, using ML algorithms has not yet been thoroughly covered, as the scope of developed algorithms is still quite wide, and further exploration is still possible. These methods are adaptable to the various data flows generated in the context of CAVs and can model high-dimensional data. However, there is a noticeable lack in the use of RL algorithms. Another interesting point is that most of the surveyed works do not conduct a temporal evaluation in their experiments, which is critical in this context.

In our work, we focus on using reinforcement learning for the detection of misbehavior in a way that exploits its learning from interactions, dynamicity, and multi-agent easiness as it strongly correlates with the CAV environment.

In the next section, we briefly introduce reinforcement learning and present its application in C-ITS.

2.5 Reinforcement learning

Reinforcement learning as defined in [6] is learning what to do (how to map situations to actions) to maximize a numerical reward signal. The learner is not told which actions to take, but instead, must discover which actions yield the most reward by trying them. The problem of RL is usually formalized as a Markov decision process (MDP). Research on the RL topic is conducted in several domains. For instance, its applications to connected autonomous vehicles (CAVs) are investigated in [126] and extend to subjects such as autonomous driving, resources optimization, and security [127–133].

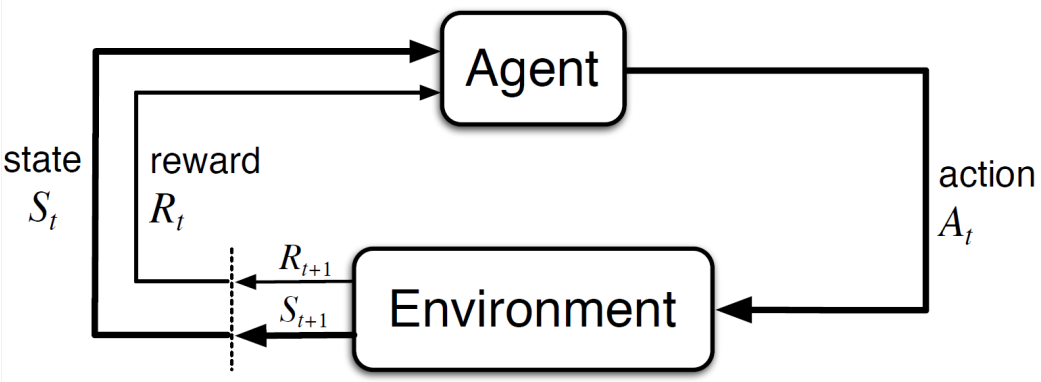


Figure 2.5: The agent–environment interaction in a Markov decision process [6]

The decision maker in RL is called the agent. The agent interacts with the environment by performing actions A_t at the current state S_t and receiving rewards R_{t+1}. In practice, the state is some representation of the environment that the agent observed. After each interaction, the environment along with the agent moves further to a new state S_{t+1}. The goal of the agent is to maximize the received rewards over time by performing the best actions in each state. This is formulated as finding a policy $\pi(s, a)$ —mapping from states to actions— that maximizes the received rewards.

2.5.1 Markov decision process

MDPs are a class of stochastic sequential decision processes under the Markov property, that is, future states of the process only depend on the current state

(and the current action). At each point in time, the decision maker, observes the state of the system, chooses an action, receives an immediate reward, and transits to the next state [134]. An MDP can be formulated as 4-tuple (S, A, P_a, R_a), where: :

- S is the set of states.
- A is the set of actions.
- P_a is the transition conditional probability distribution over all stats and actions (system dynamics).
- R_a is the immediate reward due to a as the system transits.

A reinforcement learning agent attached to such MDP will try to find the best policy π^* to maximize the return G_t:

$$G_t = \sum_t \gamma^t r_t$$

where r_t is the immediate reward received at time t and $\gamma \in [0, 1)$ is a discount factor used to prioritize immediate reward over future rewards.

To express the quality of states and actions, reinforcement learning algorithms generally involve estimating functions of states $V_\pi(s) = E_\pi[G_t|S_t = s]$ (or of state–action pairs $Q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a]$) that estimate how good it is for the agent to be in a given state (or how good it is to perform a given action in a given state) under policy π [6]. RL tries to find the policy π^* that maximizes the value function :

$$\pi^* = \underset{\pi}{argmax}\, V_\pi(s)$$

In the next section, we present related works that apply RL to misbehavior detection problems in C-ITS and related contexts.

2.5.2 Related works

Huang et al., [135] proposed a policy-based reinforcement learning for anomaly detection in time series. The state derived from the time series is composed of m previous actions of the agent and the m previous observations of the time series and is regarded as an infinite state space. The RL agent is required to perform the binary detection by outputting 0 or 1 where 0 represents normal behavior and 1 for an anomaly. The authors then define the reward signal by following the confusion matrix. The agent receives a positive scaled reward

when it correctly detects normal and anomalous behaviors (True positive and true negative). Otherwise, the agent receives a negative reward. The model was tested on 7 datasets and compared to a benchmark of algorithms including statistical methods, recurrent neural networks, and value-based reinforcement learning. The performance is measured using F-score. The model outperformed all benchmark algorithms on 4 of the 7 datasets and obtained a better overall score.

The model from [135] was afterward used in [136] where the authors effectively detect misbehaviors from the VeReMi dataset using RL-based misbehavior detection achieving 100% f-score on position-related misbehaviors and precision of 0.7692 and recall of 0.8571 for speed-related misbehaviors.

Xu et al., [137] proposed a sequential anomaly detection based on temporal-difference learning (TD-learning). The method can be used for network or host-based detection systems. The state is defined as the previous m observations from the sequential data. Observation is either a system call in host-based detection systems or a connection feature vector in network-based. The agent is required to raise an alarm in case of an anomalous sequence. A reward value of 1 is given whenever the agent correctly detects an anomalous sequence, otherwise, no reward is given. The system was evaluated on anomaly detection for host computers using system call data. The proposed anomaly detection method can achieve better detection accuracies than other approaches including SVMs, and HMMs

Pang et al., [138] proposed a novel deep reinforcement learning approach able to explore rare and novel anomalies beyond the ones present in the dataset itself. The authors define their method for any state $s \in R^d$ where the agent takes the action of labeling each state as anomalous or normal. The reward signal is composed of 2 parts. The first part quantifies the reward for detection. Accordingly, the agent receives a +1 reward if it correctly detects an anomalous state. It receives a +0 reward when it correctly detects a normal state. Otherwise, a -1 reward is given. The second part of the reward quantifies the abnormality of the state using an unsupervised model. For instance, the authors use iForest. Thus, the agent receives a real-valued reward $\in [0, 1]$ describing the abnormality of the state. The authors then evaluate the model on 48 real-world datasets where some classes of anomalous data are only present in the testing phase and never seen by the agent in the training phase. The model is compared with five state-of-the-art semi/unsupervised anomaly detectors. Overall, the proposed model outperforms the state-of-the-art detectors.

Zha et al., [127] proposed an anomaly detection algorithm using a meta-policy with reinforcement learning. Given any data set the authors first map the raw features to a new feature space by extracting a predefined

set of meta-features. These features are composed of "detector features", "anomaly features" and "normality features". This allowed the others to define a reinforcement learning anomaly detection algorithm that can be used on any dataset. The state is defined as the extracted features of the current observation. The agent is then required to take the action of selecting the observation for analysis or not. The agent receives a reward of +1 if the selected observation is anomalous, a reward of -0.1 if the selected observation is normal, and a 0 reward if the agent does not select the observation. The authors evaluated their proposal on 24 datasets and compared it with a set of baseline methods including semi-supervised and unsupervised learning. The proposed anomaly detection method outperformed all baselines on average over all data sets.

Feng et al., [128] proposed an anti-jamming countermeasure for V2V communications. They use reinforcement learning for power control and channel selection. The channel selection is formulated as a multi-armed bandit (MAB) problem. The state defined in this problem includes metrics on previously selected channels and evaluated risks. The upper confidence bound algorithm is used to solve the problem and achieves better performance than random and static policies.

Rasheed et al., [129] proposed a deep reinforcement learning algorithm for the detection of attacks on autonomous vehicle sensors and communications. The RL agent tries to keep an optimal safe distance from the vehicle in front of it by estimating the optimal velocity. The state is composed of the m previously received distance measurements from all sensors and communications (camera, LIDAR, V2X). The agent then estimates the optimal velocity. The reward is perceived as a variation from the optimal safe distance. The problem is solved using DQ-Learning in a generative adversarial network environment. Note we only consider the RL used for defending and that another RL is used to generate the attacks. The model is compared to a Kalman filter method and a state-of-the-art deep reinforcement learning method. The model achieves the lowest deviation from an optimal safe distance under different attacks. Similarly [130] solve the same problem using DQ-learning and LSTM.

As part of their security system, Ferdowsi et al., [131] used the MAB method to detect data injection attacks on the sensors of the vehicle. The MAB interacts with the vehicle dynamics to detect which sensor is being attacked at each time step. The authors define regret as the difference between the reward of the best possible arm at each step, and the generated reward of the arm that is played by the algorithm. Here, playing an arm is represented by the selection of a subset of the vehicle's sensors. The proposed detection approach obtained a substantially better performance compared to a Kalman

filter.

In [132], the authors defined an MDP framework over vehicular communications and considered an RL-based misbehavior detector at the RSU nodes following the MDP. The state is composed of previous actions and current messages, the actions set is 0 for genuine messages and 1 for false data, and the reward is weighted over actual labels. They used a value-based Q-learning method that takes the vehicular time-series data and outputs actions to take at each time t. They used the VeReMi dataset to train their model to detect variant DOS attacks. The results show that the model efficiently detects dos attacks with a 99.5% f-score.

In [133], the authors used RL to correct errors in vehicular messages. They used Q-learning to model the vehicles' behaviors. Firstly, a discretization of the vehicle attributes (distance, speed, acceleration. angle) has been made to allow for a finite number of states (72 states). The action space consists of two actions (0 for the genuine message, and 1 for the erroneous message). The reward is modeled by the difference between previous and current messages, and also the difference between messages before and after correction. They simulated the scenarios using the SUMO simulator. The results show that their approach can detect error V2X data up to 80%.

In our work, we first formulate the V2X context using stream graphs which allows us to motivate and understand the use of RL methods in such context. We focus on modeling our misbehavior detection problem using deep Q-learning as our proposed solution.

Surveyed research uses either real or simulated datasets. However, these datasets are usually kept private, which limits further studies using the same datasets.

In the next section, we present a set of tools and datasets that are freely available and useful in different subjects in the context of CAVs.

2.6 Simulators and datasets

The evaluation and validation of an MBD method usually require experimentation on a real or simulated scenario. In this section, we survey the main open-source simulators and datasets that can be used to test and validate MBD methods.

2.6.1 Classification

The MBD community uses several simulators of CAVs. We list the main ones in Table 2.9 where we classify these simulators through different criteria:

Ref	State	Agent and actions	Reward	Method
[135]	Past actions and observations	0 : normal observation. 1 : anomalous observation	A if true positive. -B if false positive. -C if false negative. D if true negative. A,B,C,D are positive numbers	Policy-based RL using LSTM network. (PTAD)
[137]	Past observations	Raise an alarm. Do not raise an alarm.	1 if an anomaly is detected. 0 otherwise.	TD-learning
[138]	$S \in R^d$	Label as normal Label as anomalous	$R_1 = 1$ if anomaly is detected. 0 if normal is detected -1 otherwise $R_2 =$ abnormality score $\in [0, 1]$ Reward $= R_1 + R_2$	Deep Q learning and iForest
[127]	$S \in R^d$ (a set of d predefined meta-features)	Select observation. Do not select observation.	+1 if selected observation is anomalous. -0.1 if selected observations is norm. 0 if no observation is selected	Proximal Policy Optimization (PPO)
[129]	$S \in R^{m \times d}$ history of measurements.	the optimal velocity V	δ = Variation from optimal distance	Deep Q-learning & LSTM-GAN
[131]	/	Select a subset of sensors	Difference between the reward of the best possible arm and the reward of the played arm	MAB
[133]	discretized vehicle attributes (distance, speed, acceleration. angle)	0 for genuine message, and 1 for erroneous message	the difference between messages before and after correction	Q-learning
[132]	previous actions and current messages	0 for genuine and 1 for false data	A if true positive. -B if false positive. -C if false negative. D if true negative. A,B,C,D are positive numbers	value-based Q-learning

Table 2.8: Overview of RL application to anomaly detection

- V2X data: whether the simulator (or an extension of it) provides data corresponding to the C-ITS communication stack (e.g., CAM messages).
- Sensor data: whether the simulator (or an extension of it) provides data corresponding to the flow of information perceived through the different sensors in the vehicle (e.g., images from the camera).
- Adversary module: whether the simulator (or an extension of it) provides the option to inject some type of attack (e.g., DoS, tampering) in the simulation.
- Security module: whether the simulator provides some security mechanisms (e.g., packet verification).

Simulator	V2X data	Sensor data	Adversary module	Security
VEINS	✓	✗	✓	✗
iTetris	✓	✗	✗	✗
CARLA	✗	✓	✗	✗
Matlab	✓	✓	✗	✓
Ventos	✓	✗	✓	✗
Artery	✓	✗	✗	✗
Vanetza	✓	✗	✗	✓

Table 2.9: Open-source simulators classification

As illustrated in Table 2.9, the simulators provide different kinds of outputs and thus can be used for diverse use cases. We give a brief presentation of each simulator in the next section.

2.6.2 Simulators

2.6.2.1 VEINS

Vehicles in network simulation is an open-source simulation framework [139]. The V2X communication module integrates several standardized V2X protocol stacks and the network simulator OMNeT++ [140]. VEINS integrates a traffic simulator named *Simulation of Urban MObility* (SUMO) [141]. Unfortunately, VEINS does not have an ADAS module and therefore, cannot simulate sensor measurements. However, VEINS provides a security module with an exhaustive library of security mechanisms related to MBD [142] or privacy preservation [143].

2.6.2.2 iTETRIS

An Integrated Wireless and Traffic Platform for Real-Time Road Traffic Management Solutions (iTETRIS [144]) is an open-source simulation framework. The V2X communication module uses NS-3 [145] as its network simulator and is compliant with the protocols stack defined by the ETSI standard. The Traffic mobility module uses SUMO as its traffic simulator. Unfortunately, iTETRIS does not provide an ADAS module or a security module.

2.6.2.3 CARLA

Car Learning to Act (CARLA [146]) is an open-source simulator for autonomous driving research. The simulator does not include a V2X communication module. However, some extensions managed to implement a V2X module for CARLA [147]. Also, CARLA integrates its traffic simulator for both vehicles and pedestrians. Moreover, CARLA integrates an ADAS module, which includes several sensor models and tools to conceive perception, planning, and control systems. However, CARLA does not include any security modules.

2.6.2.4 Matlab

Matrix laboratory (Matlab [148]) is a multi-paradigm numerical computing environment and proprietary programming language. Recently, Matlab has provided libraries to simulate, conceive, and evaluate systems related to an ADAS context. Matlab does not include a V2X module. However, some extensions propose an open-source V2X module. The traffic module uses basic traffic modeling based on the object trajectory. With that said, Matlab does provide driving scenarios or tools to set the object trajectory.

2.6.2.5 VENTOS

Vehicular Network Open Simulator (VENTOS [149]) is a V2X simulation framework based on IEEE V2X standards. VENTOS integrates *SUMO* and *OMNeT++*. However, VENTOS does not include sensor models and therefore, does not support Cooperative ADAS applications. Thus, the VENTOS scope is limited to the connected vehicle.

2.6.2.6 Artery

Artery is a V2X simulation framework based on ETSI ITS-G5 protocols (Unlike *VEINS*) [150]. Just as *VEINS*, *Artery* integrates *SUMO* and *OM-*

NeT++.

2.6.2.7 Vanetza

An open-source implementation of the ETSI ITS-G5 stack. *Vanetza* [151] provides several classes to generate messages such as CAM and DENM and PKI certification. *Vanetza* is a communication-driven simulator, as it does not implement any vehicular sensors. Besides PKI certification, the simulator does not provide other security/adversary modules.

Several domain applications can benefit from this set of simulators such as traffic modeling, VANET applications, network evaluation, and testbeds for cybersecurity.

2.6.3 Datasets

Similarly, several datasets are used by the MBD community. In the following, we list the main datasets found.

1. Next Generation Simulation (NGSIM) [152] has recorded vehicle movements on various US roadways.

2. Safety Pilot Model Deployment (SPMD) [153] dataset has more than 5.6 TB of recorded BSMs.

3. The Wyoming Connected Vehicle Pilot (WCVP) [117] dataset has V2X data collected on US road I-80.

4. VeReMi [9] has several simulated misbehavior types (extended in [155] with new misbehavior types).

5. Dimensions dataset [59] contains dimensions of several thousands of vehicles and injected implausible dimensions. The resulting dataset can be used for MBD.

6. Bristol dataset [154] contains CAM exchanged between two On-Board Units and four RSUs.

Then, in Table 2.10, we classified each dataset based on several characteristics: covered safety message fields, the presence of failure causes, physical signal data, metadata, and the CAV context.

In this section, we briefly presented different simulators and datasets that can be used to evaluate an MBD technique. We compared these materials using criteria that can help choose the necessary material for a given task.

Data Features	Description	Dataset					
		NGSIM [152]	SPMD [153]	SPMD [117]	VeReMi [9]	Dimensions [59]	Bristol [154]
Standard Elements							
DSRCmsgID	Identifier for message type	✗	✓	✓	✓	✗	✓
SecMark	Timestamp	✓	✓	✓	✓	✗	✓
MsgCount	Message Number for a sequence	✗	✓	✓	✓	✗	✓
TemporaryID	Network ID	✗	✓	✗	✗	✗	✓
Latitude	Position along latitude axis	✗	✓	✓	✓	✗	✓
Longitude	Position along longitude axis	✗	✓	✓	✓	✗	✓
Elevation	Elevation to the sea level	✗	✓	✓	✓	✗	✗
Speed	Object speed	✓	✓	✓	✓	✗	✓
Heading	Angle between object & North	✗	✓	✓	✗	✗	✓
Yaw Rate	Heading per second	✗	✓	✗	✗	✗	✗
Lat. Accel	Acceleration along latitude axis	✗	✓	✗	✗	✗	✗
Long. Accel	Acceleration along longitude axis	✗	✓	✗	✗	✗	✗
Vet. Accel	Acceleration along vertical axis	✗	✓	✗	✗	✗	✗
Positional Accuracy	Accuracy at one standard deviation	✗	✓	✗	✗	✗	✗
Brake System Status	Status of the Brake System	✗	✓	✓	✗	✗	✗
Vehicle Length	Vehicle Length	✓	✗	✗	✗	✓	✗
Vehicle Width	Vehicle Width	✓	✗	✗	✗	✓	✗
Physical Signal Data							
RSSI	Received Signal Strength Ind.	✗	✗	✗	✓	✗	✓
Metadata							
Sender ID	Emitter Identifier	✗	✓	✓	✓	✗	✓
Gentime	Time of message creation	✗	✓	✓	✓	✗	✓
Receiver Data							
Receiver ID	Receiver Identifier	✗	✓	✗	✓	✗	✓
Receiver Position	Position along the X-Y-Z axis	✗	✗	✗	✓	✗	✓
Cause of Perception Failures							
Attacks	Presence of attacks	✗	✗	✗	✓	✓	✗
CAV Type							
#Type	Diversity of CAV types	✗	✗	✗	✗	✓	✗

Table 2.10: Datasets for CAV with available features

For example, for the detection of anomalous 3D objects in the perceived images, one would use CARLA or MATLAB instead of the others since it provides sensor data, as illustrated in Table 2.9.

2.7 Conclusion

In this chapter, we presented the context of CAV and C-ITS. We introduced the main aspects, including, ITS-stations, communications between these stations, and also vehicle components. We provided an overview of state-of-art definitions and classifications of misbehaviors in the context of connected and autonomous vehicles. Moreover, we presented classical and current machine learning algorithms for misbehavior detection in the context of CAV. Finally, We also surveyed tools and data sets that are freely available for use in MBD. We found limitations to the current misbehavior definition and model and also to the current misbehavior detection classification in the context of CAVs, and thus, in this chapter, our first contributions are a new misbehavior definition and a new model covering both current attacks and faults. Furthermore, we propose classifications of both misbehavior and misbehavior detection methods in chapter3 extending the proposed definition and model in this chapter.

In the next few chapters, we will discuss and analyze the current state of the art and propose improvements in several topics. Specifically, as we observed the lack of a concise framework for the detection of misbehavior in C-ITS, we propose in chapter 4 a misbehavior detection architecture tailored for the V2X communications stack as it takes advantage of the communication layers' structure to define several modules aiming to detect, classify, and report misbehaviors. To implement our architecture, we noticed the unavailability of simulation frameworks that provide C-ITS data such as V2X, camera feed, and map knowledge in one environment. Thus, relying on the tools described in this chapter, we present in chapter 6 our simulation framework that provides such data and allows for attack injection, detection, and visual interpretation. In our work, we mainly focus on applying machine learning to detect misbehavior, and we noticed that the classical use of machine learning in the current literature usually leads to data leakage. We propose in chapter 6 our solution in the form of a spatiotemporal cross-validation approach to avoid such a problem. We further investigate reinforcement learning in chapter 5 as we develop its theoretical and applicable framework with respect to the C-ITS context. We show that it shares some fundamental characteristics of the vehicular environment and propose our own RL model for the detection of misbehavior.

Chapter 3

Characterizing misbehavior and its detection in C-ITS

In this chapter, we extend to our contribution from chapter2 where we proposed a definition and a model for misbehavior in C-ITS. We provide here classifications of both misbehavior and misbehavior detection methods.

3.1 Classifying misbehavior

We overviewed in chapter 2 the technologies involved in CAV and C-ITS and their security issues and solutions. We observed the continuous evolution of possible misbehavior in the context of CAV and also research and industrial effort done to provide efficient misbehavior detection mechanisms. But, these researches usually target specific security attacks and failures in the system. And thus, with respect to the current literature on misbehavior and its classification, we find it difficult to classify such research works due to limited coverage of present faults and attacks in Tables 2.1, 2.2, and 2.3. In particular, they explicitly show security and safety issues that should be considered when developing misbehavior detection mechanisms and architectures.

We sense the need for structuring these practical examples of attacks and failures into a concise model from which we can derive detection mechanisms for a class of faults and attacks rather than for specific examples of attacks or faults. We believe this will help us understand the causes and effects of these issues and facilitate the development of new security mechanisms. We discuss the limitations of the current literature on misbehavior classification when applied to attacks and failures presented in chapter2. We propose a hierarchical misbehavior classification to describe simple (resp., complex) misbehavior affecting an Item (resp., multiple items) of a CAV that is easily extensible

	Paper	[47]	[42]	[48]	[49]	[35]	[38]	Ours
Attack	**Component details**	✓	✓	✓	✓	✗	✗	✓
	Threat classification	✓	✓	✓	✗	✗	✗	✓
	Context attacks	✓	✗	✗	✗	✗	✗	✓
Fault	**Component details**	✗	✗	✗	✗	✓	✓	✓
	Threat classification	✗	✗	✗	✗	✗	✗	✓
	Context Fault	✗	✗	✗	✗	✓	✗	✓

Table 3.1: Literature: misbehavior classifications analysis

to newly defined components and constitutes a basis for the classification of research done in developing misbehavior detection mechanisms [17].

The surveyed misbehavior classifications from chapter 2 [35, 38, 42, 47–49] do not generalize to all existing misbehavior types. In Table 3.1, we compare these taxonomies based on several criteria:

- Faults/Attacks: the taxonomy accounts for both.
- Component description: the taxonomy describes the targeted and failing components.
- Threat classification: the taxonomy categorizes the threat raised by the misbehavior.
- Context: the taxonomy considers failures or attacks in the vehicle's surroundings that may cause the CAV to fail.

We thus propose a new misbehavior model and a new classification that better expresses the roots and threats of misbehavior and covers both malicious actions and internal faults of a component in the CAV.

Based on our misbehavior definition 2.7 from the last section, we developed a misbehavior classification with mainly two parts: intentional (Figure 3.1) and unintentional (Figure 3.2).

3.1.1 Intentional misbehavior

Our classification covers simple misbehavior affecting one vehicle's item and complex misbehavior that may affect multiple items at the same time. The classification is extendable to consider other vehicle items that might be added in future generations of CAV. The IMB raises several threats that we characterize using the STRIDELC threat model. Figure 3.1 shows a hierarchical tree, characterizing any given (unit) intentional misbehavior by one

path from the tree root to a leaf. A set of such paths may describe complex misbehavior. It also shows how we divide it into two categories: physical (where a physical component has been intentionally tampered with) and non-physical categories, and for each, we select subcategories according to C-ITS standards and vehicular architectures. We put the whole classification under a security threat model (STRIDELC). If we take a Sybil Attack as an example, we can draw a path for it in the tree (i.e., intentional, non-physical, outside, wireless connectivity, facilities, spoofing, and confusion). Based on this tree and its set of paths, we define a new CAV classification for IMB. In the case of newly discovered misbehavior, the classification is easily extendable to support it by adding leaves or intermediate ITS-Ss under the corresponding category on the current tree.

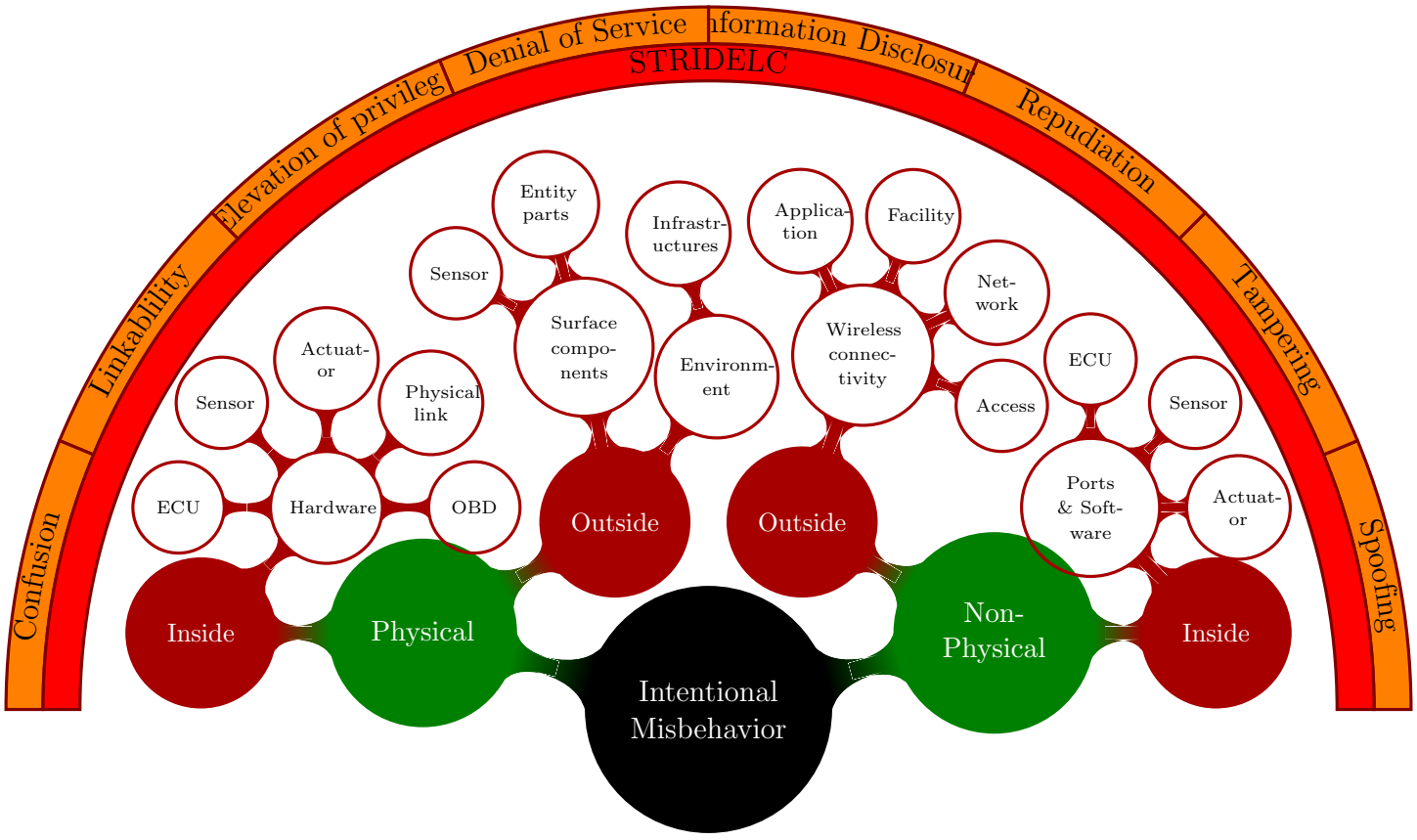


Figure 3.1: Intentional misbehavior classification.

3.1.2 Unintentional misbehavior

The same issues from the last section arise for UMB, in addition to faults in the CAV components, we consider extravehicular factors such as weather, road conditions, and signs. We similarly draw a hierarchical tree to identify misbehavior that is not the outcome of a malicious attack. In the same manner, as the one for IMB, we can represent the misbehavior where a planner fails to anticipate the other driver's behavior (Unintentional, Nonhuman, Element/item, Software fault, Perception). Figure 3.2 shows haw we divide it into three categories: Item fault for errors occurring because of an item, Design Fault and External Environment Fault categories and derive subcategories according to C-ITS standards.

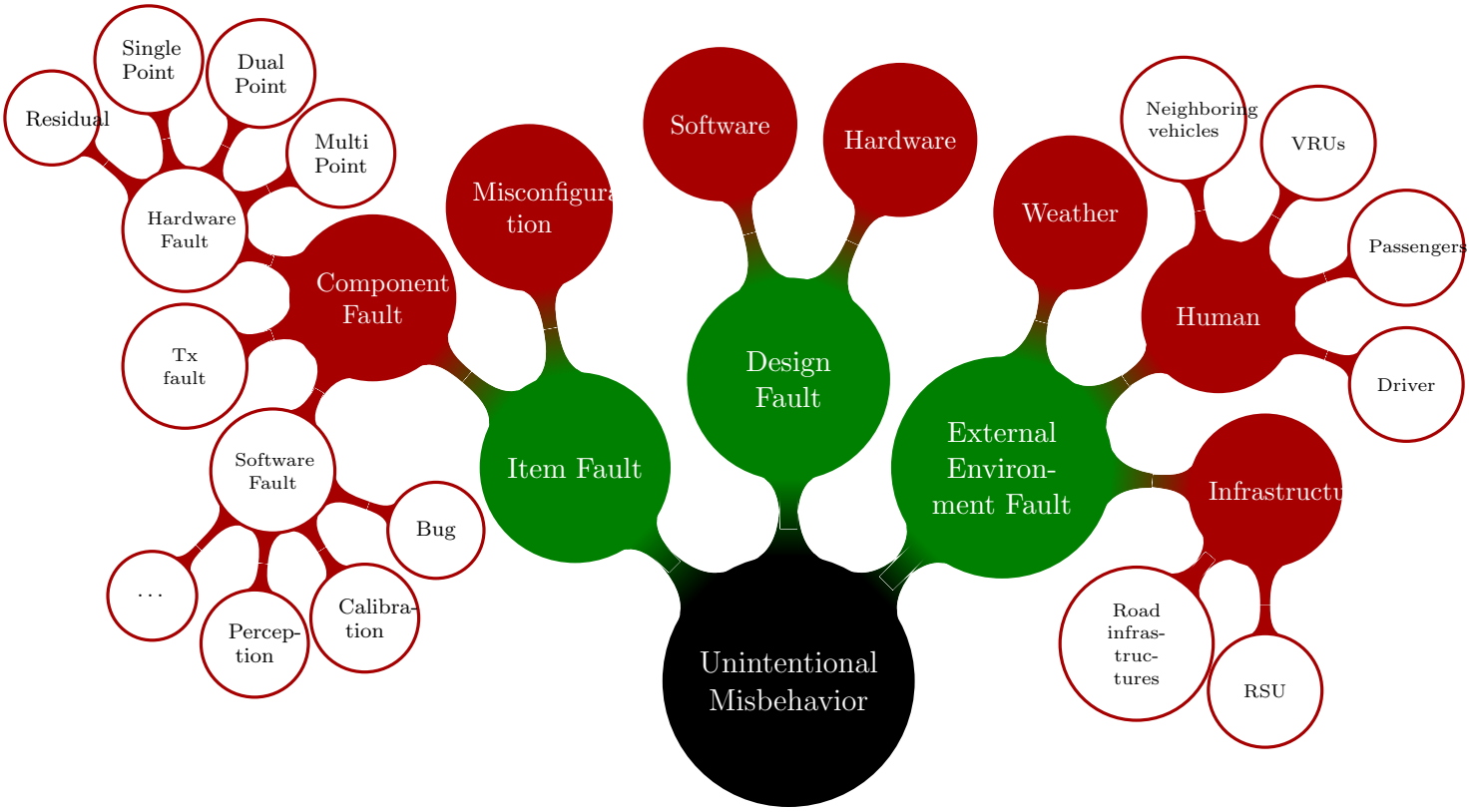


Figure 3.2: Unintentional misbehavior classification.

3.1.3 Discussion

Compared with the taxonomy proposed in [47], our misbehavior model describes the misbehavior with the information available in real-time and is

most useful for anomaly detection and ML algorithms while keeping as much relevant information as possible. It's worth noting that knowledge of tools used for the attack or information about manufacturers of the car is either not available at the time of the event or not relevant for detection. Also, our model covers the UMB resulting from faults in the system.

In [42], the taxonomy covers the stages of an attack and the connected elements (attacker, target, motive, consequence). However, this taxonomy lacks a hierarchical dependency between the target and the attack vector, which can clarify the limits of some attacks. Also, the taxonomy does not cover physical attacks (e.g., vandalism) and indirect attacks on infrastructures. As in [47], the taxonomy does not cover faults and UMB due to the vehicle's context. The improvements on [42] proposed in [48] do not solve these limits, as they are made to include attacks on data streams only. [49] gives a listing of possible attacks on autonomous systems, but does not specify the component targeted by the attack or CAV context-specific threats and also does not cover components' faults.

The taxonomy proposed in [35], simplified in [38], covers the primary sources of failure. Nevertheless, it lacks details on which parts of the CAV fail to perform and their hierarchical order. For instance, we would put mechanical failure under hardware. Moreover, the construction zone seems too specific to be at the same level as weather and road conditions. For details, we prefer to use the ISO 26262 standard definition of hardware faults and failures. Also, this taxonomy only covers CAV's failures and omits malicious attacks.

Our proposed misbehavior model, definition, and classification consider the previously mentioned limits in the literature on misbehavior taxonomies and give a better representation of misbehavior in CAV. It can be used in real-time MBD since the features composing the model can be easily acquired in real-time. This helps in describing the misbehavior that occurs on the CAV and enables us to determine the severity of the misbehavior and the flow of actions needed to maintain the safety and security of the CAV and its passengers. For example, a warning message informing of a failure of the emergency braking system due to some internal error may help the driver take decisive action.

Our model and classification can also be used for offline MBD and access revocation. Indeed, some misbehaviors might not be fully described in a real-time scenario and must be analyzed offline to facilitate their detection in the future. The knowledge stored by our model can be used in the form of a global misbehavior report sent periodically to a Misbehavior Authority that can analyze and act on this knowledge.

We've defined in this section the misbehavior, its model, and classification

in the context of CAVs. Ideally, when misbehavior occurs, it should be detected, labeled, and characterized. MBD mechanisms should provide such characterization whenever they identify misbehavior. Our model acts as output features for MBD mechanisms and, in particular, the ML-based ones can rely on these features to characterize their results. In the next section, we present ML-based algorithms, characterize them, and give insight into their use for MBD.

3.2 Characterizing misbehavior detection in C-ITS

As stated in the previous section, misbehavior in the CAV context presents a high risk for the vehicle and its passengers. Thus, it needs special handling, first by detecting it and then by applying the favorable reaction to ensure the efficiency and security of the CAV.

In the following, we analyze recent advances in MBD using ML in the context of CAVs and propose a taxonomy that covers all the research. In contrast with the survey [50], where classical methods such as node-centric and data-centric methods are classified, we focus on ML methods.

As thorough as they are, the different taxonomies in the literature are not adapted for our purpose. We focus on three main aspects: the classification depth, the methods covered, and the adaptation to the context of CAVs. In contrast, the authors in [68] gave a high-level classification of ML techniques without an explicit focus on MBD and CAVs context, and thus their classification does not give sufficient details about the underlying algorithms in each category. Moreover, in [67], the taxonomy only mentions global categories, and no examples of actual techniques are given. In [69], a large listing of techniques is given, but lacks hierarchical structuring and does not help us select and understand the underlying techniques. In [70, 71], only unsupervised techniques are covered, and the authors discuss advances in neural network-based methods applied to aviation. Inspired by the surveyed taxonomies, we propose our new taxonomy of ML-based MBD techniques that provides a wider scope and covers previously mentioned works and anomaly detection for CAVs.

3.2.1 Proposed taxonomy of ML based MBD techniques

In this section, we present a new classification of ML-based algorithms. In particular, our goals here are as follows:

- to highlight the most fundamental choices in ML algorithms for MBD in the context of CAVs.
- to offer insight into which algorithm to use.
- to provide an overview of modern MBD techniques.

Throughout this section, anomaly detection, novelty detection, outlier detection, and MBD are equivalent notations. Our goal is to provide insight into choosing ML algorithms, we first formulate the different characteristics of the data used in our context.

In the context of CAV, the data extracted can be of several types and inhibits several characteristics.

- *Single hop*: data transmitted only once from one ITS-S to another.
- *Multi-hop*: data transmitted through several ITS-Ss.
- *Broadcast*: same data transmitted to multiple ITS-Ss.
- *High frequency*: data frequently transmitted by ITS-Ss.
- *Multi-frequency*: data transmitted by multiple sources with different frequencies at each source.
- *Spatiotemporal*: data constrained to time and space.
- *Multi pseudo-id*: use different identifiers to send data.
- *Numerical*: valued numerical data.
- *Categorical*: a dataset containing values/observations that can be sorted according to category.
- *Univariate* : this consists of single (scalar) observations recorded sequentially over equal time increments [156].
- *Multivariate* : multiple observations recorded sequentially over equal time increments [157].
- *Discrete*: a dataset containing values/observations that are distinct and separate.
- *Continuous*: a dataset containing values/observations that take on any value within a finite or infinite interval.

For example, the flow of images extracted from the vehicle's camera contains numerical data constrained to temporal dependencies. These images contain multiple variable pixels and, as such, are considered multivariate time series. The same data could also include object classification, that is to say, a categorical variable. Another example would be the CAM transmitted between the vehicles. A CAM contains mainly numerical data that evolve over time and space that generates the CAM spatiotemporal time series data. These characteristics of the data are crucial for identifying which method fits best for the corresponding data. We constructed in Table 3.2 a set of characteristics of the data used in the literature to help to provide insight on how to use the ML classification proposed in this paper. As can be seen in the last column, we classified each surveyed method with respect to the proposed classification in Figure 3.3.

Data Type	Data characteristics	Malicious actions and errors	Algorithm, paper, and category
Messaging data: Vehicle data, Environment data (e.g., CAM)	Single hop, Multi hop, Broadcast, High frequency, Multi-frequency, Spatiotemporal, Multi pseudo-id, Numerical, Categorical, Multivariate	Tampered content, Erroneous content, DoS, Abnormal frequencies, Abnormal behaviors, Delay, Authentication	**Supervised**: • Neighbor : K-NN [60, 93, 94, 99, 116] • TreeEnsemble : DT [94], RF [59, 94, 99, 100], A-Boost [59], XGB [15, 100],LGBM [100], Bag [94] • Neural networks : ANN [83,88,100], MLP [15,59,97,158], LSTM [15,98,100], CNN [98] • Statistical : SVM [15, 60, 82, 84, 93, 95], LR [94, 95] **Unsupervised**: • Clustering : K-means [72, 77], Fuzzy STS [75], Fuzzy C-means [97] • Domain : OC-SVM [77]
Network data: extracted measures from monitoring the communication (e.g., delivery rate)	Multivariate time series, Univariate time series, Discrete, Continuous, Computed, Aggregated.	DoS, transmission error, Packet loss	**Supervised**: • Neighbor : K-NN [74, 107] • TreeEnsemble : DT [74, 107], RF [74, 107], A-Boost [74, 107] • Neural Networks : ANN [73, 79, 83, 88, 92], MLP [115] • Statistical : SVM [76, 81, 82, 92, 96, 110], NB [74, 107] **Unsupervised**: • Clustering : K-means [77], Fuzzy C-means [87] • Projection/Reconstruction : Deep-Autoencoders (DAE) [96]
Physical data: received signal strength (RSS)	Continuous, Time series, Physical interference, Raw, Aggregated	fake position, phantom objects	**Supervised**: • Neighbor : K-NN [74, 93, 107] • TreeEnsemble : DT [74, 107], RF [74, 107], A-Boost [74, 107] • Statistical : SVM [76, 93, 95, 96], NB [74, 107], LR [95] **Unsupervised**: • Projection/Reconstruction : DAE [96]
Plausibility checks data: discrete constraint satisfaction	Discrete, Time series, Raw, Aggregated	Erroneous if computed on genuinely faked data	**Supervised**: • Neighbor : K-NN [60, 74, 93, 94, 107] • TreeEnsemble : DT [74, 94, 107], RF [74, 94, 107], A-Boost [74, 107], Bag [94] • Neural Networks : ANN [83] • Statistical : SVM [60, 93, 95], NB [74, 107], LR [94, 95]
Fuzzy plausibility checks data: constraint satisfaction	Continuous, Time series.	Erroneous if computed on genuinely faked data.	**Supervised**: • TreeEnsemble : XGB [15, 100], RF [100], LGBM [100] • Neural Networks : MLP [15, 100, 158], LSTM [15, 100] • Statistical : SVM [15]

Table 3.2: Data characteristics

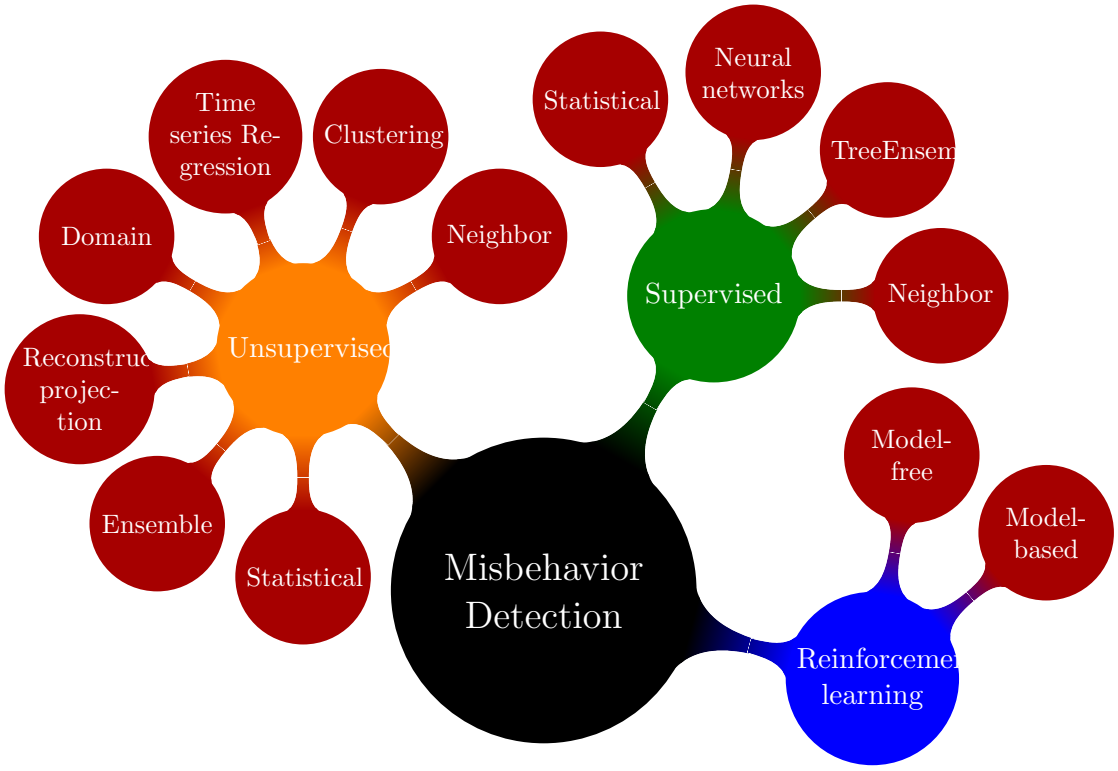


Figure 3.3: Proposed taxonomy of Misbehavior detection using ML

Our classification divides MBD into three main categories: unsupervised/semi-supervised, supervised learning, and RL (see figure 3.3). In the next few sections, for each category, we elaborate on four points. We first give a brief presentation of the category, followed by a comparison of the algorithms found in the category. We then provide references where such methods were used and offer insights into their use in the context of CAVs.

3.2.1.1 Unsupervised/semi-supervised learning

In this category, we derive several subcategories based on the nature of the methods, namely, neighborhood, clustering, time series regression, domain, projection, ensemble, and statistical.

(i) *Neighborhood-based category*
This category covers methods defining a distance measure of a point to its local neighbors for each point in the data. We put in this category the method Local Outlier Factor (LOF) [159] and its variants (Connectivity based outlier factor [160] and Local Outlier Probability [161]). Seemingly, neighbor-based approaches work better when the data exhibit underlying spatial dependencies (positional data) such as CAM.

We notice from the works [162–164] that these methods can detect implausible positions that differ largely from the spatial distribution of genuine data. We recommend using such methods when validating positional knowledge and whenever possible anomalies are distant from normal data. Other neighbor-based approaches exist; we refer the readers to algorithms such as INFLO [165], and LOCI [166].

(ii) *Cluster-based category*
This category covers methods that identify clusters of points in the data and are robust to outliers.
Similar to neighbor-based methods, cluster-based methods are applied to structural data where genuine points form a cluster. The main difference is that these methods identify clusters in the data, whereas the neighbor-based methods compute a score of abnormality for each point, which makes it easier to extract the genuine behavior. We recommend using cluster-based methods when it is assumed that there are no clusters of anomalies and that the normal data form identifiable clusters. Cluster-based methods were used to detect anomalies in the communication messages and in network data. Authors [167] applied DBSCAN to detect anomalies in network data from vehicle communications. DBSCAN was also successfully applied to vehicle communication for anomaly detection in positional data in [168].

(iii) *Time series regression-based category*
These category models the general observed phenomena in a forecasting fashion and then compare the newly obtained data to the forecasted one. If the difference is higher than a certain threshold, the data point at that timestamp is considered an anomaly.
Time-series regression-based methods fit better for sequential data that inherit a time dependency between subsequent observations, such as the evolution of speed over time. We recommend using these methods when such assumptions are observable. In the context of CAV, time-series regression-based methods were used to detect anomalies and incidents in traffic-related data extracted from V2V communications [169] and for modeling the vehicle's mobility pattern [168]. It was also used for detecting attack injection in the CAN network [170].

(iv) *Domain based category*
This category includes the One-Class Support Vector Machine (OC-SVM [171]). OC-SVM is an unsupervised algorithm that models genuine data. In a way, OC-SVM aims to separate inliers from outliers by creating boundaries over genuine data points, thus defining the dominant class's domain. It is also seen as a function that is positive inside the domain and negative outside. Points that are outside of the domain

are labeled as anomalies. In the context of CAV, OC-SVM is used to detect DoS attacks leading to sensor anomalies [122, 124].
We recommend using OC-SVM on numerical data where genuine points are found in higher-density areas on the feature space because this method finds boundaries over such areas, which construct the domain of genuine data.

(v) *Projection/ reconstruction based category*
In this category, the methods do not explicitly label data as anomalies. Instead, they represent the data in a way that renders the detection of anomalies easier. Particularly, deep belief networks (DBN) which is a novel deep learning based architecture composed of multiple hidden layers greedily (layer by layer) trained to reconstruct the input, have achieved good results in misbehavior detection. The features learned in the hidden layers allow for easier separation of different attacks. Projection methods were used for location anomaly detection [96, 172], Sybil attack detection [173], and on vehicle communications and anomalous objects detection [122, 174, 175].
Since these methods project the data in a different space, we recommend using these methods when it is constraining to work on the original data feature, whether due to high dimensionality or the difficulty of detecting anomalous data in the original space.

(vi) *Ensemble-based category*
This category combines multiple ML algorithms to solve the problem as an ensemble.
These methods were applied to detect anomalies in automotive sensors [122], in a CAN [176], in V2X communications [177], and in driving behavior [178]. Its use depends on the methods used to create the ensemble.

(vii) *Statistical category*
This category covers density estimation, histogram-based, and statistical features. We also include Robust Kernel Density Estimation (RKDE [179]).
These methods were used to identify the location of unwanted objects on the road, based on V2V messages in [180], and which were used for monitoring road traffic and vehicle behavior [181]. Similarly to cluster-based methods, this should be used when it is assumed that there is significant density for the genuine data.

3.2.1.2 Supervised learning

Under supervised ML, we also derive subcategories, namely, neighbor, treeensemble, neural networks, and statistical methods.

(i) *Neighbor based category*
The supervised version of neighbor-based methods relies on the labels provided in the training set to identify new points. The new point is given a label depending on the most used label in its neighborhood in the training set. Authors in [93, 94] used K-NN to detect message falsification. In [99], it was used to detect jamming in the vehicle network. It was also used for intrusion detection [80] and Sybil attack detection [116]. K-NN can be applied easily to most data types.
For MBD, it is best to use neighbor-based methods when the misbehaviors have similar characteristics (e.g., attacks) since these methods rely on the closest data to decide. Note that neighbor-based methods tend to misclassify these data points when the misbehavior acts as noise (multiple data points spread in the feature space).

(ii) *Tree/ensemble-based category*
This category is based on building a decision tree that maps input features to the target based on a set of splitting rules applied to the values of each feature. These methods were used effectively for the classification and detection of multiple misbehaviors in V2X communications [15, 59, 100]. They fit a set of small models to the underline data. Each model can specialize in detecting one class of misbehavior. This helps to achieve better performance, as noticed in the mentioned works. We recommend using such methods for separating misbehavior classes among themselves.

(iii) *Neural-network based category*
Neural networks are connected units that receive, process, and forward signals (real numbers) from other units. The 'process' component is a weighted mean of all received signals, where weights represent the strength of the signal. These computations can relate input features to desired target features and are thought of as a function of the input. Since this function is neither specified nor understood, neural networks are black boxes. This category includes the Multi-Layer Perceptron (MLP), Long short-term memory (LSTM), and also novel graph convolutional networks (GCN), and graph attention networks. In the context of CAV, neural networks were used extensively to detect misbehavior anomalies in V2X communications and sensor

data [15, 79, 83, 88, 90, 92, 98, 100, 123]. Neural networks offer excellent performance but require large sets of data and intensive computations. In such a setup, we recommend using these methods when data is available at a low cost and when real-time requirements can be met.

(iv) *Statistical category*
We find in this category SVMs, Logistic regression (LR), and Naive Bayes (NB) algorithms. Statistical methods consist of the baseline methods and are thus extensively used for misbehavior detection in the context of CAV [15, 60, 82, 84–86, 92, 93, 95, 96].
These methods are easily computed and explainable, which helps understand the data and the misbehavior. Using such methods imply that a certain set of assumptions on the data are met (e.g., i.i.d variables, Bayes assumption). We recommend using statistical methods such as SVM when these assumptions can be made.

3.2.1.3 Reinforcement learning

RL algorithms aim to learn the best action to take given the current state of an agent in a given environment. The algorithm finds the most rewarding action at a given state. These algorithms are classified into two subcategories, namely, model-free and model-based RL.

(i) *Model-based RL category*
These algorithms require a model of the environment (i.e., the transitions and rewards between all possible states). Such a model is usually unavailable to the agent, and the agent must learn the model in this case.
Since model-based RL requires a model environment, it is not easy to apply such methods in CAVs where the environment is complex and includes anything that interacts with the vehicle.

(ii) *Model-free RL category*
These algorithms do not require a model of the environment. They optimize a policy, i.e., the action to take a state or an approximator Q-function for the optimal action-value function through trial and error. The difficulty of applying such methods is in the method's capability to understand the current context and the presence of a reward for each action it performs, and thus applicable to any data.

3.2.2 Discussion

In C-ITS, we can find multiple kinds of misbehavior that constrain some methods more than others. Also, the data available at every moment for the MBD may vary depending on the context. For that, choosing a supervised, unsupervised, or reinforcement learning method is critical. We highlight some of the main criteria for such a choice. Firstly, the problem solved differs from one class of methods to others; RL solves a goal-oriented problem based on a reward metric without looking for structures in the data as in the case of unsupervised learning. Supervised learning solves sub-problems based on the available labels. Secondly, since supervised learning is based on labels, these labels must be available and that constrains the method to detect only misbehavior classes already known. Unsupervised learning doesn't require such labels and only finds structural dependencies in the data. Though, it requires some post-classification to understand and/or classify the structures of misbehavior. Similarly, RL methods don't require a label but need an appropriate reward metric from the environment and also post-classification to understand the misbehaviors. Finally, to apply RL, an interactive environment is required to train the method. The real environment is a high risk in this context, as for several steps the RL method will be exploring the C-ITS environment. Thus, it's usually done through simulation first.

Our classification covers supervised, unsupervised, and RL methods that can be applied to MBD and puts them into different subcategories. This classification aims to help the reader search for an appropriate algorithm for the task at hand and give a general overview of the methods used for MBD.

With the same mindset of providing insights and tools for applying machine learning to MBD, we observed a need for data in this context. In chapter 2 we presented a set of tools and datasets found in the literature that can be utilized to apply machine learning, we presented some advantages and disadvantages of such tools. One common disadvantage of those is the lack of a concise simulation or data set that provides synchronized multi-sources of data (camera, LiDAR, V2X communications, weather, ... etc.). In particular when it comes to applying reinforcement learning where interaction is needed.

Now that we understand misbehavior in C-ITS and know how to classify and describe it, we move on to detecting misbehavior. In the next chapter, we present our architecture CAV-MBDA for misbehavior detection for CAVs. We use the classification of detection methods proposed in this chapter to define our detection module and the classification and model of misbehavior for our decision module. Finally, we show an example pipeline of detection using CAV-MBDA.

Chapter 4

CAV-MBDA: a multi-layered architecture for misbehavior detection in C-ITS

We discussed in the previous chapters how important it is to secure the C-ITS system from the vehicle's point of view. We stressed the importance of monitoring this data for any anomalies to guarantee the safe operation of the C-ITS system. We described our definition and model of misbehavior that will help us build our detection architecture. In this chapter, we describe our solution CAV-MBDA; a multi-layered architecture for the detection of such misbehaviors locally within the vehicle that takes into account all previously discussed kinds of data, and the different communication layers. We define the whole pipeline from sensing knowledge to detecting misbehavior.

4.1 Overview

We want to design an architecture for the vehicle that, from the information sensed and received in the environment, is capable of processing and preparing resourceful data, detecting misbehaviors in the data at different levels, and acting on that detection by issuing decisions, reactions, and reports. From this idea, we build the first overview of our architecture that includes three connected modules: preprocessing, detection, and decision. Each module will then be expressed through a set of functions, levels, and layers.

Recalling the physical architecture from chapter 2, we want our architecture to be able to monitor and secure all flows of data in the vehicle, that might come from components such as actuators, and sensors. Thus, we place

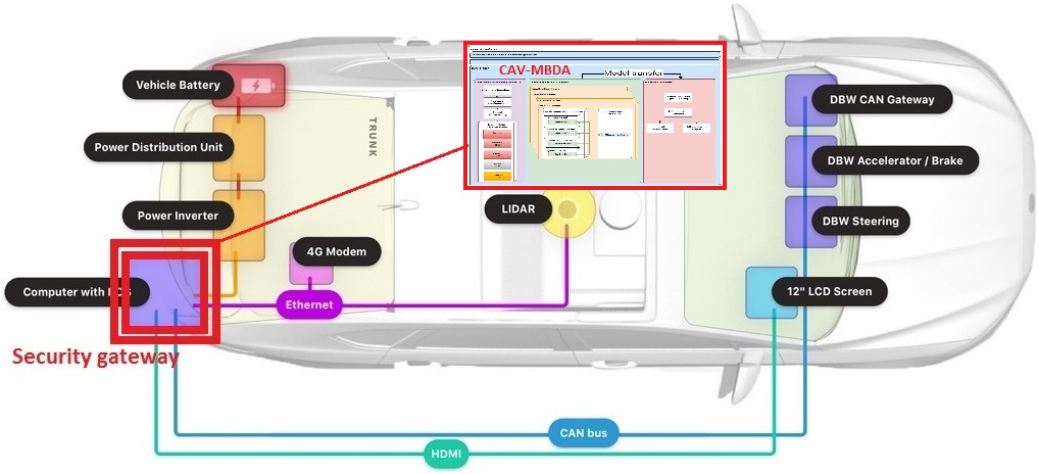


Figure 4.1: Physical architecture of a self-driving car with our CAV-MBDA as part of the central security gateway [2]

our architecture modules around the main computing unit as shown in Figure 4.1. The architecture will then process all/required incoming feeds to detect misbehaviors in the system.

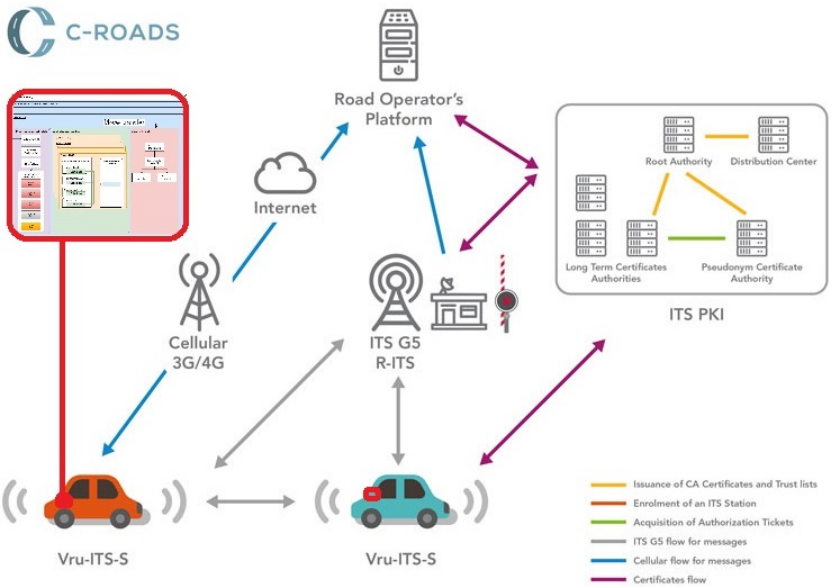


Figure 4.2: CAV-MBDA as part of C-Roads project architecture [7]

It is also possible to distribute the architecture modules over the vehicle as we will show after defining the architecture modules. We assume that the architecture CAV-MBDA is placed as shown in Figure 4.1 as a module of

a central security gateway. Similarly, we show in Figure 4.2 how our CAV-MBDA would be implemented as part of the whole C-ITS system. CAV-MBDA would be able to communicate with PKI and road operator platforms to communicate and exchange detection reports.

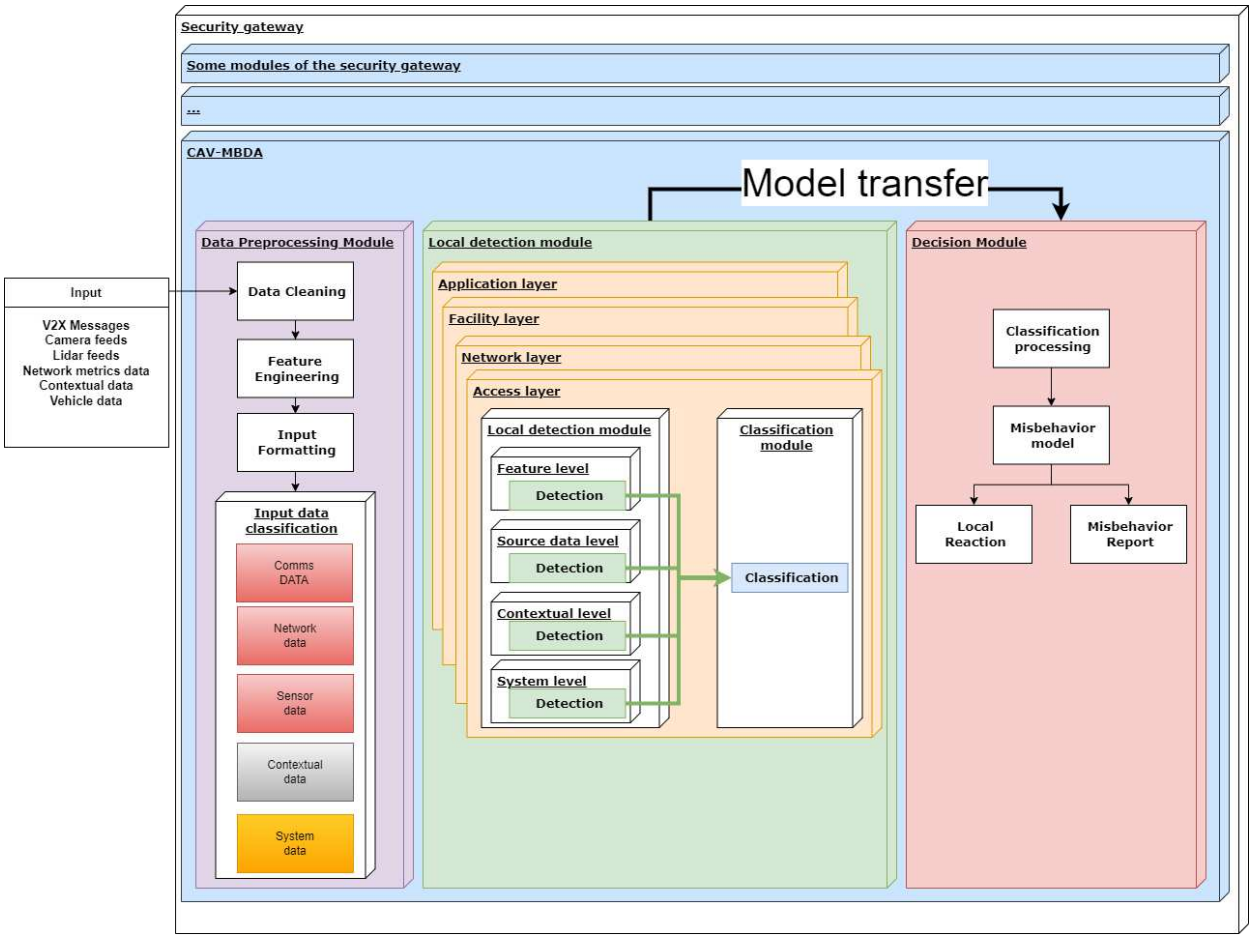


Figure 4.3: CAV-MBDA architecture overview

Figure 4.3 shows different levels of one layer of the detection module: feature level, source data level, contextual level, and system level, which we will discuss in detail in the next couple of sections. In Figure 4.3, we place our CAV-MBDA architecture as part of the security gateway modules. . The data after the acquisition is preprocessed (purple box), then is subject to the detection of misbehaviors (green box) and finally, a decision is made, and a reaction is taken to finalize the process (red box).

In comparison to similar works in [182, 183], our architecture considers the physical architecture of the vehicle as its starting point, then builds on

the standard ETSI stack to define its modules. It is key to point out that by doing so, CAV-MBDA provides simultaneous interoperability with AUTOSAR and in accordance with the ETSI communication stack. In contrast to [182], we consider CAV components data as inputs from the physical architecture of the vehicle rather than considering the whole vehicle as the sole provider. This allows us to enrich our knowledge of the vehicle state i.e., multiple data sources, and also to point out the affected and misbehaving components inside the vehicle. Compared to [183] framework, we consider that misbehaviors can affect multiple data flows (e.g., network metrics, message semantics, sensors noise), and thus, misbehaviors should be detected at different layers. As pointed out, we build our architecture based on the different ETSI stack layers. This allows us to tailor detection mechanisms for each layer separately, resulting in better detection performances.

In the next few sections, we define and explain each step of our architecture.

4.2 Defining the architecture

We define here each of the modules of the CAV-MBDA architecture. But first, we categorize the data into 5 types as follows :

- Communication data: data containing exchanged knowledge between ITS-Stations (e.g., CAM, DENM, CPM).
- Network data: network metrics computed during the communication of the vehicle (e.g., packet loss, transmission rate).
- Sensor data: data received from the vehicle's sensors (e.g., camera, LiDAR).
- Contextual data: knowledge about the environment (e.g., maps, the speed limit on current roads, infrastructures)
- System data: Data exchanged between different levels of C-ITS (e.g., certificates, revocation list)

4.2.1 Data Preprocessing Module

The data processing module is the entry point of the security gateway (Figure 4.4). This module receives multiple inputs (sensed information) such as V2X messages, Camera and LiDAR feeds, and contextual information such as weather, maps, and infrastructures. And it consists of multiple functions :

- Data Cleaning: These inputs are cleaned by removing wrongly structured data, imputing/removing missing data, and synchronizing different input entries.

- Feature engineering: A set of new features are computed on the raw data to help in the process of misbehavior detection.

- Input formatting: The set of features is then put into the correct format to be processed by the misbehavior detection methods at the detection module.

- Input data classification: The features are then tagged depending on their usability at different layers and levels of misbehavior detection.

Once the preprocessing module function ends, tagged data is used in the detection module.

4.2.2 Detection module: specifying architecture layers

The detection module is the main part of the CAV-MBDA architecture. And thus all types of data are processed at this module. It is responsible for checking all data for misbehaviors and anomalies. We rely on the C-ITS communication stack from figure 2.3 to define the layers of the architecture. By definition, each layer has different kinds of data and can be processed independently to achieve specific misbehavior detection. For example, at the access layer, the message signature can be checked and compared to black lists to decide whether to accept the message. At the networking and transport layer, data such as transmission rates can be computed and compared to predefined ones to decide whether the sender is behaving normally. At the facilities layer, standardized messages can be checked for misbehaviors. And finally, at the application layer, C-ITS services data such as MAP and SPAT can be validated using feeds from camera and LIDAR segmentation. These layers all include local detection and classification processes.

For each layer, four levels of detection are defined. Since misbehaviors come in different forms that can either directly affect a feature in the data, for example, a blank set of pixels in a camera image. We call this detection at the feature level. Other misbehaviors affect also the coherency of two or more features for the same data source, for example, a fake speed in a V2X CAM message affects the coherency between speed and distance traveled between two subsequent messages. Depending on the fake value of the speed, the distance would have to be bigger if the fake speed is higher than the actual speed or lower in the other case. We call this detection at the source level.

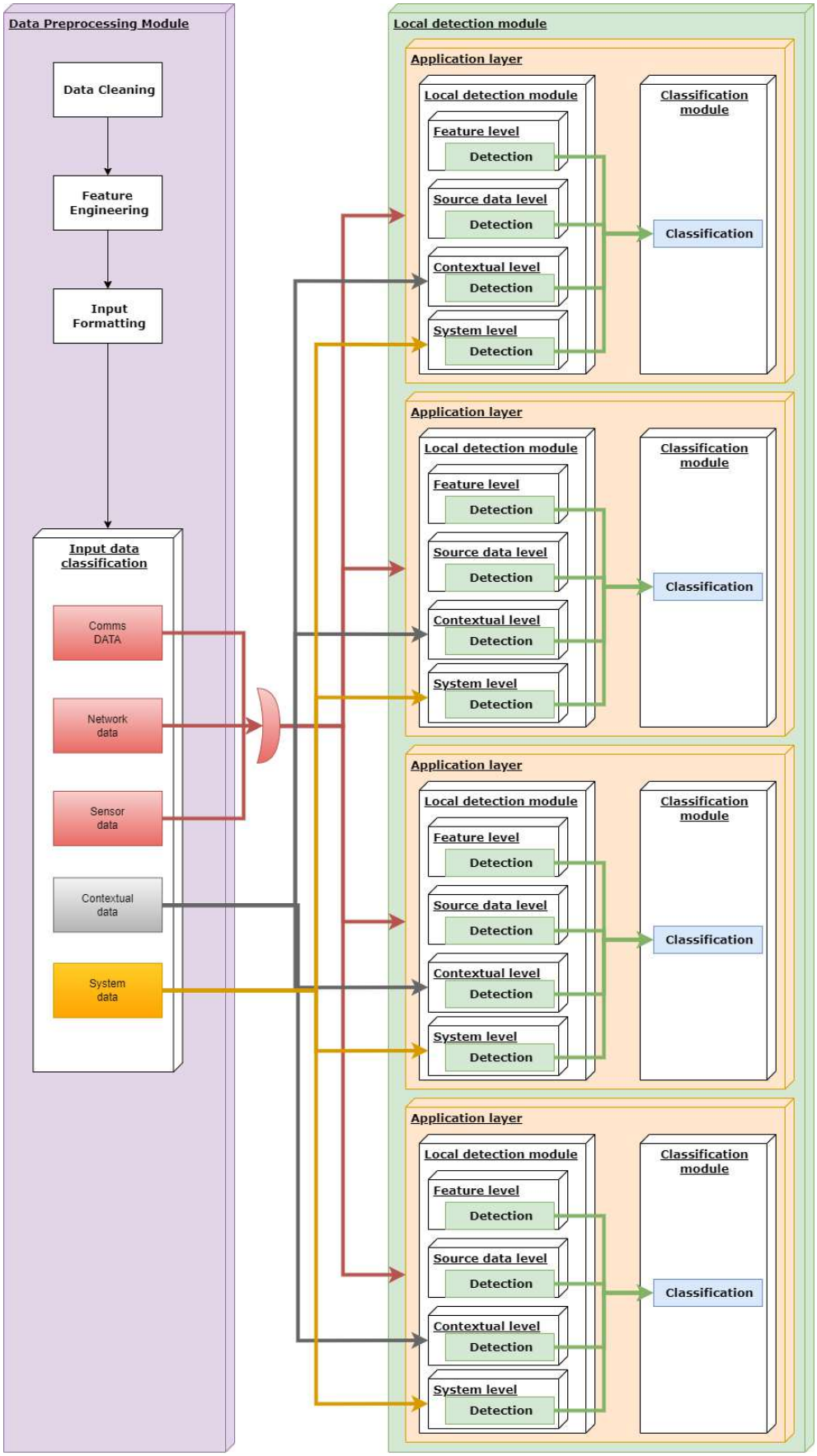


Figure 4.5: Detection module: overview

An increment on that is when the fake data affect the coherency with the context, for example, and out-of-road fake position. We call that detection at the contextual level. And finally, when requiring metadata of the C-ITS system, such as black lists of previously detected misbehaving entities. we call that detection at the system level.

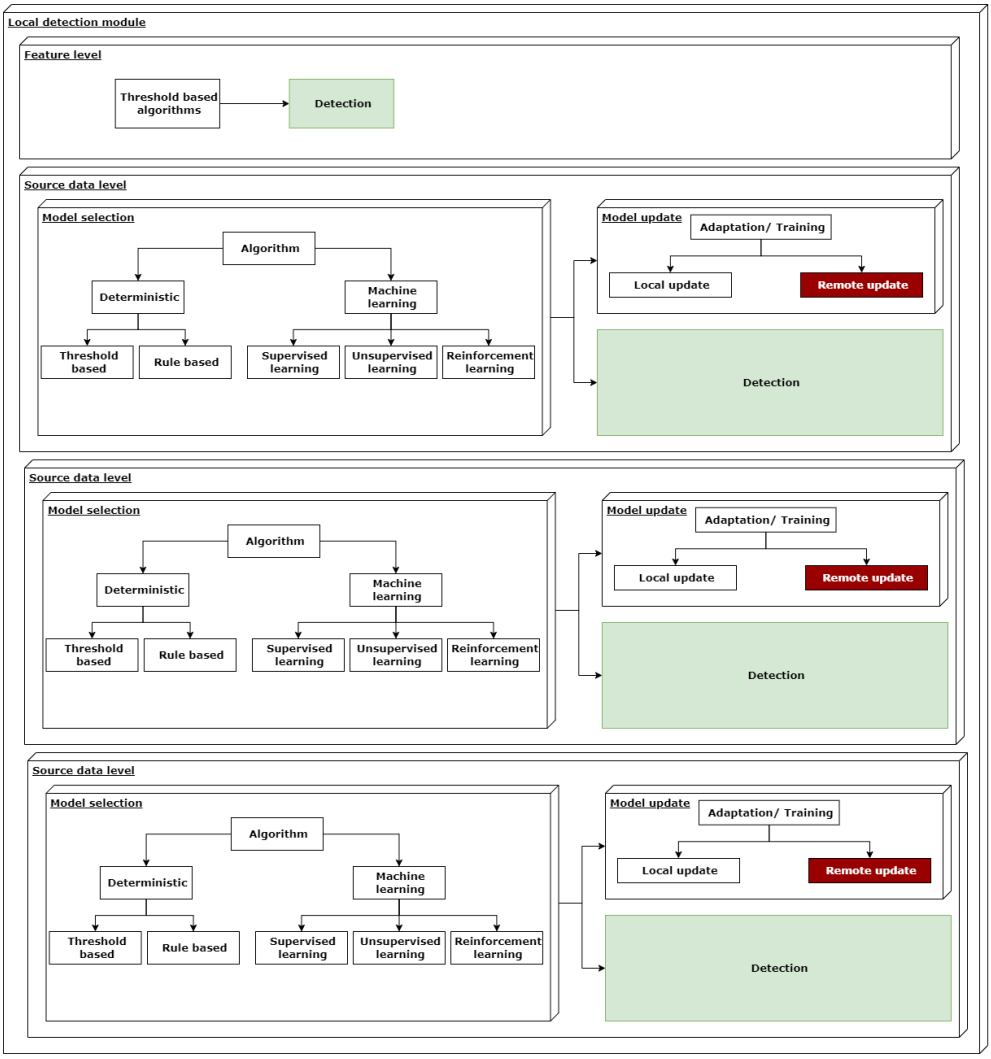


Figure 4.6: Local detection module: detection levels

At each level, a set of detection methods can be deployed. We describe these levels in Figure 4.6 :

- Feature level: this level consists of a set of threshold-based rules applied to each input feature to check its plausibility.

- Source data level: In this level, both deterministic and machine learning based algorithms can be used to generate a local misbehavior detection model. This model can be trained/adapted on the fly using timed feedback from the environment. The model can also be updated by a remote source (typically a misbehavior detection authority).

- Contextual level: This level is conceptually similar to the source data level. The difference resides in the model, as the contextual level has a dedicated input type and is required to detect contextual misbehavior.

- System level: this level is also conceptually similar to previous ones, with the difference that the misbehavior it needs to detect is at the system level.

Whereas some levels are conceptually similar, we find that separating these levels allows for specialized misbehavior detection mechanisms with respect to possible misbehavior. As solutions require various resources to detect certain types of misbehavior, this also helps in optimizing the detection process starting from a basic solution (i.e., feature level) up to a complex solution (i.e., system level) when required. Note that it is not mandatory to implement all levels. Depending on the requirements and constraints of the task, one can just implement the source data level.

After the local detection, the results of the implemented levels are used for a classification process. The process uses the detection along with raw data features to classify the misbehavior. The classification consists of a machine learning model that merges all the results from different levels and layers and provides classes for the detected misbehavior. Similarly, This model can be remotely updated in case of new emerging misbehavior for which there was no classification before. After classifying the misbehavior comes the detection.

4.2.3 Decision module

The final step in the process of misbehavior detection is to take proper reactions to the detected misbehavior and to create a descriptive misbehavior report (Figure 4.7). The classification results are processed in this step and the misbehavior model from table 2.6 defined in chapter 2 is used to describe the misbehavior. A local reaction is then performed, and a misbehavior report is built using the misbehavior model. The detection model is also included in the report as required by the misbehavior authority.

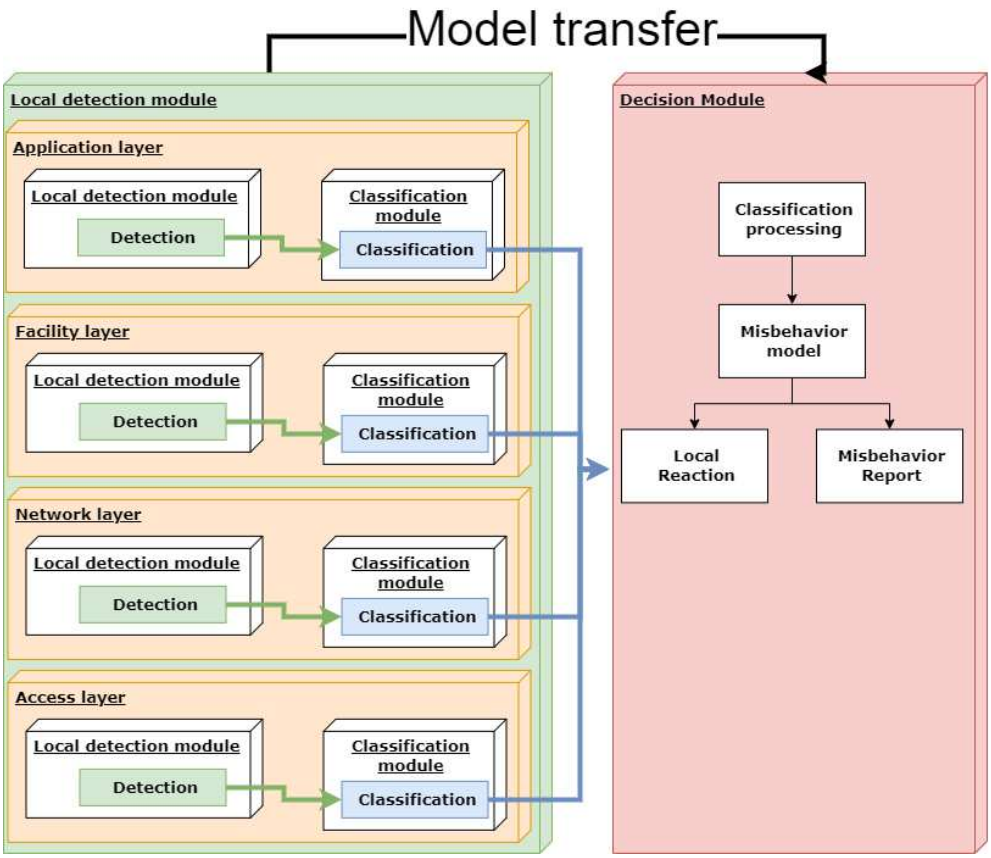


Figure 4.7: Decision module

4.3 Describing pipeline of detection using the architecture

With respect to our work, we implement the CAV-MBDA architecture as part of the autonomous vehicle inside our simulation framework. We define one vehicle to be the detector vehicle, which will monitor input data from the simulation and apply a detection method to react to misbehavior on-the-fly. We defined the architecture to be very generic with respect to data inputs and outputs. Though, Practically, we only partially implement and develop inside the architecture. For our use cases, the detection pipeline is as follows.

- Data is collected from V2X communications, mainly, and from other simulator statistics.
- Data is then processed using a specific preprocessing defined in section 6.3.2 and new features are created to enrich the data.

- Processed data is then fed as input to machine learning models at the source data level and at the feature level.
- The detection model then outputs the results of the detection.
- In some cases, a classification model is used to classify specific attacks from the detection result.
- Since we implement an RL model as our detection model, the detection output is also matched with the action's output into the simulation to control the vehicle and affect the environment, as we will describe in chapter 5.

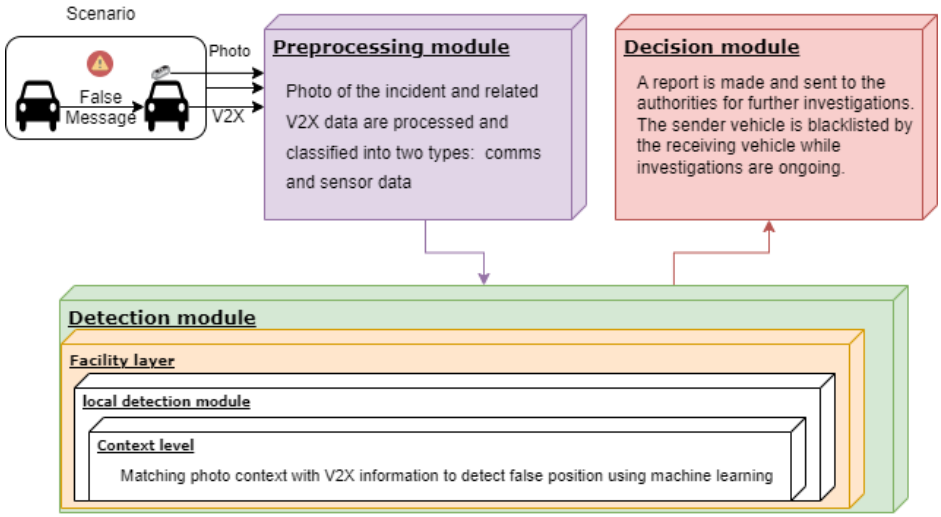


Figure 4.8: CAV-MBDA architecture example

Figure 4.8 show a practical example of how the architecture can be used to detect a false position from two sources of data. First, the scenario is observed, a photo is collected through the camera and a CAM message is received. The data is then processed and classified into comms and sensors data types. The detection module receives this data through its facility layer and uses the contextual level model to detect misbehavior. The contextual level is able to use both comms and sensor data together and finds a mismatch between the two sources and thus it alerts the decision module of this detection. The decision module then issues a report with required proof to the authorities for further investigation of the sender's behavior and also blacklists the sender locally.

In the next few chapters, we will partially build the remaining steps of the architecture. First, we'll introduce our detection module based on RL in chapter 5 that will be implemented in the architecture. Then, we'll define specific preprocessing for vehicular data and extract valuable knowledge from it in chapter 6. We describe in chapter 6 how we simulate data for our task of misbehavior detection. This simulation feeds the architecture we described in this chapter.

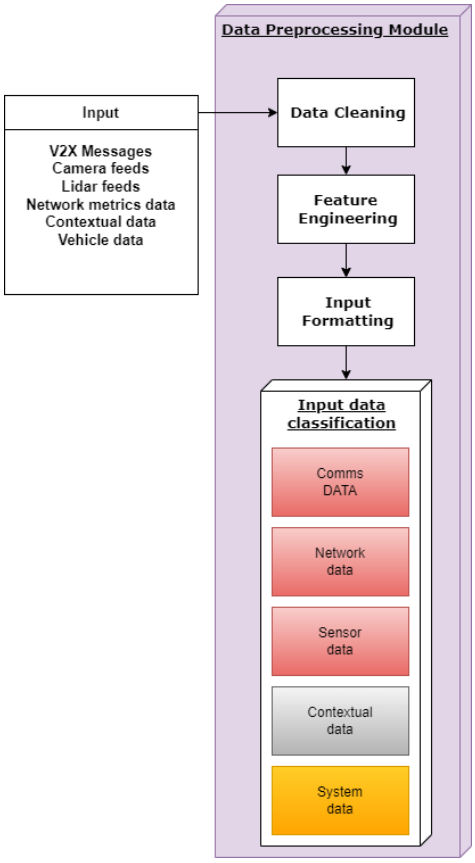


Figure 4.4: Data processing module

Chapter 5

Reinforcement learning based misbehavior detection

In the preceding chapter, we introduced the CAV-MBDA architecture for vehicles. In this chapter, we shift our focus to its use for the vehicle. We examine the current state-of-the-art reinforcement learning for misbehavior detection in C-ITS and acknowledge its limitations. We utilize partially observable Markov decision processes to formalize the detection problem. Moreover, we introduce a novel Q-learning-based model architecture that can effectively detect misbehavior for CAVs.

5.1 C-ITS as a stream graph

5.1.1 Notation

Recall that we categorized the data in C-ITS into 5 types: Communication data, Network data, Sensor data, Contextual data, and System data. At every moment, the vehicle has access to such data, and we denote the collection of all this knowledge about the vehicle as the state $s_{t,v}$ of the vehicle v at time t. The state can be written as $s_{t,v} = \{c_v(t)|c \in C\}$ where C is the set of vehicle components outputting data in the vehicle using a function $c_v(t)$. In the next section, we formalize how this state is used to feed the C-ITS system with data, particularly in the form of standardized messages.

5.1.1.1 Vehicles, States, Messages

In C-ITS, a set of vehicles V interact dynamically with each other. These interactions take the form of a message m (let M be the set of all possible messages with $d' = dim(M)$), where a vehicle $v \in V$ at a time $t \in T$

sends a projection of its state $s_{t,v}$ using a predefined set of features (e.g., speed, position, gas level, camera-captured photos). The state $s_{t,v}$ contains all information available to the vehicle at the time t, including data from sensors, cameras, and other accessible vehicle components. let $S_v = T \times V$ be the set of all possible states for vehicles with $d = dim(S_v)$ and $d' < d$.

$$p : S_v \to M \tag{5.1}$$

The projection p represents in practice any standardized message in V2X (CAM, for example).

5.1.1.2 Communication

Communication between vehicles is constrained to signal strength, and communication range depending on the environment and component of the vehicle. Let be two vehicles $u, v \in V$. The distance between u and v at time t is denoted by $d : T \times V \times V \to \mathbb{R}$ and quantifies the spacial and contextual (e.g., weather, infrastructures) distance between the two vehicles with respect to radio signal strength. A message m sent from u to v at time t is denoted as $m_{t,u,v} \in T \times V \times V \times M$ only if the distance $d(t, u, v) < CR$ where CR is the communication range. With possible delay [184], we can write :

$m_{t,u,v} = p(s_{t-1,u})$ with $\delta t : t - 1 \to t$ denoting all possible delays (transmission delay based on distance and environment).

Let $d_{max} < CR$ be the maximum distance for which messages are not delayed, meaning :

$$m_{t,u,v} = \begin{cases} p(s_{t,u}) & iff\ d(t,u,v) \leq d_{max} \\ p(s_{t-1,u}) & otherwise \end{cases} \tag{5.2}$$

Note that this abstract distance and delay can be represented using the model of radio communication and delay from [184] with probabilities of communication success as a function of the real distance.

5.1.1.3 V2X graph

Using the previous notations, We can represent the V2X communications as a stream graph [185], $G(T, V, S, E)$ where:

- T is the set of time instances
- V is the set of vehicles
- S_v is the set of all vehicles stats at all times $S_v = T \times V$

- E is the set of active links at each time t when two vehicles u, v are able to communicate (i.e., $d(t, u, v) < CR$) $E = T \times V \times V$

Note that V is considered as the set of vehicles for simplicity. The formulation is also valid considering any communicating C-ITS station rather than just vehicles.

We also denote W the matrix containing all exchanges of messages following a projection p as :

$W = T \times V \times V \times M$ where $M = p(S_v)$ and thus $m_{t,u,v} \in W$

5.1.2 Cooperative awareness message (CAM) as a projection of vehicle state

Cooperative Awareness Messages (CAMs) are messages exchanged in the ITS network between ITS-Ss to create and maintain awareness of each other and to support the cooperative performance of vehicles using the road network. A CAM contains the status and attributes information of the originating ITS-S. CAMs may be sent by the originating ITS-S to all ITS-Ss within the direct communication range. [27]

In a formal way, we consider CAMs to be projections on S_v and we denote using equation 5.1 p_{cam} the projection function associated with it. and so :

$m^{cam}_{t,u,v} = p_{cam}(s_{t',u})$ is the CAM message sent from vehicle u to vehicle v

Specific behavior

Since CAM is a broadcast message that can also be delayed, we have a set of possible behaviors and constraints.

- The same message is broadcasted to all vehicles in the communication range of the vehicle u at the time t': $m^{cam}_{t',u,v} = p_{cam}(s_{t',u})$ for all $v \in V |\, d(t', u, v) < CR$.
- With a delay, the message can be received at some other vehicle at a later time. $m^{cam}_{t,u,v} = p_{cam}(s_{t',u}) |\, d(t', u, v) > d_{max}, t' < t$.

In the remainder of this document, we refer to m^{cam} simply as m.

5.2 Defining examples of misbehaviors in C-ITS

The complexity of the C-ITS system with the autonomous vehicle and all the information available at every moment allows for different kinds of misbehav-

iors as seen in section 2.3.1. In this section, we give insight into misbehavior as a function of the state of the vehicle and provide some examples that can be easily simulated.

The state of the vehicle, or precisely, the projection of the state $s_{t,v}$ that is shared with other vehicles, needs to be faithful to the actual state of the vehicle. So what happens when a vehicle decides to deceive others?

We first define an attack function f_a as follows :

$$f_a : S_v \rightarrow M' \tag{5.3}$$

The function f_a is very similar to our projection p from equation 5.1. the only difference is that M' is not faithful to S_v in this case. And by, faithful, we mean that M' contains information that contradicts the states S_v. In this case, the receiving vehicle might build a false perception of its context. Note that, the same effect can happen in the case of a faulty component outputting faulty data, $c'_v(t)$ in which case the state $s_{t,v}$ is poisoned or faulty.

CAM falsification

For the sake of clarity, let's define p_{cam} as follows :

$$p_{cam} : s_v \rightarrow (t, x, y, v_x, v_y, a_x, a_y) \tag{5.4}$$

where t is time and x, y are positional coordinates, v_x, v_y, a_x, a_y are velocity and acceleration along axis x and y respectively.

- A positional falsification attack might define a function f_{pa} as :

  $$f_{pa} : (x', y', p_{cam}(s_v)) \rightarrow (t, x + x', y + y', v_x, v_y, a_x, a_y) \tag{5.5}$$

  where x', y' can change the transmitted position to any value. for example, misbehavior from VeReMi dataset [9] can be expressed in this manner:

  - Constant attack as $f_{pa}(-x + 5560, -x + 5820, p_{cam}(s_v))$
  - Constant offset as $f_{pa}(250, -150, p_{cam}(s_v))$
  - Random as $f_{pa}(\mathbb{U}(0, 6300), \mathbb{U}(0, 6300), p_{cam}(s_v))$
  - Random offset as $f_{pa}(x+\mathbb{U}(-300, 300), y+\mathbb{U}(-300, 300), p_{cam}(s_v))$
  - Eventual stop as $f_{pa}(-x + x_{ev}, -y + y_{ev}, p_{cam}(s_v))$ where $x_{ev} = x, y_{ev} = y$ while the stop probability is not reached, then $x_{ev} = x_s, y_{ev} = y_s$ where x_s, y_s are the saved positions at the eventual stop time.

Where $\mathbb{U}$ is the uniform distribution.

- Equivalently, A speed falsification attack might define a function f_{pa} as:

$$f_{sa} : (v'_x, v'_y, p_{cam}(s_v)) \rightarrow (t, x, y, v_x + v'_x, v_y + v'_y, a_x, a_y) \tag{5.6}$$

One can define various logics of attacks as f_a functions and in chapter6, we simulate C-ITS genuine and misbehaving data using the f_a functions.

In the following section, we will demonstrate how reinforcement learning shares specific characteristics of the environment and we formalize a misbehavior detection agent using RL.

5.3 RL Formulation

As defined in section 2.5, reinforcement learning is *the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment* [186]. The environment can be partially observable. In fact, partial observability is "in some sense" included in the function approximation framework for RL. Nevertheless, a more explicit treatment of partial observability might be needed [6].

In particular, we want to highlight the main characteristics of the vehicular environment and its underlying system and actors and demonstrate how reinforcement learning shares similar characteristics. To this end, we find that :

- Vehicular environment is dynamic and multi-agent. We can feel that by observing and thinking of how many components, actuators, and agents are in the system and how every one of those has its own goals and processes. Signal light phases policies aim to make sure of fluid traffic, drivers want to reduce travel time, and pedestrians want to be safe at all times. All these independently evolving agents are part of the vehicular environment and make it highly dynamic.

- The vehicular environment is interactive. In order to live in the same environment, the various agents have to interact and communicate and abide by several rules. Pedestrians can ask for a red signal to allow them to pass, vehicles use their lights to show they want to turn, and in a C-ITS system, vehicles communicate important information to ensure safety. These interactions affect agents in different ways, which introduce more dynamics in the environment.

- The vehicular environment is partially observable from the agent's point of view. No agent is capable of knowing all other agents and how they interact and affect the environment. A vehicle has no knowledge that a different vehicle down the same road had just made a crucial action, resulting in an accident that will affect the road and all vehicles on it. Though, the whole system, with all agents collectively, is assumed to give full observation of the environment.

With respect to these characteristics of RL and the vehicular environment, we believe that RL can solve various problems in the C-ITS system, including autonomous driving, signal phase planning, and misbehavior detection.

5.3.1 RL for Autonomous vehicles' misbehavior detection

In the context of reinforcement learning, we consider the vehicles to be agents, their actions such as labeling a message as misbehavior and blacklisting other vehicles are the set of actions. The set of messages including the recently received message forms the state, and we formulate a reward function based on how well the agent detects misbehavior and drives through the environment.

We associate a Markov decision process to the vehicle context from chapter 2 as follows:

- The state S represents the stats of all vehicles at time t. $S = \{s_{t,v} | v \in V\}$
- A is the set of actions. $\{a_0$: Remove vehicle v from S, a_1: vehicle dynamics$\}$
- P_a is the transition conditional probability distribution over all stats and actions (system dynamics assuming deterministic policies for all vehicles).
- R_a is the immediate reward due to a as the system transits. r = local time loss.

We assume this MDP exists in the vehicular context, all agents of the context collectively define the next step for this MDP without necessitating historical knowledge. Though, considering one vehicle RL agent, the state S cannot be fully observable. Related works such as [129, 132, 135, 137] work around this MDP as follows.

5.3.1.1 The state representation

In [127, 133, 138], the state is represented using only one projection of a vehicle (e.x., one received message) as raw features or measures computed on that feature. This type of agent is called a "reactive agent" in RL [187] which does not rely on any historical memory. Using such an agent in this context limits the solution. In fact, the partial observability nature of the context and the non-markovian characteristic of building the state with one observation in this sense makes finding an optimal policy using a reactive agent intractable [188]. Due to this limit, the authors tried to include the history of building the state.

In [129, 132, 135, 137] the state is represented by a projection of some vehicles' states as a fixed set of previous observations and/or measurements and actions. This agent is labeled as a "Finite history windows" agent. The use of a window of previous observations to build the state helps in making the state "more markovian" and thus makes searching for an optimal policy easier in a sense. The problem with using a finite history is that we forget important past observations that might be relevant for the current choice of action [187]. For example, in the vehicular context, it is possible to receive knowledge about road hazards much sooner than when the road hazard actually affects the agent. And if the history used is not big enough, critical information is lost. A second problem is that using bigger windows is computationally inefficient and cannot realistically be implemented [6].

One solution to use the historical observations intelligently is using recurrent neural networks such as Elman networks to construct a convenient state for the task at hand. The network will, starting from the first observation, construct a representation of a state using subsequent observations that include relevant information about the history that should be beneficial to the task at hand [187–190]. Thus, to improve on the limits shown in the literature for V2X misbehavior detection, we propose in section 5.4 an RL architecture that uses such a network to achieve misbehavior detection in C-ITS.

5.3.1.2 The actions

The actions used in [127, 132, 135, 137, 138] is whether to label an observation as misbehavior or not (binary classification). Others use task-specific actions, such as outputting the optimal velocity for control of the safest distance from the vehicle ahead [129], selecting a subset of sensors for measurements to avoid faulty ones [131].

Using binary labeling as the only action results in a weak feedback signal

from the environment (assuming no further actions are based on the label). This makes it difficult to construct a functional reward signal and requires strong assumptions. Specifically, these actions have little impact on the next observation, so we need to use actions that have a stronger effect on the environment in order to compute an efficient reward signal. In addition to the RL method proposed in Section 5.4, we propose to use more impactful actions in the simulation to generate better feedback.

5.3.1.3 The reward measurement

The main reward function used in the literature for RL misbehavior detection in C-ITS is a function of "supervised labels" [127,132,135,137,138]. The main issue with such a reward is that it is assumed that this feedback is available right after taking an action, which can be difficult to have. One can think of a misbehavior authority in C-ITS that would be able to provide such feedback, but it would also take several time instances until that feedback is received, and the vehicle will continue to take actions in the system which might be slow. One way to have instant feedback is to compute a difference between theoretically optimal and practical measurements, such as in [129] where the difference between optimal and actual distances is used to compute the reward. Similarly, [133] use the difference between corrected and actual messages to construct feedback. In the same sense, locally within the vehicle observable context, we construct feedback based on observations and optimal behavior in the current situation and use it to learn in our RL model in section 5.4.

5.3.2 Partially observable Markov decision process

As seen in the last section, the environment is highly dynamic, interactive, and partially observable. These characteristics make the Markov decision process defined in section 5.3.1 only partially observable and thus, we associate a partially observable MDP (POMDP) to our problem by augmenting the previous MDP as follows.

Given that every vehicle receives messages (projections of vehicle states) from neighboring vehicles, we can formulate these observations as :

- Ω is the set of all observations and is equals to $P_{cam}(S_v)$ with $v \in \mathbb{V}$
- O is the conditional observation probabilities.

A vehicle only observes camera projections of its surrounding vehicles at a given moment. Which makes up the observable (data and knowledge

available for the vehicle) for CAM messages at time t for vehicle v a set: $P_{cam}(S_{v'})|v' \in \mathbb{V};,;d(t,v,v') \leq CR$. As previously mentioned, the abstract distance function can be represented by the physical model of radio communication and delay from [184], and the same model can be used to represent the conditional observation probabilities O.

In the next section, we present the proposed architecture for RL misbehavior detection that we associate with this POMDP.

5.4 RL for misbehavior detection

Following our reasoning from the last section, we define our RL model and framework as follows.

5.4.1 Data

Using the simulation from chapter 6, we design our RL model locally within one vehicle of the simulation and with respect to the data available. Since we mainly want to detect misbehavior in V2X data, our state representation is heavily constructed using V2X CAM data. Other traffic data is also used to create feedback to train the RL model.

5.4.2 State

Our state is constructed using a function approximation Φ_θ, incrementally built from the history of inputs. Given an Observation O_i and a state S_i :

$$S_i = \begin{cases} O_i & iff \quad i = 0 \quad (Initial\ state) \\ \Phi_\theta(O_i, S_{i-1}) & \quad\quad\ \ i > 0 \end{cases} \tag{5.7}$$

where θ are the trainable parameters for the update function Φ. O_i is constructed using the observable CAM message received from any vehicle v as $O_i = P_{cam}(S_v)$ at step i. In practice, we represent the update function Φ_θ using a neural network, precisely an Elman network is shown in figure 5.1.

The update function Φ constructs the current state incrementally from the previous state and current observation, with the main objective that the constructed state contains information from the history that is most relevant to the task. For that, the function is trained together with the RL model for misbehavior detection.

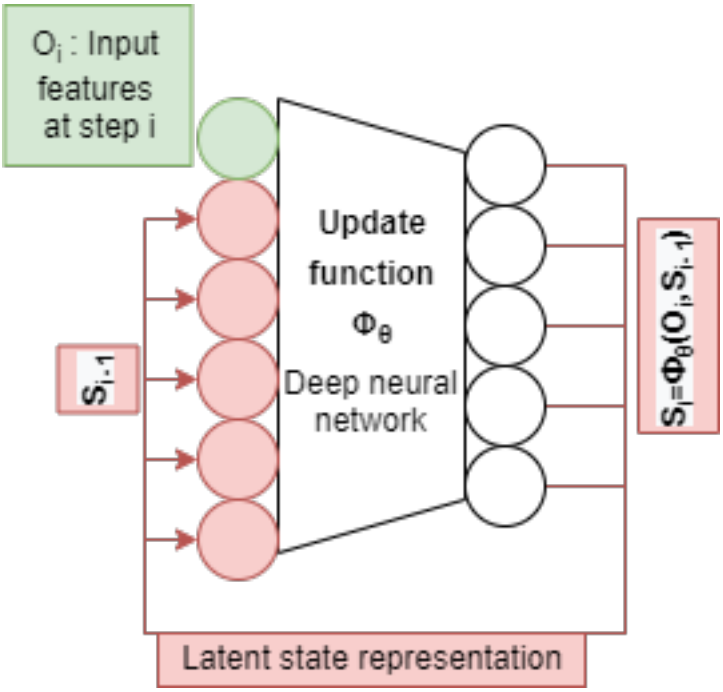


Figure 5.1: The update function Φ_θ

5.4.3 Actions and reward

The agent, which is the vehicle in our simulation, is allowed mainly to blacklist vehicles that our model conceives as misbehaving. The blacklisted vehicles will not be allowed to interact with other vehicles and no V2X messages from them are accepted.

- a0: accept the message and perform driving actions with that perception.
  - outcome: time loss if the sender vehicle is misbehaving.
- a1: blacklist the vehicle ID of the sender.
  - outcome: lower perception and penalty if the sender is not misbehaving.

The timeloss outcome in seconds is computed as the difference between the actual and expected speed at each step. This value is implemented in the sumo simulator as :

$$timeloss \; = \; \alpha(1 - v/v_{max}), \tag{5.8}$$

where v is the vehicle speed, v_{max} is based on the vehicle max speed and the road speed limit, and α is a simulation time to real time factor.

The reward is then formulated as :

$$r = timeloss + \beta, \tag{5.9}$$

where β is the penalty for trying to blacklist a non-misbehaving vehicle.

The reward r characterizes the deviation from optimal behavior locally around the vehicle. The agent tries to maximize this reward using the set of actions allowed for it with the goal of ensuring optimal behavior. The way misbehaviors are defined is that they are assumed to affect negatively the vehicles causing this deviation from optimal behavior. And in that sense, the agent with its set of actions becomes robust against such misbehavior and should be capable of actively detecting and nullifying them, as we'll show in the next chapter. One apparent limit is that the agent acts in a selfish manner. The reward r is built around the behavior of only the agent itself, and maximizing the reward for the agent doesn't usually mean that all other vehicles are benefiting from it. In fact, theoretically, if misbehavior has a positive effect on the agent, then the agent won't act against it.

5.4.4 Deep Q-learning architecture

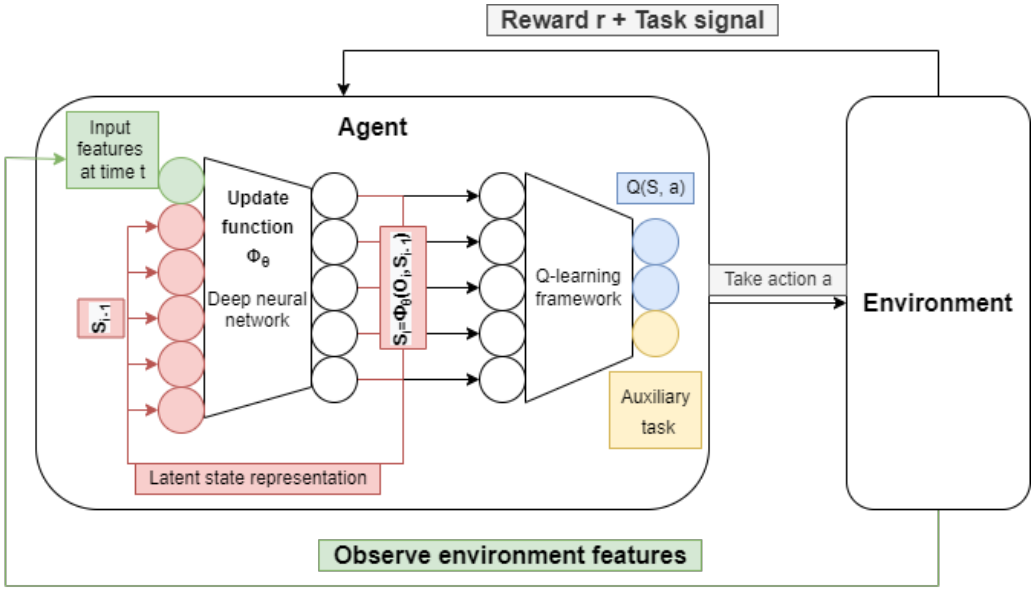


Figure 5.2: The RL architecture: q-learning with update function Φ_θ

Using the update function Φ defined in the previous section, we define the architecture of our RL model by plugin the function Φ into an RL q-learning framework as shown in Figure 5.2. We use the output of the update function as input to a deep q-learning model and also as input to the next step update function to incrementally create the state. Note that we don't have a specific loss for the update function, and that's because we want the agent to learn the best state representation for its tasks. The state is passed through the deep q-learning network and the latter outputs the chosen action (i.e., q-values for each possible action). At the same time, using the same network, we make the agent learn some auxiliary tasks such as average road speed in a supervised manner. That helps the agent to better represent the context and learn from it. The errors on these tasks and in choosing the best action are then backpropagated, first, through the q-learning network to learn the q-function, and then through the update function to learn how to construct the state. In summary, the feedback from the environment is used to learn both a q-learning model and a state representation to enable the best performances.

5.4.5 Real-world limitations

RL solutions are very promising for detecting misbehaviors in CAVs, nevertheless, when considering real-world scenarios, several limitations are encountered.

- **Data Efficiency and Sample Complexity:** RL solutions often require a significant amount of data, which might be challenging to acquire in real-world scenarios due to safety constraints and component costs.
- **Training Time and Resources:** Training RL models can be computationally intensive and time-consuming creating deployment and continuous learning challenges.
- **Real-Time Requirements:** RL solutions must make decisions in real-time to ensure timely responses in CAVs; however, the computational demands might hinder their ability to meet strict real-time constraints
- **Trade-off Between Exploration and Exploitation:** Balancing exploration with the exploitation of learned knowledge is crucial to building an efficient RL solution. However, exploration in a safety-critical environment i.e., the vehicular environment is difficult and costly.

When building an RL solution, we must consider such limitations to ensure the efficiency of the model and environmental safety.

5.5 Discussion and Conclusion

RL is the current art of learning from interactions [6], and observing the C-ITS system as we did in this chapter gives insight that RL should be a go-to formulation for most C-ITS challenges. Hence, the increasing number of research applying RL for C-ITS. Though, one must create an environment that fits RL requirements and assumptions. And currently, in the real world, and especially for security, it is not possible to learn an RL model directly from real interaction because of the lack of deployment, and the real risk of human and physical damage. And that's mainly the fruit of fairly new technology. Thus, the need for other tools such as datasets and simulators. And we explained that using a dataset hinders the learning of an RL model because of little to no feedback from interactions with a dataset because it is static knowledge. Though, given an artificially powerful reward, the model can learn as we saw in different research works [127, 132, 135, 137, 138], but, this learning is closer to a supervised model than to an RL model due to the fact of using the labels solely as rewards and the static nature of a dataset. Throughout our work, we also applied RL to a static dataset, but the best model we got was only slightly worse than a supervised random forest model (Chapter 6). And that's simply because there were very few interactions to learn from and none to change and affect the environment. And so, using simulation was very critical to learning an appropriate RL model. Thus, we made the simulation from chapter 6, and within, we implemented the proposed RL model in this chapter. Our proposal actually learns from interactions, since every action has an immediate impact on the next state of the environment, as opposed to using a static dataset. The model also learns from feedback based on the current state of the environment rather than just a predefined label. With that, our formulation and environment for the model fit the learning from interaction art that is RL.

In this chapter, we highlighted the limits of state-of-the-art RL solutions for misbehavior detection. We proposed our own formalism and solution based on q-learning. In the next few chapters, we implement and evaluate our proposal of an MBD architecture, including the proposed RL model in this chapter, and compare it with other models.

Chapter 6

Implemention, Evaluation and analysis

In the previous few chapters, we described our misbehavior detection architecture and defined our RL detection model. In this chapter, we simulate data through the synchronization of simulators and inject misbehavior. We implement the MBD architecture from chapter 4. We first recall how V2X data in C-ITS is disseminated. Then we analyze a state-of-the-art dataset for misbehavior detection, VeReMi. We identify data leakage in some applications of the ML to such data and propose a specific preprocessing as a solution to the problem. We benchmark supervised machine learning applications on VeReMi. And finally, we implement our RL model in the simulation and present and analyze the results.

6.1 SiMBD: A simulator for misbehavior detection in C-ITS

Now that we have defined the state $s_{t,v}$, the available data types, and some misbehaviors using the set of functions f_a, we recall the available simulators that use and provide such data. Note that, different simulators from section 2.6.2 do not offer all types of data. For example, CARLA offers sensor data and contextual data. SUMO offers network data (with OMNET++) and contextual data. ARTERY (with VANETZA) offers communication data. The remaining simulators all offer some type of data, but not all.

As we aim to offer a framework where different types of data and misbehaviors are all available simultaneously for detection tasks and for building machine learning algorithms in a realistic environment, we had to find a way to realize that. And thus, we looked at the advantages of the three simulators

CARLA, SUMO, and ARTERY, and we tried to combine and synchronize them to provide all types of data simultaneously [19].

We built the *cooperative perception simulation* module that synchronizes Artery [150], CARLA [146], and SUMO [141]. The co-simulation (CARLA and SUMO) allows the synchronization of all vehicles, offering realistic traffic simulations with 3D objects for sensor data. This allows for the definition of specific traffic scenarios with different feeds from cameras, radar, LiDARs, and also context-related information on the map, infrastructures, and the weather. We use this existent co-simulation and increment it with Artery simulation for V2X data following the standards. Finally, we defined attack and detection modules, which led us to the implementation in Figure 6.1. We describe these modules in the following sections.

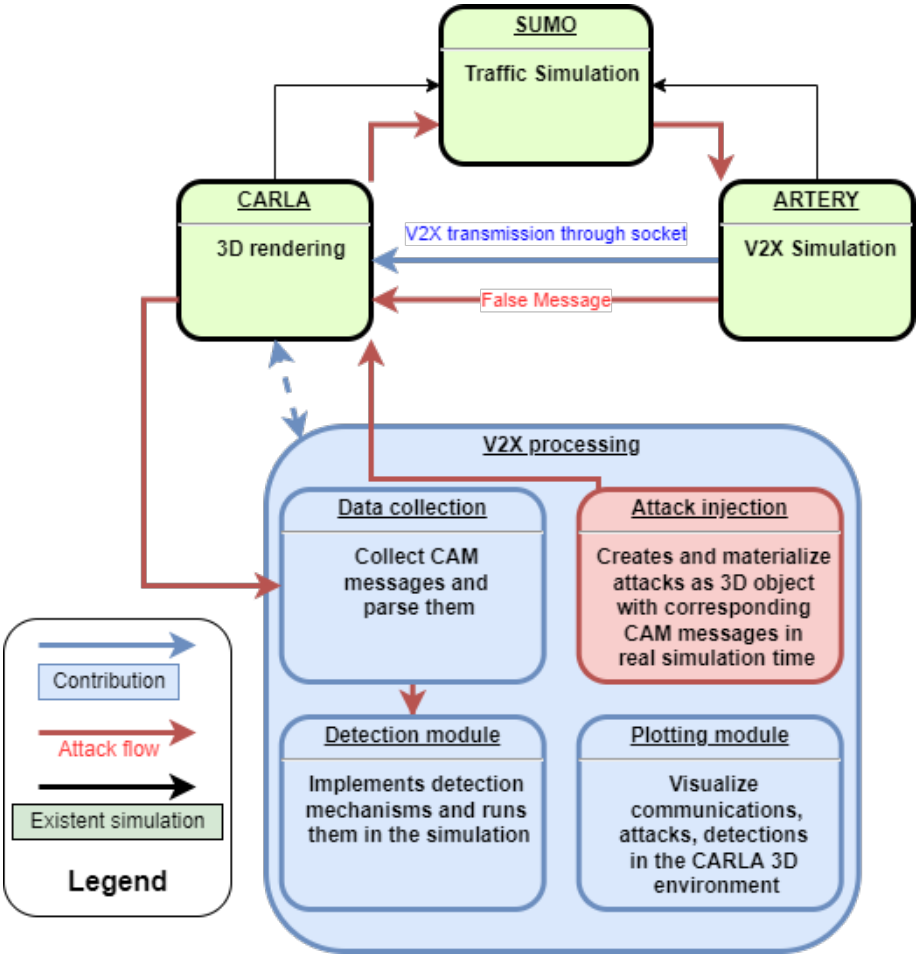


Figure 6.1: SiMBD: Simulator's architecture with main contributions

6.1.1 V2X data collection module

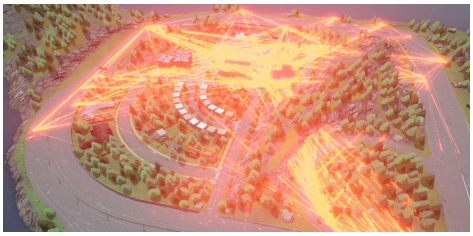
This module collects and parses Cooperative Awareness Messages (CAMs). Whenever a vehicle in the simulation generates a CAM message in ARTERY, this CAM is transmitted through a socket (a client-server low latency two-way communication) to the CARLA's client, which processes it for autonomous driving and misbehavior detection-related use cases. Other V2X messages can be recovered in the CARLA client in the same manner. At each step of the simulation, every vehicle has access to a representation of its context in terms of V2X messages, RGB/LiDAR images, weather, and infrastructures. The data is then logged into a structured dataset that can be used for vehicular tasks such as autonomous driving, supervised misbehavior detections, and data analysis.

6.1.2 V2X attack injection module

This module creates attacks targeting CAM messages. The module materializes, in real simulation time, attacks as 3D objects based on their corresponding malicious CAMs allowing us to observe the attacks' impact on road users. Examples of attacks from section 5.2 have been implemented. The module also implements a simple attack that creates a new CAM for a ghost vehicle ahead of the attacker's vehicle. Different attacks can be implemented by inheriting from the generic attack model. The generic model only requires that a perform_attack function be implemented with whatever attack logic is needed. The impact of the attack will then be observable in data where it is possible, it could be physical using ghost vehicles, or behavioral, showing reactions of other vehicles and changes in traffic statistics.

6.1.3 V2X detection module

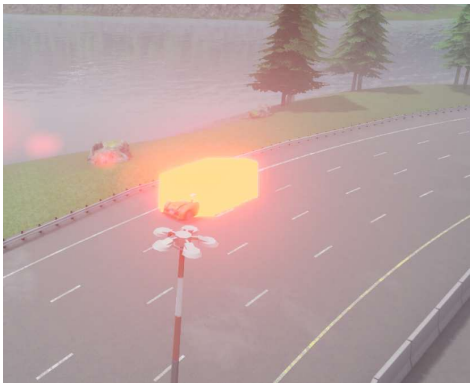
This module implements and runs V2X detectors. At each step of the simulation, the module looks at the available data at that step and performs a detection. The module returns a set of data points that it believes are false or causing misbehavior. Implementing a detection method only requires creating a check function with the wanted logic. For demonstration, our simulator runs a simple speed check [50]. One can also think of using multiple detection algorithms and implementing a voting mechanism or a Dempster–Shafer theory belief fusion operators to take decisions on the detection. We also implement our contribution of a reinforcement learning model described in chapter 5.



(a) V2X communication exchanges

(b) Detection

(c) Ghost vehicle



(d) Attacker vehicle

Figure 6.2: Plotting module visualization examples.

6.1.4 V2X plotting module

This module provides visualization for V2X communications, attacks, and detectors. We show in Figure 6.2 their representations. In particular, communications between vehicles are shown using a red artificial link mapping receiver and sender vehicles (Figure 6.2a). The attacker vehicles are plotted with a red box around them (Figure 6.2d), and a purple box shows the materialization of their attack if possible (Figure 6.2c). When detected a blue box is plotted on top of the purple box to show that the attack is successfully detected (Figure 6.2b). The plotting module also serves as a debugging tool to further understand what are the methods detecting, and how misbehaviors impact the whole C-ITS system.

6.2 Dataset analysis: V2X broadcast

Communicating vehicles in C-ITS exchange data in different manners. One propriety of exchanging CAM messages is that the same message is sent to all neighboring vehicles. We recall from section 5.1.2 that we defined the messages sent from the vehicle u at the time t' as $m^{cam}_{t,u,v} = p_{cam}(s_{t',u})$, $t' \leq t$

for each neighboring vehicle v. These messages are collected at each vehicle and augmented with the vehicle information in the dataset VeReMi.

6.2.1 VEREMI dataset

In order to create the training dataset for the machine learning model, the broadcast data is collected from different vehicles. Figure 6.3 depicts an example where each vehicle broadcasts CAM messages to the surrounding vehicles, then receives, processes, and logs these CAM messages in local data storage. Afterward, at the global level, servers will be able to store and process local logs received from vehicles. In particular, Figure 6.3 shows a

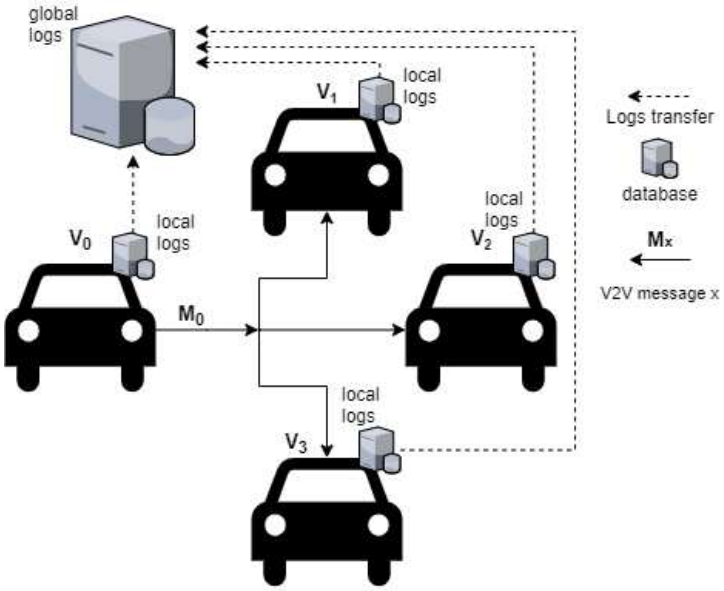


Figure 6.3: Broadcast scenario

scenario where a vehicle v_0 broadcasts a message $m_0 = p_{cam}(s_{v0})$ to its neighbors (v_1, v_2, v_3). These vehicles receive this message and store it. Afterward, the messages are collected in global storage which contains 3 different copies of the original message at different timestamps related to the reception times.

Note that VeReMi also stores additional information on receiving vehicles for each received message. Let $\mathbb{D}$ be the VeReMi dataset. Elements of $x \in \mathbb{D}$ are defined as a couple of projections of receiving and sending vehicles' states. For two communicating vehicles u, v and a cam message sent from u to v and received at time t, $m^{cam}_{t,u,v} = p_{cam}(s_{t',u})$, $t' \leq t$, the corresponding element in $\mathbb{D}$ is $x_{t,u,v} = (p_{cam}(s_{t',u}), p_{VeReMi}(s_{t,v}))$ where p_{VeReMi} is the additional data VeReMi add to received CAM messages from receiving vehicle state (e.x.,

receiver position and speed). The additional information helps in extracting useful features for misbehavior detection.

6.2.2 Practical VeReMi and simulation parameters

Parameter	Value
Mobility	SUMO LuST (DUA static)
Simulation start	(3,5,7)h
Simulation duration	100s
Attacker probability	(0.1, 0.2, 0.3)
Simulation Area	2300,5400-6300,6300
Signal interference model	Two-Ray Interference
Obstacle Shadowing	Simple
Fading	Jakes
Shadowing	Log-Normal
MAC implementation	802.11p
Thermal Noise	-110dbm
Transmit Power	20 mW
Bit rate	6 Mbps
Sensitivity	-89dBm
Antenna Model	Monopole on roof
Beaconing Rate	1Hz

Table 6.1: Veins simulation parameters for VeReMi [9]

We use the last version of VeReMi simulated dataset [93] which presents the same characteristics as a real dataset. This dataset has proven reliable and is used by several researchers [59,60,94,95,142]. This dataset is generated based on 225 simulations with five different attacks presented in the next section. Simulation parameters are summarized in table 6.1. Each simulation uses three different attacker densities. Each parameter set is repeated 5 times with different random seeds. Each simulation was made in the area of Luxembourg and lasts for 100 seconds, with a communication bit rate of 6 Mbps. Our work uses the sub-dataset with the following parameters: attack probability of 0.3, simulations start at $7am$ which corresponds to the highest density.

6.2.3 Features

As expected, VeReMi contains both features from CAM messages and the state of the receiving vehicle. Actually, only a fraction of CAM features

Features	Description	Symbol
Type	identifier for message type	ID
Reception Time	time BSM was received by the receiver	Rt
Receiver ID	Id of the receiving vehicle	RID
Receiver X position	receiving vehicle x coordinate	RXP
Receiver Y position	receiving vehicle y coordinate	RYP
Receiver Z position	receiving vehicle z coordinate	RZP
Transmission Time	time BSM was emitted by the emitter	Tt
Transmitter ID	Id of the transmitting vehicle	TID
BSM ID	Id of the message	MID
Transmitter X position	transmitting vehicle x coordinate	TXP
Transmitter Y position	transmitting vehicle y coordinate	TZP
Transmitter Z position	transmitting vehicle z coordinate	TZP
Transmitter X velocity	transmitting vehicle x velocity	TXV
Transmitter Y velocity	transmitting vehicle y velocity	TYV
Transmitter Z velocity	transmitting vehicle z velocity	TZV
RSSI	received Signal Strength Indicator	RSSI
Label ID	(0=Normal Behavior)	L

Table 6.2: VeReMi dataset for connected and automated vehicles

are available in the VeReMi dataset. In table 6.2, the available features are presented.

We derive a statistical analysis of these features in the next few sections.

6.2.4 Attacks

The VeReMi dataset includes five types of attacks. Each attack has a label and a set of generation parameters. These attacks tamper with the location of the transmitting vehicle in different ways. First, an attacker can transmit a fixed location (Constant). Second, an attacker could transmit a fixed offset of its real location (Constant Offset). An attacker sends a uniformly random position (Random). An attacker sends a random location in an area around the vehicle (Random offset). Lastly, an attacker behaves normally and then transmits the same location repeatedly (Eventual Stop).

Label ID	Description	Parameters
1: Constant	Fixed location	x = 5560, y = 5820
2: Constant Offset	Fixed offset location	Δx = 250, Δy = -150
4: Random	Random location	uniformly random in playground
8: Random Offset	Randomly offset location	Δx, Δy are uniformly random from [-300,300]
16: Eventual Stop	Attacker behaves normally for some time and then attacks by transmitting the same position repeatedly	Stop probability increases by 0.025 each position update

Table 6.3: Attack Definition

6.2.5 Descriptive statistical analysis

We compute several statistics and analyze each feature of our dataset[1]. Table 6.4 summarizes the features' statistics.

		mean	std	min	Q1	Median	Q3	max
Receiver	Time	21798,3	105,2	21600,0	21704,3	21817,4	21885,8	21959,9
	X position	4294,37	1018,19	2325,41	3597,82	3673,08	5023,85	6324,99
	Y position	5510,63	258,74	5180,04	5255,00	5470,59	5738,24	6079,99
Transmiter	Time	21798,3	105,2	21600	21704,3	21817,4	21885,8	21959,9
	X position	5020,44	3288,29	0,41	3602,46	3858,48	5560	27278,9
	Y position	5939,26	2325,82	7,79	5254,50	5538,36	5819,28	22998,1
	X velocity	0,04	8,63	-40,38	-3,95	0	3,83	41,12
	Y velocity	1,51	20,67	-44,25	-4,36	0	11,93	48,90
	RSSI	2,5E-07	2,9E-06	1,2E-09	3,3E-09	8,4E-09	3,1E-08	2,3E-03

Table 6.4: Summary statistics

From Table 6.4, we observe that the standard deviation of the receiver and the transmitter position distributions differ largely (10 times bigger in Y position). However, both features are emitted from vehicles interacting in the same context, i.e, vehicles are located in the same geographic area and hence they should have similar coordinates. Thus, we assume the presence of some anomalies in either or both position features.

[1]Details about the dataset can be found here: https://anonymous.4open.science/r/5de28865-7f74-4360-b3fa-daa68c97bd83/

We plot the histogram of some features in Figure 6.4 to study their distributions. Our observations are as follows:

- The distributions of the receiver X and Y positions compared to the transmitter ones are very different. This confirms our initial intuition. We can see that transmitter X and Y position values are spread in a wider set of values $[0, 28000]$ while most of these values are in the same interval as the receivers ones $[2000, 7000]$ (Figure 6.4). We can assume that only a portion of values is anomalous.

- We observe that the transmitter X and Y velocity distributions are symmetric w.r.t 0 for both sets of values, which we assume correctly since most roads have two ways. We also observe a high number of null velocities, which might be the result of the traffic lights or stop signs.

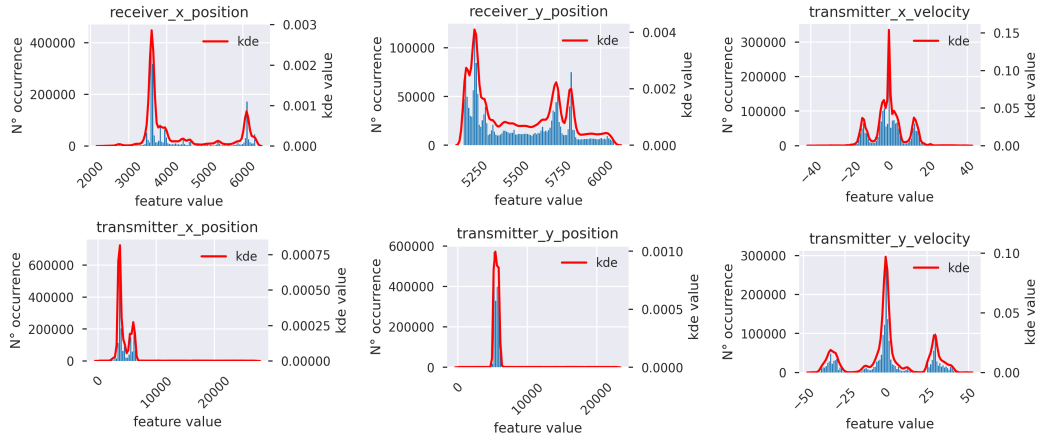
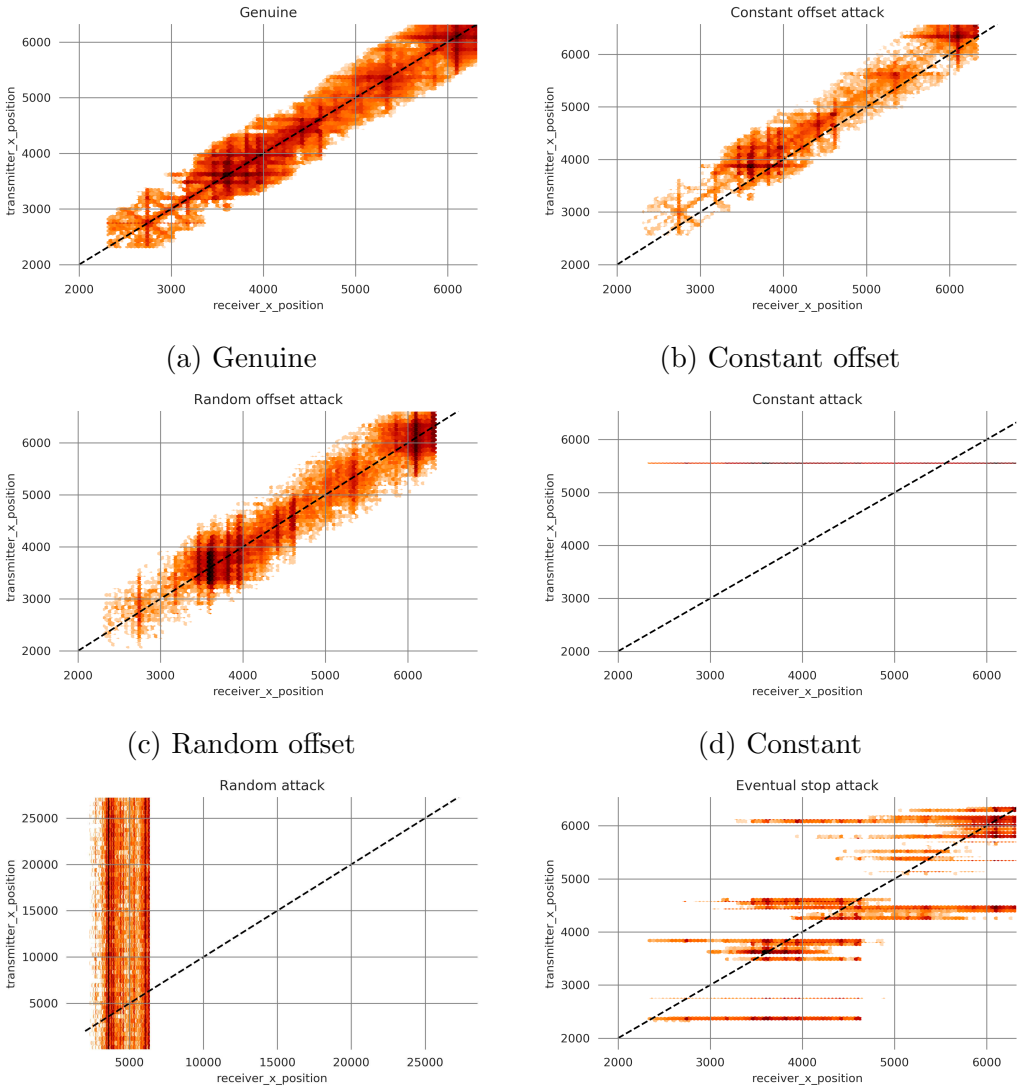


Figure 6.4: VeReMi's features distribution analysis

Now that we established that there are anomalies in the transmitted positions, we plot transmitter and receiver position values against each other for each attack and for genuine data. In Figure 6.5, we show a density plot of these values and we make the following observations:

- As expected, plotting position values for genuine data form a linear scatter plot following the diagonal line $y = x$. Note that the width of this line reflects the transmission range of the vehicles.

- For the attack "constant offset", the figure shows a translated version of the genuine one that is caused by the constant change applied by the attacker.



(a) Genuine

(b) Constant offset

(c) Random offset

(d) Constant

(e) Random

(f) Eventual stop

Figure 6.5: Comparison of X position of receiver and transmitter values

- For the attack "random offset", we observe an almost identical plot as we did for genuine data. The difference lies in the left and right extremities of the observed data. We identify in these extremities a sparse structure formed along the maximum transmission range. This is due to the fact that some attacker vehicles positioned at the edge of the transmission area of a benign can transmit a position that is outside that area.

- For the attacks “constant”, “random” and “eventual stop”, the difference in the data is clearly shown. Only a small portion of the transmitter values intersects with those of the receiver.

These computed statistics do not capture the broadcast aspect of the data. Hence, we compute the number of copies of each unique message, i.e. the same message being received by multiple receivers. In the following sections, we refer to the original CAM message as the unique message. Figure 6.6 shows a histogram of the number of copies created from each unique message and the density estimation per number of copies in the data. We observe that the area under the curve for messages with a number of copies higher than 10 dominates the density estimation, meaning that most of the dataset contains redundant messages (more than 50%). Such characteristics of the data must be handled when modeling a machine-learning solution.

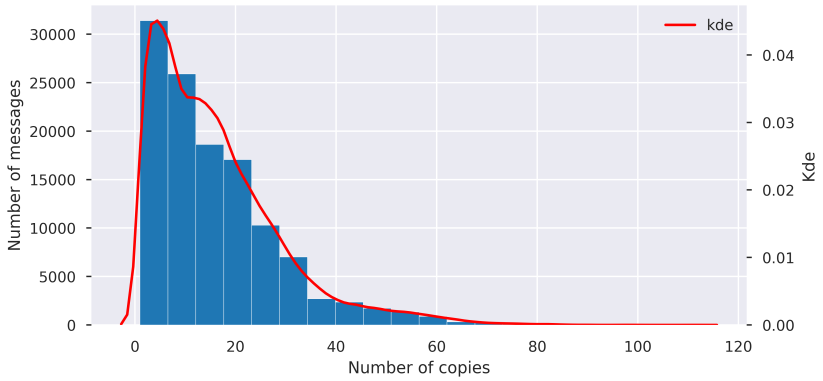


Figure 6.6: Number of CAM copies for each unique message

From the aforementioned analysis, we conclude that the transmitted position values include outliers in an anomalous way compared to receivers' positions and are proof of tampering. The data allowed us to assume that the transmitted time is coherent. Thus, we use the positions to compute the transmitter speed rather than relying on the transmitted velocities. Comparing the values gives an interesting metric to use for detection.

Moreover, the dataset needs preprocessing such as removing constant and id features detected in the analysis because these features do not correlate with the target task and only induce additional computation costs. A last point is that Figure 6.6 presents a high redundancy of data that needs to be processed accordingly (see Section 6.3.2). In the next section, we create new features to improve misbehavior detection.

6.2.6 Feature Engineering

Based on our statistical analysis, we rely on the following plausibility checks output as dataset features. Note that ($T_{X,Y}$, $T_{t,X,Y}$,$T_{t,v}$) represents the position vector, the position at time t and the speed at time t for the transmitter vehicle (receiver vehicle respectively).

Acceptance Range Threshold (ART)

This check verifies if the distance between the CAM emitter and receiver is above the maximum theoretical communication range.

$$ART(M) = \begin{cases} 1 & if\ d(T_{X,Y}, R_{X,Y}) > \Delta_r \\ 0 & otherwise \end{cases} \tag{6.1}$$

where $d(x, y)$ is the Euclidean distance and Δ_r denotes the theoretical value of the communication range.

Sudden Appearance Warning (SAW)

This check verifies if a vehicle suddenly appeared within a certain range. In normal traffic conditions, it can be assumed that new vehicles first appear at the boundary of the communication range. Thus, if a vehicle appears at a distance below the communication range without prior message emission then the SAW value equals to 1.

$$SAW(M_0) = \begin{cases} 1 & if\ d(T_{X,Y}, R_{X,Y}) < \rho \\ & \cap\, M_0\ is\ the\ first\ message \\ 0 & otherwise \end{cases} \tag{6.2}$$

where ρ is the threshold value for the first appearance.

Simple Speed Check (SSC)

This check verifies if the difference between the transmitter speed and the estimated speed from the position and time differences Δ_t between two consecutive messages is less than a threshold value Δ_v.

$$SSC(M_t) = \begin{cases} 1 & if\ \frac{d(T_{t,X,Y}, T_{t-1,X,Y})}{\Delta_t} - T_{t,v} > \Delta_v \\ 0 & otherwise \end{cases} \tag{6.3}$$

where Δ_v is the threshold for the speed difference. M_t is the message at the time t.

We notice that all the extracted features are related to position, speed, and displacement. Indeed, based on our statistical analysis, the position represents the main anomalous feature. In the next section, we discuss how to apply ML to this dataset and propose specific preprocessing to ensure its good application.

6.3 Applying machine learning on VeReMi

To apply ML to VeReMi, or specifically supervised ML to the dataset $\mathbb{D}$, we need first to split the data into training and validation sets. Note that the dataset $\mathbb{D}$ is equivalent to the matrix W defined in section 5.1.1.3 as we can consider it an output of the VANET temporal graph defined in the same section. We'll use W to refer to the collected data to express that the results that will be shown here are valid for any CAM dataset.

Let be an index set $I = \{(t_0, u_0, v_0)...\}$ on W where $m_{t_0,u_0,v_0} = W[t_0, u_0, v_0]$. We define two subsets on I, namely, *train*, *validation* such us :

- $train \cap validation = \emptyset$
- $train \cup validation = I$

In ML, we use the train set to learn a model for our given task, then we use the validation set to evaluate the model and validate the performance and tune the model. And finally, an unseen test set is used to test the final model and analyze its performance. The way these sets are constructed is, from some applications, critical for obtaining valid models. We show in the next section how randomly choosing these sets induces data leakage and invalidates the model for the CAM dataset.

6.3.1 Random splitting and data leakage

Let's suppose that the two subsets *train* and *validation* are constructed randomly from the set I. In this case, it's possible to have $(t_i, u_j, v_k) \in train$ and $(t_{i'}, u_j, v_{k'}) \in validation$ and $t_{i'} < t_i$. And due to the broadcast nature of the data and the fact that it is possible to have delayed messages, it is possible to have :

$W[t_i, u_j, v_k] = W[t_{i'}, u_j, v_{k'}] = p_{cam}(s_{t_{i'},u_j})$

This means that the same data (i.e., $p_{cam}(s_{t_{i'},u_j})$) is present both in the *train* set (i.e., $W[t_i, u_j, v_k]$) and in the *validation* set (i.e., $W[t_{i'}, u_j, v_{k'}]$).

Hence, we have a form of data leakage between train and validation sets using random splitting. We learn and evaluate the model on the same data, which makes the performance of the validated model look better and hides overfitting.

6.3.2 Specific C-ITS preprocessing

The reason that random splitting *train* and *validation* sets cause data leakage is two folds. broadcast related; as duplicated messages are received by different vehicles. Temporal, as the messages sent from any vehicle, can be considered a time series, and breaking this temporal constraint makes the wrong assumption that past and future messages are independent. Note that, in the case of the matrix W or the data set VeReMi, we look at the receiving end of the vehicle, and thus we only receive parts of other vehicles' time series depending on the spatial constraints of receiving messages. Furthermore, we look at all receiving ends of all vehicles, which makes the data a mixture of redundant parts of vehicles' time series (see figure 6.6).

We can see that in practice, using the VeReMi dataset, Figure 6.7 shows the issues that arise from using a classical random split (hold-out) on the full dataset with a 10% validation proportion. Results show that copies of a unique message are present multiple times in both validation and train sets (refer to black and white dots in Figure 6.7) where each colored column represents a unique message and a black dot means that a copy of the message is present in the train set (a white dot for the validation set). Note that the expected result should have only white (resp. black) colored dots for each unique message. This means that the observations are repeated in both sets, which leads to a biased performance.

This brings up two questions, why don't we just remove redundant data? and if we don't remove it, how to avoid data leakage? Actually, the redundant part of the data is only related to sender vehicles, so only the $p_{cam}(s_{t',u})$ part of the data. And thus the additional data added by VeReMi for example $p_{VeReMi}(s_{t,v})$ is unique. Meaning that for all copies of a message $p_{cam}(s_{t',u})$ received by multiple vehicles, the $p_{VeReMi}(s_{t,v})$ part of the data is different. This additional unique information actually has a huge impact on the performances of the MBD models, as we'll show in the next few sections. Particularly, it is actually used to extract features as defined in section 6.2.6. In summary, what we want to say is, looking at the same piece of information from different perspectives gives valuable information about it that cannot be achieved from just one perspective. So, the answer is no, we cannot remove redundant data.

So, how can we split data to apply ML methods and avoid data leakage

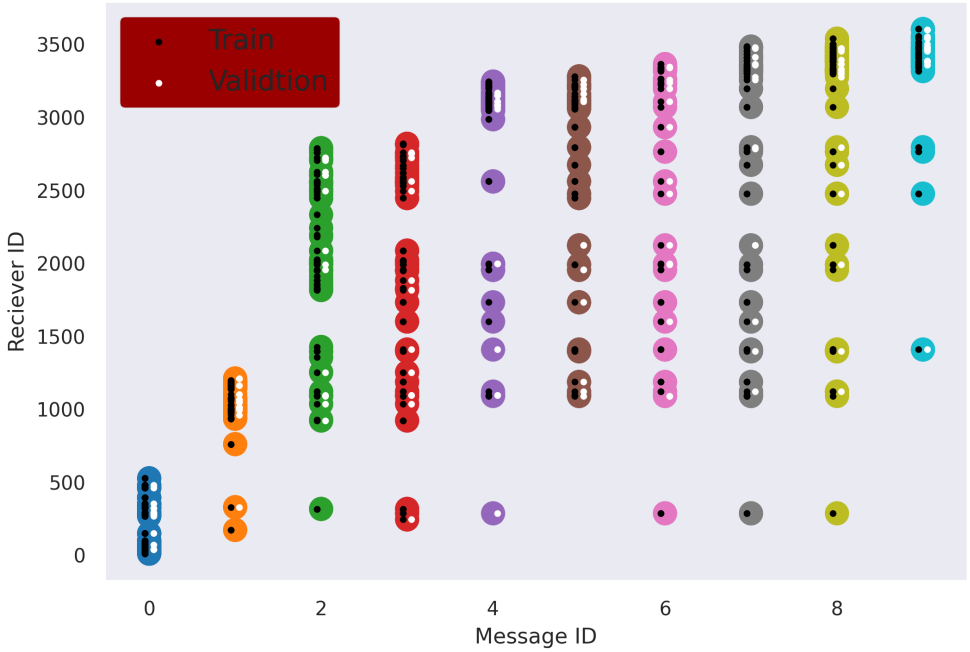


Figure 6.7: Sample of messages with a random split..

in this case? We already said that in the vehicles' time series, breaking the temporal constraint makes the wrong assumption that past and future messages are independent. So that's one part of the answer that we should extend to the mixture of our partially redundant time series. The second part would be to ensure that the spatial constraints also are not broken by the split. It turns out it's not possible always have both. We describe in the next section, our algorithm for splitting the data and avoiding data leakage.

6.3.2.1 Spatio-temporal cross-validation

To avoid data leakage from random splits, we include temporal and spatial constraints on building both *train* and *validation* sets [18].

- Temporal constraint : $\forall(t_i, u_j, v_k) \in train, \forall(t_{i'}, u_{j'}, v_{k'}) \in validation, t_i < t_{i'}$. let $\tau \in T$ such as $t_i \leq \tau < t_{i'}$ be the chosen temporal split moment that verifies our temporal constraint.
- Spatial constraint: we apply a correction to the validation indexes as follows :
  - $\forall(t_{i'}, u_{j'}, v_{k'}) \in validation$ if $t_{i'} = \tau + 1$ and $d(\tau, u_{j'}, v_{k'}) > d_{max}$ and $d(t_{i'}, u_{j'}, v_{k'}) > d_{max}$ then we move $(t_{i'}, u_{j'}, v_{k'})$ to *train* set.

– Note that this correction invalidates our temporal constraint on some elements.

Constructing the train and validation sets in this way ensures that the previously discovered data leakage issue with random splitting is no longer present. Indeed :

- $\forall(t_i, u_j, v_k) \in train$, $W[t_i, u_j, v_k] = p_{cam}(s_{t,u_j}) : t <= \tau$
- $\forall(t_{i'}, u_{j'}, v_{k'}) \in validation$, $W[t_{i'}, u_{j'}, v_{k'}] = p_{cam}(s_{t',u_{j'}}) : t' > \tau$
- thus, there cannot be any $t \in T, u_j \in V, (t_i, u_j, v_k) \in train, (t_{i'}, u_j, v_{k'}) \in validation$ such that $W[t_i, u_j, v_k] = W[t_{i'}, u_j, v_{k'}] = p_{cam}(s_{t,u_j})$

So we are able to avoid data leakage by ensuring spatial constraints are always intact and slightly breaking the temporal constraint for our mixture of the partially redundant time series (note that the temporal constraint of vehicles' time series is actually preserved at the vehicle level).

In practice, the temporal split helps to reduce the data leakage, yet, it does not solve the problem entirely, i.e., a portion of messages still ends up in both train and validation sets because of possible delays.

To visualize this our solution, let us consider a message M sent from a vehicle to other 20 vehicles within the range of 500 meters. All messages M_0 till M_{19} have different receiving times due to the surrounding context and the recording time (spatial constraints). T_{early}, T_{late} represent respectively the earliest and latest recorded messages. Any time split T_{split} between T_{early} and T_{late} puts copies of the message M in both train and validation sets, violating the assumption of unseen data.

Figure 6.8 presents a scenario where this issue is observable. The x-axis represents the time, the y-axis represents the vehicle's position and the distance between messages is the one defined in section 5.1.1.2. The scenario consists of four different messages represented by black dots, from which two copies are created each time (blue boxes). To perform a temporal split, we must choose a discrete time to split the data. If T_1 is chosen for the temporal split, then the broadcast copies of the message M_1 have reception times earlier and later than T_1 as shown in Figure 6.8, which means that the temporal split on T_1 would put the message M_1 in both train and validation sets, which leads to data leakage.

The proposed solution outputs a fairly simple algorithm 1 that keeps the assumption of an unseen validation/test set valid.

The algorithm first sorts the data w.r.t time to ensure the temporal flow of the data (Algorithm 1, line 2). The algorithm then splits the data with

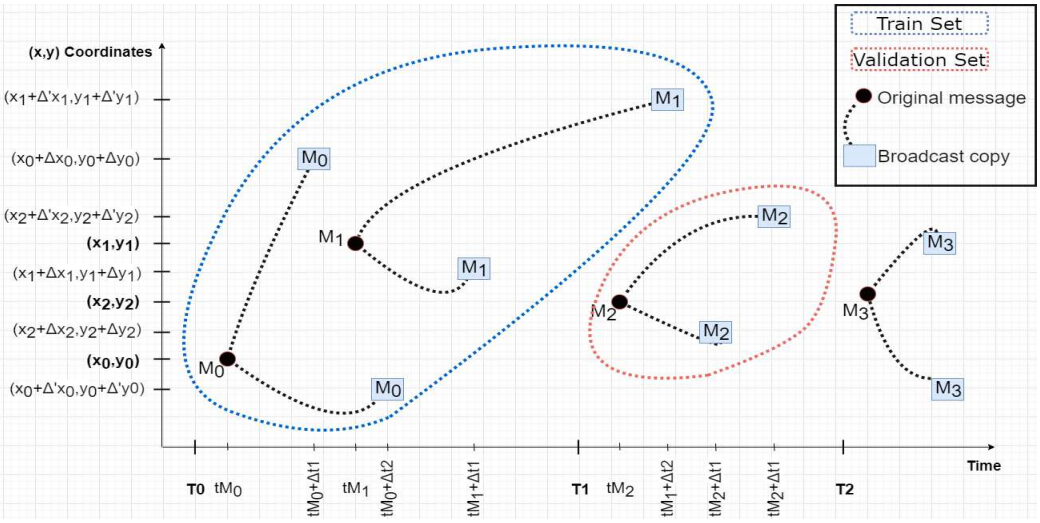


Figure 6.8: Spatial and temporal data split

Algorithm 1 Proposed split approach

Require: $M\{msg_{id}, time, sender_{id}, ...\}, train\_prop$

1: **procedure**
2: **sort**$(M, time)$
3: $T_{split} = M.time[train\_prop * sizeof(M)]$
4: $T_{set} = M[M.time \leq T_{split}]$
5: $V_{set} = M[M.time > T_{split}]$
6: $intersect_{id} =$ **intersect**$(T_{set}.msg_{id}, V_{set}.msg_{id})$
7: $intersect = V_{set}$.**pop**$(V_{set}.msg_{id} \in intersect_{id})$
8: T_{set}.**add**$(intersect)$
9: **return** T_{set}, V_{set}
10: **end procedure**

respect to a given *time_split* (Algorithm 1, lines 3-5). Then, we check for spatial dependencies between the train and validation sets (Algorithm 1, line 6) by computing the possible intersections between copies of the same message in the train and validation sets. The Algorithm then ensures the respect of these spatial dependencies (Algorithm 1, lines 7-8) and outputs the resulting train and validation sets.

6.4 Benchmarking supervised machine learning

To prove the benefit of the proposed data splitting, we run multiple ML methods on the VeReMi dataset while varying the data split approaches. We present and analyze the results of each method with each setup. For both split methods, we fit our models with the same standard Scikit-learn parameters from [191]. We tested the following ML methods that are commonly used in the vehicular community, AdaBoost, Decision Tree, Naive Bayes (NB), Nearest Neighbors, Neural Net, and Random Forest (RF)[2].

To evaluate both data splits, we first split the data into train and test sets, considering the temporal and broadcast aspects of the data. Then, the train set is split into training and validation sets using both random and our splitting approach separately for performance comparison. Also, we evaluate the MBD performance with and without using the previously extracted features (section 6.2.6). Overall, we consider 5 attacks, 6 ML methods, 2 splitting methods, and 2 sets (validation and test) with (or without) engineered features.

6.4.1 Metrics

In MBD, there are fewer bogus than normal messages, resulting in an imbalanced class problem. Therefore, we use the harmonic mean of precision and recall, named $F_1 - score$.

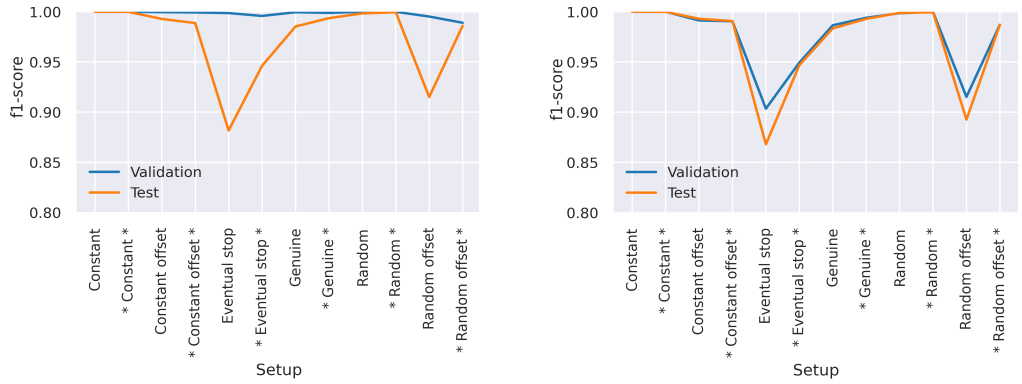
$$F_1 - score = 2. \frac{precision \,.\, recall}{precision + recall} \tag{6.4}$$

Thus, a high $F_1 - score$ value means that the model classifies both the positive (attack) and negative (benign) classes successfully.

6.4.2 Results and discussion

Results show that a random splitting approach gives biased performances, i.e., the model does not generalize the learned behavior during the learning phase and gets a significantly lower (10%) performance during the testing phase. To observe this, we plot the best performances obtained among all methods for each attack in the learning phase in Figure 6.9a. This shows the obtained $F_1 - score$ at the learning phase (validation) and testing phase

[2]The sources and dataset for our work are provided here https://github.com/mohammedLamine/Spatial-and-temporal-cross-validation-strategy-for-misbehavior-detection-i



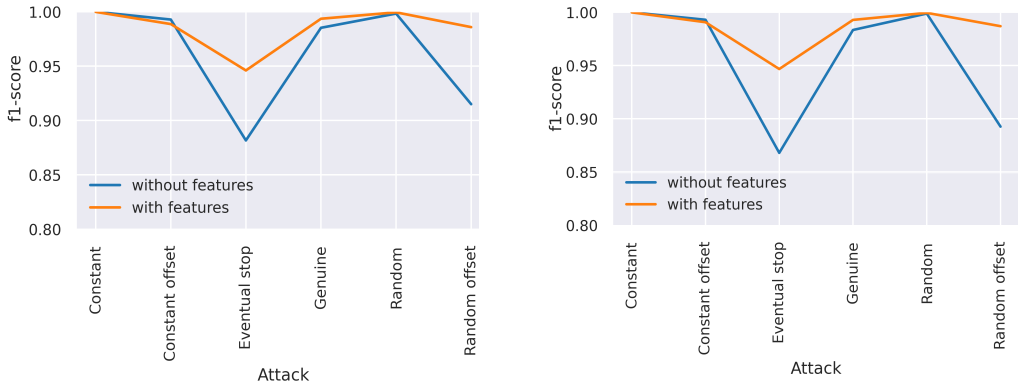
(a) Results of a random splitting

(b) Results of our splitting approach

Figure 6.9: Splitting methods' comparison on models' generalization

(test). We notice that the learning phase's performance fits perfectly for each attack, with a minimum of 0.99 $F_1 - score$. We expect the model to have a similarly good performance in the testing phase.

However, the model tested on unseen data, i.e., the test set obtained worse performance in the testing phase than what was expected ($F_1 - score$ drops to 0.88 in case of eventual stop attack which makes a performance 10% worse). This questions the high performance observed in the learning phase and can be explained by the data leakage caused by the random splitting. Therefore, our proposed splitting fixes this problem, and our learned model generalizes better. Thus, the model should perform in the testing phase as well as in the learning phase.



(a) Results of a random split

(b) Results of our approach

Figure 6.10: Features influence on models performance

Compared to the random splitting (Figure 6.9b), the performance in the

learning phase for our splitting is not a perfect fit for the data ($F-1score$ varies between 0.9 and 1 depending on the attack). Compared to the performance on the testing phase, we notice that both performances are similar (3% lower performance in the worst case).

The result confirms that random splitting is not a good approach when modeling broadcast data as it induces data leakage in the learning phase. Our approach fits better in the broadcast scenario and mitigates data leakage by ensuring respect of spatial and temporal dependencies of the broadcast.

Furthermore, to assess the impact of the extracted features on the models, we plot the performance of each model with and without the extracted features. Figure 6.10 shows the obtained $F1-score$ for all attacks and splitting methods. Figure 6.10 shows improved results for each attack in the testing phase. We notice an improvement between 0.07 and 0.1 in the case of an eventual stop attack.

Thus, the extracted features (ART, SAW, SSC) improved the detection of attacks, especially the ones that were more difficult to detect.

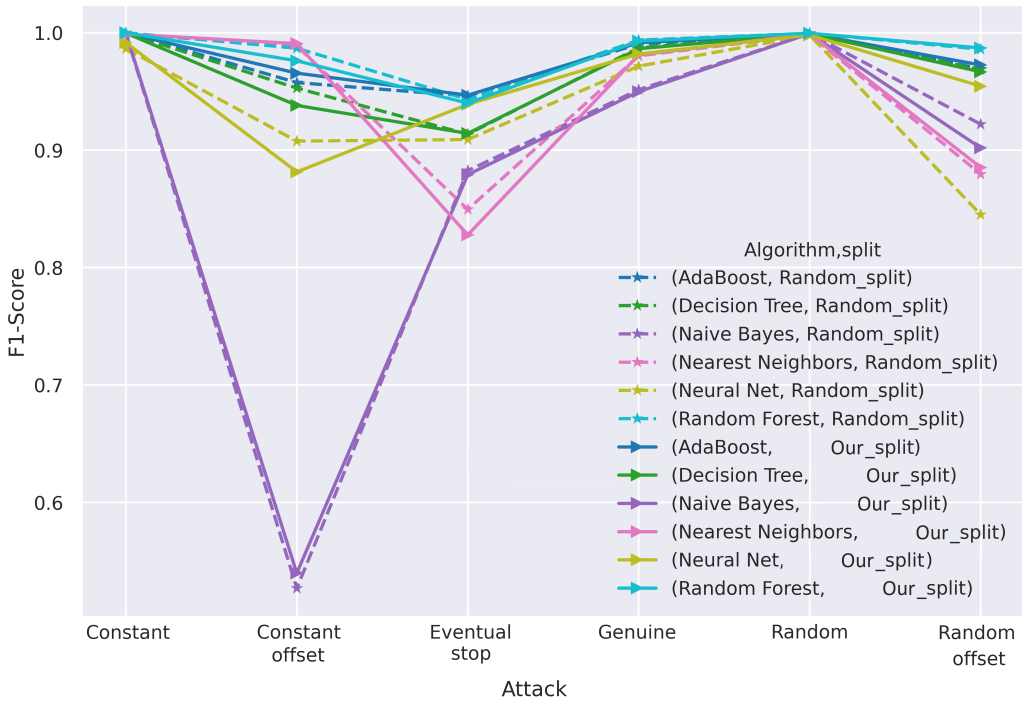


Figure 6.11: $F_1-score$ using different splitting and ML methods

In Figure 6.11, we plot the performance of each method, where the $F_1-score$ is computed on the test set for each data splitting approach. As seen, the performance using a random splitting is slightly better than

the performance obtained using our approach for some pairs of methods and attacks, particularly for NB and RF methods and the Random attack. For RF, the difference in $F_1 - measure$ is very small (less than 0.02), meaning that the two approaches are relatively equivalent. The same observation can be made on the random attack with the NB method.

On the other hand, the performance using our approach is better using neural networks with random offset attacks; in fact, we can see an 0.1 increase in performance for this setup (0.954 for our approach and 0.845 for a random splitting). We observe that the NB method has the worst performance in most cases (for each attack). The methods RF and AdaBoost have the best overall performance, with RF performing slightly better (about 0.01 increase in $F1 - score$). This suggests that boosting-based methods perform better in the task of MBD, because those methods fit many small classifiers that can model the specificity of each attack.

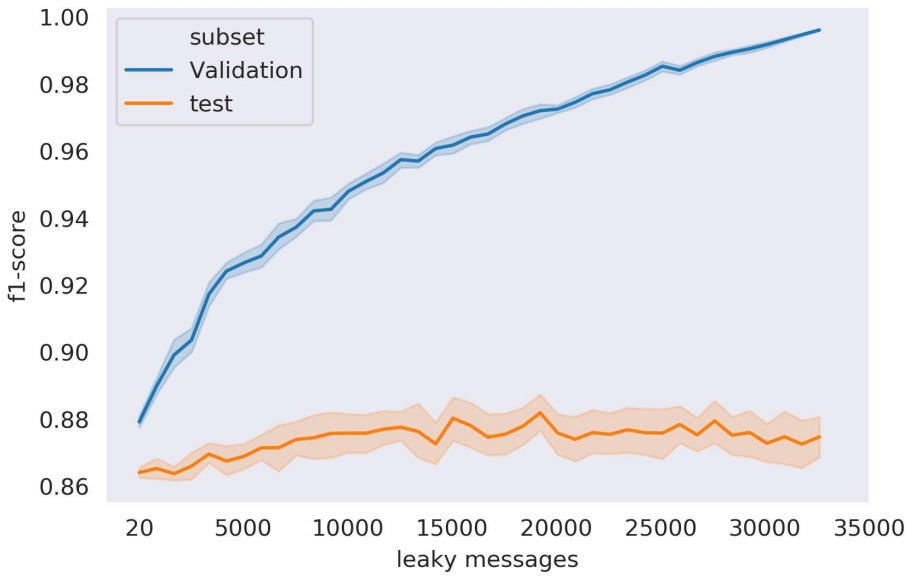


Figure 6.12: Evolution of $f_1 - score$ values for the validation and test sets subject to the number of leaked messages

Figure 6.12 shows the performance obtained by the decision tree algorithm for several data splits. For each split, we purposely leak a certain number of messages between validation and train sets to emphasize the effects of data leakage on performance. We observe that the validation score gets better as we increase the number of leaked messages (from a 0.88 score

to nearly 1.0). The test score on the other hand stays the same (roughly between 0.86 and 0.88). The observed gap in the score at the highest number of leaks leads to an overestimation of the model and an incoherent performance in the real world.

Overall, random splitting datasets of a broadcast time series overestimate the learning performance and lower the performance on the test set. However, the use of engineered features improves the performance.

6.5 Analysing RL on VEREMI

To apply reinforcement learning on the dataset VeReMi, we must first create an artificial simulation environment for the RL agent where the dataset feeds the data of this simulation. To do so, we wrote a simple algorithm that works as follows. We apply algorithm 2 for all different vehicles in the VeReMi

Algorithm 2 Simulating from a dataset

Require: $M\{msg_{id}, time, sender_{id}, ...\}, agent_{id}$

1: **procedure**
2: $\quad scenario_{data} = M[reciever_{id} == agent_{id}]$
3: $\quad$ **sort**$(scenario_{data}, time)$
4: $\quad$ **yield** $scenario_{data}$
5: **end procedure**

dataset. The RL model implemented in vehicles will receive sequences of messages from neighboring vehicles and will learn to detect false messages from genuine ones. For that, a reward signal is defined based on labels [127, 132, 135, 137, 138] as follows.

Reward

- The agent receives a positive reward +4 if a false message is correctly detected.
- The agent receives a negative reward −3 if a false message is wrongly detected.
- The agent receives a positive reward +1 if a true message is correctly detected.
- The agent receives a negative reward −2 if a true message is correctly detected.

Of course, as discussed in section2.5, defining the reward function in this manner strongly assumes that a misbehavior authority is capable of providing such data in real time. The reward defined in this section will train our RL model in the next section. In the next section, we define the hyper-parameters used to build our RL model and how we train the model with respect to the literature.

6.5.1 Defining and training the model

We used our architecture from chapter 5 to create the RL model to be trained on VeReMi. And thus, a set of hyper-parameters must be tuned to find an appropriate model. We summarize the set of hyperparameters in table 6.5. We first describe here how an RL model and in particular a deep q-learning model is trained.

Parameter	Domain and description	chosen value
num deep-Q layers	Number of layers of the q-learning $\in \mathbb{N}$	20 000 000
num update layers	Number of layers of the update function $\in \mathbb{N}$	20 000 000
num iterations	Number of training iteration $\in \mathbb{N}$	20 000 000
initial collect steps	data collection steps before training $\in \mathbb{N}$	1000
iteration collect steps	data collection steps during training$\in \mathbb{N}$	32
replay buffer length	maximum number of collected training data $\in \mathbb{N}$	20 000
batch size	number of data points used for each training step $\in \mathbb{N}$	256
num batch per iter	number of batch runs every step $\in \mathbb{N}$	1 (unused)
learning rate	leaning factor $\in \mathbb{R}^+$	$5 * 10^{-6}$
decay learning rate	whether to decay the learning rate during training	true
epsilon	probability of choosing a random action [0,1]	0.1
target update period	number of steps for target update $\in \mathbb{N}$	5
target update tau	ratio of target update[0,1]	0.95
gamma	reward discount factor [0,1]	0.97
loss	loss function	Huber loss

Table 6.5: RL model hyperparameters

It is acknowledged that training a deep q-learning model often leads to

instability [192, 193]. To achieve stability, a replay memory (or experience replay) is used to store the trajectory of the MDP (state, action, reward, next state). Then, a mini-batch of data points is sampled from the memory. The main reason for using such memory is to avoid the temporal dependency of successive observations used to train the model [193]. We implement a replay memory to train our RL model. The memory is first initialized with a set of observations and then updated at every step with new observations. In addition to the replay memory, a duplicate network (called target network) is used to obtain an unbiased estimator of the error used to train the model (the target model is used to compute future action values, while the original network is used to compute the action-value for the current state and actions. We synchronize the target network with the originally trained network every few steps [193]. In our implementation, we partially synchronize the two networks. We use TensorFlow agents [194] for all our implementations. The values used for all hyperparameters regarding training can be looked up in table 6.5.

6.5.2 Model returns analysis

Using the hyperparameters from the previous section, we train several model architectures with the aim of maximizing the returns. The reward of each step of an episode is summed and normalized to episode length, and we visualize the evolution of this metric during training.

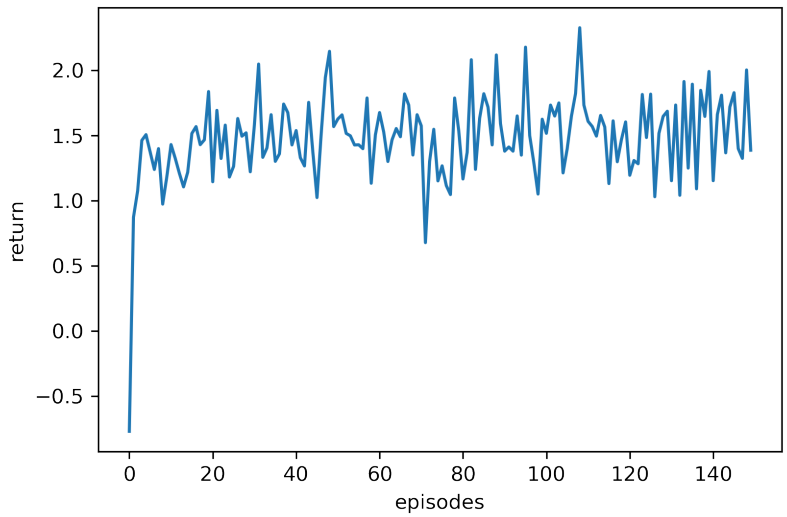


Figure 6.13: Average returns per episode over training period

Figure 6.13 show the average reward per episode during the training. In the beginning, the amount of return obtained by the model is low (less than -0.5 per action) then slowly inches up to a better average value (1.5 per action) as the model learns the dynamics of the environments until the model is stable. Similarly, the loss obtained throughout the training of the model is shown in Figure 6.14.

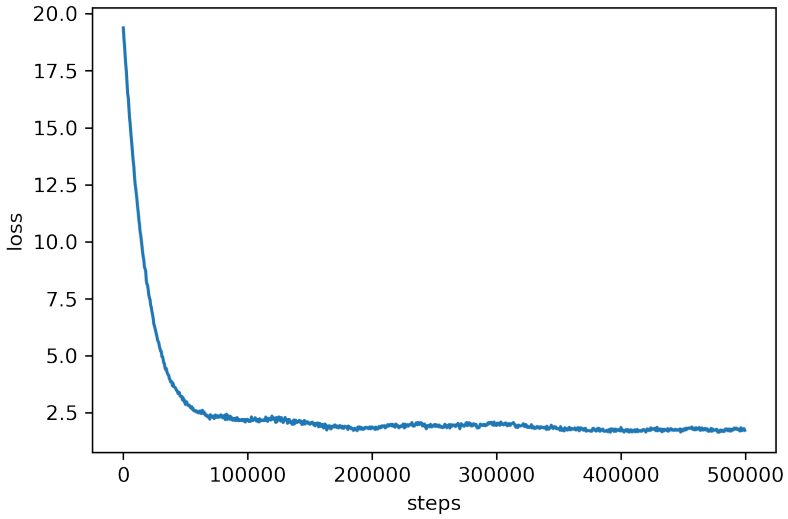


Figure 6.14: Loss per step over training period

We observe the loss going lower as the model is learning. The loss keeps fluctuating near the reached sub-optimal value (i.e., 1.7) due to the nature of training an RL model as both the compared q-values change are obtained through the two networks (original and target networks from the last section). The final network is then kept and tested on a separate dataset and the results are shown in the next section.

6.5.3 Model results

	Genuine	Malicious
Genuine	56525	947
Malicious	4757	24679

Table 6.6: Confusion matrix: RL model

We generate using the same algorithm 2 a set of scenarios to test the trained RL model from the last section. The model actions are interpreted as labels for each received message in the scenario. We construct a confusion matrix using the generated actions and the ground truth labels in table 6.6. The model was able to correctly predict 24679 attacks out of 29436 attacks in the test scenarios, successfully achieving 96% precision and 84% recall scores. An f1-score of 90% with respect to the malicious label and an overall precision of 93% (table 6.7). The overall results show that the RL model is quite capable of detecting Malicious messages from the VeReMi dataset.

	Precision	Recall	F1-score	Accuracy	Support
Genuine	92%	98%	95%	93%	57472
Malicious	96%	84%	90%		29436

Table 6.7: Results: RL model

For comparison, we train multiple supervised learning models on the VeReMi dataset and compare the obtained performances. Firstly, we train the strongest supervised model on this task, which is random forests. The random forests model confusion matrix is shown in table6.8. The model is able to correctly detect 24893 attacks (+214 compared to the RL model) and also achieves high scores on precision and recall, 97% and 85% respectively, and an overall accuracy of 94% (table 6.9). Overall, the random forests model achieves a marginally better performance of around 1% in our metrics than the RL model.

	Genuine	Malicious
Genuine	56670	802
Malicious	4543	24893

Table 6.8: Confusion matrix: Random forests model

	Precision	Recall	F1-score	Accuracy	Support
Genuine	93%	99%	95%	94%	57472
Malicious	97%	85%	90%		29436

Table 6.9: Results: Random forests model

Secondly, we train other supervised learning models, namely, decision trees (DT), support vector classification (SVC), nearest neighbors (KNN), AdaBoost, multi-layered perceptron (MLP), and naive Bayes. The summary

Malicious	Precision	Recall	F1-score	Accuracy
RL	96%	84%	**90%**	*93%*
RF	97%	85%	**90%**	**94%**
DT	87%	85%	86%	91%
SVC	100%	47%	64%	82%
KNN	85%	56%	67%	82%
AdaBoost	95%	72%	82%	89%
MLP	97%	74%	84%	90%
Naive Bayes	100%	45%	62%	81%

Table 6.10: Summary of precision, recall, f1-score and accuracy measures of trained models on VeReMi for malicious messages prediction

of the results obtained by all models is shown in table 6.10. As pointed out, random forests obtained the best performance in terms of accuracy 94% and f1-score 90%. Decision trees, which are a similar model to random forests, obtain a slightly worse performance, achieving 86% f1-score as the second-highest score of supervised models. Other models struggle to detect the malicious messages, with ranging recall from 45% to 74%, meaning that at least one out of 4 attacks is not detected.

Unsurprisingly, the sophisticated random forests model performs best on the VeReMi dataset and should be the go-to model in this case. Though, one would wonder how it would perform on a novel attack outside this restricted environment, restricted to simulated episodes from a fixed dataset with fixed misbehaviors. What happens when the trained model is confronted with unseen misbehaviors?. Is the knowledge learned from the dataset enough to achieve good performance? In the next section, we confront the model in such a situation using the simulator. We show that there's a drop in performance in both RL and RF models. And we exploit the learning from interactions feature of the RL model to perform better than the RF model in simulation.

6.6 Analysing RL model

We showed in the previous section that the RL model proposed in chapter 5 was able to overperform most supervised models, except for random forests, which performed slightly better on the dataset VeReMi. In this section, we continue the analysis of the proposed RL model on a simulated environment, and we show how exploiting the online learning of the RL model allows us to achieve better performance than random forests.

6.6.1 Simulation setup

We use our simulation to simulate the C-ITS environment for our RL agent. To recreate similar episodes to the ones simulated from VeReMi in the previous section, we implemented some of VeReMi attacks in our simulation, and then we add an additional attack for testing. The main simulation parameters are as follows :

- Number of vehicles: 100+ vehicles
- Number of attackers: 10% of vehicles
- Number of agents: 1 agent
- Agent: randomly chosen at the start of the episode
- Episode time: 15 minutes or 3000 agent steps

The chosen parameters for our simulation increase the randomness of actual episodes. We only allow one vehicle to detect misbehaviors in the simulation with respect to the MBD architecture built in chapter4, though, one can allow for multiple agents at the same time if the aim is to implement a multi-agent system or a distributed learning mechanism. The agent is chosen randomly in each episode. 10% of the vehicles are selected randomly at the start of the episode with different attack implementations. The episode ends if either the episode or the agent acts 3000 times in the simulation, i.e., 5 minutes of presence if dissemination of CAM messages is optimal at 10Hz frequency. Overall, different runs of the simulation yield different episodes.

6.6.2 Training

We reimplemented the RL architecture from figure5.2 inside our simulation and feed as input exchange messages between vehicles and additional metrics on the messages, such as distance between the vehicles and, as per the model conception, the latent state representation. As mentioned before, only one vehicle is implementing the model in the same episode. The agent will then go through the simulation and take action for each received message. The actions include labeling the received message and blacklisting the sending vehicle in the simulation so that all messages from that vehicle are ignored. The agent is then rewarded and penalized accordingly. Following are the reasoning and outputs of our penalization and rewarding of the agent.

- The aim is to optimize the trip for the agent by reducing its time loss as it drives through the simulation. Unsurprisingly, we use the time

loss metric as part of the optimization goal of our agent. We assume that, mainly, misbehaviors and attacks increase time loss and thus can indicate the presence of misbehavior in the exchanges. Though, this can raise an unwanted selfish-like behavior of our agent. Using the time loss of the agent alone might orient to learning a policy that might undermine the global network if it allows the agent to reduce its time loss. For example, an attack that affects vehicles around the agent causing them to slow or change lines might open up a line for the agent and allow it to speed up and reduce its time loss. As it is, the agent will be encouraged to allow such attacks to happen. To avoid such a policy, we introduce a tradeoff between the agent time loss and the local neighborhood time loss. The agent will then be able to pursue its goal of detecting misbehaviors in the C-its system.

- Additionally, for labeling, we use misclassification in the computation of the final reward signal and penalize the actions of the agent when blacklisting other vehicles. Note that the actions of the agent have immediate consequences in the episode. For example, black listing an attacking vehicle that, as part of its attack, creates a ghost vehicle (only exists in the messages with no physical presence) helps unblock and improve the affected vehicles' behaviors. These actions also increase the randomness of the created episodes and help the agent in the learning process.

- Furthermore, the model is trained first in a simulation where only VeReMi attacks are present. The idea here is to be able to compare the RL model to supervised models on known attacks and then novel unseen attacks that could emerge in the future.

We adjust the parameters for training our model as shown in table6.11. Note that most of the parameters are the same except for a bigger replay buffer, collection steps per iteration, and batch size. As we are able to generate quite different episodes and by extension different training examples, we had to increase the buffer to hold more of these examples and to generate more per step. The model is then able to see lots of variable examples at each step which helps in learning the general behavior of its context as it is fed with better knowledge and also leads to better detection performance. We show in figure6.15 the loss for training the RL model. Similarly to training on the data set, the loss starts decreasing from 24 as the model learns the context of simulation and starts achieving better detection performance and finally fluctuates around 1 as the training ends. In the next section, we present the results obtained by the model.

Parameter	Domain and description	chosen value
num deep-Q layers	Number of layers of the q-learning $\in \mathbb{N}$	2
num update layers	Number of layers of the update function $\in \mathbb{N}$	2
num iterations	Number of training iteration $\in \mathbb{N}$	20 000 000
initial collect steps	data collection steps before training $\in \mathbb{N}$	2
iteration collect steps	data collection steps during training$\in \mathbb{N}$	256
replay buffer length	maximum number of collected training data $\in \mathbb{N}$	100 000
batch size	number of data points used for each training step $\in \mathbb{N}$	128
num batch per iter	number of batch runs every step $\in \mathbb{N}$	100 (unused)
learning rate	leaning factor $\in \mathbb{R}^+$	$5 * 10^{-6}$
decay learning rate	whether to decay the learning rate during training	true
epsilon	probability of choosing a random action [0,1]	0.1
target update period	number of steps for target update $\in \mathbb{N}$	5
target update tau	ratio of target update[0,1]	0.95
gamma	reward discount factor [0,1]	0.97
loss	loss function	Huber loss

Table 6.11: RL model hyperparameters for simulation

6.6.3 Model testing

After training the model the first time on VeReMi attacks in the simulation, another attack creates ghost vehicles ahead of the attacker vehicle to slow down the traffic introduced to the simulation. These ghost vehicles reproduce the attacker states (speed, breaking state, heading, ...) except for the position which makes the attack quite similar to VeReMi attacks on positions. The RL model is then allowed to adapt and adjust to it by online learning inside the simulation. Table6.12 shows the resulting confusion matrix. The RL model was able to detect 21617 attacks and failed to detect the remaining 9408. At the same time, it was able to correctly classify 57379 of genuine data and only misclassify 1596 of them. Given that the attack is fairly new to the model and the genuine data is the same as for training, it was already predicted that the models will perform much better on genuine data. In summary, as presented in table6.13, the model was able to achieve 80% f1-

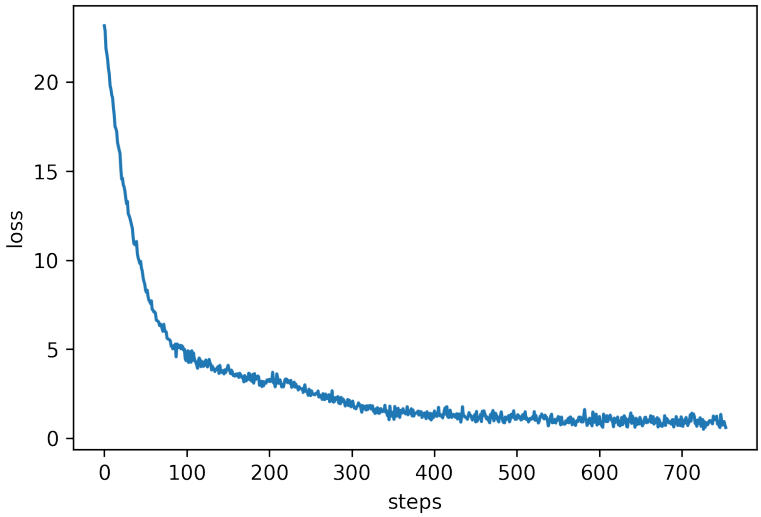


Figure 6.15: Loss per step over training period in simulation

score on malicious data detecting 70% of them which represents a fairly good performance on unseen attacks. We believe as the agent keeps running inside the simulation, these metrics can improve.

	Genuine	Malicious
Genuine	57379	1596
Malicious	9408	21617

Table 6.12: Confusion matrix: RL model on simulation

	Precision	Recall	F1-score	Accuracy	Support
Genuine	86%	97%	91%	88%	58975
Malicious	93%	70%	80%		31025

Table 6.13: Results: RL model on simulation

To compare our RL model to random forests, we logged all data examples while training the RL model and created a dataset out of them. The data is then used to train the random forests model, and we show in table6.14 the confusion matrix corresponding to it. The random forests model was able to detect 18975 attacks and failed to detect 12050 of them. Similar to our RL model, the random forests greatly performed on genuine data,

correctly classifying 58626 examples and misclassifying only 349 of them. Table6.15 show the resulting metrics based on the results of random forests. The model was able to achieve 75% f1-score and only detected 61% of the attacks. Compared to our RL model, random forests dropped 9% in detecting the new attack.

	Genuine	Malicious
Genuine	58626	349
Malicious	12050	18975

Table 6.14: Confusion matrix: Random forests model on simulation

	Precision	Recall	F1-score	Accuracy	Support
Genuine	83%	99%	90%	86	58975
Malicious	98%	61%	75%		31025

Table 6.15: Results : Random forests model on simulation

Unsurprisingly, our RL model performed better than supervised models in the simulation compared to a dataset, as we showed in the previous section. Exploiting the online learning of the RL framework was key to achieving such a result. Furthermore, learning a rich representation of the context of the vehicle, updated and adapted to the changes in this context, and aiming to improve the detection of misbehaviors gave the RL model an upper hand on performance compared to learning a distribution of dataset examples of an attack. And it shows that detecting misbehaviors in a dynamic interactive context is better done using a similarly dynamic and interactive solution. Finally, we believe the current performance of our RL model is good but can be improved by further investigating distributed learning and cooperative detection that might be harder to achieve and regulate.

With regard to the explainability of such a solution, it is difficult in this case to understand the rationale behind misbehavior detection decisions, the model does not output any helpful data about how the decision was made. Some works tackled the problem of explainability in cybersecurity [195, 196]. Some suggest using explainable methods such as decision trees where the output is a set of understandable rules at the cost of some performance decay when affordable. Others try different explanations, in particular in our case, we are interested in the use of SHAP (SHapley Additive exPlanations) [197] to try and generate explanations of the RL model. In future work, we

want to tackle both explaining the model parameters and the latent state representation that our model outputs.

In this chapter, we implemented the MBD architecture, we analyzed the state-of-the-art dataset for misbehavior detection, VeReMi. We showed how specific preprocessing in our MBD architecture solves the problem of data leakage in some supervised ML applications. We analyzed supervised machine learning and reinforcement applications on VeReMi. And finally, we implemented our RL model in the simulation and showed how using our architecture, we are able to achieve better performances than state-of-the-art algorithms specifically in detecting novel emerging attacks.

In the next chapter, we conclude and give some future research perspectives.

Chapter 7

Conclusion and perspectives

7.1 Conclusion

In this thesis, we have been interested in securing connected and autonomous vehicles. CAVs and V2X are currently being researched, developed, and deployed all over the globe exploiting the significant potential to increase traffic safety, optimize traffic flow, and reduce traffic congestion and emissions. Increasing knowledge and semantics about the environment is the key idea to what CAVs V2X brings to the future of intelligent transportation systems. Nevertheless, it gives raise to security threats that have not been an issue before, including mainly, semantic-based attacks and failures. To face such threats, it is required to enhance the system with intelligent security. In our thesis, we propose a multi-layered architecture for the detection of abnormal behaviors with automatic learning to secure the connected and autonomous vehicles' communications, sensors, and internal components.

Our architecture is generic, in the sense that it allows several algorithms such as rule-based, machine learning, and deep learning to be used at different communication levels. And thus, can be made very simple using the simplest algorithms such as rule-based. And also can be very complex using multiple different algorithms at all levels. We provided a definition of misbehavior to cover the literature on cyber security threats, attacks, and failures and we built a corresponding misbehavior model to classify and understand these misbehavior covering security as well as failures/faults. Then, we built a simulation environment capable of generating V2X and sensor data and also injecting and detecting misbehaviors in real time which can be used to evaluate different detection methods. Moreover, we showed and developed a specific preprocessing that must be used when processing V2X broadcast data, and we evaluated several algorithms such as decision trees, random forests,

and multilayered perceptron that resulted in data leakage when only classic preprocessing was done. Furthermore, the architecture allows us to propose a novel reinforcement learning based neural architecture for the detection of misbehaviors that yield better results compared to current state-of-the-art algorithms.

More specifically, we considered the detection of semantic misbehaviors and novelties that can emerge without prior observations. We laid out a mathematical framework using partially observable Markov decision processes, on which we derive our RL model. We designed an update function that allows the vehicle to monitor and learn its environment and store critical knowledge. Furthermore, we designed the RL model to detect misbehaviors using our update function, and we showed in a simulated environment, through evaluation, that the model is capable of detecting novel misbehaviors and performs better than current state-of-the-art algorithms.

To conclude, we will share some possible future directions of research based on the work carried out in this thesis.

7.2 Perspectives

We share in this section some short and long-term perspectives of our work.

7.2.1 Short-term

7.2.1.1 Misbehavior classification and Explainability

We mainly focused in our thesis on detecting misbehavior, and to do that we worked on preprocessing the data in the vehicular context. Though, it would be interesting to classify these detections and understand how, why, and where did the misbehavior happen. We introduced a model for misbehavior which we wanted to learn while detecting the misbehavior. We believe that learning such a model can help explain misbehavior and give insight into how to better detect them. Another angle at this is to try and explain the machine learning model learned to detect misbehavior. In particular, it would be interesting to look inside the latent state representation learned by the model and understand what knowledge was kept over several iterations to help detect misbehavior.

7.2.1.2 Evaluation and improvement of the RL model

The RL model we defined was able to correctly predict a set of attacks in our simulation and achieved better performances on detecting novel unseen

misbehaviors. One can ask how it would perform over a wider range of attacks and even unintentional failures in the system. We want to implement such attacks to test the model and improve upon the current performance. Additionally, one may seek to expand the POMDP framework, and in particular the state representative for one vehicle with cooperative knowledge. The partially observed state would then increase from being just what our agent observes, to what all neighboring vehicles are observing. The cooperative perception message can be the stepping stone for such future work. In order to improve on our empirical results, one may resort to more advanced architectures such as LSTMs for the definition of both the update function network and the q-learning network.

7.2.1.3 CPM integration

In our work, we heavily used CAM messages for injecting and detecting attacks. The CPM message introduces another level of complexity to both the detection and injection of attacks and would be interesting to see how our architecture in its current partial implementation performs using this kind of data. The CPM will also increase the knowledge about the vehicle's context in our latent space representation which we would want to evaluate its benefit.

7.2.1.4 Simulation environment

The simulation used in this thesis is limited in terms of the variety of attacks, possible actions, and reactions of the vehicular network. It might be beneficial to introduce such powerful additions to the simulation and helps learn a better model, and face a stronger threat, which would make the simulation a good environment for collaboration on misbehavior detection. It would allow the research community to focus on improving their misbehavior mechanisms instead of collecting data or improving their misbehavior simulators.

7.2.2 Long-term

7.2.2.1 Full implementation of the architecture

Our architecture is generic, and it was not possible to implement each and every module, level, and layer of it during this thesis. We want to incrementally implement these modules to hopefully achieve a most robust misbehavior detection for CAV. The implementation of this architecture would need an adequate environment where standard C-ITS data is available as well as

components data. The modules can then be built independently to detect, classify and report misbehaviors.

7.2.2.2 Real use case integration

The proposed methods and architecture in this thesis were only tested and applied to simulated data and environments. We want to test our solutions on real use cases. Abiding with the real world and real-time constraints is as challenging as it is necessary. We want to see how our solutions perform in such cases so that we improve and adapt them to create impactful results.

7.2.2.3 Standardized evaluation

Overall in this work and the current literature on misbehavior detection for connected and autonomous vehicles, we evaluate our solutions mainly from detection rate metrics. Defining and possibly standardizing specific metrics for the evaluation of such solutions with respect to goals and objectives for CAVS will be very rewarding to the vehicular community. It would also help for comparison and benchmarking purposes and will push research further.

7.3 Open Issues

Throughout this work, we surveyed existing papers on MBD. We focused on ML and existing simulators and datasets in order to assess state-of-the-art and investigate novel development tracks. We can assert that an important research activity in MBD has been done. However, as CAVs are rapidly evolving, certain limitations and open problems arise and require further exploration.

7.3.1 Evaluation, validation, and reproducibility

The surveyed set of papers uses different datasets with different evaluation methods, making comparing the different works difficult. Their results are usually not reproducible due to the use of private datasets, private prepossessing, or confidential hyper-parameters. A first attempt to solve this problem is available in [9]. However, a common evaluation framework is mandatory for evaluating recent findings in the domain. It is also noticeable that some proposed methods lack formal validation while others are hardly interpretable. These black-box methods usually perform well, but do not offer insight into how the problem is being solved.

7.3.2 Security

A major issue to consider is the security of ML algorithms [54, 198, 199].In particular, these algorithms might be fooled by adversarial attacks perturbing the inputs and misleading the MBD. Attacks against ML algorithms can occur at the training (e.g., model poisoning attack) [200] and test stages (evasion attack) [201].

Moreover, new advanced services such as cooperative perception must consider MBD at the design phase.

7.3.3 Misbehavior Dataset and preprocessing

A major open issue is the lack of complete state-of-the-art datasets for MBD. Currently, simulated datasets do not include all physical behaviors of a ground truth dataset. For instance, these physical behaviors may include GNSS dead reckoning or V2X signal obfuscation. As a result, the obfuscation of the V2X signal may be interpreted as a discrete jamming attack. Therefore, a major contribution to the community would be to propose an open dataset that considers the three mentioned issues while being approved by the misbehavior community. Overall, this dataset would allow the research community to focus on improving their misbehavior mechanisms instead of collecting data or improving their misbehavior simulators. On the other hand, preprocessing these data should be done considering all the characteristics of the data. Indeed, broadcasting V2X data from the vehicle generates duplicated messages that might cause data leakage when applying ML methods [18].

7.3.4 Simulation Platform

Currently, the evaluation of algorithms for CAVs is costly, incomplete, unoptimized, or proprietary. Thus, it is difficult to correctly evaluate MBD solutions without an open-source environment dedicated to modeling and testing. Moreover, this platform may help to provide realistic datasets. Several contributions are paving the way in this direction. However, they do not address one of the following aspects: cryptography operations, sensor processing, or data fusions. In absence of one of these elements, it is difficult to demonstrate that MBD fulfills the safety requirements (e.g., detection time) associated with autonomous driving. Indeed, poor performances from the MBD module may occur.

7.3.5 Standardization

Globally, standards developing organizations (SDO) are defining the functional requirements for MBD. Currently, the main standardization effort is driven by the ETSI which published a report TR 103 460 and is working on a technical standard TS 103 759. We highlight potential open points that could be addressed in the forthcoming standards.

First, a baseline set of standards should define both MBD mechanisms and a misbehavior reporting service. For each MBD mechanism, specific data types and sources may be used, and if these are to be reported, the standard must specify all requirements on the sources and how they are combined to identify misbehaviors. For instance, in some instances, the data from a safety message (e.g., BSM) may be sufficient to detect an attack. In other instances, the system might use the list of detected objects around the vehicle, using vehicle-based sensors in order to verify if the safety message is correct. Secondly, it is conceivable that vehicles may include the opinion of other surrounding vehicles in their MBD system. This should only be done if the mechanisms for communicating are thoroughly specified and standardized. If this opinion is not trustworthy, the MBD system may be negatively affected by a majority of dishonest opinions.

Overall, ongoing standardization efforts are setting the core requirements for implementing MBD. We are confident that each issue listed above will be addressed in due course.

Contributions

- [19]A simulator for cooperative and automated driving security, M. L. Bouchouia, J.-P.Monteuuis, H. Labiod, O. Jelassi, W. B. Jaballah, J. Petit "Workshop on Automotive and Autonomous Vehicle Security (AutoSec), ndss-symposium (2022)."
- [18]Spatial and temporal cross validation strategy for misbehavior detection in C-ITS ML Bouchouia, JP Monteuuis,O Jelassi,H Labiod,WB Jaballah, J Petit "RCIS the Fifteenth International Conference on Research Challenges in Information Science 11 - 14 May, 2021"
- [17] A Survey on Misbehavior Detection for Connected and Autonomous Vehicles ML Bouchouia,H Labiod, O Jelassi,JP Monteuuis,WB Jaballah, J Petit , Z Zhang
- Reinforcement Learning Framework and Application for V2X Misbehavior Detection (ongoing)

Bibliography

[1] I. J. of Law and P. Policy, "Emerging ontology of driverless cars and plausible legal implications." https://ijlpp.com/emerging-ontology-of-driverless-cars-and-plausible-legal-implications/#_ftn1. Accessed: 2023-01-30.

[2] O. Cameron, "Under the hood of a self-driving taxi," Jul 2017.

[3] J.-P. Monteuuis, *Resilience by design & failures forecasting for a connected autonomous vehicle*. PhD thesis, Institut Polytechnique de Paris, 2020.

[4] A. Festag, "Cooperative intelligent transport systems standards in europe," *IEEE communications magazine*, vol. 52, no. 12, pp. 166–172, 2014.

[5] M. Arshad, Z. Ullah, N. Ahmad, M. Khalid, H. Criuckshank, and Y. Cao, "A survey of local/cooperative-based malicious information detection techniques in vanets," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, p. 62, 2018.

[6] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*, vol. 135. MIT press Cambridge, 1998.

[7] V. Taillandier, "Connecting autonomous vehicles and smart level crossings," Mar 2022.

[8] S. S. Banerjee, S. Jha, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "Hands off the wheel in autonomous vehicles?: A systems perspective on over a million miles of field data," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 586–597, IEEE, 2018.

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

[10] R. S. Hallyburton, Y. Liu, Y. Cao, Z. M. Mao, and M. Pajic, "Security analysis of Camera-LiDAR fusion against Black-Box attacks on autonomous vehicles," in *31st USENIX Security Symposium (USENIX Security 22)*, (Boston, MA), pp. 1903–1920, USENIX Association, Aug. 2022.

[11] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *20th USENIX security symposium (USENIX Security 11)*, 2011.

[12] C. Nast, "Hackers Remotely Kill a Jeep on the Highway—With Me in It," *WIRED*, July 2015.

[13] S. Nie, L. Liu, and Y. Du, "Free-fall: Hacking tesla from wireless to can bus," *Briefing, Black Hat USA*, vol. 25, no. 1, p. 16, 2017.

[14] A. Chernikova, A. Oprea, C. Nita-Rotaru, and B. Kim, "Are self-driving cars secure? evasion attacks against deep neural networks for steering angle prediction," in *2019 IEEE Security and Privacy Workshops (SPW)*, pp. 132–137, IEEE, 2019.

[15] J. Kamel, I. Jemaa, A. Kaiser, L. Cantat, and P. Urien, "Misbehavior detection in c-its: A comparative approach of local detection mechanisms," in *Vehicular Networking Conference (VNC)*, 2019.

[16] M. Jagielski, N. Jones, C.-W. Lin, C. Nita-Rotaru, and S. Shiraishi, "Threat detection for collaborative adaptive cruise control in connected cars," in *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pp. 184–189, 2018.

[17] M. L. Bouchouia, H. Labiod, O. Jelassi, J.-P. Monteuuis, W. B. Jaballah, J. Petit, and Z. Zhang, "A survey on misbehavior detection for connected and autonomous vehicles," *Vehicular Communications*, p. 100586, 2023.

[18] M. L. Bouchouia, J.-P. Monteuuis, O. Jelassi, H. Labiod, W. Ben Jaballah, and J. Petit, "Spatial and temporal cross-validation approach

for misbehavior detection in c-its," in *Research Challenges in Information Science* (S. Cherfi, A. Perini, and S. Nurcan, eds.), (Cham), pp. 452–468, Springer International Publishing, 2021.

[19] M. L. Bouchouia, J.-P. Monteuuis, H. Labiod, O. Jelassi, W. B. Jaballah, and J. Petit, "A simulator for cooperative and automated driving security," *Workshop on Automotive and Autonomous Vehicle Security (AutoSec), ndss-symposium*, 2022.

[20] ISO, "Iso 26262 : Road vehicles — functional safety," 2018.

[21] J. SAE, "3016: 2021 taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," *Society of Automotive Engineers*, 2021.

[22] A. Nkoro and Y. A. Vershinin, "Current and future trends in applications of intelligent transport systems on cars and infrastructure," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 514–519, IEEE, 2014.

[23] ETSC, "Briefing : Cooperative intelligent transport systems (c-its)." https://etsc.eu/wp-content/uploads/ETSC-Briefing-on-Cooperative-Intelligent-Transport-Systems-C-ITS.pdf. Accessed: 2022-06-15.

[24] ETSI, "Automotive intelligent transport systems (its)." https://www.etsi.org/technologies/automotive-intelligent-transport. Accessed: 2023-01-30.

[25] ETSI, "Technical specification 102 894-1 v1.1.1 (2013-08) : Intelligent transport systems (its); users and applications requirements; part 1: Facility layer structure, functional requirements and specifications." https://www.etsi.org/deliver/etsi_ts/102800_102899/10289401/01.01.01_60/ts_10289401v010101p.pdf. Accessed: 2022-06-15.

[26] Convex-Project, "Deliverable d4.1: Roadside its station specification." https://convex-project.de/onewebmedia/D4.1_Roadside-ITS-Station-Specification_rev1.pdf. Accessed: 2022-06-15.

[27] ETSI, "European standard 302 637-2 v1. 3.1-intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service," *ETSI*, 2014.

[28] E. ETSI, "319 411-1 v1.2.2. electronic signatures and infrastructures (esi); policy and security requirements for trust service providers issuing certificates; part 1: General requirements," *ETSI, Sept*, 2018.

[29] C. Dukes, "Committee on national security systems (cnss) glossary," *CNSSI, Fort Meade, MD, USA, Tech. Rep*, vol. 4009, 2015.

[30] R. Ross, M. McEvilley, and J. Oren, "Systems security engineering: Considerations for a multidisciplinary approach in the engineering of trustworthy secure systems," tech. rep., National Institute of Standards and Technology, 2016.

[31] J.-P. Monteuuis, J. Petit, J. Zhang, H. Labiod, S. Mafrica, and A. Servel, "Attacker model for connected and automated vehicles," in *ACM Computer Science in Car Symposium*, 2018.

[32] C. Yan, W. Xu, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," *DEF CON*, vol. 24, 2016.

[33] S. Parkinson, P. Ward, K. Wilson, and J. Miller, "Cyber threats facing autonomous and connected vehicles: Future challenges," *IEEE transactions on intelligent transportation systems*, vol. 18, no. 11, pp. 2898–2915, 2017.

[34] J. Petit and S. E. Shladover, "Potential cyberattacks on automated vehicles," *IEEE Transactions on Intelligent transportation systems*, vol. 16, no. 2, pp. 546–556, 2014.

[35] P. Bhavsar, P. Das, M. Paugh, K. Dey, and M. Chowdhury, "Risk analysis of autonomous vehicles in mixed traffic streams," *Transportation Research Record*, vol. 2625, no. 1, pp. 51–61, 2017.

[36] H. Hasrouny, A. E. Samhat, C. Bassil, and A. Laouiti, "Vanet security challenges and solutions: A survey," *Vehicular Communications*, vol. 7, pp. 7–20, 2017.

[37] A. Rawat, S. Sharma, and R. Sushil, "Vanet: Security attacks and its possible solutions," *Journal of Information and Operations Management*, vol. 3, no. 1, p. 301, 2012.

[38] J. Cui, L. S. Liew, G. Sabaliauskaite, and F. Zhou, "A review on safety failures, security attacks, and available countermeasures for autonomous vehicles," *Ad Hoc Networks*, vol. 90, p. 101823, 2019.

[39] M. N. Mejri, J. Ben-Othman, and M. Hamdi, "Survey on vanet security challenges and possible cryptographic solutions," *Vehicular Communications*, vol. 1, no. 2, pp. 53–66, 2014.

[40] M. Amoozadeh, A. Raghuramu, C.-N. Chuah, D. Ghosal, H. M. Zhang, J. Rowe, and K. Levitt, "Security vulnerabilities of connected vehicle streams and their impact on cooperative driving," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 126–132, 2015.

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

[42] V. L. Thing and J. Wu, "Autonomous vehicle security: A taxonomy of attacks and defences," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 164–170, IEEE, 2016.

[43] J.-P. Monteuuis, A. Boudguiga, J. Zhang, H. Labiod, A. Servel, and P. Urien, "Sara: Security automotive risk analysis method," in *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security*, pp. 3–14, ACM, 2018.

[44] M. Howard and S. Lipner, *The security development lifecycle*, vol. 8. Microsoft Press Redmond, 2006.

[45] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," *Black Hat Europe*, vol. 11, p. 2015, 2015.

[46] C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, "Darts: Deceiving autonomous cars with toxic signs," *arXiv preprint arXiv:1802.06430*, 2018.

[47] F. Sommer, J. Dürrwang, and R. Kriesten, "Survey and classification of automotive security attacks," *Information*, vol. 10, no. 4, p. 148, 2019.

[48] C. L. Krishna and R. R. Murphy, "A review on cybersecurity vulnerabilities for unmanned aerial vehicles," in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pp. 194–199, IEEE, 2017.

[49] F. Jahan, W. Sun, Q. Niyaz, and M. Alam, "Security modeling of autonomous systems: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–34, 2019.

[50] R. W. van der Heijden, S. Dietzel, T. Leinmüller, and F. Kargl, "Survey on misbehavior detection in cooperative intelligent transportation systems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 779–811, 2018.

[51] N. Bißmeyer, *Misbehavior detection and attacker identification in vehicular ad-hoc networks*. PhD thesis, Technische Universität, 2014.

[52] F. A. Ghaleb, A. Zainal, and M. A. Rassam, "Data verification and misbehavior detection in vehicular ad-hoc networks," *Jurnal Teknologi*, vol. 73, no. 2, 2015.

[53] J. D. Howard and T. A. Longstaff, "A common language for computer security incidents," tech. rep., Sandia National Labs., Albuquerque, NM (US); Sandia National Labs ..., 1998.

[54] G. Dede, R. Hamon, H. Junklewitz, R. Naydenov, A. Malatras, and I. Sanchez, "Cybersecurity challenges in the uptake of artificial intelligence in autonomous driving, eur 30568 en," *EUR 30568 EN, Publications Office of the European Union*, 2021.

[55] R. Mariani, "An overview of autonomous vehicles safety," in *2018 IEEE International Reliability Physics Symposium (IRPS)*, pp. 6A–1, IEEE, 2018.

[56] I. O. for Standardization, *ISO/SAE 21434: 2021: Road Vehicles: Cybersecurity Engineering*. ISO, 2021.

[57] U. N. I. of Standards and Technology, "Nist cybersecurity framework 2.0 concept paper: Potential significant updates to the cybersecurity framework." https://www.nist.gov/system/files/documents/2023/01/19/CSF_2.0_Concept_Paper_01-18-23.pdf. Accessed: 2023-01-30.

[58] I. T. S. J. P. O. I. J. of the US Department of Transportation (USDOT), "How to use the cybersecurity framework profile for connected vehicle environments." https://www.its.dot.gov/research_areas/cybersecurity/docs/1_How_to_Use_the_CSF_Profile_for_CVE.pdf. Accessed: 2023-01-30.

[59] J.-P. Monteuuis, J. Petit, J. Zhang, H. Labiod, S. Mafrica, and A. Servel, ""my autonomous car is an elephant": A machine learning based detector for implausible dimension," in *2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC)*, pp. 1–8, IEEE, 2018.

[60] S. So, P. Sharma, and J. Petit, "Integrating plausibility checks and machine learning for misbehavior detection in vanet," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 564–571, IEEE, 2018.

[61] ETSI, "Etsi tr 103 460 v2.1.1 (2020-10): Intelligent transport systems (its);security; pre-standardization study on misbehaviour detection; release 2," 2020.

[62] ETSI, "Etsi ts 103 759 v2.1.1 (2023-01): Intelligent transport systems (its);security; misbehaviour reporting service; release 2," 2023.

[63] M. Raya, "Data-centric trust in ephemeral networks," tech. rep., EPFL, 2009.

[64] S. Ruj, M. A. Cavenaghi, Z. Huang, A. Nayak, and I. Stojmenovic, "On data-centric misbehavior detection in vanets," in *2011 IEEE Vehicular Technology Conference (VTC Fall)*, pp. 1–5, IEEE, 2011.

[65] R. Hussain, S. Kim, and H. Oh, "Privacy-aware vanet security: Putting data-centric misbehavior and sybil attack detection schemes into practice," in *International Workshop on Information Security Applications*, pp. 296–311, Springer, 2012.

[66] K. Zaidi, M. Milojevic, V. Rakocevic, and M. Rajarajan, "Data-centric rogue node detection in vanets," in *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 398–405, IEEE, 2014.

[67] G. Loukas, E. Karapistoli, E. Panaousis, P. Sarigiannidis, A. Bezemskij, and T. Vuong, "A taxonomy and survey of cyber-physical intrusion detection approaches for vehicles," *Ad Hoc Networks*, vol. 84, pp. 124–147, 2019.

[68] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in iot security: current solutions and future challenges," *IEEE Communications Surveys & Tutorials*, 2020.

[69] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy and survey of intrusion detection system design techniques, network threats and datasets," *arXiv preprint arXiv:1806.03517*, 2018.

[70] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PloS one*, vol. 11, no. 4, p. e0152173, 2016.

[71] L. Basora, X. Olive, and T. Dubot, "Recent advances in anomaly detection methods applied to aviation," *Aerospace*, vol. 6, no. 11, p. 117, 2019.

[72] M. Raya, P. Papadimitratos, I. Aad, D. Jungels, and J.-P. Hubaux, "Eviction of misbehaving and faulty nodes in vehicular networks," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 8, pp. 1557–1568, 2007.

[73] D. Tian, Y. Wang, G. Lu, and G. Yu, "A vehicular ad hoc networks intrusion detection system based on busnet," in *2010 2nd International Conference on Future Computer and Communication*, vol. 1, pp. V1–225, IEEE, 2010.

[74] J. Grover, V. Laxmi, and M. S. Gaur, "Misbehavior detection based on ensemble learning in vanet," in *International Conference on Advanced Computing, Networking and Security*, pp. 602–611, Springer, 2011.

[75] N. Dutta and S. Chellappan, "A time-series clustering approach for sybil attack detection in vehicular ad hoc networks," in *Int. Conf. Adv. Veh. Syst., Technol. Appl*, pp. 21–26, 2013.

[76] H. Sedjelmaci and S. M. Senouci, "A new intrusion detection framework for vehicular networks," in *2014 IEEE International Conference on Communications (ICC)*, pp. 538–543, IEEE, 2014.

[77] L. A. Maglaras, "A novel distributed intrusion detection system for vehicular ad hoc networks," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 6, no. 4, pp. 101–106, 2015.

[78] H. Sedjelmaci and S. M. Senouci, "An accurate and efficient collaborative intrusion detection framework to secure vehicular networks," *Computers & Electrical Engineering*, vol. 43, pp. 33–47, 2015.

[79] K. M. A. Alheeti, A. Gruebler, and K. D. McDonald-Maier, "An intrusion detection system against black hole attacks on the communication network of self-driving cars," in *2015 sixth international conference on emerging security technologies (EST)*, pp. 86–91, IEEE, 2015.

[80] K. M. A. Alheeti and K. McDonald-Maier, "An intelligent intrusion detection scheme for self-driving vehicles based on magnetometer sensors," in *2016 International Conference for Students on Applied Engineering (ICSAE)*, pp. 75–78, IEEE, 2016.

[81] O. A. Wahab, A. Mourad, H. Otrok, and J. Bentahar, "Ceap: Svm-based intelligent detection model for clustered vehicular ad hoc networks," *Expert Systems with Applications*, vol. 50, pp. 40–54, 2016.

[82] M. Kim, I. Jang, S. Choo, J. Koo, and S. Pack, "Collaborative security attack detection in software-defined vehicular networks," in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 19–24, IEEE, 2017.

[83] F. A. Ghaleb, A. Zainal, M. A. Rassam, and F. Mohammed, "An effective misbehavior detection model using artificial neural network for vehicular ad hoc network applications," in *Application, Information and Network Security (AINS), 2017 IEEE Conference on*, pp. 13–18, IEEE, 2017.

[84] P. Gu, R. Khatoun, Y. Begriche, and A. Serhrouchni, "Support vector machine (svm) based sybil attack detection in vehicular networks," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2017.

[85] K. M. A. Alheeti, A. Gruebler, and K. McDonald-Maier, "Using discriminant analysis to detect intrusions in external communication for self-driving vehicles," *Digital Communications and Networks*, vol. 3, no. 3, pp. 180–187, 2017.

[86] S. Sharanya and S. Karthikeyan, "Classifying malicious nodes in vanets using support vector machines with modified fading memory," *ARPN Journal of Engineering and Applied Sciences*, 2017.

[87] H. Amirat, N. Lagraa, C. A. Kerrach, and Y. Ouinten, "Fuzzy clustering for misbehaviour detection in vanet," in *2018 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, pp. 200–204, IEEE, 2018.

[88] B. Subba, S. Biswas, and S. Karmakar, "A game theory based multi layered intrusion detection framework for vanet," *Future Generation Computer Systems*, vol. 82, pp. 12–28, 2018.

[89] P. Sharma, J. Petit, and H. Liu, "Pearson correlation analysis to detect misbehavior in vanet," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pp. 1–5, IEEE, 2018.

[90] P. K. Singh, M. K. Dash, P. Mittal, S. K. Nandi, and S. Nandi, "Misbehavior detection in c-its using deep learning approach," in *International Conference on Intelligent Systems Design and Applications*, pp. 641–652, Springer, 2018.

[91] E. Eziama, K. Tepe, A. Balador, K. S. Nwizege, and L. M. Jaimes, "Malicious node detection in vehicular ad-hoc network using machine learning and deep learning," in *2018 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, IEEE, 2018.

[92] Y. Zeng, M. Qiu, Z. Ming, and M. Liu, "Senior2local: A machine learning based intrusion detection method for vanets," in *International Conference on Smart Computing and Communication*, pp. 417–426, Springer, 2018.

[93] S. So, J. Petit, and D. Starobinski, "Physical layer plausibility checks for misbehavior detection in v2x networks," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 84–93, ACM, 2019.

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

[95] P. K. Singh, S. Gupta, R. Vashistha, S. K. Nandi, and S. Nandi, "Machine learning based approach to detect position falsification attack in vanets," in *International Conference on Security & Privacy*, pp. 166–178, Springer, 2019.

[96] X. Wang, I. Mavromatis, A. Tassi, R. Santos-Rodriguez, and R. J. Piechocki, "Location anomalies detection for connected and autonomous vehicles," in *2019 IEEE 2nd Connected and Automated Vehicles Symposium (CAVS)*, pp. 1–5, IEEE, 2019.

[97] N. Kaja, *Artificial Intelligence and Cybersecurity: Building an Automotive Cybersecurity Framework Using Machine Learning Algorithms*. PhD thesis, University of Michigan-Dearborn, 2019.

[98] Y. Zeng, M. Qiu, D. Zhu, Z. Xue, J. Xiong, and M. Liu, "Deepvcm: A deep learning based intrusion detection method in vanet," in *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pp. 288–293, IEEE, 2019.

[99] D. Kosmanos, D. Karagiannis, A. Argyriou, S. Lalis, and L. Maglaras, "Rf jamming classification using relative speed estimation in vehicular wireless networks," *arXiv preprint arXiv:1812.11886*, 2018.

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

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

[102] H.-Y. Hsu, N.-H. Cheng, and C.-W. Tsai, "A deep learning-based integrated algorithm for misbehavior detection system in vanets," in *Proceedings of the 2021 ACM International Conference on Intelligent Computing and its Emerging Applications*, pp. 53–58, 2021.

[103] B. Luo, X. Liu, and Q. Zhu, "Credibility enhanced temporal graph convolutional network based sybil attack detection on edge computing servers," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, pp. 524–531, IEEE, 2021.

[104] C. Catal, H. Gunduz, and A. Ozcan, "Malware detection based on graph attention networks for intelligent transportation systems," *Electronics*, vol. 10, no. 20, p. 2534, 2021.

[105] M. A. Elsayed and N. Zincir-Heywood, "Boostguard: Interpretable misbehavior detection in vehicular communication networks," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, IEEE, 2022.

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

[107] J. Grover, N. K. Prajapati, V. Laxmi, and M. S. Gaur, "Machine learning approach for multiple misbehavior detection in vanet," in *International Conference on Advances in Computing and Communications*, pp. 644–653, Springer, 2011.

[108] S.-Y. Wang and C.-C. Lin, "Nctuns 5.0: A network simulator for ieee 802.11 (p) and 1609 wireless vehicular network researches," in *2008 IEEE 68th Vehicular Technology Conference*, pp. 1–2, IEEE, 2008.

[109] S. R. Garner *et al.*, "Weka: The waikato environment for knowledge analysis," in *Proceedings of the New Zealand computer science research students conference*, vol. 1995, pp. 57–64, 1995.

[110] W. Li, A. Joshi, and T. Finin, "Svm-case: An svm-based context aware security framework for vehicular ad-hoc networks," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, pp. 1–5, IEEE, 2015.

[111] K. M. A. Alheeti, A. Gruebler, and K. D. McDonald-Maier, "On the detection of grey hole and rushing attacks in self-driving vehicular networks," in *2015 7th Computer Science and Electronic Engineering Conference (CEEC)*, pp. 231–236, IEEE, 2015.

[112] K. Ali Alheeti, A. Gruebler, and K. McDonald-Maier, "Intelligent intrusion detection of grey hole and rushing attacks in self-driving vehicular networks," *Computers*, vol. 5, no. 3, p. 16, 2016.

[113] K. M. A. Alheeti, R. Al-Zaidi, J. Woods, and K. McDonald-Maier, "An intrusion detection scheme for driverless vehicles based gyroscope sensor profiling," in *2017 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 448–449, IEEE, 2017.

[114] K. M. A. Alheeti and K. McDonald-Maier, "An intelligent security system for autonomous cars based on infrared sensors," in *2017 23rd International Conference on Automation and Computing (ICAC)*, pp. 1–5, IEEE, 2017.

[115] K. M. A. Alheeti, A. Gruebler, and K. D. McDonald-Maier, "An intrusion detection system against malicious attacks on the communication

network of driverless cars," in *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 916–921, IEEE, 2015.

[116] P. Gu, R. Khatoun, Y. Begriche, and A. Serhrouchni, "k-nearest neighbours classification based sybil attack detection in vehicular networks," in *2017 Third International Conference on Mobile and Secure Services (MobiSecServ)*, pp. 1–6, IEEE, 2017.

[117] USDOT, "Wyoming cv pilot basic safety message one day sample: Department of transportation - data portal."

[118] Z. Zhu, Z. Hu, W. Dai, H. Chen, and Z. Lv, "Deep learning for autonomous vehicle and pedestrian interaction safety," *Safety science*, vol. 145, p. 105479, 2022.

[119] T. Okamura, K. Sato, *et al.*, "Misbehavior detection method by time series change of vehicle position in vehicle-to-everything communication," *Journal of transportation technologies*, vol. 11, no. 02, p. 284, 2021.

[120] H. Guo, J. A. Crossman, Y. L. Murphey, and M. Coleman, "Automotive signal diagnostics using wavelets and machine learning," *IEEE transactions on vehicular technology*, vol. 49, no. 5, pp. 1650–1662, 2000.

[121] M. Realpe, B. Vintimilla, and L. Vlacic, "Sensor fault detection and diagnosis for autonomous vehicles," in *MATEC Web of Conferences*, vol. 30, p. 04003, EDP Sciences, 2015.

[122] Y. Fang, H. Min, W. Wang, Z. Xu, and X. Zhao, "A fault detection and diagnosis system for autonomous vehicles based on hybrid approaches," *IEEE Sensors Journal*, 2020.

[123] F. van Wyk, Y. Wang, A. Khojandi, and N. Masoud, "Real-time sensor anomaly detection and identification in automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[124] Y. Wang, N. Masoud, and A. Khojandi, "Real-time sensor anomaly detection and recovery in connected automated vehicle sensors," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[125] K. N. Tripathi, A. M. Yadav, and S. Sharma, "Fuzzy and deep belief network based malicious vehicle identification and trust recommendation framework in vanets," *Wireless Personal Communications*, pp. 1–30, 2022.

[126] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[127] D. Zha, K.-H. Lai, M. Wan, and X. Hu, "Meta-aad: Active anomaly detection with deep reinforcement learning," *arXiv preprint arXiv:2009.07415*, 2020.

[128] S. Feng and S. Haykin, "Anti-jamming v2v communication in an integrated uav-cav network with hybrid attackers," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[129] I. Rasheed, F. Hu, and L. Zhang, "Deep reinforcement learning approach for autonomous vehicle systems for maintaining security and safety using lstm-gan," *Vehicular Communications*, p. 100266, 2020.

[130] A. Ferdowsi, U. Challita, W. Saad, and N. B. Mandayam, "Robust deep reinforcement learning for security and safety in autonomous vehicle systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 307–312, IEEE, 2018.

[131] A. Ferdowsi, S. Ali, W. Saad, and N. B. Mandayam, "Cyber-physical security and safety of autonomous connected vehicles: Optimal control meets multi-armed bandit learning," *IEEE Transactions on Communications*, vol. 67, no. 10, pp. 7228–7244, 2019.

[132] R. Sedar, C. Kalalas, J. Alonso-Zarate, and F. Vázquez-Gallego, "Multi-domain denial-of-service attacks in internet-of-vehicles: Vulnerability insights and detection performance," in *2022 IEEE International Conference on Network Softwarization (NetSoft)*, 2022.

[133] L. Zhao, H. Chai, Y. Han, K. Yu, and S. Mumtaz, "A collaborative v2x data correction method for road safety," *IEEE Transactions on Reliability*, 2022.

[134] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[135] C. Huang, Y. Wu, Y. Zuo, K. Pei, and G. Min, "Towards experienced anomaly detector through reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[136] R. Sedar, C. Kalalas, F. Vázquez-Gallego, and J. Alonso-Zarate, "Reinforcement learning-based misbehaviour detection in v2x scenarios," in *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, pp. 109–111, IEEE, 2021.

[137] X. Xu, "Sequential anomaly detection based on temporal-difference learning: Principles, models and case studies," *Applied Soft Computing*, vol. 10, no. 3, pp. 859–867, 2010.

[138] G. Pang, A. v. d. Hengel, C. Shen, and L. Cao, "Deep reinforcement learning for unknown anomaly detection," *arXiv preprint arXiv:2009.06847*, 2020.

[139] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing (TMC)*, vol. 10, pp. 3–15, January 2011.

[140] A. Varga, "Omnet++," in *Modeling and tools for network simulation*, pp. 35–59, Springer, 2010.

[141] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018.

[142] J. Kamel, M. R. Ansari, J. Petit, A. Kaiser, I. B. Jemaa, and P. Urien, "Simulation framework for misbehavior detection in vehicular networks," *IEEE Transactions on Vehicular Technology*, 2020.

[143] K. Emara, "Poster: Prext: Privacy extension for veins vanet simulator," in *2016 IEEE Vehicular Networking Conference (VNC)*, pp. 1–2, IEEE, 2016.

[144] M. Rondinone, J. Maneros, D. Krajzewicz, R. Bauza, P. Cataldi, F. Hrizi, J. Gozalvez, V. Kumar, M. Röckl, L. Lin, *et al.*, "itetris: a modular simulation platform for the large scale evaluation of cooperative its applications," *Simulation Modelling Practice and Theory*, vol. 34, pp. 99–125, 2013.

[145] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*, pp. 15–34, Springer, 2010.

[146] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[147] T.-K. Lee, T.-W. Wang, W.-X. Wu, Y.-C. Kuo, S.-H. Huang, G.-S. Wang, C.-Y. Lin, J.-J. Chen, and Y.-C. Tseng, "Building a v2x simulation framework for future autonomous driving," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–6, IEEE, 2019.

[148] MATLAB, *9.7.0.1190202 (R2019b)*. Natick, Massachusetts: The MathWorks Inc., 2018.

[149] M. Amoozadeh, B. Ching, C.-N. Chuah, D. Ghosal, and H. M. Zhang, "Ventos: Vehicular network open simulator with hardware-in-the-loop support," *Procedia Computer Science*, vol. 151, pp. 61–68, 2019.

[150] R. Riebl, H.-J. Günther, C. Facchi, and L. Wolf, "Artery: Extending veins for vanet applications," in *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pp. 450–456, IEEE, 2015.

[151] R. Riebl, C. Obermaier, S. Neumeier, and C. Facchi, "Vanetza: Boosting research on inter-vehicle communication," *Proceedings of the 5th GI/ITG KuVS Fachgespräch Inter-Vehicle Communication (FG-IVC 2017)*, pp. 37–40, 2017.

[152] U. D. of Transportation Intelligent Transportation Systems Joint Program Office, "Next generation simulation (ngsim) vehicle trajectories and supporting data: Department of transportation - data portal," 8 2018.

[153] USDOT, "Safety pilot model deployment data: Open data."

[154] A. Tassi, I. Mavromatis, and R. J. R. Piechocki, "A dataset of full-stack its-g5 dsrc communications over licensed and unlicensed bands using a large-scale urban testbed," *Data in brief*, vol. 25, p. 104368, 2019.

[155] J. Kamel, M. Wolf, R. W. van Der Heijden, A. Kaiser, P. Urien, and F. Kargl, "Veremi extension: A dataset for comparable evaluation of misbehavior detection in vanets," in *IEEE International Conference on Communications (ICC)*, 2020.

[156] "Nist/sematech e-handbook of statistical methods."

[157] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods*. Springer-Verlag, 1987.

[158] J. Kamel, A. Kaiser, I. Jemaa, P. Cincilla, and P. Urien, "Catch: A confidence range tolerant misbehavior detection approach," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, 2019.

[159] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

[160] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 535–548, Springer, 2002.

[161] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Loop: local outlier probabilities," in *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 1649–1652, 2009.

[162] Z. Ning, F. Xia, N. Ullah, X. Kong, and X. Hu, "Vehicular social networks: Enabling smart mobility," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 16–55, 2017.

[163] K. Gu, X. Dong, and W. Jia, "Malicious node detection scheme based on correlation of data and network topology in fog computing-based vanets," *IEEE Transactions on Cloud Computing*, 2020.

[164] J. Ning, J. Wang, J. Liu, and N. Kato, "Attacker identification and intrusion detection for in-vehicle networks," *IEEE Communications Letters*, vol. 23, no. 11, pp. 1927–1930, 2019.

[165] W. Jin, A. K. Tung, J. Han, and W. Wang, "Ranking outliers using symmetric neighborhood relationship," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 577–593, Springer, 2006.

[166] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos, "Loci: Fast outlier detection using the local correlation integral," in *Proceedings 19th international conference on data engineering (Cat. No. 03CH37405)*, pp. 315–326, IEEE, 2003.

[167] Q. Ricard and P. Owezarski, "Ontology based anomaly detection for cellular vehicular communications," in *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, 2020.

[168] M. T. Garip, *Design and Mitigation of Vehicular Botnets in Vehicular Ad Hoc Networks*. PhD thesis, UCLA, 2019.

[169] Z. Iqbal and M. I. Khan, "Automatic incident detection in smart city using multiple traffic flow parameters via v2x communication," *International journal of distributed sensor networks*, vol. 14, no. 11, p. 1550147718815845, 2018.

[170] S. L. D. Dhulipala, "Detection of injection attacks on in-vehicle network using data analytics," *Master's thesis, Electr. Comput. Eng., Iowa State University, Ames, IA.*, 2018.

[171] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[172] G. Boquet, A. Morell, J. Serrano, and J. L. Vicario, "A variational autoencoder solution for road traffic forecasting systems: Missing data imputation, dimension reduction, model selection and anomaly detection," *Transportation Research Part C: Emerging Technologies*, vol. 115, p. 102622, 2020.

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

[174] A. Dairi, F. Harrou, Y. Sun, and M. Senouci, "Obstacle detection for intelligent transportation systems using deep stacked autoencoder and k-nearest neighbor scheme," *IEEE Sensors Journal*, vol. 18, no. 12, pp. 5122–5132, 2018.

[175] T. Denouden, R. Salay, K. Czarnecki, V. Abdelzad, B. Phan, and S. Vernekar, "Improving reconstruction autoencoder out-of-distribution detection with mahalanobis distance," *arXiv preprint arXiv:1812.02765*, 2018.

[176] C. C. Schappin, "Intrusion detection on the automotive can bus," *Technische Universiteit Eindhoven*, 2017.

[177] M. A. Javed, M. Z. Khan, U. Zafar, M. F. Siddiqui, R. Badar, B. M. Lee, and F. Ahmad, "Odpv: An efficient protocol to mitigate data integrity attacks in intelligent transport systems," *IEEE Access*, vol. 8, pp. 114733–114740, 2020.

[178] M. Matousek, M. Yassin, R. van der Heijden, F. Kargl, *et al.*, "Robust detection of anomalous driving behavior," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pp. 1–5, IEEE, 2018.

[179] J. Kim and C. D. Scott, "Robust kernel density estimation," *Journal of Machine Learning Research*, vol. 13, no. Sep, pp. 2529–2565, 2012.

[180] S. Concas, M. Kamrani, *et al.*, "Development of a real-time roadway debris hazard spotting tool using connected vehicle data to enhance roadway safety and system efficiency," *Center for Transportation, Equity, Decisions and Dollars (CTEDD)(UTC)*, 2019.

[181] S. Ucar, C. Patnayak, P. Oza, B. Hoh, and K. Oguchi, "Management of anomalous driving behavior," in *2019 IEEE Vehicular Networking Conference (VNC)*, pp. 1–4, IEEE, 2019.

[182] M. Chowdhury, M. Islam, and Z. Khan, "Security of connected and automated vehicles," *arXiv preprint arXiv:2012.13464*, 2020.

[183] J. Kamel, *Misbehavior detection for cooperative intelligent transport systems (C-ITS)*. PhD thesis, 2020. Thèse de doctorat dirigée par Urien, Pascal et Ben Jemaa, Inès Informatique, données, IA Institut polytechnique de Paris 2020.

[184] A. Vinel, V. Vishnevsky, and Y. Koucheryavy, "A simple analytical model for the periodic broadcasting in vehicular ad-hoc networks," in *2008 IEEE Globecom Workshops*, pp. 1–5, IEEE, 2008.

[185] M. Latapy, T. Viard, and C. Magnien, "Stream graphs and link streams for the modeling of interactions over time," *Social Network Analysis and Mining*, vol. 8, no. 1, pp. 1–29, 2018.

[186] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[187] K. P. Murphy, "A survey of pomdp solution techniques," *environment*, vol. 2, no. 10, 2000.

[188] M. L. Littman, "Memoryless policies: Theoretical limitations and practical results," in *From Animals to Animats 3: Proceedings of the third international conference on simulation of adaptive behavior*, vol. 3, p. 238, MIT Press Cambridge, MA, USA, 1994.

[189] L.-J. Lin and T. M. Mitchell, *Memory approaches to reinforcement learning in non-Markovian domains*. Citeseer, 1992.

[190] S. D. Whitehead and L.-J. Lin, "Reinforcement learning of non-markov decision processes," *Artificial intelligence*, vol. 73, no. 1-2, pp. 271–306, 1995.

[191] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, pp. 2825–2830, 2011.

[192] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *Machine Learning Proceedings 1995*, pp. 30–37, Elsevier, 1995.

[193] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep q-learning," in *Learning for Dynamics and Control*, pp. 486–489, PMLR, 2020.

[194] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[195] N. Capuano, G. Fenza, V. Loia, and C. Stanzione, "Explainable artificial intelligence in cybersecurity: A survey," *IEEE Access*, 2022.

[196] B. Mahbooba, M. Timilsina, R. Sahal, and M. Serrano, "Explainable artificial intelligence (xai) to enhance trust management in intrusion detection systems using decision tree model," *Complexity*, 2021.

[197] S. Neupane, J. Ables, W. Anderson, S. Mittal, S. Rahimi, I. Banicescu, and M. Seale, "Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities," *IEEE Access*, 2022.

[198] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in iot security: Current solutions and future challenges," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2020.

[199] A. Qayyum, M. Usama, J. Qadir, and A. Al-Fuqaha, "Securing connected autonomous vehicles: Challenges posed by adversarial machine learning and the way forward," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 998–1026, 2020.

[200] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 27–38, ACM, 2017.

[201] D. Li and Q. Li, "Adversarial deep ensemble: Evasion attacks and defenses for malware detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3886–3900, 2020.

ECOLE
DOCTORALE

Titre : Détection de comportements anormaux multicouches pour un véhicule connecté et autonome.

Mots clés : Véhicules connectés et autonomes, apprentissage automatique, détection de comportements anormaux, apprentissage par renforcement, cybersécurité, détection d'anomalies.

Résumé : De nos jours, le domaine des véhicules, y compris les véhicules autonomes et les villes intelligentes, est en train de se développer pour moderniser la vie humaine dans une ville où tout est connecté : les humains grâce à un smartphone, les infrastructures, les voitures et les motos. Dans un tel système, les informations sont échangées, traitées et utilisées pour le bon fonctionnement de toute entité dans le système. Cependant, la dépendance accrue à la communication véhiculaire en fait également une cible d'attaques de sécurité, ce qui pourrait entraîner la diffusion d'informations fausses ou manipulées provenant de sources malveillantes. Cela pourrait constituer une menace pour le bon fonctionnement du système et pourrait potentiellement entraîner des accidents. Pour résoudre ce problème, il est crucial de valider et de vérifier la communication pour garantir son exactitude et prévenir les attaques malveillantes. Nous avons pour objectif de formuler la détection de comportements anormaux pour les véhicules connectés et autonomes de niveau 4/5 d'automatisation. Dans notre thèse, nous proposons une architecture multicouche pour la détection de comportements anormaux avec un apprentissage automatique pour sécuriser la communication, les capteurs et les composants internes des véhicules connectés et autonomes. Cette architecture nous permet de proposer un nouveau modèle de réseau de neurones basé sur l'apprentissage par renforcement pour la détection de comportements anormaux. Nous avons montré dans un environnement simulé, à travers une évaluation, que notre modèle est capable de détecter des comportements anormaux nouveaux et fonctionne mieux que les algorithmes de l'état de l'art. De plus, nous abordons la fuite de données dans les communications véhiculaires et proposons une méthode de validation croisée pour éviter cette fuite dans les applications d'apprentissage automatique. Lors de l'évaluation des résultats de notre thèse, nous avons développé une simulation pour les environnements de véhicules, capable d'injecter et de détecter des comportements anormaux. Enfin, les idées développées dans cette thèse ont donné lieu à plusieurs publications.

Title : Multi layered Misbehavior Detection for a connected and autonomous vehicle

Keywords : Connected and autonomous vehicles, machine learning, misbehavior detection, reinforcement learning, cybersecurity, anomaly detection.

Abstract : In recent years, the vehicular field has undergone significant advancements with the development of autonomous vehicles and smart cities. These advancements have brought about a modernization of human life, where everything is interconnected - from individuals through smartphones to infrastructure, cars, and motorcycles. In such a system, information is exchanged and processed, and used to ensure the proper functioning of all entities. However, the increased reliance on V2X communication also makes it a target for security attacks, which could lead to the dissemination of false or manipulated information from malicious sources. This could pose a threat to the proper functioning of the system and can potentially result in accidents. To address this problem, it is crucial to validate and verify the communication to ensure its accuracy and prevent malicious attacks. We aim to formulate misbehavior and misbehavior detection for connected and autonomous vehicles of level 4/5 automation. In our thesis, we propose a multi-layered architecture for the detection of abnormal behaviors with automatic learning to secure the connected and autonomous vehicles' communications, sensors, and internal components. The architecture allows us to propose a novel reinforcement learning based neural architecture for the detection of misbehaviors where we showed in a simulated environment, through evaluation, that the model is capable of detecting novel misbehaviors and performs better than current state-of-the-art algorithms. Furthermore, we tackle data leakage in V2X data and propose a cross-validation method to avoid said leakage in machine learning applications. We also developed a simulation for vehicular environments capable of injecting and detecting misbehaviors for the evaluation of our thesis results. The ideas developed in this research have resulted in several publications and have the potential to significantly enhance the security and reliability of vehicular systems.

Institut Polytechnique de Paris
91120 Palaiseau, France