Misbehavior detection for cooperative intelligent transport systems (C-ITS)

Joseph Kamel

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2020IPPAT024

Thèse de doctorat

Misbehavior Detection for Cooperative Intelligent Transport Systems (C-ITS)

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

École doctorale n°626 l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat: Informatique, données et intelligence artificielle

Thèse présentée et soutenue à Palaiseau, le 21/07/2020, par

JOSEPH KAMEL

Composition du Jury :

Sidi Mohammed Senouci Directeur du laboratoire DRIVE, ISAT	Président du jury
Isabelle Chrisment Professeur à TELECOM Nancy, Université de Lorraine	Rapporteur
Oyunchimeg Shagdar Chef d'équipe de recherche REVECOM, VEDECOM	Examinateur
Frank Kargl Directeur de l'Institut des systèmes distribués, Université d'Ulm	Examinateur
Jonathan Petit Directeur de l'ingénierie, Qualcomm Technologies Inc.	Examinateur
Pascal Urien Professor, INFRES, Télécom Paris	Directeur de thèse
Ines Ben Jemaa Ingénieur de recherche, SCA, IRT SystemX	Co-encadrement
Arnaud Kaiser Chef de Projet, SCA, IRT SystemX	Invité
Brigitte Lonc Chef de Projet, Innovation Multimédia et Connectivité, Renault	Invité
Pierpaolo Cincilla Expert, Processus de cybersécurité et conformité, Renault	Invité

Cybercriminal activity is one of the biggest challenges that humanity will face in the next two decades.

- Steve Morgan, Editor-in-Chief, Cybersecurity Ventures

Acknowledgements

I would like to take the opportunity to thank all those who supported me during this experience.

First, I would like to thank my PhD director, Professor Pascal Urien, for his consistent guidance and valuable discussions. I would like to thank my PhD supervisors, Doctor Arnaud Kaiser and Doctor Ines Ben Jemaa, for sharing their invaluable knowledge, experience and advice. My director and supervisors were the key to the successful completion of my thesis. I would like to thank Professor Frank Kargl for hosting my internship in Ulm, it broadened my horizons on various scientific subjects and gave me an alternative view of my thesis. I would like to thank Professor Jonathan Petit for his help in one of the most valuable and fruitful collaborations. I would like to thank Doctor Pierpaolo Cincilla for initially encouraging me to pursue the scientific domain and take on the challenge of a PhD.

I would like to extend my gratitude to the reviewers, Professor Sidi Mohammed Senouci and Professor Isabelle Chrisment for accepting to evaluate my thesis. My earnest thanks are also due to Professor Aline Viana, Doctor Brigitte Lonc and Professor Oyunchimeg Shagdar for accepting to be a member of the jury.

Thanks to all of my colleagues in IRT SystemX, especially Loic for his valuable help and technical discussions. Thanks to all my colleagues in the SCA team, especially Farah, Nabil, Hafeda, Reda, Francesca, Hassane, Marios and Michel for creating a great work atmosphere.

I would also like to thank my friends Marie, Oceane and Alex for their support and all our happy get-togethers and for uplifting my spirit during the challenging times faced during this thesis. I am also grateful to my cousins Michel, Rima, Nour and Sami, and to my friends, Reza, Imane, Alexis, Nathalie and Alex for their priceless friendship and precious support, and for continuously giving me the courage to make this thesis better.

Last but definitely not least, I am forever indebted to my awesomely supportive parents and my wonderful sister Sarah for inspiring and guiding me throughout the course of my life, and hence this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Road transportation is the most used mode of travel in the world. However, the resulting road accidents remain a major issue causing huge economic and human losses. According to the National Highway Traffic Safety Administration (NHTSA) and the Directorate-General for Mobility and Transport (DGMT), more than three million people are injured each year due to traffic accidents in the United States and the European Union [1] [2]. To combat this problem, manufacturers are adding more safety features in their vehicles.

Modern safety features are often technologically advanced embedded electronic systems. For instance, blind spot monitor increases the drivers' awareness by issuing a warning when a vehicle approaches the driver's sides or rear. An area notoriously tricky to check using traditional side mirrors. Lane keep assist systems prevent oversteering and helps keep the vehicle centered. It works by alerting the driver when needed and could even take automatic measures to keep the vehicle in the lane. Collision avoidance systems aims to prevent or reduce the severity of a collision. These systems rely on sensors like radars, lasers and cameras to detect dangerous situations in the surrounding environment. They warn the driver of incoming threats or act automatically on the vehicle with actions like emergency braking or swerving.

The outcomes of these vehicular safety systems, from timely alerts to swift measures, are often life saving for the driver and passengers of a vehicle. To quantify their added value, Haus et al. [3] estimated the benefits of automated emergency braking in the United States. Their results show a decreased fatality risk between 84% and 87% and injury risk between 83% and 87%. Given their great advantage on high-stakes use cases, that involve human lives, safety systems are becoming less of a commodity and more of a necessity. However, because of their critical use-case, the cyber-security mechanisms that ensures the functionality of these safety systems need to be on par with the high risk in case of failure. After all, these safety systems could assume control of the vehicle. Sensors could fail and embedded electronics of a vehicle could malfunction. Moreover, malicious actors could target safety systems with cyber-attacks which would have catastrophic results on unsecured vehicles.

Koscher et al. [4] were the first to demonstrate a cyber-attack on the embedded system of commercial vehicles.

They use the integrated controller area network or CAN bus to modify the vehicle's displays, control the breaks or even stop the engine completely. Checkoway et al. [5], working with the same research group, were able to prove that some of these cyber-attacks could be done over-the-air without the need for a wired connection. Miller et al. [6] demonstrated to journalists some of these cyber-attacks on a 2014 Jeep Cherokee. A story that went viral on social and traditional media outlets. The journalists were surprised and frightened when the attackers were able to remotely kill the engine while the vehicle was cruising on the highway.

In this thesis, our goal is to ensure the cyber-security of a new vehicular safety system. Cooperative Intelligent Transport Systems (ITS) (C–ITS) is a new technology that aims to reduce traffic accidents and improve road safety in general. The technology is based on inter-vehicle communications in the form of safety messages. These messages can contain information about the vehicle like the position and the current speed or various warnings about the current traffic condition. This exchanged information could prove critical for the vehicle to insure the safety of the passengers or surrounding pedestrians. For instance, safety applications can use the received data to warn the driver about upcoming road works or about the need for an imminent emergency brake. Furthermore, C–ITS could be used to improve the overall traffic conditions through applications like the cooperative adaptive cruise control. C–ITS could even reduce traffic jams and contribute to shockwave damping on the highway by providing the driver with speed advice.

Safety mechanisms in C–ITS are cooperative in nature. Therefore, it is essential to establish interoperability between different vehicle manufacturers. For instance, vehicles from different brand should use the same communication technologies and communication formats to ensure the largest possible benefits. Likewise, safety applications like cooperative adaptive cruise control require a common protocol to function properly. Standardization bodies could provide a common solution to this problem. The European Telecommunications Standards Institute (ETSI) in Europe and the Institute of Electrical and Electronics Engineers (IEEE) in the US are pushing unified standards for common communication protocols and formats. These formats include the type of transmitted messages and their general modes of operation. Currently, the ETSI and IEEE provide different standards in Europe and the US. Even though the formats are different, interoperability between the two platform is still possible in the future.

For instance, the ETSI published the Cooperative Awareness Message (CAM) [7] to enable vehicles to share kinematic information and the Decentralized Environmental Notification Message (DENM) [8] to share warnings. Whilst the IEEE published the Basic Safety Message (BSM) [9] providing similar functionalities to users. Both these systems are currently being tested through pre-deployment projects in Europe and the US. As a result, users could start to benefit soon from C–ITS functionalities.

1.1 Problem statement

The reliable operation of the C–ITS-based safety applications require secure Vehicle–to–Everything (V2X) messages. To this end, the IEEE and the ETSI created and standardized the Vehicular Public Key Infrastructure (PKI) [10] [11]. The PKI issues digital certificates to the different actors of the C–ITS system, called ITS Stations (ITS–Ss). The certificates are used by the ITS–S to sign each transmitted message thus ensuring its authenticity, integrity and non-repudiation. Furthermore, the adoption of such systems requires, especially in our current global environment, the protection of the users' privacy. For this reason, the ITS–S are allowed to regularly change their signatures using temporary digital certificates. These temporary certificates are called pseudonyms and they would allow the user to avoid being tracked [12].

Nevertheless, despite the protections provided by the PKI infrastructure, security challenges still exist in the current C–ITS system. In particular, digital signatures do not ensure the accuracy and validity of a message. For instance, a malicious vehicle with a valid certificate could send inaccurate or false data over the V2X network. Consequently, a misbehavior detection mechanism is needed to protect the system and mitigate the effects of these malicious or otherwise faulty ITS–Ss. Although Misbehavior Detection is a well-researched topic and despite the large number of published studies, many research challenges still exist in the misbehavior detection domain. C–ITS is an ever evolving system with different developing components both on the technical and regulatory side. In this thesis, our goal is to identify the remaining challenges that need to be addressed and propose solutions to the various components of the misbehavior detection system.

1.2 Contributions

In this section we cite and summarize all the contributions of this thesis. We start with a state of the art evaluation section. Then the following contributions are divided into sections corresponding to the misbehavior detection process: local, reporting and global detection. Finally, we also add a general section for contributions spanning the entire detection process.

1.2.1 State of the art evaluation

Feasibility Study [13]: We started our work with a feasibility study of the different misbehavior detection mechanisms in C–ITS. In this study, we extract and classify the mechanisms used in the published works on misbehavior detection. We then analyze the feasibility of such mechanisms in the current European ETSI and American IEEE ecosystems. Then, we discuss the remaining challenges that need to be addressed by the community. Some of these challenges were addressed in the course of the thesis.

1.2.2 Contributions to the local misbehavior detection

CaTch Detectors [14]: This contribution to the local detection component concerns the plausibility and consistency checks. We notice that both the ETSI CAM and the IEEE BSM messages integrate a field called *confidence range* for each mobility parameter. This field is included based on the fact that sensor measurements could be inaccurate due to physical limitations or environmental characteristics. However, this information was not taken into consideration during the misbehavior detection checks computation. This may result in a high number of false positives in a realistic scenario. Therefore, we created CaTch (Confidence range Tolerant misbehavior detection approach), a misbehavior detection library which takes into consideration the *confidence range*. We implement the CaTch detectors in a simulator and show that taking into consideration the sensors' inaccuracy during the checks computation process increases the detection quality.

Fusion Applications Comparative [15]: This contribution to the local detection component concerns the fusion application of the plausibility checks. We notice that many fusion application proposals already exist in the literature. However, the various published detection results are often difficult to compare since it's done on different data and with different scenarios. To this end, we re-implement the complete detection process along with the different detection applications in an extension of the Vehicles in Network Simulation (VEINS) simulator [16]. Our comparative results show a trade-off between the accuracy of the detection mechanisms and the calculation latency.

VeReMi Dataset Extension [17]: This contribution concerns the complete local detection process. A large number of studies are aimed at providing local misbehavior detection solutions. However, the results of these studies are still difficult to compare, reproduce and validate. This is due to the lack of a common reference dataset. For this reason, the original Vehicular Reference Misbehavior Dataset (VeReMi) was created. The goal of VeReMi is to allow for comparable evaluation of misbehavior detection in C–ITS. It is the first public misbehavior detection dataset allowing anyone to reproduce and compare different results. VeReMi is used in a number of studies and is currently the only dataset in its field. We extend the dataset by adding realistic a sensor error model, a new set of attacks and larger number of data points. Additionally, we provide benchmark detection metrics using a set of local detectors and a simple misbehavior detection mechanism.

1.2.3 Contributions to the misbehavior reporting

Misbehavior Reporting Protocol [18]: Despite the consensus on a need for a unified communication protocol between the ITS–S and the Misbehavior Authority (MA), a misbehavior reporting protocol is still missing. In this work, we propose a Misbehavior Report (MBR) message format and identify the relevant information that the report should include. For each detected misbehavior type we propose the corresponding proofs to be included in the

report as well as an assigned confidence level. The latter is an indication that enables to differentiate non-forged proofs and self-forged proofs (i.e. if a proof could be forged by the reporting entity).

Misbehavior Reports Dataset [19]: Following our misbehavior reports proposal, we need to propose a simple method for researchers to use this protocol within their systems. To this end, we publish a dataset of misbehavior reports derived from the local embedded detection of misbehaving entities. This dataset can be used to further develop and evaluate the MA detection component. The sets include different road topology, varying attacker rates and attack scenarios.

1.2.4 Contributions to the global misbehavior detection

Machine Learning Based MA [20]: The MA is responsible for protecting the ITS system by mitigating the effects of an attack. Accordingly, the MA must first identify the type of attack. We believe that the MA will benefit from using Artificial Intelligence (AI) solutions such as Machine Learning (ML) to perform this task. To this end, we implement the complete misbehavior detection process in an extension of VEINS simulator [16]. Then we evaluate different ML approaches for the MA. Our results show that the use of ML enables the MA to precisely classify the reported ITS–S and identify the different types of misbehavior.

Global Sybil Attack Detection [21]: We believe that an attacker performing a Sybil attack (i.e. using multiple fake identities) is better detected at the global level. Therefore, we propose a misbehavior detection process at the MA, which is able to identify and detect both Sybil and other types of attacks. It is based on advanced ML algorithms. In addition, we evaluate our solution by integrating it in both ETSI and IEEE C–ITS standard architecture.

Release of an Open Source Framework for Misbehavior Detection (F^2MD) [22]: In order to test all our previously proposed solutions, we implemented F^2MD. It is a complete misbehavior detection solution that includes a real time simulation of the whole misbehavior detection process from local to global detection. F^2MD is a VEINS module. VEINS is a well-known an open source framework for running vehicular network simulations [23]. VEINS is based on the Objective Modular Network Testbed in C++ (OMNeT++) [24] for network simulation and the Simulation of Urban MObility (SUMO) [25] for road traffic simulation. F^2MD is also open source and could be found and used freely on GitHub [26].

Transfer to ETSI standardization body [27]: Because of its cooperative nature, interoperability is essential for the correct functionality of the C–ITS systems. Therefore, the ETSI and IEEE standardization bodies have already standardized large functions of the system. Misbehavior detection is a security layer for C–ITS. As a result, some standard functionalities are required. Specifically, the operation of the cloud component, the connection between

the MA and the PKI and the misbehavior reporting process. To this end, throughout the course of this thesis, we participated in the ETSI ITS-W5 and contributed to the TR-103-460: Pre-standardization Study on Misbehavior Detection.

1.3 Work environment

This research work has been carried out in the Secure Cooperative Autonomous systems (SCA) Project in the Technological Research Institute SystemX. SCA is research project with both industrial and academic partners. The industrial partners include vehicle manufacturers, automotive suppliers and cybersecurity services providers: Atos (IDnomic), Oppida, Groupe PSA, Groupe Renault, Transdev, Trialog, Valeo and YoGoKo. The academic partner is the Institut Mines-Télécom, specifically Télécom Paris, amongst the top public higher education and research engineering establishments in France.

In addition to the work in SCA, this thesis included a research visiting hosted by the Institute of Distributed Systems at Ulm University. The visiting was under the supervision of Professor Frank Kargl and focused mainly on misbehavior detection in Cooperative Adaptive Cruise Control (CACC). However, with the help of their privacy experts, we were additionally able to analyze the privacy aspect of the CAMs within the current ETSI C–ITS system. Finally, we also jointly published the extension to the VeReMi dataset.

Chapter 2

State of The Art

This chapter includes a state of the art description of the various systems related to misbehavior detection, that are referred to in this thesis. First, we present the C–ITS model and current security architecture. Next, we discuss some of the current C–ITS pre-deployment projects. Then, we detail the insider attacker model that is common in this domain. After that, we cite and describe the published related works. Last, we end this section with a feasibility study on the different misbehavior detection mechanisms in C–ITS.

2.1 C-ITS model

Figure 2.1 shows the current C–ITS system structure. In C–ITS vehicles should exchange safety messages with a cost-efficient and low-latency communication system. However, this communication system is not yet agreed upon and two possible technologies currently exist:

- **IEEE 802.11p** is an amendment to the IEEE 802.11 (the common Wi-Fi standard) that enables wireless communications in a vehicular environment [28]. The IEEE 802.11p is the base technology for the European ITS-G5 and the American WAVE. ITS-G5 stands for Intelligent Transport Systems operating at the 5.9 GHz frequency band. It is specified in the ETSI EN 302 571 and the ITS Directive 2010/40/EU [29]. WAVE stands for Wireless Access in Vehicular Environments. It is specified in IEEE 1609 family of standards [30]. WAVE enables the Dedicated Short-Range Communications (DSRC) standards and its family of safety applications. Both ITS-G5 and WAVE have been tested in pre-deployment projects in the US and Europe.
- **3GPP Release 14** introduced cellular LTE support for V2X services [31]. In this standard, the 3rd Generation Partnership Project (3GPP) also introduced an optimized LTE Direct technology for automotive applications. It is commonly referred to as Cellular-V2X or C-V2X. Additionally, 3GPP Release 15 introduces 5G support for C-V2X. In C-V2X, two modes of operations are available: (1) in range communication or mode 3 where the

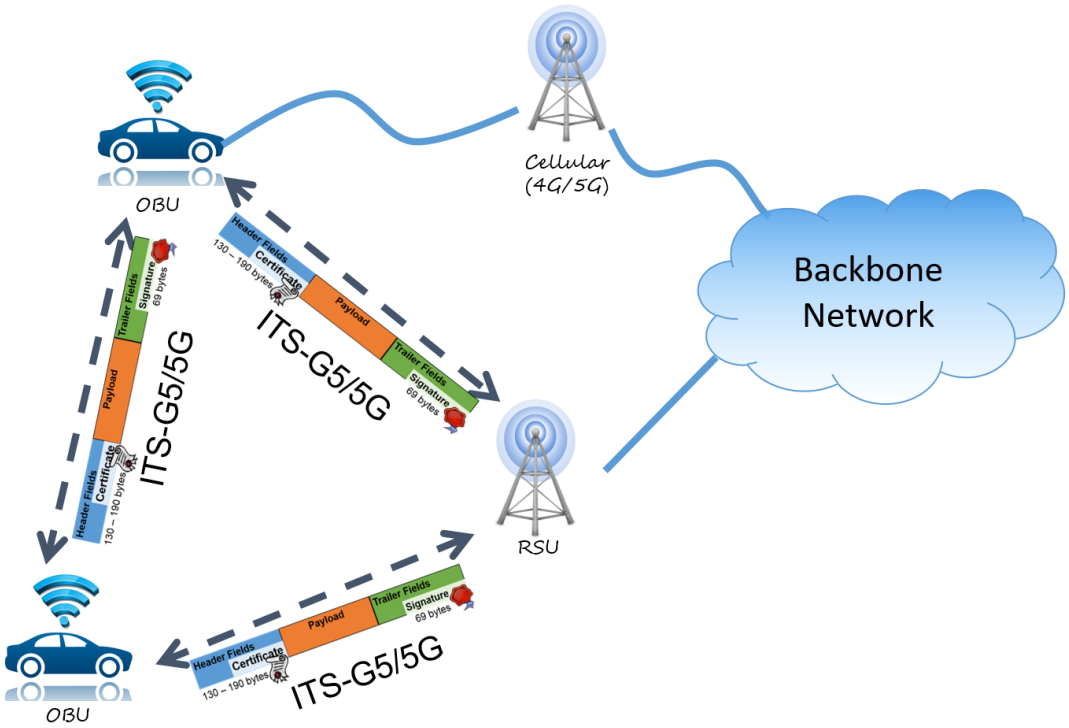


Figure 2.1: C–ITS model

transmission is assisted by a base station (eNodeB) on the LTE-Uu interface, (2) out of range communication or mode 4 where direct Device to Device (D2D) transmission occurs without edge or back-end interference over the PC5 or Sidelink interface. C-V2X is also being tested in various pre-deployment projects.

In April 2019, the European Commission proposed the Delegated Act on C-ITS that would push to the adoption of ITS-G5 in the European member states. However, the European Parliament voted against this proposal, adopting a technology neutral approach instead. Regardless of the communication technology however, the C–ITS safety protocols have been well developed and standards have already been released.

In Europe, the ETSI ITS Working Groups proposed and published several safety messages formats. Cooperative Awareness Message (CAM) is a periodically transmitted beacon message [7]. The CAM contains status and attribute information about the ITS–S. The status includes information like the position, motion state, time and activated systems. The attribute includes information like the dimensions, vehicle type and role in the road traffic. Decentralized Environmental Notification Message (DENM) is another type of message that contains information and warnings related to the current traffic and road conditions. This information is then presented to the driver of the receiving ITS–S who is able to react and potentially avoid dangerous situations. In the US, the IEEE also published safety messages formats. The Basic Safety Message (BSM) is the IEEE periodic beacon equivalent to the CAM message. However, unlike the ETSI system, warnings could also be appended to BSMs. Therefore, the BSM also replicates the functionalities of the DENMs without the need for a different specific message.

Similarly to the exchange messages formats, the standardization bodies are also working on proposals for

several safety applications. Even though most safety applications are currently still in the pre-standardization phase and not readily available, a lot of progress has already been made. As a result, some road-safety and traffic-efficiency applications are well developed. Here are some examples:

- **Road safety applications**:
  - *Collision Avoidance Systems [32] [33] [34]:* These systems are considered primary road safety applications. The C–ITS CAM and DENM services are utilized to broadcast specific warnings to neighboring ITS–Ss to prevent collisions. These warning include Intersection Collision Risk Warning (ICRW), Road Hazard Signaling (RHS) and Longitudinal Collision Risk Warning (LCRW).
  - *Collective Perception Service (CPS) [35] [36]:* CPS enables different ITS–Ss to share data of their surrounding environment. This data is acquired by physical sensors such as radars, lidars and cameras. CPS also defines a specific type of message called the Collective Perception Message (CPM). CPMs include specific containers that allows a ITS–S to efficiently encode perceived objects that could be quickly and securely transmitted and decoded by neighboring ITS–Ss.
- **Traffic efficiency applications**:
  - *Cooperative Adaptive Cruise Control (CACC) [37]:* CACC is an extension of the in-vehicle Adaptive Cruise Control (ACC) system. CACC uses the Vehicle–to–Vehicle communication (V2V), specifically a CAM container, to broadcast specific information such as the target speed, acceleration control and breaking capacity. This information allows for safer and more efficient cruising with a shorter time gap with respect to the preceding vehicle.
  - *Platooning [38]:* Platooning is a similar application to CACC with more advanced functionalities. Specifically, platooning considers lateral and longitudinal control of the vehicle, whereas only longitudinal control is considered with CACC.
  - *Electronic Payment Applications [39]:* Electronic payment is a C–ITS application that allows payment over V2X messages. This application enables seamless Electronic Toll Collection (ETC) for highways and Electronic Fee Collection (EFC) for parking spaces or access to city centers.

2.2 Pre-deployment projects

Various V2X pre-deployment projects are planned, ongoing or have been completed in the United States and Europe. Pre-deployment projects for connected vehicles started as early as 2012. In this section, we discuss some of these projects.

Safety Pilot Model Deployment (SPMD) - 2012 Is an early pilot program started by the U.S. Department of Transportation (USDOT) National Highway Traffic Safety Administration (NHTSA) and is conducted by the University of Michigan Transportation Research Institute (UMTRI). Its main objective was to demonstrate connected vehicle technologies in a real-world environment. The deployment took place in Ann Arbor, Michigan and spanned over 100 kilometers. It included approximately 2800 vehicles including 2305 cars, 60 trucks and 85 transit buses as well as 29 Road–Side Units (RSUs). The deployment was based on the DSRC technology and the exchange of BSM messages. It included the following safety applications: Forward Collision Warning (FCW), Lane Change Warning/Blind Spot Warning (LCW/BSW), Emergency Electric Brake Light Warning (EEBL) and Intersection Movement Assist (IMA). The deployment also included Signal phase and timing (SPaT) information on 21 different intersections.

Cooperative Mobility Pilot on Safety and Sustainability Services for Deployment (COMPASS4D) - 2013 Is an early deployment project funded by the European Union. It included a total of 31 partners from 10 countries and was coordinated by the European Road Transport Telematics Implementation Coordination (ERTICO). It took place in many European cites: Bordeaux (France), Copenhagen (Denmark), Helmond (Netherlands), Newcastle (UK), Verona (Italy), Vigo (Spain) and Thessaloniki (Greece). This deployment included 344 equipped vehicles and 111 RSUs. The deployment was based on ITS-G5 and included the following safety applications: Red Light Violation Warning (RLVW) service, Road Hazard Warning (RHW) service and Energy Efficient Intersection (EEI) service.

Tampa Hillsborough Expressway Authority (THEA) - 2016 is pilot program that assembles a large implementation team including the University of South Florida Center for Urban Transportation Research and Siemens. The project also has a number of key partners including the Florida Department of Transportation and the Hillsborough Area Regional Transit Authority. This project includes more than 1000 privately owned vehicles, 10 buses, 8 trolleys equipped with On–Board Units (OBUs) as well as 47 RSUs. It provides various V2V and Vehicle–to–Infrastructure communication (V2I) based safety features including: Emergency Electronic Brake Lights (EEBL), Forward Collision Warning (FCW), End of Ramp Deceleration Warning (ERDW), Wrong Way Entry (WWE), Intelligent Traffic Signal System (I-SIG) and Vehicle Turning Right in Front of a Transit Vehicle (VTRFTV).

Système COOpérative Pilote En France (SCOOP@F) - 2016 is a pilot project for the deployment of C–ITS systems in France and is co-funded by the European Commission. IT involves 19 partners including road operators (SANEF, LD38), car manufacturers (Renault, PSA), research institutes (CEREMA, FSTTAR, LAB, ITS Bretagne) and security and telecommunication experts (IDnomic, Orange). The project aimed to include 3000 vehicles and RSUs over 2000 km of roads with five test sites in: Ile-de-France, Paris-Strasbourg highway, Isère, Bordeaux and Bretagne. This project successfully tested the PKI access verification and the ETSI Plugtests. It was based on a combination of wireless access technologies ITS-G5, Cellular (3G, 4G) and Bluetooth. It includes the ETSI standard CAM and

DENM messages that are signed with pseudonyms delivered from the PKI. It provides various alert services like the road works warning, information about current interventions, slippery road or emergency brakes.

Many more V2X projects are currently being deployed or are planned for the future. For instance, the European Union is putting in place a Connected Roads (C-Roads) Platform for the deployment of harmonized and interoperable C–ITS services in Europe. Additionally, on the US side, a large C-V2X deployment supported by Qualcomm is planned on the roads of Colorado by Ford and in Virginia by Audi.

2.3 Communications architecture

In this section, we describe the communications architecture of the ITS station. As an example, we follow the ETSI architecture of communications in ITS (ITSC) described in detail the *ETSI EN 302 665* [40]. Figure 2.2 shows the ITS station reference architecture. It is an extended version of the Open Systems Interconnection (OSI) model [41] with included ITS applications.

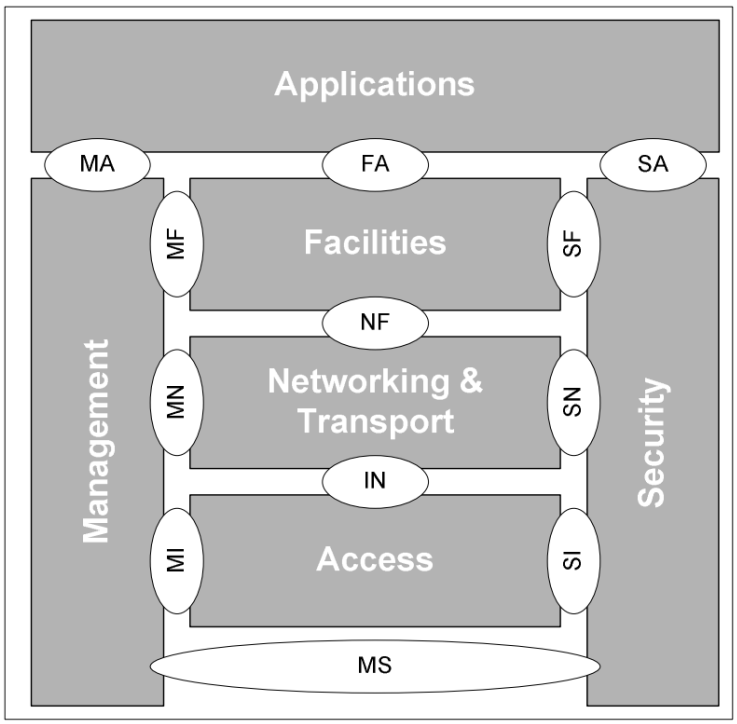


Figure 2.2: ETSI ITS station reference architecture

Here is a brief description of the functionalities of each ITSC block:

- **Applications:** This block contains the classes of applications described in section 2.1. Different classes of application have different requirements depending on their reliance on the communication. These requirements could vary with respect to reliability, latency and security.

- **Management:** This block contains all the management entities necessary for the ITS functionalities like the regulatory information management, cross layer management, station management and application management. It is common to the all the protocol stack layers.
- **ITSC OSI protocol stack:** In the current ITSC ITS architecture, multiple layers of the OSI communication protocol stack are combined into one block:
  - **Access:** The ITSC access block includes the functionality of the OSI physical and datalink layers, usually separated into layers 1 and 2. This block contains the communication technologies described in section 2.1, the IEEE 802.11p and 3GPP PC5 interface.
  - **Networking:** The ITSC Networking & Transport are combined into one layer whilst they are depicted by layers 3 and 4 in the OSI representation. The Networking protocols includes the IPv6 and GeoNetworking [42]. The Transport protocols include UDP/TCP and other dedicated ITSC protocols.
  - **Facilities:** The ITSC Facilities layer combines the OSI session, presentation and application layers, usually layers 5, 6 and 7. This layer includes generic functions and requirements which could be shared between different ITS applications. For instance, this layer manages the services like the cooperative awareness which could be used by various safety applications.
- **Security:** This block contains the functionalities relating to the communication security. It includes the firewall and intrusion management, authentication and profile management as well as crypto key and certificate management. These functionalities are used in the vehicular PKI system described in section 2.4.1.

2.4 C-ITS security

In the current C–ITS system, ITS Stations (ITS–Ss) like vehicles and RSUs cooperate by exchanging messages. These messages are used a in large number of security applications (see Section 2.1). Therefore, any errors in these messages could cause the safety applications to malfunction, which in turn could lead to catastrophic results. Consequently, the messages and their contents must be secured. As a result, the ETSI and IEEE created and standardized the vehicular PKI. However, the PKI is not enough to protect the system, specifically from internal attackers. The scientific community proposed misbehavior detection as a mitigation technique against these types of attacks. Currently, misbehavior detection is starting to be integrated in the latest ETSI and IEEE security architectures.

2.4.1 Vehicular PKI system

In this section, we describe the vehicular PKI system. The PKI is a form of asymmetric or public-key cryptography where every user owns at least one pair of keys, one public and one private. It is based on the mathematical one-way

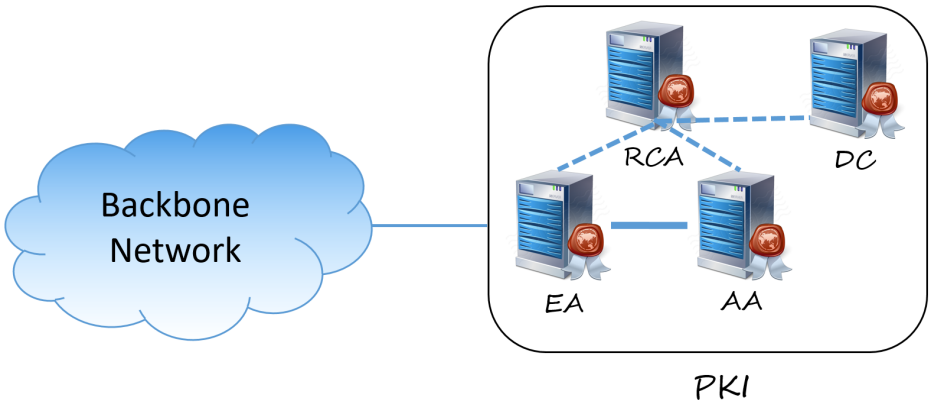


Figure 2.3: ETSI Vehicular PKI system

functions. The private key is used to encrypt data which could only be decrypted using the public key and vice-versa. Basically speaking, the sender encrypts the hashed message using his private key, this is called the signature. The receiver then uses the sender public key to decrypted the signature, then checks if it matches the hash of the received message. This enables the receiver to verify that the message was not altered during transmission. The vehicular PKI mainly updates the traditional PKI system with privacy protection and key distribution mechanisms.

As an example, we describe the ETSI Security Credential Management System (SCMS) for V2X communications. The ETSI SCMS implements a PKI published in three technical specification standards. The *ETSI TS 102 940* [11] includes a high level description of the security requirements and adopted architecture. The *ETSI TS 102 941* [43] includes a more detailed description of the protocol and the exchanged messages. The *ETSI TS 103 097* [44] contains the Abstract Syntax Notation One (ASN.1) description of the certificates and messages formats.

For the purpose of this thesis, we provide a brief general description of the functionalities the ETSI PKI illustrated in Figure 2.3. It includes four main components:

Authorization Authority (AA): The AA is the only security management entity with a direct communication with the enrolled ITS–S. The role of this entity is to issue and monitor the use of Authorization Tickets (ATs). An AT enables an ITS–S to send authentic V2X and to use specific C–ITS services.

Enrolment Authority (EA): The EA has only one direct contact with the ITS–S upon enrolment. The role of this entity is to manages the Enrolment Credentials (ECs) of ITS–S. The ECs are not used to sign messages sent to neighboring ITS–Ss. Instead, they enable the vehicle to request more ATs.

Root Certificate Authority (RCA): The RCA is the highest authority in the security management hierarchy. The role of this entity is to provide the EA and the AA with the certificates needed to be trusted by the ITS–S. These certificates allow then to issue valid ECs and ATs.

Distribution Center (DC): The DC provides the Certificate Trust List (CTL) and the Certificate Revocation List (CRL) to the local ITS–Ss. These lists enable the ITS–Ss verify the legitimacy of the certificate authorities.

In short, a new ITS–S should first find the address of an EA included in the CTL and verify that its certificate is signed by a valid RCA. Then, the ITS–S proceeds to request one EC, also called the long term identity, from the EA. This step is generally carried out in the factory. The ITS–S then acquires the address of an AA from the CTL and verifies its certificate. The ITS–S sends a request to this AA including the public key of the EC that is encrypted with the public key of the EA. Upon receiving this request, the AA will send the encrypted EC to the EA, which is able to decrypt it and verify that the vehicle is indeed enrolled. If so, the AA will issue new a AT to the ITS–S. These ATs, also called Pseudonyms, are used to sign V2X messages. An ITS–S station should regularly change its used Pseudonyms to hide its identity and avoid tracking by passive attackers.

The ETSI PKI enables local ITS–S to verify the authenticity and integrity, authorization of incoming messages from legitimate users. The certificates could also include different levels of authentication. For instance, an ambulance or a police vehicle could obtain a higher road priority. Additionally, the PKI protects the users' privacy. The users' privacy is protected on the local level through the use of Pseudonyms. The privacy is also protected on the global level. This is due to the mechanism used to request ATs. The EA and the AA are unable to track the real identities of ITS–Ss without cooperating. Finally, the PKI has been thoroughly tested by researchers for performance evaluation [45] [46] and for scalability [47]. These studies concluded that current state the ETSI PKI is feasible in a realistic environment.

Similarly to the ETSI version, the IEEE ITS–S acquire a long term identity equivalent to the EC and pseudonym identities equivalent to the ATs. Even though the IEEE version of the SCMS is not described here, it provides similar privacy and authenticity benefits to the ETSI version.

2.4.2 Misbehavior in C-ITS

Currently, neither the IEEE nor the ETSI versions of the SCMS provide protections that ensure the validity of the contents of a V2X message. An ITS–S with a valid set of ATs could send erroneous messages, all while having a valid signature. These errors are due to sensor fault, on-board unit malfunctions or even hacking. The physical sensors responsible for acquiring environmental data (e.g. GPS, Radar, etc...) could return incorrect information. The connection between the sensor and the on-board unit could be unreliable, leading to a number of bad readings. The on-board unit of a vehicle responsible for the transmission of V2X messages may contain bugs in its software. Even if we design a secure system where we minimize the possibility of the former issues, an on-board unit could still be hacked by a user with malicious intentions.

A hacker with control of an on-board unit is able to send V2X messages containing strategic information to serve a selfish purpose. For instance, an attacker could fake a traffic congestion warning on a specific road segment

to deter other vehicle from choosing it. An attacker could also fake an emergency break warning to cause other vehicles to needlessly decelerate. An attacker could even manipulate information to cause multiple vehicles to crash in a platoon. Furthermore, an attacker could use the multiple ATs in his collection to simulate the existence of multiple fake vehicles on the road. These fake vehicles could be cleverly used to take advantage of the V2X services. For instance, an attacker could fake the existence of a large number of vehicles waiting on a smart traffic light in order to gain priority at an intersection. Finally, the privacy protections provided by the vehicular PKI would make a malicious ITS–S less detectable. The attacker could perform an attack and then change his AT to regain the trust of his neighboring ITS–S. This type of potential abuse of the C–ITS services renders a misbehavior detection protocol essential to the system deployment.

2.4.3 Misbehavior detection protocol

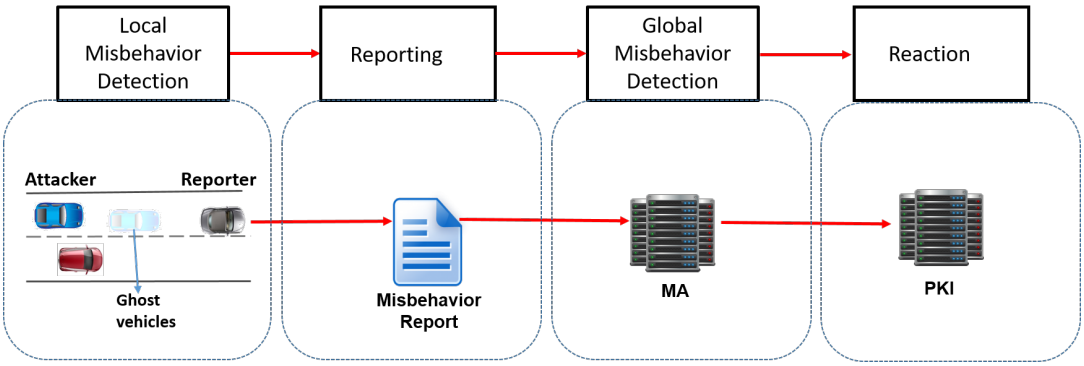


Figure 2.4: Misbehavior detection process

In this section, we describe the misbehavior detection process shown in figure 2.4. This process is divided into four steps:

Local detection: The local misbehavior detection is performed by the embedded component in every ITS–S (i.e. RSUs and Vehicle's OBUs). The misbehavior detection system is a security layer on top of the cryptography PKI component. Upon the reception of a new message and the verification of its signature, the ITS–S relays the message to the local misbehavior detection process. The message should then pass a set of simple and fast security checks to estimate the plausibility and consistency of the received data. The results of these checks are then fused using a local fusion application. The fusion application should then decide if an ITS–S is misbehaving and if a misbehavior report should be issued.

Misbehavior reporting: The misbehavior reporting process begins when the local fusion application determines that a misbehavior report should be issued. This process consists of building a Misbehavior Report (MBR) message with a set of relevant information confirming the misbehavior of the reported vehicle. More precisely, vehicles are

required to provide evidences proving the type of the detected misbehavior to the MA. This evidence consists mostly of the messages used in the detection process. After collecting enough evidence, the MBR is then sent to the global MA. The report could be transmitted to the MA via a trusted RSU or by direct cellular communication.

Global Misbehavior detection: The global misbehavior detection process begins by collecting the received MBRs. This operation is performed by the MA which is localized in the back-end security management system. Using the evidence in the MBRs, the MA should be able to recreate the local events to verify, if possible, the validity of the report. The MA then proceeds to analyze the collected MBRs to identify if a misbehavior has occurred and then precisely define the type of this misbehavior. The severity and the type of misbehavior determines the suitable reaction required to protect the system.

Misbehavior reaction: Once the detection results are obtained, the MA informs the authority in charge of enforcing the appropriate misbehavior reaction. Currently, the only discussed reaction type is the certificate revocation enforced by the PKI. In figure 2.4, we just provide an example where the PKI is in charge of proceeding to the appropriate reaction. As this is not yet standardized, the revocation and reaction procedure is still not well defined and other authorities may be in charge of the misbehavior reaction in the future.

2.4.4 General security architecture

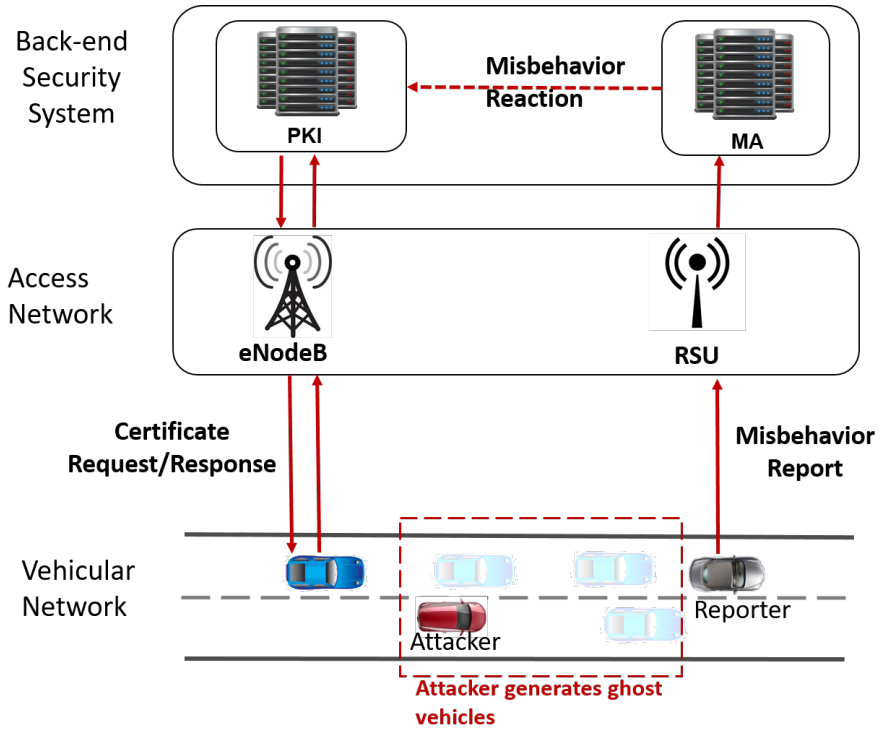


Figure 2.5: C-ITS security architecture

In this section, we recap the overall system architecture in the case of a presence of an attacker as illustrated in figure 2.5. We describe what occurs in different sections of the system.

- *Vehicular Network* consist of communicating ITS–Ss. Vehicles with valid certificates exchange signed messages that serve various purposes in ensuring road safety. For instance, a vehicle could send beacon messages, such as CAMs or BSMs, which contains kinematic information (position, heading, velocity etc...) to inform other vehicles of its presence on the road. However, an attacker could send these messages with fake information. The attack could be performed by one or multiple collaborating vehicles. Many use cases of attacks exist with different motivations and capabilities.

- *Access Network* is the layer used to relay local information to the back-end system in the cloud. This information includes certificate requests to the PKI and misbehavior reports to the MA. The communication could be done over a cellular connectivity (eNodeB) or the ITS-G5 (RSU).

- *Back-end Security System* is currently composed of the Public Key Infrastructure (PKI) and the Misbehavior Authority (MA). The PKI delivers digital certificates to the vehicles. The MA collects the MBRs form local vehicles then investigates and issues the suitable reaction.

2.5 Attacker model

In this research domain, we consider misbehavior in the use of the C–ITS communication infrastructure. A misbehavior could be the result of a faulty on-board system or physical sensors or a result of malicious intentions. The misbehaving ITS–S transmits signed messages with a modified payload. Generally, the attacker model has the following characteristics:

1. **Insider:** The attacker has the required cryptographic credentials to communicate on the C–ITS network. The attacker possesses a valid set of ECs and is able to request valid ATs from the AA. These ATs are used to sign the misbehaving messages.

2. **Active:** The attacker actively participates in the C–ITS communication and sends bogus data over the V2X network. Unlike the passive attacker whose main role is data collection and who therefore operates without transmitting any messages.

3. **Message payload modification:** The attacker can modify any field in its outgoing V2X messages. This is possible if an attacker has complete access to his vehicle's CAN bus. The attacker could also have changed the software of his vehicle's OBU. The attacker could also mount a man-in-the-middle attack and modify any sensor data. For instance, this should allow the attacker to modify location data received from a GPS module.

It should also allow the attacker to modify speed and acceleration information from a Telematics Control Unit (TCU). As the vehicle's OBU (V2X Unit) is connected to the CAN bus, it would consider this data genuine.

4. **Transmission rate modification:** We assume that the attacker can modify the transmission rate of his OBU. The attacker can alter the OBU in his vehicle, allowing faster or slower transmission rate depending on the type of the attack. For instance, the attacker could greatly increase his transmission rate to perform a Denial-of-Service (DoS) attack. The attacker could also double his transmission rate to pretend to be two vehicles, thus creating a ghost vehicle.

5. **Pseudonym certificates access:** The attacker has complete access over the usage of his previously acquired pseudonym certificates. We use the same premise as above, that the attacker has modified his OBU. This modification should give the attacker an unrestricted usage of his pseudonym certificates. This would enable the mounting of Sybil attacks. For example, to create four other ghost vehicles in addition to his own, the attacker would use five simultaneous certificates and increase the transmission frequency by five times. The attacker would be able to transmit signed beacons for all four Sybil ghost vehicles and his own, all while honoring required standard transmission rate.

2.6 Related works

Misbehavior Detection is a well-researched topic with studies spanning the last two decades. Golle et al. first published in 2004 a paper on detecting and correcting malicious data in the vehicular networks [48]. Their work is regarded as one of the first to introduce the research axis. They highlight the possibility of vehicles transmitting fraudulent data such as a road congestion or an erroneous vehicle position. They also emphasize the damaging effects of such attacks. Since then, this topic has been studied in much further detail. Van der Heijden et al. published in 2019 a well written and complete recent survey on the different misbehavior detection studies [49]. The survey includes a large number of well-known works and detection mechanisms. They classify the studies into two families of detection mechanisms: data-centric and node-centric. A study could make use of a combination of mechanisms but every mechanism fits under one of these families. Data-centric mechanisms rely purely on the contents of the message to estimate its plausibility and consistency. Node-centric mechanisms rely on the behavior of a node and generally assign a trust value to every neighboring ITS–S.

In this section, we examine the published studies related to all the contributions of this thesis. The studies will be sorted by the detected attack and by the family of detection mechanism.

2.6.1 Detection of Sybil attacks

With the current ETSI and IEEE C–ITS model, the PKI infrastructure provides the vehicle with multiple valid certificates. The vehicle proceeds to regularly change its identity in order to prevent easy remote tracking. An attacker could take advantage of this system to launch a Sybil attack. Sybil attacks were first introduced by Douceur in [50]. In V2X networks, a sybil attack occurs when an ITS–S actively uses multiple pseudonyms at the same time to simulate one or several ghost vehicles. Due to the important damages it may cause in C–ITS systems, researchers introduced several approaches for Sybil detection.

Pouyan et al. [51] propose three methods for local Sybil attack detection. The resource testing method assumes that a radio network entity cannot send and receive on the same channel at the same time. This detection method is not valid in the current state of vehicular networks because attackers may have multiple channels to send and receive messages. The Position verification method assumes that a vehicle can be localized at only one position at the same time. This method proposes the use of physical signal properties (i.e. Received Signal Strength Indicator (RSSI), Time Of Arrival (TOA) and Time Difference Of Arrival (TDOA)) to verify the true position. The feasibility of this method in a real scenario remains to be assessed. Last, they propose the encryption and authentication based methods, which assume that using a PKI is enough to detect Sybil attack. In the current C–ITS system, a vehicle could obtain multiple pseudonym identities at the same time. Therefore, a legitimate entity with valid key materials can still perform a Sybil attack.

Chen et al. [52] and Park et al. [53] both propose Sybil detection mechanisms where a vehicle with an OBU collects signed time stamps from RSUs. The theory is that these stamps act as proof that a vehicle had passed by a certain RSU. Each vehicle is required to broadcast collected stamps. Since a Sybil attacker can only have one physical path, a group of vehicle with a similar collection of stamps is considered a suspect of a Sybil attack. Chang et al. [54] propose a similar mechanism they call Footprint. Footprint works similarly to the previous mechanisms as vehicles also collects signatures from RSUs. However, their model uses cryptography mechanisms to hide the real vehicles trajectories. Therefore, this model is more privacy friendly. Nevertheless, all these detection mechanisms assume a Sybil attacker is moving through the system and is not performing a localized attack (i.e. fake congestion). Additionally, a Sybil attacker that refrains from broadcasting the collected time stamps would have a similar status to an ITS–Ss that had just started a new trip.

Hao et al. [55] propose a protocol that detects Sybil nodes in a cooperative way by examining the consistency between the vehicles positions and those of their neighbors. The idea is based on detecting the sudden appearance of a vehicle or of multiple vehicles as well as on evaluating the number of neighbors. When a vehicle locally detects that a neighbor is potentially malicious, it broadcasts a warning message to have the confirmation of other neighbors that an attack is occurring. When the number of vehicles that confirm that an attack is occurring is greater than a predefined threshold, the identified vehicle may be quarantined for a certain period of time or reported to

the authority. We believe that cooperative detection systems are not reliable because the attacker takes part of the community and could distort the detection procedure. Moreover, it requires an honest majority to work properly. Xiao et al. [56] propose a similar protocol that also relies on vehicles broadcasting a list of neighbors. The broadcasted list should include unique identifiers for neighbors such as the hash of the last beacon. Additionally, the neighbors list should include the RSSI. Afterwards, specific calculation determines the legitimacy of each node according to the neighbors list and the range of each vehicle. Sybil attackers could then be reported or excluded from the network.

Zhou et al. [57] introduced Privacy-Preserving Detection of Abuses of Pseudonyms (P^2DAP), a method which enables the ITS–S to verify its neighbors' pseudonym identities through edge computing. In P^2DAP, they consider edge computing through the use of RSUs, however this functionality could also be attributed to an eNodeB. In their system, the RSUs are considered a trusted entity. Accordingly, the RSU has the ability to link different pseudonyms. This linkage is done by pseudonyms which hash a common value. Linking pseudonyms would enable the detection of a Sybil attacker using the certificates issued for the same vehicle. In the current C–ITS system, the linkability of pseudonyms is not available even at the PKI level without the cooperation of the AA and the EA. Similarly, Lee et al. [58] propose Detection Technique against a Sybil Attack (DTSA), a method that is also enabled through the edge computation. DTSA suggest a different kind of pseudonyms where vehicles obtain a session key from a Vehicular Ad hoc NETwork (VANET) server to use within a limited time frame. Afterwards, each vehicle verifies the identity of neighboring vehicles with the help of the same VANET server.

Granted that the Sybil attack is ultimately a special kind of bogus information attack, it could also be detected by some types of methods designed for bogus information detection. In particular, those based on physical-layer and data-centric false beacon information detection mechanisms. This is addressed in detail in the following section.

2.6.2 Detection of bogus information attacks

Broadcasting bogus information sent on the V2X network is at the essence of misbehavior in C–ITS. Bogus information could have different implications varying from minor issues, like deteriorating the quality of infotainment services, to dramatic problems like causing accidents and potentially victims. In this section, we describe some of the bogus information detection methods that have been proposed in the literature.

2.6.2.1 Bogus beacon messages detection mechanisms

Vora et al. [59] propose a method of verification of the position in the broadcasted beacons. Their method is edge-assisted and based on a total coverage of an area by RSUs or eNodeBs. They assume a broadcasted message is received by multiple RSUs. The RSUs then uses the received signal strength to estimate the distance. Using the distance from multiple RSUs, the misbehavior detection system is able to approximate an area where the message was generated. The position included in this message should coincide within this area. Hubaux et al. [60] propose another location verification scheme for broadcasted beacons. They discuss two mechanisms: tamper-proof Global

Positioning System (GPS) and verifiable multilateration. They argue that tamper-proof GPS is not an adequate solution. They state that tamper-resistant hardware has well known weaknesses and GPS in general is vulnerable to attacks such as spoofing and jamming. Therefore, they then argue that verifiable multilateration is the more robust method to achieve position verification. Verifiable multilateration is an edge assisted method that requires the total coverage of an area by base stations (e.g. RSUs). It uses distance bounding to verify that the messages could be physically received by a certain station. Afterwards, a back-end system verifies that the claimed position is within the intersection area of all the base stations.

Sun et al. [61] proposed several physical layer features to verify a vehicle's location and mobility. In their attacker model, the attacker transmits false positions inside the beacon messages. This attacker can be independent or can collude with other attackers who will corroborate the false data. The authors consider a straight highway scenario with at least one honest vehicle in the communication range of the ego vehicle. They verify the attacker's location and mobility information using Angle of Arrival (AoA) estimation, Doppler Speed (DS) measurement, extended Kalman filter (EKF) and input from neighboring vehicles.

Schmidt at al. [62] propose VEhicle Behavior Analysis and Evaluation Scheme (VEBAS). Their mechanism uses the semantics of the messages to determine its authenticity. This includes the use of multiple data-centric mechanisms such as: Acceptance Range Threshold (ART), Minimum Distance Moved (MDM), Map-Proofed Position (MPP) and Sudden Appearance Warning (SAW). ART is based on the fact that the transmission range is physically limited. MDM supposes that a stationary vehicle should not transmit messages in order to prevent a roadside attacker. MPP is the verification that a vehicle is moving with valid trajectory with respect to the road. SAW is the monitoring of newly appearing vehicles, specifically targeting ghost vehicles. Finally, results of all these methods are combined using an Exponentially Weighted Moving Average (EWMA). Bißmeyer et al. [63], propose a method that combines the data-centric mechanism of VEBAS and a plausibility model to check for position intersections between multiple vehicles.

Leinmüller et al. [64] propose a detection scheme that combines mechanisms from VEBAS with a new data exchange based detector. They introduce a node-centric detector called Pro-Active Neighbor Exchange. This detector is based on the vehicles broadcasting a table of their neighbors. The vehicles use this information to cross-check the general claimed positions of the neighbors. Van der Heijden et al. [65] propose a subjective logic based fusion of multiple misbehavior detection schemes. They propose an improved version of the ART detector that they call enhanced Acceptance Range Threshold (eART). eART improves on the ART by setting the acceptance transmission range to a Gaussian curve instead of a fixed threshold. They claim the Gaussian approach is better for combining eART with other mechanisms. Additionally, they use the Pro-Active Neighbor Exchange detector. Finally, they combine both detectors using subjective logic. Subjective logic [66] a probabilistic approach that takes into account the uncertainty and the trust in the source of the information.

Zaidi et al. [67] propose a collaborative detection mechanism based on the information broadcasted by neigh-

bors. In this model the vehicle broadcasts a flow parameter for their vehicles in their neighboring region. The flow parameter is calculated based on the density and speed of vehicles in a fixed range. Therefore, the flow for neighboring vehicles has to be within a certain threshold. Consequently, using a statistical model, vehicles could calculate the plausibility of the neighbors' information and detect misbehaving vehicles.

2.6.2.2 Bogus warning messages detection mechanisms

Ruj et al. [68] propose a data-centric scheme of misbehavior detection in vehicular networks. Similarly to the previously discussed studies, they use an edge assisted mechanism for position verification. Their method relies on distance bounding, but additionally on the speed of light and the message timestamp to verify the distance from the source of the transmitted signal. Furthermore, they propose a data-centric verification scheme for warning messages. Their method for warning verification relies on the assumption that a vehicle emitting a warning event should behave accordingly. Ghosh et al. [69] also propose a similar scheme for the detection of fake warning messages. Their method is based on integrated root-cause analysis. For example, a vehicle issuing a blocked road warning needs to be on a proximity of the event and needs to change its path accordingly to avoid the obstacle. A vehicle issuing a warning event is thus monitored by the neighboring vehicles to determine the message authenticity and therefore the validity of the warning.

Cao et al. [70] propose a voting based validation scheme for warning messages. The validation of an event depends on the number of signatures it receives from neighboring vehicles. In this work, they created an efficient and secure protocol to collect and distribute signatures. Their protocol relies on growth codes introduced by Kamra et al. [71]. Growth codes are initially introduced to enhance efficiency and data persistence in an unreliable or failing sensor networks. They are used here to ensure that transmitted signatures arrive to their respective destinations. Hsiao et al. [72] propose a similar voting based warning validation. They consider a system where an event becomes valid if the number of witnesses exceeds a certain threshold, then proceeds to evaluate multiple threshold-based event validation algorithms.

2.6.2.3 Trust based detection mechanisms

Kim et al. [73] propose a misbehavior detection mechanism based on a Certainty of Event (CoE) curve. The CoE is calculated using a combination of six different sources: Cryptographic Authentication, Source Location, Local Sensors, Responses from Other Vehicles, Infrastructure Validation and Reputation. Cryptographic Authentication is the validation of the PKI certificates and the message signature. They consider that this authentication is enough to prevent Sybil attacks, which is no longer true in the presence of pseudonyms. Source Location is check if a vehicle geographic location is in the area of relevance of the event. Local Sensors is an on-board validation when the recipient vehicle is in direct line of sight of the event. Responses from Other Vehicles is a voting based integrity check that relies on the support or contradiction of neighbors of a certain event. Infrastructure Validation is based on

the validation of RSUs equipped with the right equipment and with direct line of sight to the event. Finally, Reputation is a local trust based on the behavioral history of a certain node. A vehicle's trust increases if it reports a true alert and decreases otherwise. It is worth noting that although the combination of different mechanisms increases the efficiency of the method, however it inherits all the feasibility challenges.

Raya et al. [74] propose Local Eviction of Attackers by Voting Evaluators (LEAVE). LEAVE allows for the local eviction of an accused by neighboring vehicles. They argue that this system is the temporary step to protect the C–ITS before the global revocation phase by the PKI. Their local eviction is a cooperative system based on an honest majority of vehicles performing local misbehavior detection checks. Their local misbehavior detection is based on detecting known attacks by monitoring specific parameters and data anomalies with plausibility and consistency checks. Zhuo et al. [75] propose Suicide-based Local Eviction Protocol (SLEP) and Permanent Revocation Protocol (PRP). SLEP is another mechanism for local of an accused vehicle. However, in SLEP only one accusation is enough to revoked a vehicle. The catch is that both the accuser and the accused are revoked. Neighboring vehicles would then transmit all accusations to a cloud entity. The cloud entity would then decide on which vehicle to permanently revoke using the PRP and would allow the other to rejoin the system. This mechanism is in place specifically to reduce the number of fake accusations.

Leinmüller et al. [76] propose a method a cooperative position verification method to defend against the roadside attacker. In their work, vehicles build local trust in their neighbors using data-centric mechanisms (like the MDM). This local trust is then broadcasted to other vehicles. Finally, vehicles combine their local trust and the trust values received by the neighbors to create the global trust. A vehicle is considered an attacker if it fails its global trust falls below a determined threshold.

Kerrache et al. [77] introduce a novel Trust architecture for Vehicular Networks using the standardized messaging services of ETSI ITS (T-VNets). T-VNets aims to detect false ETSI CAMs and DENMs type messages. Their method proposes building trust using a combination of different mechanisms: data-centric, event-based, watchdog and RSU-based. Data-centric mechanisms evaluate the quality of received messages. Event-based mechanisms evaluate the effectiveness of issued warning events. Watchdog is a mechanism where vehicles share positive of negative recommendations or neighbors. Finally, RSUs broadcast a trust value for vehicles based on current and historical behaviors evaluations. All their mechanisms are then combined to compute a global trust evaluation for every neighbor. The trust level is shared between nodes using CAMs and regularly updated.

Raya et al. [78] propose a data-centric trust establishment mechanism. Their method calculated trust without using cooperation between vehicles. This approach would reduce the risk of a Sybil attacks especially in the presence of pseudonyms. The trust evaluation is based on four indicators. The vehicle specific trust, different types of vehicles would have a different initial trust value, i.e. police vehicle, emergency vehicle. The event-specific trustworthiness which is the trust based on the relation of the event to the emitting vehicle. The dynamic trustworthiness which is based on the revocation status. Finally, the time and location indicators such as the proximity to the event. These

indicators are combined in a global trust value which should remain above a certain threshold.

Jaeger et al. [79] proposed a Kalman filter based method to verify beacon messages. Kalman filters allow for fast and robust trajectory prediction even in the presence of sensor errors. They enable accurate vehicle tracking and consequently comparison between the Kalman predicted and the beacon claimed position. This would enable the tracking of the vehicle even whilst changing pseudonyms. Therefore, Kalman filters could achieve an implicit linkability between pseudonyms. Consequently, it could be a useful tool to defend against Sybil attacks by increasing the integrity of honest nodes. Xu et al. [80] also propose a mechanism to circumvent the issue of pseudonymity for the trust establishing mechanisms. They propose the use of wireless fingerprinting for vehicle identification in the case of an identity change. They implement their proposal in a simulation environment. The result of these simulations claims a high success rate in the detection of sybil attacks.

2.6.2.4 Machine learning based detection mechanisms

Van der Heijden et al. [81] introduced VeReMi. VeReMi is a misbehavior detection dataset created by simulating the LuST network scenario. They used VEINS. Their simulation is based on VEINS a co-simulator of SUMO and OMNET++. The dataset consists of message logs for every vehicle in the simulation. The log contains information such as the vehicle position, speed and the message RSSI. Many other parameters are also provided such as the number of vehicles, the number of attackers, as well as the variation of attacker rates. VeReMi also contains four types of misbehavior: Fixed Position, Fixed Position Offset, Random Position, Random Position Offset and an Eventual stop. Many published studies discussed later use the VeReMi dataset, specifically to evaluate different machine learning algorithms.

Grover et al. [82] propose a machine learning based approach to detect misbehavior in C–ITS. For this purpose, they experiment with different classifiers implemented in the Waikato Environment for Knowledge Analysis (WEKA) toolset. Their experiments include: Naive Bayes, Instance based learner (IBK), Decision Tree (J-48), Random Forest (RF) and AdaBoost. They conduct a comprehensive comparison and show that RF and J-48 classifiers perform best. In a subsequent paper [83], they improve the detection performance by replacing the single classification algorithm with several classification algorithms. They then use a majority voting ensemble-based scheme to aggregate the results of the previously mentioned classifiers into a single stronger classifier. They show that the ensemble-based model is more robust and efficient in classifying multiple misbehaviors present in C–ITS and could achieve a better result in comparison to each individual base learner.

Ghaleb et al. [84] also propose a machine-learning misbehavior detection model to detect the location forging attacks they introduced in [85]. They create and extract multiple features from the data model including historical values and several plausibility and consistency checks. In total, their network takes 7 features as input. These features include checks for overlapping positions, maximum transmission range, Kalman filter mobility validation as well as cooperative reports from the neighboring vehicles. They use these features to train a feed forward

neural network over the Next Generation Simulation (NGSIM) [86] dataset. NGSIM contains real-world traffic data and vehicle trajectories recorded using digital video cameras every one-tenth of a second. However, the authors synthetically introduced the attacks in this dataset. Although, their model has a 99% detection rate, their analysis was performed on a static set of vehicles in the NGSIM dataset. Additionally, since the attacks were introduced synthetically, there exists a possibility of bias in these attacks.

So et al. [87] propose a machine learning based framework to detect location spoofing. They scored their location plausibility, movement plausibility and other quantitative features to feed into the machine learning models. They evaluated the performance of these models in terms of their classification accuracy and precision-recall characteristics. They used the VeReMi [81] dataset for the training and testing purposes. Consequent, they considered location spoofing attacks from the VeReMi dataset. Their study aimed to create a baseline ML solution using the K-Nearest Neighbors (K-NN) and Support Vector Machine (SVM) classifiers. Both algorithms performed similarly with SVM having a slight edge. In order to improve on this model, So et al. [88] propose three novel physical-layer plausibility checks, The First-BSM (FBSM), Majority-BSM (MBSM) and Weighted-BSM (WBSM) all based on the RSSI. They show that these checks outperform recently proposed machine learning based schemes operating at the application-layer.

Singh et al. [89] proposed another machine learning solution using the VeReMi dataset. The study tested SVM and Logistic regression. However, their features consist of the positions of the sender and the difference between the sender and the receiver. These features are not adapted for the detection of any attack. They obtain unreasonably high detection accuracy. According to our tries to recreate their results, it's likely that their system is over-fitting. Singh et al. also proposed a deep learning based solution [90]. In this study, the detection is done within an RSU where an attacker attempts to create a fake traffic jam. Additionally, they use a completely different set of features. The new features consist of the mean amount of time vehicles spent in a road segment and the number of vehicles currently in the road segment. They create their own relatively small simulation scenario. They trained and tested an MLP and an LSTM. In their model, LSTM is the better performer although at the cost of more computational time.

Zhang et al. [91] propose a misbehavior detection mechanism based on SVM and the Dempster-Shafer Theory (DST) of evidence to detect false message injection. One SVM-based classifier is used to detect false messages based on message content and vehicle attributes. Another SVM-based classifier is used to evaluate whether the vehicle is credible based on its behavior in terms of message propagation. Every vehicle would then transmit their trust assessment reports to a back end system. The back-end system would use the DST to aggregates multiple trust assessment reports about the same vehicle and derives a comprehensive trust value for each vehicle.

Gyawali et al. [92] propose a machine learning based scheme to detect two categories of attacks: false alert attack in which the malicious vehicle broadcasts false warning messages and position falsification attack where the attacker alters the location information in the beacon. The false alert attack consists of a fake congestion where the attacker lowers its speed and the segment flow value. They are generated using the VEINS simulator. The position

falsification messages are extracted from the VeReMi dataset. They use different features for the detection of the two categories of attacks. For the alerts, they use a specific broadcasted flow value for a specific road segment. For the position falsification detection, they use feature such as the position and speed change and the RSSI. In this study, they train multiple machine learning models including Logistic Regression, k-nearest neighbors, Decision Tree and Random Forest. They claim that the proposed scheme is more effective to detect internal attacks as compared to the detectors proposed in VeReMi.

2.7 Feasibility study

In this section, we assess the feasibility of existing misbehavior detection mechanisms within the current C–ITS ecosystem. We contribute to the state of the art by providing a new classification and understanding of the works on misbehavior detection. We focus on the logic behind the detection mechanisms instead of the details of every detection method. Finally, we discuss remaining challenges that have to be addressed by the community.

Feasibility is assessed on multiple levels. Currently, standardization bodies such as the ETSI and IEEE made significant progress in designing the C–ITS architecture. We believe that misbehavior detection mechanisms should be **in line with the current standards** to ensure a much needed fast and easy deployment. Furthermore, misbehavior detection mechanisms face challenges caused by the constant conflict between security and privacy. This conflict often appears in the form of **regulations or legal complications** (e.g. violation of the General Data Protection Regulation (GDPR) [93]). Finally, there is feasibility challenges in term of **the required equipment** necessary for the functionality of a detection mechanism. The equipment required for a detection mechanism may be too costly to include in each station. Additionally, the entire system may not yet have reached the maturity a mechanism requires. In this section, we will illustrate standard incompatibilities with a circle (○), legal and regulation conflicts with a square (□) and required equipment with a triangle (△).

2.7.1 Feasibility of detection mechanisms for Sybil attacks

Using the multiple studies that have targeted the detection and mitigation of the Sybil attack discussed in section 2.6.1, we extract three prominent methodologies used specifically for Sybil detection:

Path history detection mechanisms: This mechanism is based on the fact that one physical vehicle has one physical path. Therefore, requiring vehicles to collect and broadcast signed time stamps from RSUs would theoretically prevent Sybil attackers. It is proposed by Chang et al. in [54], Chen et al. in [52] and Park et al. in [53].

The current state of the standard requires a vehicle to change all of its identifiers when using a new certificate, in order to prevent linkability between pseudonyms [11]. However, beaconing a footprint history would negate this effect and facilitate the linkability of pseudonyms (○). Additionally, a new protocol has to be implemented to enable:

the RSUs to issue signed timestamps and the OBUs to beacon these timestamps (○). Although the current ETSI CAM messages include an optional path history field, this variable consists of a list of GPS coordinates and is not compatible with the signed timestamps proposed in this method [7]. Furthermore, it is unclear if requiring a broadcast of the path history is friendly to the current privacy protection laws (□). Although in [53] and [54] the privacy issue of direct traceability is addressed, this would not negate the pseudonyms linkability problem. Finally, this approach relies on a wide coverage RSUs in the C–ITS network (△).

Pseudonym Linkability based mechanisms This mechanism relies on a system where in some way pseudonyms have to be linked. Linking pseudonyms would enable the detection of a Sybil attacker using the certificates issued for the same vehicle. It is proposed by Zhou et al. in [57] and Lee et al. in [58].

Enabling pseudonym linking at the RSU level is not compliant with the privacy requirements of C–ITS. Currently, the vehicular PKI system is designed in such a way that not even the AA and the EA have the ability to link pseudonyms without cooperating [43] (□). At the present time, the ETSI standard does not specify any linkage authority and the IEEE designed linkage authority is only available to the misbehavior authority [94] (○). Moreover, it's unclear how scalable this approach is when an RSU has to link pseudonyms of a great number of vehicles (△).

Neighbor List Exchange This mechanism relies on vehicles broadcasting a list of neighbors including unique identifiers. Using this list, vehicles are able to calculate legitimate nodes and Sybil attackers could then be reported or excluded from the network. It is proposed by Hao et al. in [55] and Xiao et al. in [56]

Exchanging the neighbor lists is a distributed and simple approach. However, its efficiency could be greatly affected by the rate of the pseudonym change [95] (○). Furthermore, data protection acts could oppose broadcasting the information about other vehicles (□).

Since a Sybil attack is ultimately a special kind of bogus information attack, it could also be detected by some methods designed for bogus information detection. In particular, we note physical-layer and data-centric false beacon information detection mechanisms, which we address in detail in the next section.

2.7.2 Feasibility of detection mechanisms for bogus information attacks

A large number of studies in the literature have targeted the detection and mitigation of the bogus information attacks. In this section, we extract the specific detection mechanisms behind studies discussed in the section 2.6.2. We organize our evaluation into three sections: detection methods for false position information (the main component of the beacon message), detection methods for warning messages and detection methods that evaluate a node and thus all the messages it broadcasts.

2.7.2.1 False beacon information

RSU triangulation This mechanism relies on a total coverage of an area by RSUs. A message received by multiple RSUs would be limited to a general area therefore verifying its physical location. It is used by Vora et al. in [59] and Hubaux et al. in [60].

The triangulation technique requires a total RSU coverage of the area (△). Additionally, it requires a specific communication protocol between the RSUs to share achieve this triangulation (○). Finally, the automatic back-end information sharing of vehicle locations could be opposed by the privacy protection regulations (□).

Physical layer detection This mechanism relies on the physical aspects of the signal for the message location verification. These physical aspects include the Received Signal Strength Indicator (RSSI), Time Of Arrival (TOA), Angle of Arrival (AoA) and Doppler Speed (DS). It is used by Pouyan et al. in [51], Xiao et al. in [56], Sun et al. in [61], So et al. in [88] and Gyawali et al. in [92].

Physical detection methods are generally compatible with the standard and the corresponding laws. However, they may require specific on-board sensors (△).

Data-centric detection This mechanism uses the semantics of the messages to determine its authenticity. Specifically, this mechanism targets the plausibility and consistency of the information in the message. It is used by Schmidt at al. in [62], Bißmeyer et al. in [63], Leinmüller et al. in [64] and Van der Heijden et al. in [65].

Generally, local data-centric mechanisms don't present major feasibility issues. These mechanisms do not require any additional equipment nor changes to the standard nor present any legal challenges.

Machine learning based detection This mechanism analyzes various features of the messages with a machine learning algorithm. The features can be issued from the raw message data or can be derived from the physical signal proprieties or the data-centric checks. It is used by Grover et al. in [82, 83], Ghaleb et al. in [84], So et al. in [87], Singh et al. in [89, 90], Zhang et al. in [91] and Gyawali et al. in [92].

Similarly to data-centric mechanisms, machine learning doesn't require any changes to the standard or present any legal challenges. However, using machine-learning requires significant on board processing power (△).

Neighbors information exchange This mechanism is based on the exchange of some additional information between neighboring vehicles. This information is then used by vehicles to detect misbehaving beacon data. It is used by Leinmüller et al. in [64], Van der Heijden et al. in [65], Zaidi et al. in [67] and Ghaleb et al. in [84].

For this mechanism to work, a new protocol for exchanging additional information between neighbors has to be standardized and implemented [7] (○). Additionally, the legality of sharing certain additional data between neighbors has to be investigated (□).

2.7.2.2 False warning messages

Data-centric detection This mechanism relies on the assumption that a vehicle emitting a warning event should behave accordingly. Neighboring vehicles monitor the behavior of the emitting vehicle to determine the authenticity of a certain event. This mechanism requires a vehicle's pseudonym to remain the same after emitting the warning. It is used by Ruj et al. in [68] and Ghosh et al. in [69].

A block on the pseudonym change after generating a warning message is currently planned to be included in the standard [32]. Therefore, this method is compatible with the standard, presents no legal challenges, and does not require any special equipment. However, it is worth noting that this approach assumes that the malicious vehicle is only deceptive about the warning messages. Otherwise beacons its correct location information. Therefore, this method has to be bundled with a position verification technique.

Voting based detection This mechanism is based on voting or cooperative validation of an event in order to ensure its integrity. This mechanism is generally effective in a densely populated network with an honest majority. It is used by Cao et al. in [70], Kim et al. in [73] and Hsiao et al. in [72].

Similarly to other mechanisms, voting schemes requires a new protocol and messages architecture [7] [8] (○). Nevertheless, this protocol could be more challenging to integrate due to the effect of pseudonymity on the voting integrity (○). This effect is amplified with a higher frequency of pseudonym change [95].

2.7.2.3 General node trust evaluation

In this section, we evaluate detection methods that estimates the general trust in the vehicle instead of estimating the correctness of messages separately. Therefore, all messages from a corresponding node will be evaluated according to its trust level. It is worth noting that all methods that evaluate node trust are eventually affected by pseudonym change. This severity of this issue varies depending on the change frequency (○).

Cooperative trust establishment This mechanism is based on establishing cooperative trust through voting or through a consensus. It is used by Raya et al. in [74], Zhuo et al. in [75], Leinmüller et al. in [76], Kim et al. in [73] and Kerrache et al. in [77].

Similarly to all node-centric approaches, compatibility with pseudonyms is always a challenge (○). Additionally, cooperative trust establishment requires the modification of the current communications architecture whether to include voting or for consensus mechanisms [7] [8] (○). Last, revocation of a node from the network entails a denial of security application to the node. Legally, it is unclear if nodes of the same clearance level could deny other nodes from access to safety applications (□).

Data centric trust evaluation These mechanisms evaluate trust without a cooperation between vehicles. It is used by Raya et al. in [78].

Data-centric methods are generally compatible with the laws and the standard. However, the node-centric approach also adopted by these methods may have a conflict with the pseudonymity ensured in the standard (○).

Pseudonym linking These mechanisms are used to circumvent the pseudonymity issue for trust establishing mechanisms. These solutions aim to achieve an implicit linkability between pseudonyms. It is used by Jaeger et al. in [79] and by Xu et al. in [80].

Although the implicit linking of pseudonyms benefits greatly all node-based mechanisms, privacy protection regulations may oppose these types of methods to ensure that linkability is only feasible by a trusted authority. Otherwise, the whole concept of pseudonyms may as well be questioned (□).

2.7.3 Feasibility discussion

Table 2.1 summarizes the feasibility evaluation of Section 2.7. The first observation is that mechanisms designed to detect false beacon messages or warning messages separately, are globally feasible. Interestingly, these methods could be combined to form a global misbehavior detection framework. On the other hand, mechanisms based on node-trust face more feasibility challenges. However, the feasibility is not the only factor to consider. The performance of a mechanism should also be evaluated [49]. Some problems could be overcome if there is a big enough incentive. A system that requires changes in the regulations, in existing standards, or requires specific equipment should justify a clear major benefit to have a chance to be adopted.

A system with incompatibilities with the standard could be considered if the advantages it presents are significant. For example, several solutions [55][56][63][64][65] are based on a neighbor's information sharing mechanism. In simulation, this mechanism shows promising results [65] and does not imply major changes in standardized protocols to add the relevant fields on exchanged messages. For these reasons, we classify this method among those that have a good balance between ”costs” and benefits. On the other hand, reputation-based mechanisms that requires the C–ITS to have a stable identity, are harder to integrate. Requiring a stable vehicle identity would directly oppose the pseudonymity both practically and in principle. Therefore, the addition of a reputation protocol requires rethinking major parts of the current system and is unlikely to be adopted.

Systems that present legal issues are difficult to adopt because they need a change in the regulations. Legal issues (usually privacy violations) itself can prevent a systems' deployment despite the advantages it could presents. For instance, detection methods that rely on broadcasting a path history pose a major legal concern related to privacy. Similarly, methods that are based on the implicit local linkability of pseudonyms break the mechanisms meant to ensure the protection users' privacy. In our opinion, privacy threatening methods such as those that rely on path history broadcasting and local pseudonym linking are unlikely to be deployed, especially in the presence of

Table 2.1: Summary of the feasibility challenges.

✔: Compatible, ✚: Requires Adjustment/Study, ✖: Incompatible

	Detection Method	Used Studies	Current feasibility		
			Standard ○	Legal □	Equipment △
Sybil	Path History	[52] [53] [54]	✖	✖	✚
	Pseudo linkability	[57] [58]	✚	✚	✚
	Neighbor List	[55] [56]	✚	✚	✔
Sybil & Bogus Info	***Beacon Messages***				
	RSU triangulation	[59] [60]	✔	✔	✚
	Signal Properties	[51] [56] [61] [88] [92]	✔	✔	✔
	Data-Centric	[62] [63] [64] [65]	✔	✔	✔
	Machine-Learning	[82] [83] [84] [87] [89] [90] [91] [92]	✔	✔	✚
	Info Exchange	[64] [65] [67] [84]	✚	✚	✔
Bogus Info	***Warning Messages***				
	Data-Centric	[68] [69]	✔	✔	✔
	Voting-Based	[70] [72] [73]	✚	✔	✔
	Node-Trust				
	Cooperative	[73][74] [75] [76] [77]	✖	✚	✔
	Data-Centric	[78]	✖	✔	✔
	Pseudonym Linking	[79] [80]	✔	✖	✔

alternative, more privacy friendly methods.

Lastly, methods that require specific equipment face the simple trade off of costs and benefits. For a method to be eligible for deployment, in some way, the benefits have to outweigh the costs. With this in mind, we take the example of the RSU triangulation technique for location verification. Currently, RSU coverage is limited and thus the cost of a total RSU coverage is high compared to the benefits. Moreover, other less demanding methods exist. However, in a later stage of the connected C–ITS network, with a wider and more secure RSU coverage, the triangulation check could be easier to justify and subsequently integrated.

In essence, this study doesn't aim to evaluate the detection methods based on their current compatibility status. Instead, the goal is to evaluate the compatibility status itself. C–ITS systems are reaching the deployment stages. A misbehavior detection system, based on the current state of the art needs to be implemented and deployed in parallel to the deployment of C–ITS. The need for a robust and seamlessly compatible detection method is imminent. Moving forward, the gap between the regulations and the scientific methods needs to be bridged. The regulations

need to be adapted to accommodate for some detection mechanisms. Correspondingly, future studies need to consider the feasibility challenges while innovating new misbehavior detection mechanisms.

Chapter 3

Framework for Misbehavior Detection Simulation and Evaluation

In this chapter, we present a misbehavior detection simulation framework that enables the research community to develop, test, and compare detection algorithms. We also demonstrate its capabilities by running example scenarios and discuss their results. Framework for Misbehavior Detection (F^2MD) is open source and available for free on our GitHub [26].

3.1 Motivations

Misbehavior Detection in C–ITS is an active field of research that concentrates on developing mechanisms to detect anomalous behavior pertaining to vehicle movement, transmission, etc. Detection algorithms could be on physical sensors and on V2X messages. In this work, we concentrate on V2X-based Misbehavior Detection. In order to perform substantial research, these techniques have to be implemented on a large scale, in different scenarios, with a variety of vehicle densities and many other variations. Currently, there are only a few widespread deployments of connected vehicles (see section 2.2). A variety of problems still exist with these deployments such as the limited number of use cases and the strict regulations on the generated data. Obtaining raw, untouched data is difficult due to regulations that require anonymization and stripping of information that may be helpful in designing detection algorithms. Also, finding a suitable subset of data for analysis warrants spending huge amounts of time that could be used in actually designing detection algorithms. Hence, there is a crucial need to simulate vehicular networks and evaluate misbehavior detection algorithms in those simulations.

Simulators such as VEINS [23] provide a platform for the development of misbehavior detection algorithm. We propose F^2MD, a singular framework with which one can:

- Evaluate effectiveness of attacks (we provide 6 attacks and 9 faulty behaviors with our framework),
- Assess performance of detection algorithms (we provide 15 algorithms with our framework), using 8 metrics,
- Implement a set of new V2X attacks,
- Implement several detection algorithms for easy comparison,
- Visualize in real-time the detection algorithms performance,
- Generate dataset format to feed the common attack dataset VeReMi [81],
- Evaluate multiple misbehavior report formats and global Misbehavior Detection algorithms.

The remainder of this chapter is structured as follows. In section 3.2, we discuss the related works. In section 3.3, we explain our proposed framework in detail. In section 3.4, we run multiple examples with our framework to demonstrate the extent of results one can get from it. Finally, in section 3.5, we present our conclusion.

3.2 Related works

The current state of C–ITS infrastructure is far behind for conducting comprehensive research on misbehavior detection. This warrants implementing custom V2V simulators or retrofitting network simulators for this purpose. Therefore, it is safe to assume that simulations are a crucial part of evaluation of detection algorithms. For V2V simulations, a simulator should consist of network and mobility models that simulate real-world V2V scenarios. Simulators such as Network Simulator 3 (NS-3) and OMNeT++ provide feature-rich environments for network simulations. However, these simulators do not simulate a crucial aspect of V2X networks, a vehicle's mobility model. Likewise, MATLAB could also be used to perform simulations like in the work of Sun et al. [61] and Ghaleb et al. [84]. However, MATLAB simulation require a readily available dataset on which misbehavior detection algorithms can be run for conducting evaluations.

In C–ITS research, one commonly used network and mobility simulator is VEINS [23]. VEINS is a well-known and open source framework for running vehicular network simulations [23]. It is based on the OMNeT++ [24] for network simulation and the SUMO [25] for road traffic simulation. VEINS provides APIs to create custom applications that run locally on a vehicle. These applications can react on receiving a beacon from another vehicle and/or on changing its own position among other features. VEINS also provides the capability to generate custom datasets for different road networks. However, it does not include misbehavior detection algorithms or the capability to evaluate them.

To the best of our knowledge, there is only one simulation module that allows the simulation of misbehavior mechanisms in C–ITS, namely VeReMi [81]. VeReMi is a dataset, but the publication also offers a VEINS extension

which includes five position-based attacks and four basic detection checks. However, the purpose of this VEINS extension is mainly to feed the VeReMi dataset and is not suitable for real time detection and evaluation. In our framework, we offer a different take on the VEINS extension focusing mainly on a real time simulation. We also offer more attacks on the V2X network, advanced detection algorithms and common evaluation metrics. Note that we later also work closely with the team behind VeReMi to the extend the dataset using the F^{2}MD framework for the benefit of the scientific community (see section 4.3).

3.3 Framework features

In this section, we describe the different components of F^{2}MD. All parts of the framework that are described in the following section are also available for download in open source format [26].

3.3.1 General framework characteristics

This framework provides a complete solution for real time simulation and evaluation of a misbehavior detection system. It extends VEINS with a large panel of detection, evaluation and other general C–ITS modules. One of the main characteristics of F^{2}MD is its modularity. The architecture is organized in several functional levels: *input data*, *local detection*, *local visual output*, *report data output* and *global detection*. According to the misbehavior evaluation level, the complexity of the scenario, the attacks and the detection method may be chosen. We adopted this approach to facilitate the testing of different state of the art solutions in different modules of the framework.

Additionally, F^{2}MD is extensible. Besides the implemented detection mechanisms and attacks, it offers the possibility to extend the framework with additional modules through the existing API. A key characteristic of our framework is its integration with non-simulated modules such as external Machine Learning modules for advanced detection and external packet reports logging. Figures 3.1 and 3.2 summarize this idea while showing the different modules of the architecture. These modules are detailed in Sections 3.3.2 to 3.3.9.

Furthermore, to create a state of the art like environment, multiple other support modules should exist. These modules are either used by the detection methods (e.g. storage mechanisms) or are expected in the current C–ITS (e.g. pseudonym change policies ...). Figure 3.2 provides a visual index for these different modules. A detailed explanation is available in sections 3.3.3 and 3.3.7.

3.3.2 Framework input data

The first input required for the framework is a SUMO scenario. We do provide three scenarios out-of-the-box. The scenarios include various network topologies with a downtown area, residential roads and main arterial roads linked to highways.

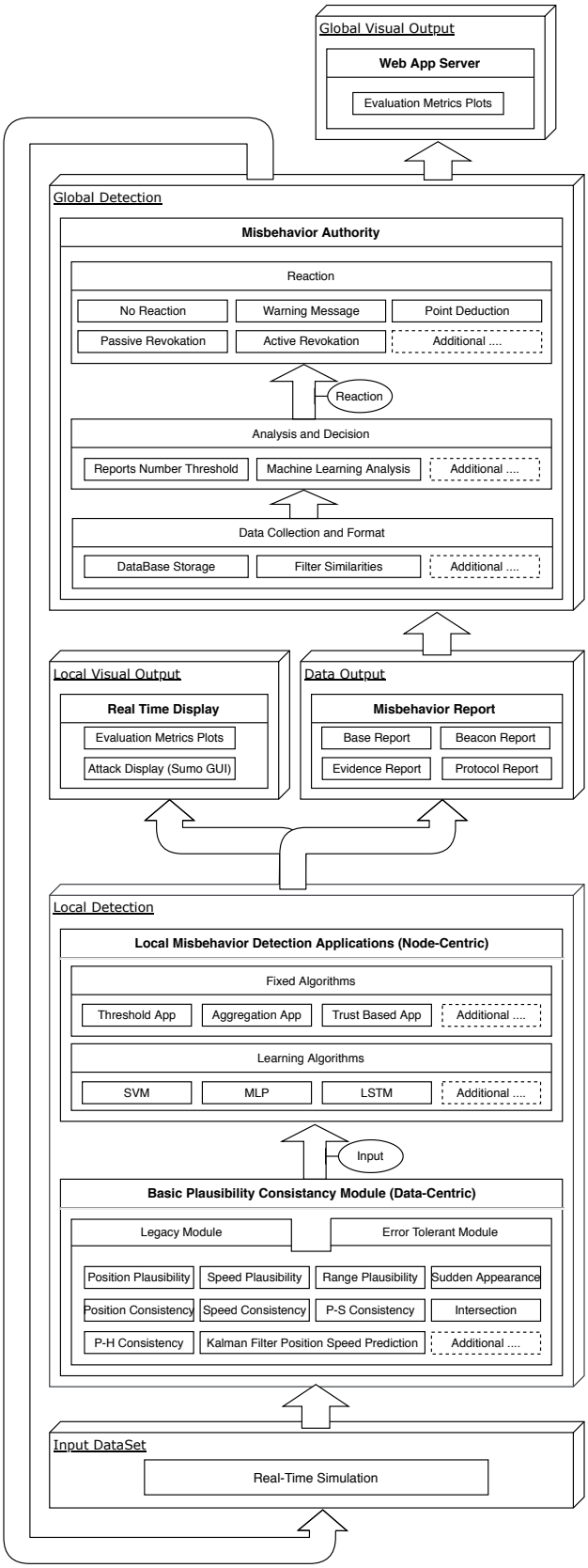


Figure 3.1: F^{2}MD diagram representation of the main modules

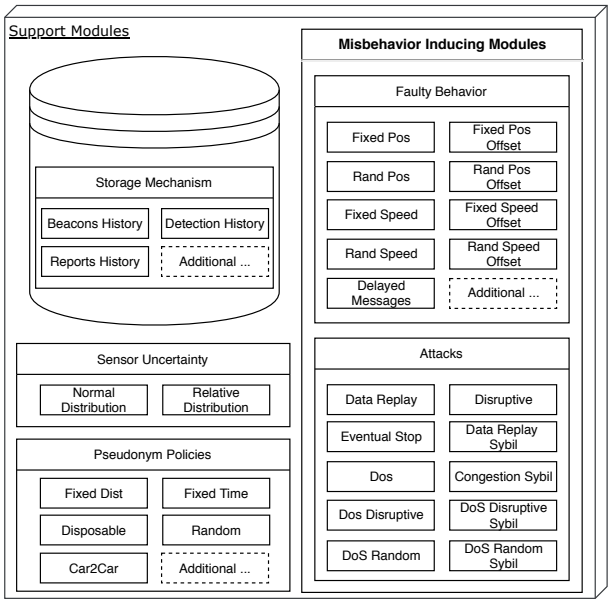
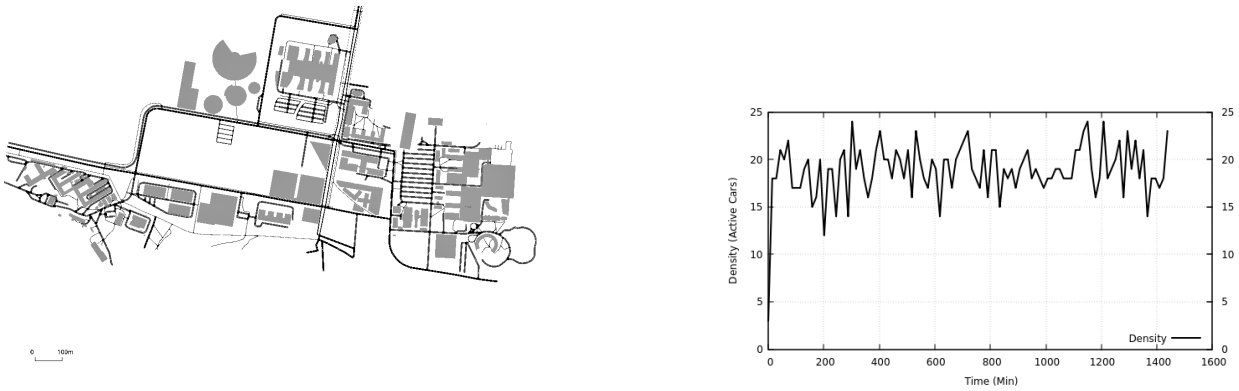


Figure 3.2: F^{2}MD diagram representation of the secondary modules

The first scenario is from the Paris-Saclay area (see figure 3.3). This network combines some suburban-like grid and some organic network properties. It is a relatively small network with randomly generated vehicle traces. Therefore, the vehicle density is somewhat stable. The network size is *1.11 km*^{2} and of density varying around *17.1 Vehicle/km*^{2}. In total, this scenario contains *12,542* vehicles with *8,475,371* exchanged V2X messages. We usually use it as our test bench for calibration and fine tuning since it provides fast and predictable results.



(a) Network (b) Vehicle Density

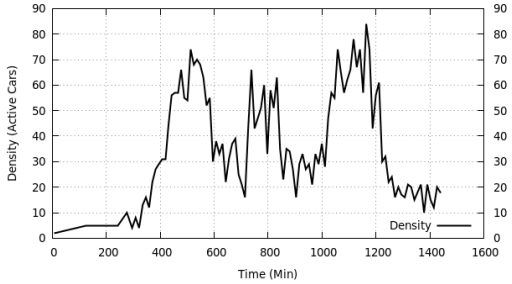
Figure 3.3: F^{2}MD Paris-Saclay scenario

The second scenario is the Luxembourg SUMO Traffic (LuST) [96] (see figure 3.4). This scenario is a SUMO network based on population census data and real traffic information of Luxembourg. Therefore, the vehicle density is somewhat realistic with morning and evening peaks. This scenario is provided in a Small and a Large version.

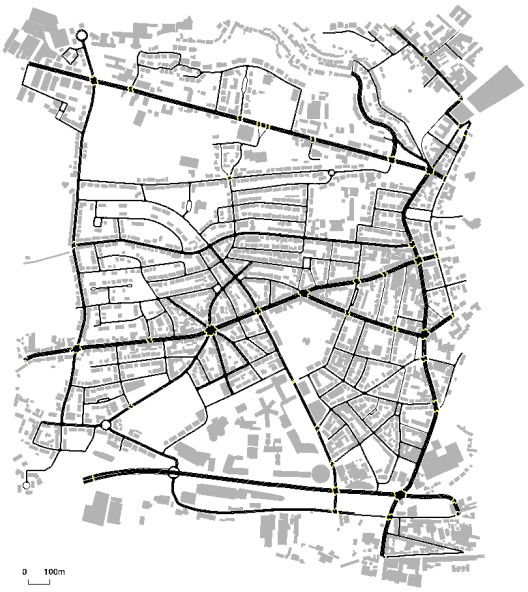
This would enable the user to choose the most suitable network size for his use case. In addition, both versions would constitute excellent training and test sets to be used in the evaluation of machine learning models.

The small LuST network has an area of *1.61 km*^{2} with a variable density peaking at *67.4 Vehicle/km*^{2}. In total, this small scenario contains *24,663* vehicles with *17,097,930* exchanged V2X messages. The large LuST scenario has a network size of *6.51* km^2 and a peak density of *104.5 Vehicle/km*^{2}. The large scenario contains *82,146* vehicles with *301,082,858* exchanged V2X messages.

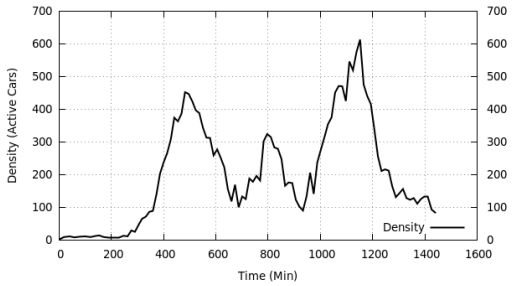
(a) Small Network



(b) Small Vehicle Density



(c) Large Network



(d) Large Vehicle Density

Figure 3.4: F^{2}MD Luxembourg scenarios

In addition to the SUMO scenario, the VEINS simulation requires an OMNeT++ configuration. The OMNeT++ configuration includes beacon parameters such as the header bit length and the beacon interval. It also includes the Network Interface Card (NIC) settings such as the txPower, bitrate, Recall and thermalNoise. We strongly

recommend leaving these settings as the VEINS defaults. The values could be found in the OMNeT++ configuration file on our github. Finally, our framework requires specific inputs like the attack types, the attacker density, the report format and the Pseudonym Change Policies (PCP). This list is not exhaustive but gives a general idea of what types of input is expected. Some inputs are even specific to the type of attack or the PCP. For example, if we choose a periodical pseudonym change, then we have to set the mean change period. These inputs could also be modified in the OMNeT++ configuration file. A full list of inputs is included in our Github repository [26].

3.3.3 Misbehavior mechanisms

In order to assess the quality of different detection methods, we need to introduce misbehavior into the system. For this reason, we implemented two categories of misbehavior: Faulty behavior and Attacks. Faulty behavior are errors on the sensor data, and Attacks are more elaborate and intentional schemes. The framework injects a predefined percentage of misbehaving vehicles. These vehicles could all present one type of misbehavior or could mix multiple types. Details of all the implemented misbehavior mechanisms are presented below.

3.3.3.1 Faulty behaviors

Each vehicle should include an on-board treatment of the data to ensure their plausibility before transmission. However, this preventive system could possibly lack some use cases and is prone to failure, especially in the case of budget vehicles. Here, we consider the case where such an on-board data pre-treatment system fails. We extracted from the literature a set of possible faulty behaviors [97]. The following set was implemented in the framework:

- **Fixed Position:** The ITS–S broadcasts a faulty fixed position (X,Y)
- **Fixed Position Offset:** The ITS–S broadcasts its real position with a fixed offset (ΔX,ΔY)
- **Random Position:** The ITS–S broadcasts a faulty limited random position ($rand(X_{min} \mapsto X_{max}), rand(Y_{min} \mapsto Y_{max})$)
- **Random Position Offset:** The ITS–S broadcasts its real position with a limited random offset ($\Delta(0 \mapsto X_{max})$,$\Delta(0 \mapsto Y_{max})$)
- **Fixed Speed:** The ITS–S broadcasts the same faulty fixed speed (V_x)
- **Fixed Speed Offset:** The ITS–S broadcasts its speed with a fixed offset (ΔV_X)
- **Random Speed:** The ITS–S broadcasts a faulty limited random speed ($\Delta(0 \mapsto V_{max})$)
- **Random Speed Offset:** The ITS–S broadcasts its real speed with a limited random offset ($\Delta(0 \mapsto V_{max})$)
- **Delayed Messages:** The ITS–S broadcasts its information delayed from reality (Δt)

3.3.3.2 Attacks

Our attack schemes vary in complexity. The following list details what is currently implemented in our framework:

- **DoS:** The ITS–S sends V2X messages at a higher frequency than what is defined in the standard. The frequency increase inflicts an overhead on the broadcasting channel. This may render the channel unusable by other vehicles.
- **DoS Random:** The ITS–S performs a *DoS* attack while simultaneously randomizing all the V2X messages data.
- **DoS Random Sybil:** The ITS–S performs a *DoS* attack while simultaneously randomizing all the V2X messages data.
- **Disruptive:** The ITS–S records the data broadcasted by neighbor ITS–Ss. The attacker then proceeds to broadcast V2X messages with data derived from previously received beacons. Given that the falsely broadcasted data is initially generated by genuine vehicles, it is plausible on some levels. The intention of this attacker is to flood the network with these type of messages thus deteriorating the quality of the C–ITS
- **DoS Disruptive:** This ITS–S performs a simultaneous *DoS* and *Disruptive* attacks
- **DoS Disruptive Sybil:** This ITS–S performs a *Dos Disruptive* attack while simultaneously changing its pseudonym on each send V2X message
- **Data Replay:** The ITS–S chooses a target and replays its data instantly with a certain minor prediction epsilons added. Consequently, for an observer it would seem that there are two vehicles in the same space-time dimension
- **Data Replay Sybil:** This ITS–S performs a *Data Replay* attack while simultaneously changing its pseudonym while changing the target vehicle. This mechanism the ITS–S avoid detection
- **Eventual Stop:** The ITS–S suddenly starts broadcasting a fixed positions and a null speed thus simulating a sudden stop
- **Traffic Congestion Sybil:** The ITS–S uses the previously acquired and stored pseudonyms simultaneously to generate a set of ghost-vehicles. The ghost vehicles data is calculated somewhat intelligently in a grid like matter while avoiding implausibility to simulate a realistic traffic congestion. To this end, the following actions are performed:
  - The position, speed and heading of the ghost vehicles are calculated according to the data of the attacking vehicle or a chosen target vehicle.

- A list of pseudonyms is generated and maintained, one pseudonym per ghost vehicle.
- The beaconing frequency of the attacker is increased according to the number of ghost vehicles.
- The ghost vehicles beacons are multiplexed such that every vehicle is sending one beacon per cycle.

This attack demonstrates the ability to use the framework to (i) manipulate pseudonyms, (ii) increase the beacon frequency on-the-fly and (iii) intelligently calculate the data to serve a specific objective.

- **MA Stress:** This attack does not target local vehicles. Instead, it targets the global entity (i.e. the MA) by sending a large number of fake reports. The reports contain the identities of the attacker's neighboring vehicles. The attacker could choose to change its identity for every report. The attacker could also increase the frequency by which the reports are sent to the MA.

3.3.4 Framework local detection

In this framework, we provide a rich module for local misbehavior detection. This module provides simple methods to customize and test different algorithms using a simple methodology. The local detection logic goes as follows. The system runs basic plausibility and consistency checks on every received message. The results are transmitted to the local misbehavior application installed in each vehicle that decides whether or not to send a report to the MA. Therefore, the local detection could be customized in two locations: the basic plausibility (often called detectors) and the more intelligent detection application (often referred to as data fusion). For this reason, we have implemented multiple versions of the basic checks and multiple misbehavior applications. We also provide a method for real-time machine learning based detection applications.

3.3.4.1 Plausibility checks

Inspired by the literature, we extracted a set of basic misbehavior detection checks. The following checks or detectors were implemented in their legacy version and in an Error Tolerant version we call CaTch. The legacy version is much faster to compute the plausibility checks and returns a binary output to show whether a certain aspect of the message is plausible or not. The CaTch version is generally slower to compute the plausibility checks but returns an uncertainty factor that reflects the scale of the message implausibility. The CaTch detectors is one of our contributions and will be discussed in more detail in section 4.1.

Here is the list of all the local plausibility checks that are implemented:

- **Range plausibility:** Check if the position of the sending ITS–S is inside of the ego ITS–S maximum range (predefined value mapped on the ego ITS–S maximum radio coverage).
- **Position plausibility:** Check if the position of the sending ITS–S is at a plausible place (e.g. on a road, no overlaps with physical obstacles, etc.).

- **Speed plausibility:** Check if the speed advertised by the sending ITS–S is less than a predefined threshold.
- **Position consistency:** Check if two consecutive beacons coming from a same ITS–S have plausible separating distance.
- **Speed consistency:** Check if two consecutive beacons coming from a same ITS–S have plausible acceleration or deceleration.
- **Position speed consistency:** Check if two consecutive beacons coming from a same ITS–S have consistent speed and separating distance.
- **Beacon frequency:** Check if the beacon frequency of a sending ITS–S is compliant with the standards.
- **Position heading consistency:** Check if the positions in two consecutive beacons coming from a same ITS–S correspond to the heading advertised by that ITS–S.
- **Intersection check:** Check if no two beacons coming from two different ITS–S have overlapping locations (i.e. both ITS–S overlap each others).
- **Sudden appearance:** Check if no ITS–S suddenly appeared within a certain range.
- **Kalman filter tracking:** Check if the ITS–S advertised information is within a plausible range of the Kalman filter predicted values [98]. This calculation is proposed in [79]. We propose to use this Kalman calculation to extract the following detector values:

  - Kalman Position Speed Consistency *(kPSC*)*,

  - Kalman Position Consistency *(kPC)*,

  - Kalman Position Acceleration Consistency *(kPAC)*,

  - Kalman Speed Consistency *(kSC)*.

3.3.4.2 Fusion applications

The misbehavior detection applications are the decision making part of the detection logic. They are also referred to as fusion applications since the decision is often based on fusion of multiple factors (the results of the plausibility checks, the node history, etc.). We implemented multiple simple examples. Some of them use a deterministic algorithms and others are based on artificial intelligence. The deterministic algorithms were implemented directly into VEINS while the learning applications are external to the simulation and were implemented in python and accessed through a specific API. We start by detailing the applications based on deterministic algorithms.

Threshold Based App: This is our simple baseline application. An ITS–S is considered misbehaving if a certain message fails at least one of the plausibility checks. A fail is determined if the result of the check falls below a certain threshold (see algorithm 1).

Algorithm 1: Threshold Based Solution

c_x: Check Value, θ: Threshold;

for $c_0 \dots c_n$ **do**

  if $c_i < c_{min}$ **then**

    $c_{min} = c_i$

  end

end

if $c_{min} < \theta$ **then**

  Misbehaving

else

  Genuine

end

Aggregation Based App: This application is based on the node history. The result of the plausibility checks of a certain message are aggregated with the results of the last tn time-steps. An ITS–S is considered misbehaving if the aggregated results falls below a certain threshold. The goal of this application is to reduce false positives with respect to the baseline solution. However, this application is still too simple for real deployment. It is used here also as a bench-marking tool.

Algorithm 2: Aggregation Based Solution

c_x: Check Value, θ: Threshold;

for $c_0 \dots c_n$ **do**

  $c_{sum} = \frac{\sum_{t=1}^{tn} c_i}{tn}$

  if $c_{sum} < c_{min}$ **then**

    $c_{min} = c_{sum}$

  end

end

if $c_{min} < \theta$ **then**

  Misbehaving

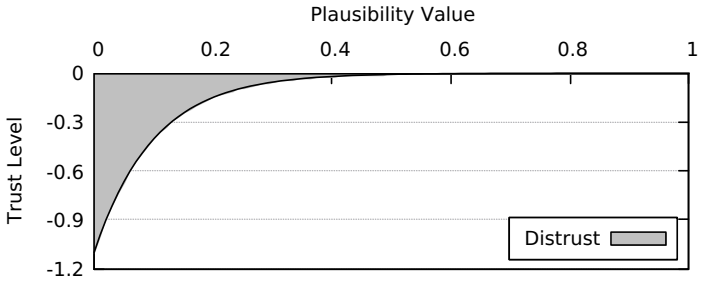
else

  Genuine

end

Non-Cooperative Trust Based App: This application is based on the gravity of the misbehavior event. According to the gravity of the misbehavior, the node is put in timeout. That means all its data is collected and then reported

to the MA. The gravity is characterized by a level of trust in the received V2X messages from a certain ITS–S. A similar approach to the logic used in [62] and in [63]. The trust is derived from the long-term trust level combined with the current calculated plausibility factors. The trust has a negatively exponential relationship with the plausibility factor (see equation 3.1 , figure 3.5). This negative exponential relationship was found by interpolating the trust plot that lead to the best accuracy on a large pre-collected data sample. Finally, a message is considered misbehaving if the cumulative global trust level falls below a certain value (see algorithm 3). Note that when a vehicle behaves correctly its trust level automatically begin rising again, albeit at a slow rate.



$$Trust(x) = -\frac{e^{(10\times(1-x))}+1}{2\times 10^4} \tag{3.1}$$

Figure 3.5: The exponential variation of the trust level function of the plausibility value

Algorithm 3: Non-Cooperative Trust Based Solution

c_x: Check Value, θ: Threshold, T_L: Long-Term Trust;

for $c_0 \dots c_n$ **do**
  if $c_i < c_{min}$ **then**
    $c_{min} = c_i$
  end
end

$T_S = -\frac{e^{(10\times(1-c_{min}))}+1}{2\times 10^4}$

if $T_S > -\varepsilon$ ***and*** $T_L < 0$ **then**
  $T_L = T_L + 0.1$
else
  $T_L = T_L + T_S$
end

if $T_L < \theta$ **then**
  Misbehaving
else
  Genuine
end

Cooperative Trust Based App: This application is based on the cooperation of ITS–Ss. The goal of this solution is to cooperatively evaluate the behavior of a node to determine a shared level of trust in this node. This approach

is similar to the one used in [76] and in [77]. The trust is calculated identically to the case of *Non-Cooperative Trust Based*. However, the global trust levels are shared between all the ITS–Ss of the network.

Machine Learning Based Apps: This application is based on the use of a machine learning model. The goal of this solution is to train an algorithm to detect if a V2X message is misbehaving.

One of the most used programming ecosystems for Machine Learning and Deep Learning is Python. We designed this part of the framework such that anyone could extend the framework with their own Machine Learning models. We have developed an interface between Python and C++ parts of our framework. This interface allows any developer to implement their Machine Learning models in Python and let the core framework calls their models during simulations.

The Python-C++ interface is designed around a typical machine learning model development process. It is divided into offline and online phases. All files are represented as separate modules in the figure. This means that an ML model is divided into two files. One file for each phase. In the offline phase, a developer can design, train and save her ML model. We support scikit-learn [99], keras [100] and Tensorflow [101] for machine learning modeling. The joblib library [102] is then used for saving the trained models. Joblib is much more efficient in saving models that use NumPy [103] than Python's built-in module pickle.

In the online phase, the ML model runs inside an HTTP server listening on a user-defined port. The simulation core (written in C++) calls the ML model (written in Python). The simulation core sends data to be tested against the ML model in an HTTP request. The server calls the ML prediction script which performs operations as follows:

- Load the data from HTTP request that needs to be processed for prediction.
- Load the model saved during offline phase. We use joblib to load the model.
- Respond to the HTTP request with prediction from the trained model.
- Optionally, the prediction script could update the model and save it back to the loaded model. Models that use back-propagation may need such functionality.

Finally, simulation core reads the prediction output from the HTTP response to perform further investigation. Based on the previous interface, we implemented a simple example with the plausibility checks as input features similarly to [87], and a simple Genuine or Misbehaving classification as output. We also included the following example learning techniques:

- **SVM Classifier:** We used the scikit-learn library to add an example of the C-Support Vector Classification (SVC) model. The SVC implementation is based on LibSVM, A Library for Support Vector Machines [104].
- **MLP Classifier:** We used the keras library to implement a simple Sequential model with three hidden Dense layers and a Rectified Linear Unit (ReLU) activation. We fitted this model with a learning rate limiter.

- **LSTM Classifier:** We used the TensorFlow library to implement a Bidirectional LSTM model that could be trained with variable time step. We also included a fit generator and train generator for the input data.

These examples should help researchers get started with any model weather they would like to use the scikit-learn, keras or TensorFlow library.

3.3.5 Reporting Protocol

One important goal of the local detection is to send a report to a central MA for post-processing. In this framework, the fusion algorithm could decide to generate a misbehavior report. The reports are pushed to a global MA via HTTP connection. Additionally, the reports can be collected in JavaScript Object Notation (JSON) or Extensible Markup Language (XML) format in a local folder.

We propose a misbehavior report format inspired by the protocol described in section 5.1. The report is composed of three containers: *Header Container*, *Source Container* and *Evidence Container*. The *Header Container* contains the basic information that should be included in every report: generation time, sender id, reported id and report type. The *Source Container* consists of the results of the plausibility and consistency checks on the reported beacon, granted that a vehicle is reported only after a received beacon has shown some implausibility. The *Evidence Container* should help the MA in its investigation and to support its conclusion. The *Evidence Container* can be composed of messages from the reported or the reporter or any neighboring vehicles if deemed helpful. It could also include some other data like a Local Dynamic Map (LDM), or direct sensor data from the reporter. The Evidence required by the MA is further detailed in [18].

The report format is however not yet standardized and is still a subject of discussion. For this reason, we have found it useful to provide several formats of misbehavior reports to facilitate further investigation and testing. The following versions are implemented in the framework:

- **Base Report:** This format includes only the *Header Container* and the *Source Container* with no evidence.
- **Beacon Report:** This version includes a base report and the reported beacon in the *Evidence Container*.
- **Evidence Report:** This version contains a more complete *Evidence Container* depending on the type of plausibility checks failure. For example, if the vehicle failed the *Speed Consistency* check, we include both inconsistent beacons as evidence.

In the previous three formats, every message flagged as misbehaving is reported. However, not every message should be reported individually. This would generate a significant network overhead especially when the vehicle is misbehaving because of a faulty component in its system. Consequently, the report format allows for omitted reports, which means that the vehicle is not constantly reported for the same behavior. This is why we propose the *Protocol Report*.

- **Protocol Report:** This version also contains a complete *Evidence Container*. However, in this version vehicles follow the reporting protocol proposed in section 5.1. The vehicle sends an initial report the instant it detects a certain misbehavior. Then it refrains from reporting the same behavior after a certain time while collecting evidence. The evidence is then sent to the MA in a single follow-up report. This protocol assumes an intelligent MA that prioritizes the content of the received reports instead of their number.

3.3.6 Global misbehavior detection

The Misbehavior Authority (MA) is the global entity that receives the reports sent by the ITS–Ss. The MA should then decide on the suitable reaction to make. We define three main components of the MA:

- **Collection and Format:** The collected report is added to a database. This action would enable the fast access to relevant reports using a certain criterion. For example, we can get all the reports accusing a certain pseudonym or all the reports from a certain region. Those queries are helpful in the analysis phase. We also have a filtering system, that if enabled could aggregate all reports signaling the same implausibly (e.g. sets of two messages with a speed inconsistency).
- **Analysis and Decision:** The MA analyzes the reports and outputs the correct level of reaction. We have implemented a simple method that has a threshold on the number of reports for every reaction level. The number of reports required to reach every level is modifiable. We set the output as levels so it would be compatible with our reaction mechanism. However, other outputs could be developed. Please note that this component will evolve into a more complex element in future versions of the framework.
- **Reaction:** The misbehavior reaction is still a widely debated subject. We propose a level-based solution with 5 levels of reaction:
  - level 0: no reaction
  - level 1: a warning message is sent to the vehicle
  - level 2: a warning point is deducted from the vehicle's score
  - level 3: passive revocation where the vehicle cannot request more certificates
  - level 4: active revocation where the current certificates of the vehicle are revoked.

  Currently, the reactions do not cause a change in the behavior of the vehicle, however we expect to have a more intelligent system where the vehicle would change its behavior according to the reaction level (e.g. a vehicle with a faulty sensor would re-calibrate upon a warning from the MA).

Please note that this global solution for misbehavior detection that is included in this framework is only a draft proposal. We included the code of this proposal with the simulator to help the scientific community get started with research in this domain using our simulator. Our contribution to this research field are discussed in chapter 6.1.

3.3.7 Privacy

The use of pseudonyms has been included in the IEEE and ETSI standards [10, 11, 95]. However, when and how a pseudonym change happens is still a research challenge. Scientific studies have suggested multiple methods to determine the location and rate of change of pseudonyms [12].

Node-centric detection mechanisms rely on a consistent identity of the treated vehicles. This approach is greatly affected by privacy-preserving mechanisms based on pseudonyms. For this reason, we have implemented the following Pseudonym Change Policies (PCP) in our framework:

- **Periodical:** The vehicle changes its pseudonym after a predefined period of time.
- **Distance:** The vehicle changes its pseudonym after predefined number of kilometers.
- **Disposable:** The pseudonym is used for a fixed number of messages (including beacons and warnings).
- **Random:** The pseudonym has predefined chance of change at every sent message.
- **Car2car:** The first pseudonym change happens between 800 and 1500 meters after ignition. Afterwards, each 800 meters will trigger a change in 2 to 6 minutes. A version of this policy was initially proposed in the CAR 2 CAR Communication Consortium Basic System Profile [105]. However, it is no longer included in the subsequent versions of this document.

3.3.8 Evaluation metrics

In a given scenario, a *vehicle* transmitting messages could be misbehaving or genuine, and a local detection mechanism could classify *messages* as misbehaving or genuine. Therefore, as illustrated in Table 3.1, the evaluation of detection mechanisms could be partitioned into four groups: True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (PN) (see Table 3.1). Using this partition, we are able to calculate the following performance metrics for classification problems usually used in the evaluations of machine learning applications: *Accuracy*, *Precision*, *Recall*, F_1 *score*, *Bookmaker Informedness*, *Markedness*, *Matthews Correlation Coefficient* and *Cohen's kappa*.

Table 3.1: Detection output partition

	Genuine	**Misbehaving**
Detected	FP	TP
Not Detected	TN	FN

The *Accuracy* (ACC) (3.2) is the rate of positive agreement, which in our case refers to the ratio of true detection in the system. It does not constitute a good detection metric in the case of an unbalanced data, i.e. more genuine nodes exist than attackers.

$$Accuracy(ACC) = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.2}$$

The *Precision* (3.3) measures the proportion of messages correctly flagged as misbehaving out of all flagged messages. It indicates the classifiers ability to distinguish between misbehaving and genuine nodes, for example a low precision means the system is yielding a lot of false positives.

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

The *Recall* (3.4) measures the proportion of correctly identified misbehaving messages out of all received misbehaving messages. It marks the classifiers ability to detect a misbehaving node, i.e. a low recall means an attack is difficult to detect.

$$Recall = \frac{TP}{TP + FN} \tag{3.4}$$

The F_1*score* (3.5) is the harmonic mean of *Recall* and *Precision*. It could be used as a single metric to evaluate the system's performance if we attribute the same importance to the *Recall* and *Precision*. If needed, we can attribute more weight to one metric by calculating an F_β*score*. This metric could be interesting since one could argue that *Recall* is more important than *Precision* in some cases.

$$F_1 score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{3.5}$$

The *Bookmaker Informedness* (BM) (3.6) characterizes the probability of an informed decision. It shows how much the decision of this system is better than a random guess.

$$BM = \frac{TP}{TP + FN} + \frac{TN}{TN + FP} - 1 \tag{3.6}$$

The *Markedness* (MK) (3.7) is the probability that the detection is ascertained by the classification as opposed to by chance.

$$MK = \frac{TP}{TP + FP} + \frac{TN}{TN + FN} - 1 \tag{3.7}$$

The *Matthews Correlation Coefficient* (MCC) (3.6) is the geometric mean of the *Informedness* and the *Markedness*. It is especially useful when the measured classes are of very different sizes, which is often the case with C–ITS attackers.

$$MCC = \frac{TP \times TN + FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.8}$$

Cohen's kappa (K) (3.9) is a measure of the positive agreement, similar to the $Accuracy$, but where we subtract the agreement by chance.

$$K = \frac{ACC - \frac{(TP + FP) \times (TP + TN) + (TN + FP) \times (TN + FN)}{(TN + TP + FP + FN)^2}}{1 - \frac{(TP + FP) \times (TP + TN) + (TN + FP) \times (TN + FN)}{(TN + TP + FP + FN)^2}} \tag{3.9}$$

3.3.9 Visualization

One of the goals of our framework is to facilitate the evaluation of any detection mechanisms or any changes that could affect the detection rate. In order to achieve that, the simulator writes a snapshot of the current state of the running mechanisms at every time interval. Then, a script parses the data, calculates, and plots the aforementioned evaluation metrics in real-time. To further facilitate the evaluation and comparison of mechanisms, the simulator and the script support running two simultaneous mechanisms on the same system. Figure 3.6 shows a real-time comparison between two different detection applications running on top of a different set of checks.

However, the visualization is not limited to the detection results. The attacks and detection system are also visualized in SUMO's GUI. The simulator uses SUMO's Traffic Control Interface (TraCI) [106] to color the vehicles according to the intended role as specified in Figure 3.7.

Finally, since the MA is implemented as an HTTP server, we provide a web interface. The web interface runs in real time and serves as a display of different metrics and evaluations on the current state of the MA (Figure 3.8). We currently display three metrics:

- Cumulative and instantaneous prediction accuracy.
- Number of received reports per pseudonym for some of the most relevant identifiers.
- Radar chart of the cumulative percentage of the issued reactions.

3.4 Examples

In order to demonstrate the capabilities of the framework, we run multiple example scenarios. Each example is based on showcasing the capabilities of a different module of the framework. In the following scenarios we use our

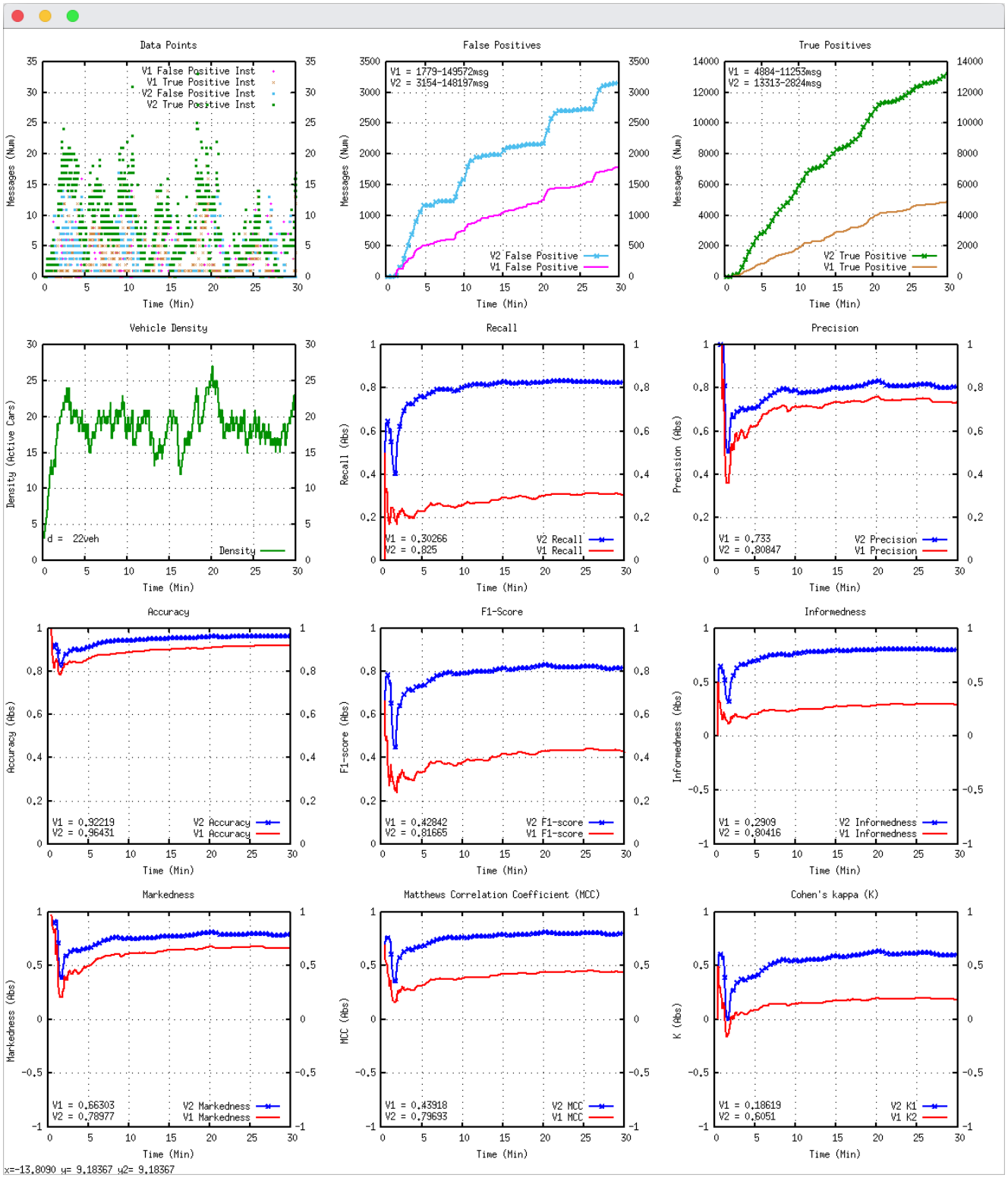


Figure 3.6: F^{2}MD GUI: Real-Time Evaluation Metrics Plots (Data Points, TP, FP, Density, Recall, Precision, Accuracy, F_1score, BM, MK, MCC, K)

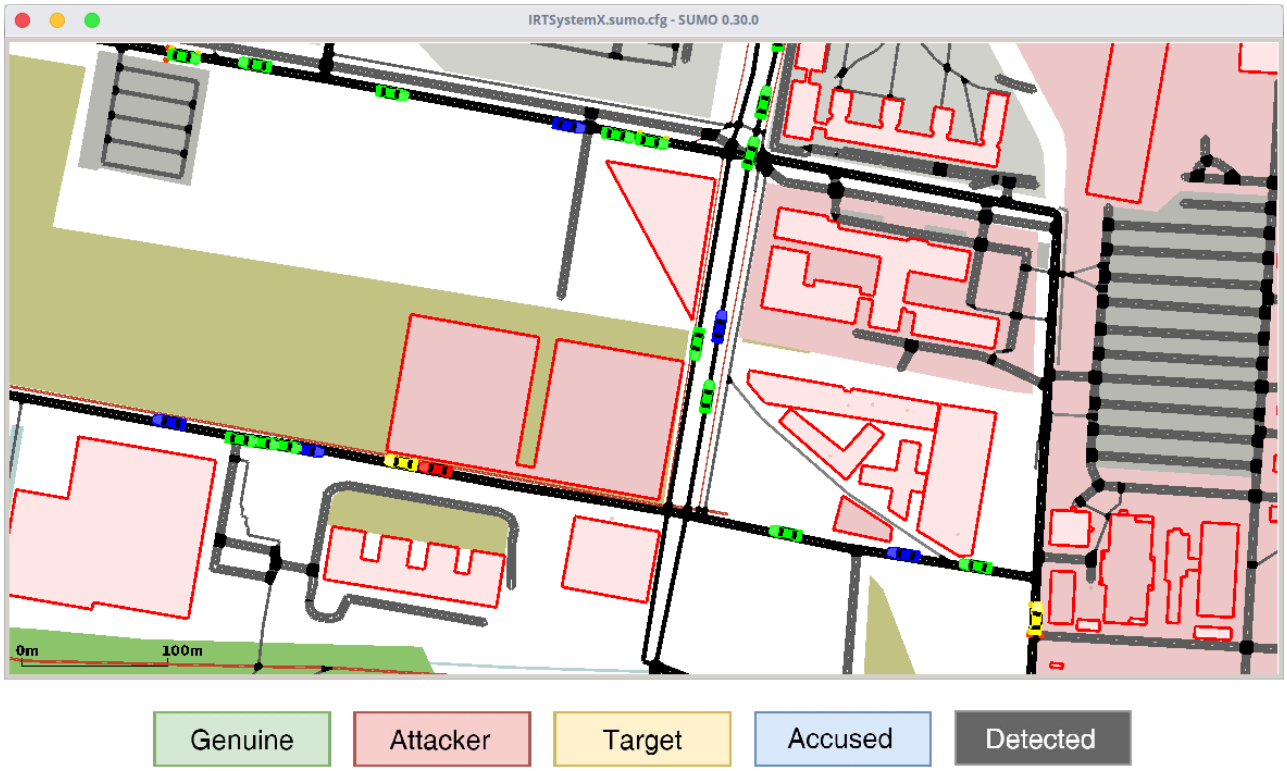


Figure 3.7: F^{2}MD GUI: Vehicle Color Profiles with SUMO

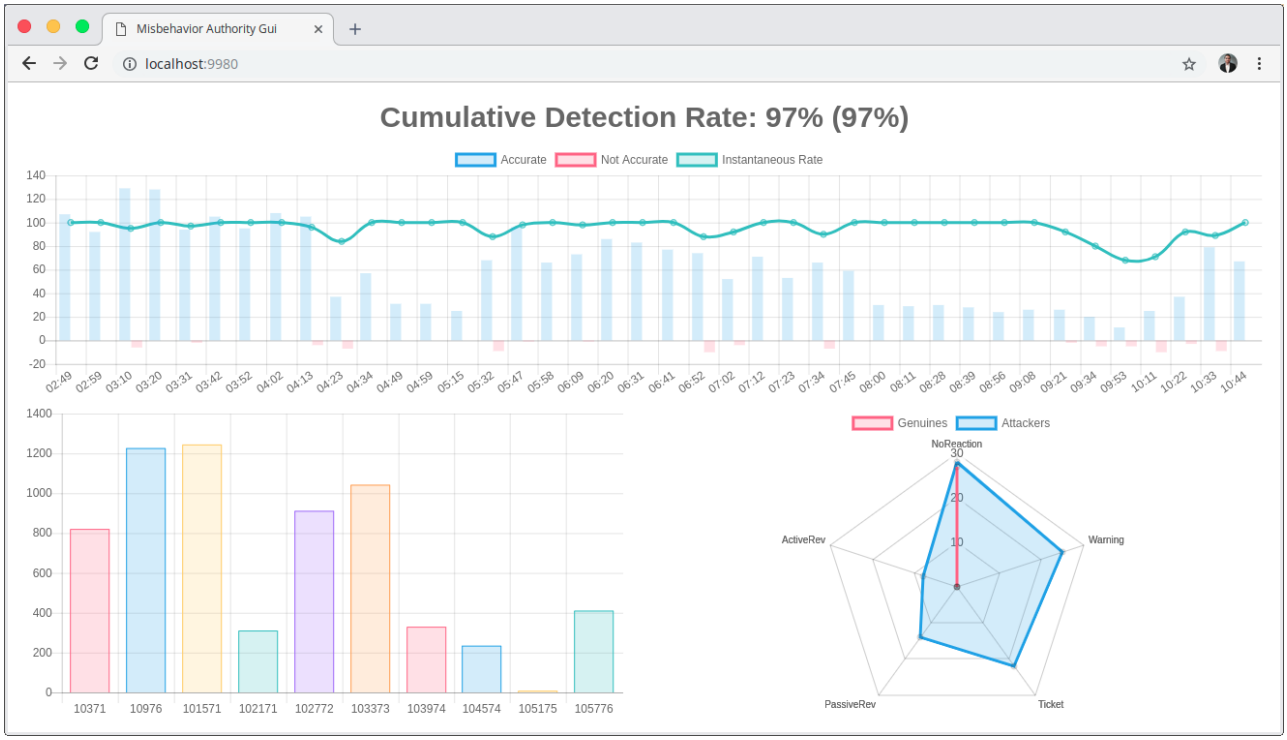


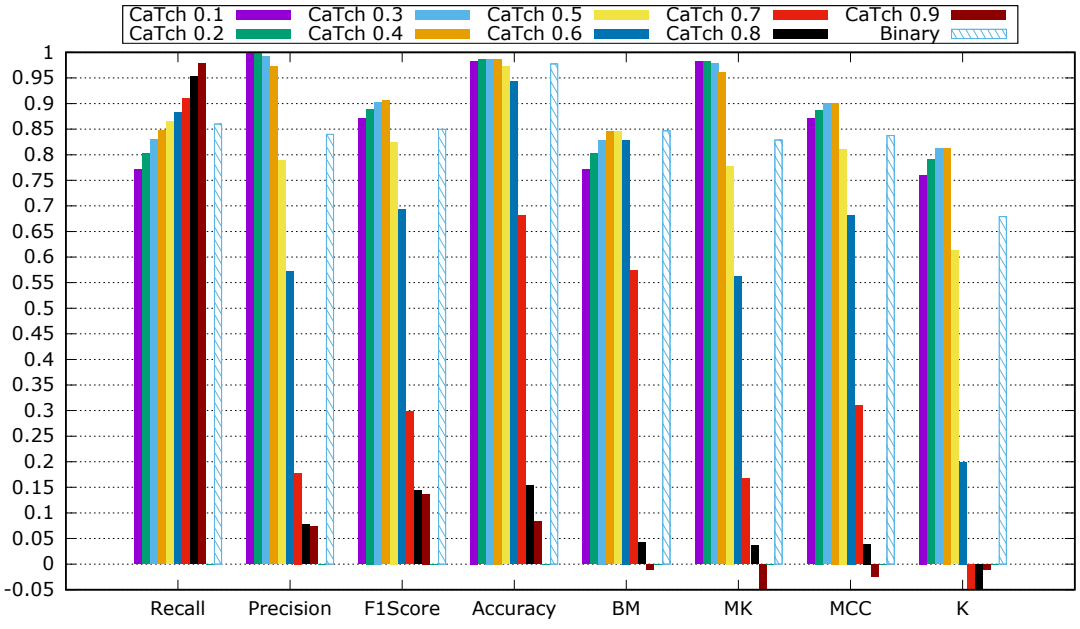
Figure 3.8: F^{2}MD GUI: MA web interface

benchmarking Paris-Saclay scenario network described in Section 3.3.2. We introduce misbehaving entities with a density of 5%. Each misbehaving vehicle introduced to the system randomly chooses its type of attack as detailed in Section 3.3.3. The following results are reproducible using the implementation and scenarios provided on our Github [26].

3.4.1 Plausibility detectors example

As discussed earlier, we implemented two versions of the detectors. The legacy version outputs a binary value while the CaTch version assigns a factor in uncertain scenarios. The plausibility factor is a score assigned to the each plausibility check done on the message. This score is calculated using the value and the error range advertised for each field in a certain message. The score varying between completely implausible (0) and definitely plausible (1).

To better understand the effect of this value, we run both versions of the detectors simultaneously coupled with a threshold-based detection application. We vary the threshold between 0.1 and 0.9. As shown in Figure 3.9, we

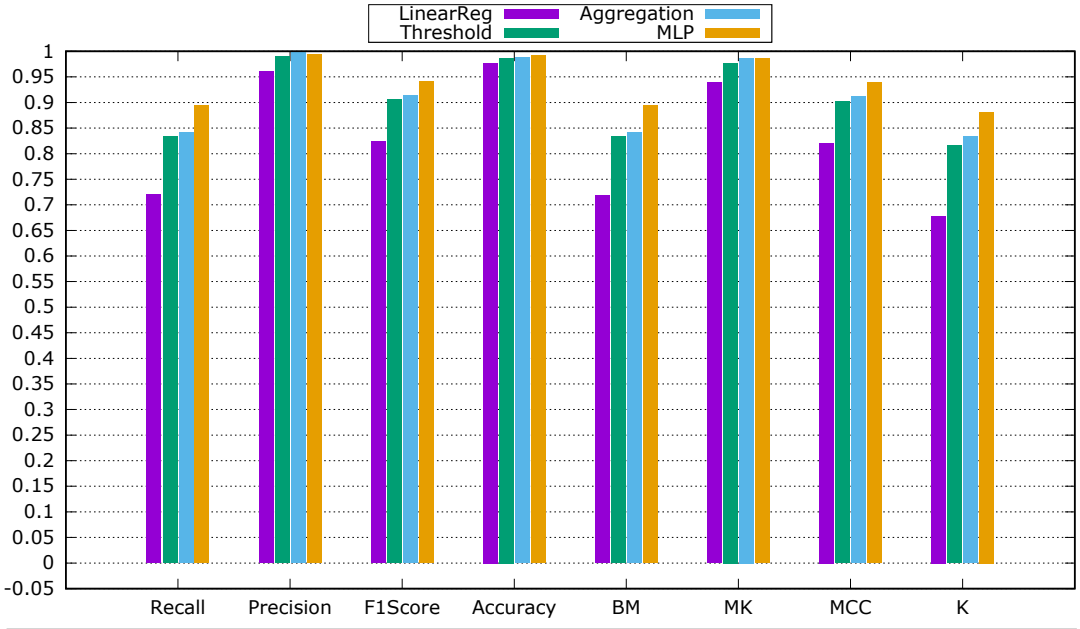


Misbehavior Types	Evaluation Metrics							
	Rec	**Pre**	**F1s**	**Acc**	**BM**	**MK**	**MCC**	**K**
CaTch 0.1	0.7713	0.9995	0.8707	0.9830	0.7712	0.9814	0.8700	0.7603
CaTch 0.2	0.8023	0.9974	0.8893	0.9852	0.8021	0.9818	0.8874	0.7902
CaTch 0.3	0.8290	0.9913	0.9029	0.9868	0.8284	0.9778	0.9000	0.8121
CaTch 0.4	0.8466	0.9728	0.9053	0.9869	0.8447	0.9606	0.9008	0.8130
CaTch 0.5	0.8644	0.7885	0.8248	0.9727	0.8459	0.7776	0.8110	0.6127
CaTch 0.6	0.8815	0.5710	0.6931	0.9420	0.8284	0.5611	0.6817	0.1991
CaTch 0.7	0.9098	0.1781	0.2978	0.6815	0.5730	0.1673	0.3096	-0.9083
CaTch 0.8	0.9520	0.0774	0.1431	0.1537	0.0417	0.0362	0.0388	-0.0858
CaTch 0.9	0.9787	0.0735	0.1368	0.0829	-0.0103	-0.0607	-0.0250	-0.0110
Binary	0.8598	0.8398	0.8497	0.9774	0.8467	0.8285	0.8376	0.6788

Figure 3.9: F^2MD results: variable checks threshold evaluation plot

can see a clear trade-off between precision and recall for the CaTch version of the detectors. This validates our hypothesis that the uncertainty factor quantifies the plausibility of a certain message. The plausibility factor therefore provides a more informative view of the implausible scenario. This could prove useful to an intelligent application trying to detect an attack. More insights and results on the CaTch detectors could be found in section 4.1.

3.4.2 Local fusion application example



Misbehavior Types	Evaluation Metrics							
	Rec	**Pre**	**F1s**	**Acc**	**BM**	**MK**	**MCC**	**K**
LinearReg	0.72004	0.96014	0.82294	0.97701	0.71765	0.93815	0.82052	0.67778
Threshold	0.83332	0.99020	0.90501	0.98687	0.83265	0.97684	0.90187	0.81545
Aggrigation	0.84187	0.99941	0.91390	0.98809	0.84183	0.98673	0.91141	0.83246
MLP	0.89379	0.99359	0.94105	0.99169	0.89333	0.98515	0.93811	0.88097

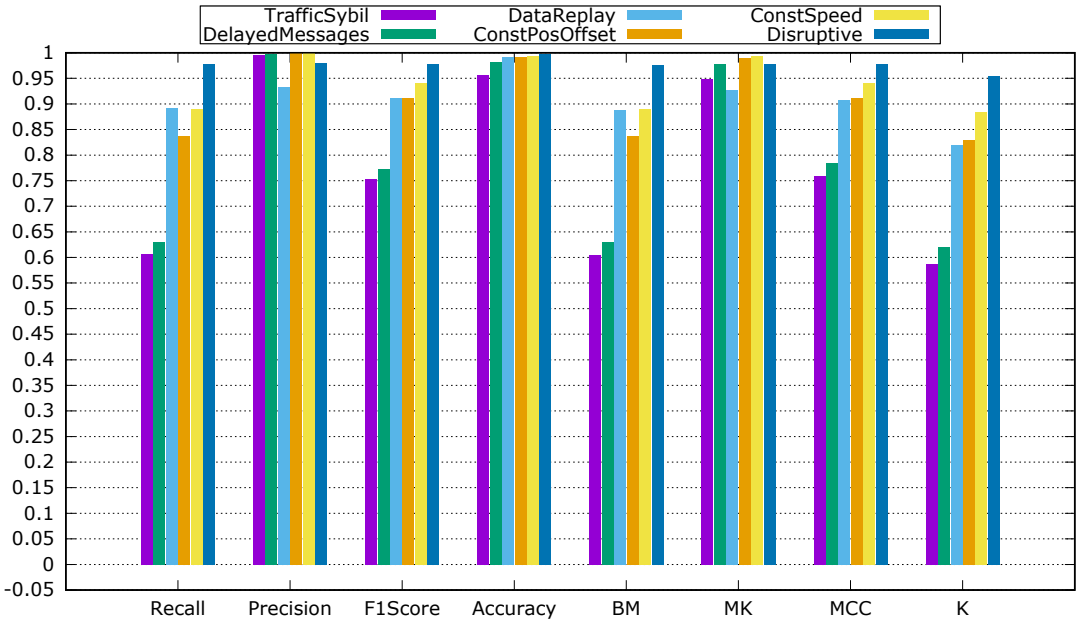
Figure 3.10: F^2MD results: local fusion application evaluation plot

Next, we demonstrate the capabilities of the fusion applications module. We tested some of the fusion applications available with our framework. We tested the Threshold and Aggregation approach as well as some machine learning models, the MLP and a simple Linear Regression. The results of the plausibility checks are used as the input features for the learning models.

Figure 3.10 shows the evaluation metrics of the different detection scenarios. These metrics are collected from plots of the GUI described in Section 3.3.9. One interesting note about the results is that not all machine learning algorithms outperform the fixed ones. The algorithm should be carefully chosen to be adapted to the application and Linear Regression is not suitable for our use case. More insights and results on the fusion applications could be found in section 4.2.

3.4.3 Attacks example

Until now, our evaluations were based on an even mix of misbehavior types. However, the detection rate is strongly dependent on the type of misbehavior. To demonstrate this fact, we tested the ability our benchmark fusion application, the simple threshold, to detect individual attacks. Figure 3.11 shows that the traffic Sybil and the delayed messages misbehavior types are somewhat harder to detect with our detectors. This result is in line with our expectations since the messages transmitted here are have a high plausibility. For instance, the delayed message contains information that was correct just few seconds earlier. Therefore, the delayed messages do not have many implausible features. One should note that the SAE J2945/1 [107] specifies that BSM generated 30 seconds in the past (and in the future) pass time relevance check. On the other hand, we see that the Disruptive attack is the easiest to detect. This type of attack replays a mass of messages without data consistency. Consequently, there is a large amount of implausible features that were easily detectable with our plausibility checks.



Misbehavior Types	Evaluation Metrics							
	Rec	**Pre**	**F1s**	**Acc**	**BM**	**MK**	**MCC**	**K**
TrafficSybil	0.6054	0.9949	0.7527	0.9569	0.6050	0.9491	0.7578	0.5858
DelayedMessages	0.6305	0.9966	0.7724	0.9810	0.6304	0.9771	0.7848	0.6207
DataReplay	0.8918	0.9327	0.9118	0.9912	0.8883	0.9269	0.9074	0.8202
ConstPosOffset	0.8367	0.9990	0.9107	0.9916	0.8367	0.9903	0.9103	0.8298
ConstSpeed	0.8892	0.9993	0.9410	0.9943	0.8892	0.9934	0.9398	0.8839
Disruptive	0.9775	0.9797	0.9786	0.9978	0.9764	0.9785	0.9775	0.9551

Figure 3.11: F^2MD results: detection evaluation by attack

3.4.4 Reporting and global detection example

Finally, we demonstrate the reporting and the global detection of the framework. We start by examining the different type of reports. Our different report formats have different containers and should vary in size. Using the same scenario as in Section 3.4.3, we measure the size of the reports transmitted to the MA in JSON. This measurement is done for each of the formats described in Section 3.3.5. Table 3.2 shows the average sizes of the different reports. We can see that the sizes are consistent with the reports format, and the differences are smaller when the information is compressed. Please note that the sizes are only used for comparison purposes, however, these are not to scale with a real eventual report. All C–ITS messages of ETSI and IEEE have a security header a signature and multiple other layer that are not included here [107] [44].

Table 3.2: Average report size comparison

Report Type	Report Size (Bytes)	
	Uncompressed	lzma Compressed
Base	512.45	313.27
Beacon	1090.00	479.73
Evidence	1979.84	545.89

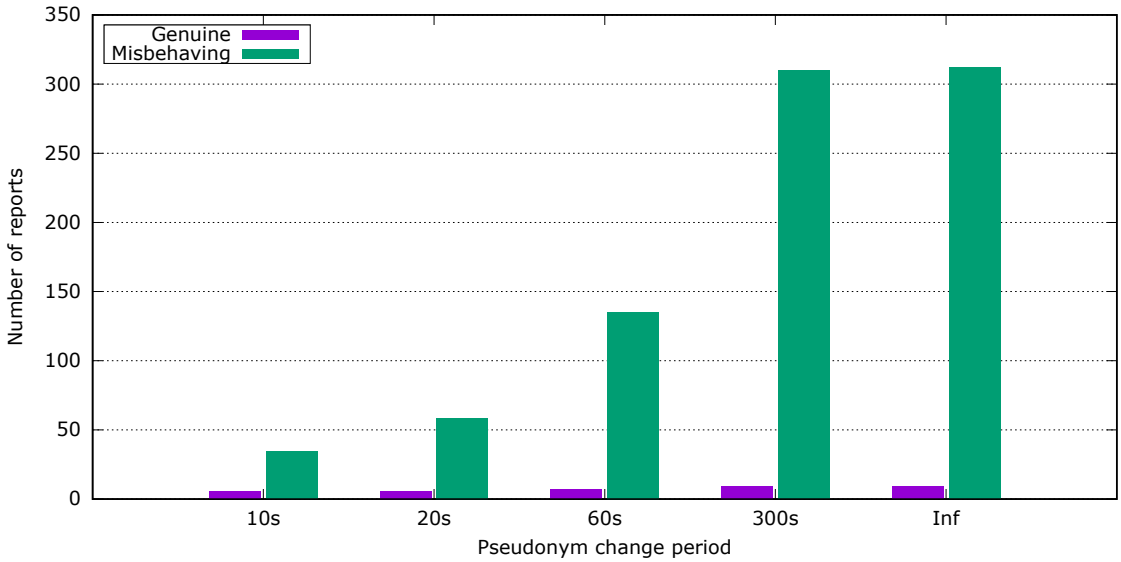


Figure 3.12: F^{2}MD results: average number of reports by pseudonym received by the MA for different pseudonym change period

For the global detection, our current simple MA is still based only on the number of received reports for a certain pseudonym. Consequently, a PCP would dramatically affect the result. To show this effect, we enable a periodical PCP. We then simulate the same scenario with multiple pseudonym change periods. Figure 3.12 shows the average number of reports received by the MA for genuine and misbehaving nodes per pseudonym. We can see that the

number of reports is significantly affected by more frequent pseudonym change. Indeed, the MA does not receive enough reports with the same reported ID, certainly affecting the detection quality.

In order to mitigate this effect, the MA can analyze the contents of the report. A report with *Evidence Container* is advantageous for this process. Additionally, currently proposed reporting protocols do not send a report for each message. Instead, to reduce overhead, the vehicle collects evidence and then sends a more complete report. This process would make the global detection based on the number of reports obsolete. The need for an intelligent MA is therefore paramount. More details on the reporting protocol and format could be found in section 5.1.

3.5 Conclusion

Cooperative Intelligent Transport Systems are susceptible to false data injection attacks that could jeopardize road users' safety. In this chapter, we proposed a simulation framework and source code called F^2MD, which enables the research community to develop, test, and compare misbehavior detection algorithms. We implemented in the framework (i) a comprehensive list of attacks, (ii) an extensive set of basic and advanced detection algorithms, (iii) a Python/C++ bridge to allow import of artificial intelligence algorithms, (iv) basic Pseudonym Change Policies, (v) a visualization tool to analyze real-time performance of the misbehavior detection system, and (vi) a Misbehavior Authority and Misbehavior Report formats. This framework is used to evaluate the solutions and contributions that we propose throughout this thesis.

Chapter 4

Local Misbehavior Detection

This chapter includes the contributions to the local component of the misbehavior detection model described in section 2.4.3. In section 4.1, we introduce a new set of error tolerant local detectors. These detectors integrate the confidence range of embedded sensors to improve the detection quality. In section 4.2, we extract misbehavior fusion application from the state of the art. We re-implement the solutions on our F^2MD simulator and we compare their performance on a common scenario. In section 4.3, we introduce improvements to VeReMi, a dataset of V2X messages that could be used by researchers to evaluate and compare their local detection solutions.

4.1 CaTch detectors

The standard ETSI CAM [7] and IEEE BSM [9] messages integrate a field called *confidence range* for each mobility parameter. This field is included based on the fact that sensor measurements could be inaccurate due to physical limitations or environmental characteristics. However, to the best of our knowledge, so far this information is not taken into consideration during the misbehavior detection checks computation.

We introduce CaTch (Confidence range Tolerant misbehavior detection approach), a misbehavior detection library which takes into consideration the *confidence range*. Then we show through extensive simulations, that taking into consideration sensors inaccuracy during the checks computation increases the detection quality of the system.

4.1.1 Local detectors and mobility data uncertainty

4.1.1.1 The local misbehavior detectors

In this work, we consider plausibility and consistency verification of the content of the received beacon message. Note that a *plausibility* check requires only one received message whereas a *consistency* check requires two successive messages coming from the same source. For instance, when performing a plausibility check on the radio

transmission range, a vehicle would judge that a neighbor which sends a position that is not in its communication range is suspicious. A typical example of consistency checks is when a vehicle receives two beacons from its neighbor at different times indicating speed information which are conflicting with the travelled distance information.

A complete set of plausibility detectors for CAM is detailed in Table 5.3. Additionally, the implemented checks are detailed in section 3.3.4.1. However, in this work we only consider the following checks: *Range plausibility*, *Position plausibility*, *Speed plausibility*, *Position consistency*, *Speed consistency*, *Position speed consistency*, *Position heading consistency*, *Intersection check* and *Sudden appearance*.

4.1.1.2 Misbehavior detectors uncertainty integration

In this work, we propose a novel solution, *CaTch*, which takes into consideration the mobility information uncertainty while performing the basic misbehavior checks. In the following sections, we detail how we integrate the confidence range of the beacon message contents in the basic checks calculation.

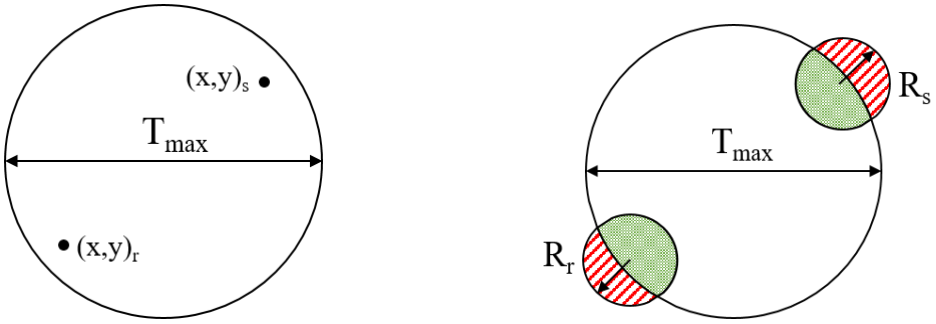
For each plausibility and consistency test, we compare our calculation (CaTch version) with the state-of-the-art one (legacy version). We calculate the uncertainty factor f for each of the misbehavior checks. The uncertainty factor f is a real number between 0 and 1, with 0 being certainly malicious and 1 having no signs of misbehavior. We assign a value of f to the nodes whose data are partially implausible. Notice that the confidence range is illustrated by green for the plausible section and red for the implausible one. We use the following common notations as depicted in Table 4.1.

Table 4.1: Common Notations

R_x	$\triangleq$	Position confidence range in beacon x
V_x	$\triangleq$	Claimed speed in beacon x
C_x	$\triangleq$	Speed confidence range in beacon x
D_x	$\triangleq$	Claimed heading in beacon x
Δt_{ij}	$\triangleq$	Time separating beacons i and j
d_{ij}	$\triangleq$	Distance separating beacons i and j
A_x	=	πR_x^2

4.1.1.2.1 Range plausibility check

We assume that the communication range is based on a disk model. The initial method to detect an out of transmission range node is done by simply checking if the distance between the sender and the receiver is less than the maximum plausible range as illustrated in figure 4.1a. In our approach, we take into consideration the confidence range of the sender's position broadcasted in the beacon R_s and the confidence range assessed internally by the receiver R_r. We calculate first the position confidence range area of the sender and the receiver, respectively A_s and A_r. We consider T_{max} as the maximum diameter in which the sender and the receiver could communicate. T_{max} coincides with the transmission range of the sender and the receiver. Accordingly, $A_{T_{max}}$, is the area of the



(a) Legacy Version (b) CaTch Version

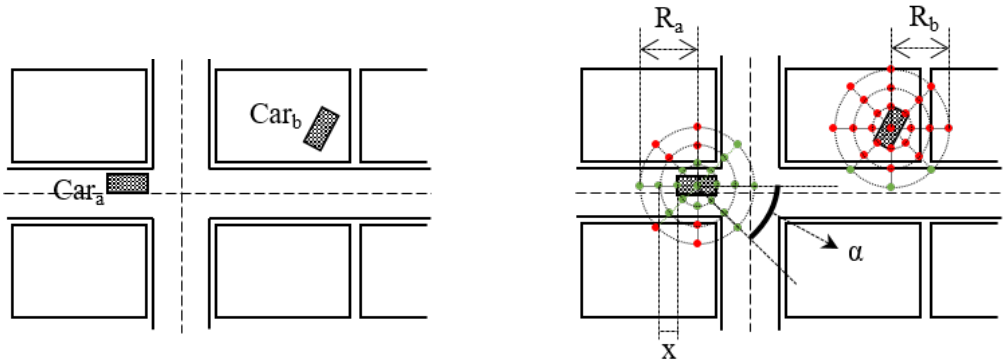
Figure 4.1: Local check: range plausibility

circle formed by T_{max}. Notice that any node which is within $A_{T_{max}}$ could communicate with both the sender and the receiver. Second, we calculate the intersection area a_s and a_r for the sender and the receiver respectively with $A_{T_{max}}$ (the green shaded zone as illustrated in figure 4.1b). Finally, we compute the uncertainty factor of this check as follows:

$$f = \frac{(a_r + a_s)}{(A_r + A_s)}$$

T_{max}	$\triangleq$	Communication Diameter
$A_{T_{max}}$	=	$\frac{\pi T_{max}^2}{4}$
a_r	=	$A_{T_{max}} \cap A_r$
a_s	=	$A_{T_{max}} \cap A_s$

4.1.1.2.2 Position plausibility check



(a) Legacy Version (b) CaTch Version

Figure 4.2: Local check: position plausibility

The position plausibility is determined using a straightforward check between the claimed position and the geographic map of the environment. For instance, in figure 4.2a, Car_b broadcasts a position that is overlapping with a building. This makes its position definitely non-plausible. In CaTch, the position plausibility is calculated for multiple

points that are within the position confidence ranges, R_a and R_b of Car_a and Car_b respectively, as depicted in figure 4.2b. Every segment is separated by an angle α and every point is separated by a distance x. These points are plotted on multiple segments along the radius of the confidence range. The green points in the figure are the possible plausible positions for both Car_a and Car_b and the red points are the non-plausible ones. Let n be the number of green points in the figure and let N be the total number of the considered points.

The uncertainty factor of this check is computed as follows:

$$f = \frac{n}{N}$$

n	$\triangleq$	Plausible Tested Points
N	$\triangleq$	All Tested Points

Note that if a vehicle is positioned outside of a road and its advertised velocity is zero, the check does not fail (i,e., we consider the car as parked). However, if the advertised velocity is different from zero, the test fails.

4.1.1.2.3 Speed plausibility check

(a) Legacy Version (b) CaTch Version

Figure 4.3: Local check: speed plausibility

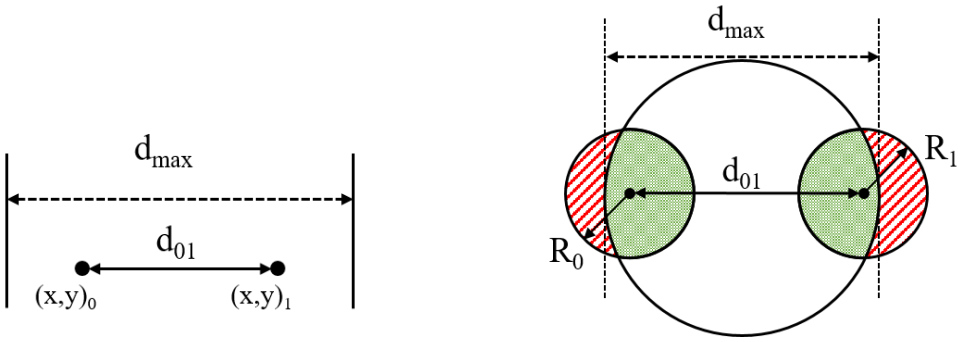
The speed plausibility is obtained by comparing the claimed speed V_0 with the maximum predefined speed V_{max} as shown in figure 4.3a. To integrate the speed confidence range C_0 into the calculation, the CaTch version checks for the plausible partition of the claimed speed confidence range compared to the maximum speed (fig 4.3b). The uncertainty factor f of this check is computed as follows:

$$f = \frac{(V_{max} - V_0 + C_0)}{2C_0}$$

V_{max}	$\triangleq$	Max Plausible Speed

4.1.1.2.4 Position consistency check

The position consistency is computed by comparing the distance between two consecutive broadcasted positions d_{01} with the maximum plausible Euclidean distance d_{max} calculated taking into account the time of the reception of the 1^{st} beacon and the time of the reception of the 2^{nd} beacon on the road as illustrated in figure 4.4a. The maximum plausible distance is computed based on the position, speed and heading information received in the $beacon_0$ message. CaTch integrates the position confidence range of two successive received beacons, respectively R_0



(a) Legacy Version (b) CaTch Version

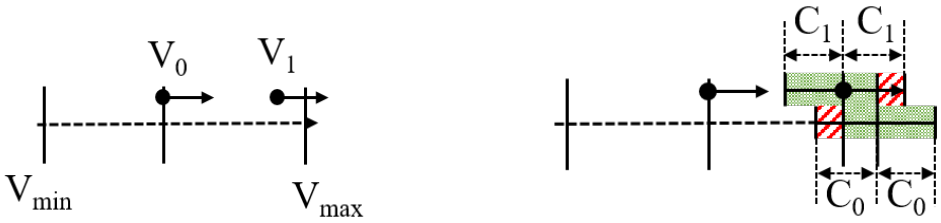
Figure 4.4: Local check: position consistency

and R_1 by calculating the position consistency similarly to the range plausibility check. CaTch uses the intersection between the area $A_{d_{max}}$ whose diameter is the maximum possible distance d_{max} (figure 4.4b) and the areas A_0 and A_1 which are respectively the areas of the of the position confidence of the 1^{st} received beacon and the 2^{st} received beacon. The uncertainty factor f of this check is computed as follows:

$$f = \frac{(a_0 + a_1)}{(A_0 + A_1)}$$

d_{max}	$\triangleq$	Maximum plausible distance
$A_{d_{max}}$	$\triangleq$	Area of the maximum plausible distance
a_n	=	$d_{max} \cap R_n$

4.1.1.2.5 Speed consistency check



(a) Legacy Version (b) CaTch Version

Figure 4.5: Local check: speed consistency

The speed consistency is obtained by checking if the speeds V_0 and V_1 from two consecutive beacons are consistent with the maximum possible acceleration or deceleration, as illustrated in figure 4.5a. Instead of a direct comparison, CaTch takes into account the confidence range C_0 and C_1 of the speeds of the consecutive beacons as illustrated in figure 4.5b. The uncertainty factor f is calculated as the overlap of the maximum speed V_{max} and the minimum speed V_{min} with the broadcasted speeds ranges. The uncertainty factor if thus calculated as follows:

$$f_{max} = \frac{V_{max} - V_1 + C_0}{4C_0} + \frac{V_{max} - V_1 + C_1}{4C_1}$$

$$f_{min} = \frac{V_1 - V_{min} + C_0}{4C_0} + \frac{V_1 - V_{min} + C_1}{4C_1}$$

$$f = \left\{ \begin{array}{ll} f_{min}, & \text{when } V_1 \leq V_0 \\ f_{max}, & \text{when } V_1 > V_0 \end{array} \right\}$$

V_{min}	$\triangleq$	Minimum plausible speed when decelerating
V_{max}	$\triangleq$	Maximum plausible speed when accelerating

4.1.1.2.6 Position-speed consistency check

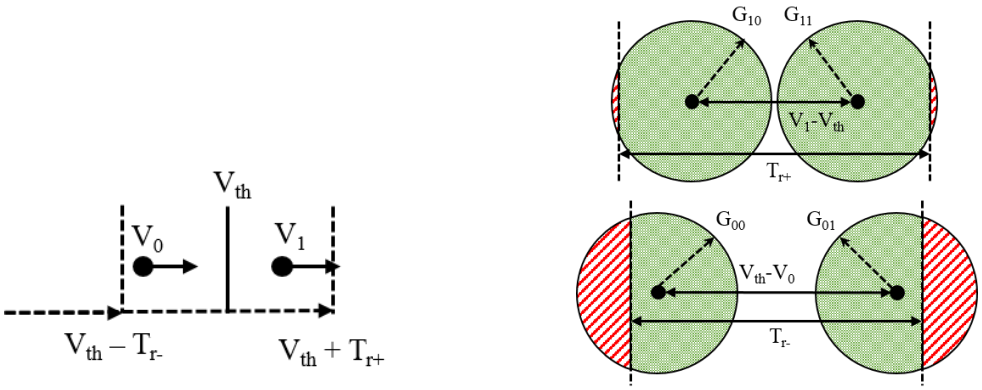
The time and distance separating two beacons result in a theoretical speed V_{th} (computed by considering the Euclidean distance). The claimed speed is consistent with the distance separating two beacons if it falls within a range denoted here as T_{r+} and T_{r-} around this theoretical speed as depicted in figure 4.6a. This check becomes tricky if we consider the confidence ranges on both the position and the speed. To this end, we calculate a new theoretical range G_x formed with a combination of the speed and the position confidence. Next we find the maximum and minimum between the speed of the first and the second beacon. We then check the plausibility of the claimed speed and the tolerance range with respect of this theoretical confidence range (fig 4.6b). The calculation of the uncertainty factor is as follows:

$$f_{max} = \frac{2\int_{lb_{max}}^{G_{max1}} \sqrt{G_{max1}^2 - x^2}dx + 2\int_{-G_{max0}}^{-lb_{max}} \sqrt{G_{max0}^2 - x^2}dx}{A_0 + A_1}$$

$$f_{min} = \frac{2\int_{lb_{min}}^{G_{min0}} \sqrt{G_{min0}^2 - x^2}dx + 2\int_{-G_{min1}}^{-lb_{min}} \sqrt{G_{min1}^2 - x^2}dx}{A_0 + A_1}$$

$$f = \left\{ \begin{array}{ll} f_{min}, & \text{when } f_{min} > f_{max} \\ f_{max}, & \text{when } f_{min} \leq f_{max} \end{array} \right\}$$

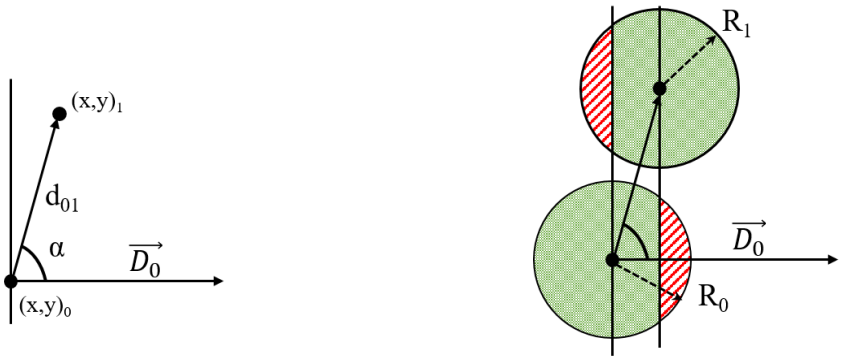
V_{th}	$\triangleq$	Theoretical speed based on the separating distance
V_{min}	$\triangleq$	Min speed of the 1^{st} & 2^{nd} beacon
V_{max}	$\triangleq$	Max speed of the 1^{st} & 2^{nd} beacon
T_{r+}	$\triangleq$	Tolerance range on excess speed
T_{r-}	$\triangleq$	Tolerance range on dearth speed
Δt	$\triangleq$	Time separating 1^{st} & 2^{nd} beacon
G_{min0}	=	$C_{min} + R_0/\Delta t$
G_{min1}	=	$C_{min} + R_1/\Delta t$
G_{max0}	=	$C_{max} + R_0/\Delta t$
G_{max1}	=	$C_{max} + R_1/\Delta t$
lb_{min}	=	$V_{th}/2 - V_{min}/2 - T_{t-}$
lb_{max}	=	$-V_{th}/2 + V_{max}/2 - T_{t+}$



(a) Legacy Version (b) CaTch Version

Figure 4.6: Local check: position-speed consistency

4.1.1.2.7 Position-heading consistency check



(a) Legacy Version (b) CaTch Version

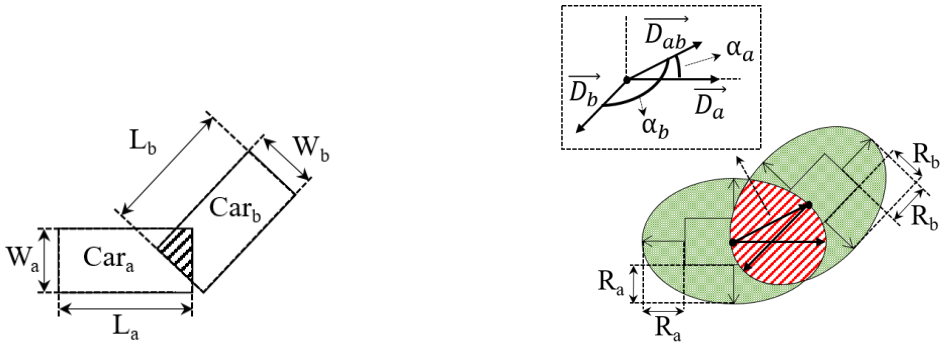
Figure 4.7: Local check: position heading consistency

To check the consistency of the heading, we calculate the angle between the advertised heading vector $\vec{D_0}$ and the real heading in the next position. This angle should be less than a predefined threshold. We set this threshold to $\pi/2$ assuming that this is a universal limit, non-dependent on vehicle specific characteristics (figure 4.7a). R_0 and R_1 are respectively the position confidence range of the 1^{st} and 2^{nd} beacon. The confidence range however could heavily affect this angle. CaTch calculates the plausible portions of the confidence range on both the 1^{st} and 2^{nd} beacon (the shaded green area in figure 4.7b). A position of a beacon is considered plausible if it forms an angle less than $\pi/2$ with the center of the other beacon. Thus, the calculation of the uncertainty factor goes as follows:

$$f = \frac{2\int_{d_{01}\cos\alpha}^{R_0}\sqrt{R_0^2 - x^2}dx + 2\int_{-R_1}^{-d_{01}\cos\alpha}\sqrt{R_1^2 - x^2}dx}{A_0 + A_1}$$

$\vec{D_{01}}$	$\triangleq$	Vector formed between the center of the positions of the 1^{st} & 2^{nd} beacons
α	$\triangleq$	Angle between $\vec{D_0}$ & $\vec{D_{01}}$

4.1.1.2.8 Intersection check



(a) Legacy Version

(b) CaTch Version

Figure 4.8: Local check: intersection

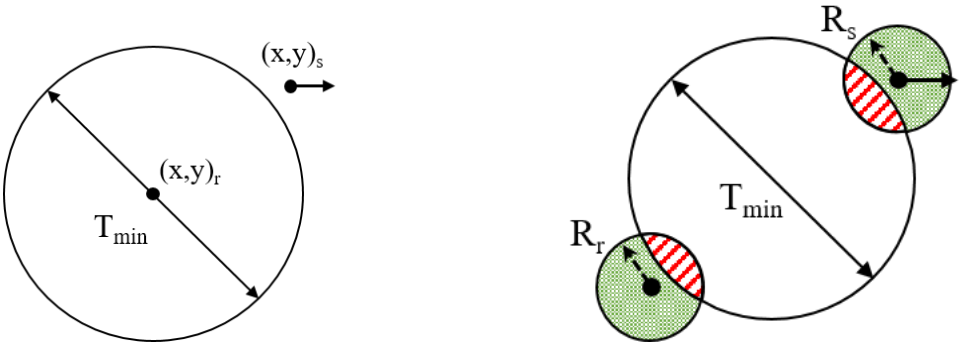
The usual intersection check, models each vehicle as a rectangle. It uses the broadcasted length and width of the vehicle in the beacons as illustrated in figure 4.8a. CaTch models each vehicle as an ellipse with an increased width and length according to the confidence range of the position (fig 4.8b). To calculate the uncertainty factor, first the intersection area between the two ellipses is calculated (fig 4.8b). However, this factor alone is not enough since the severity of the intersection depends greatly on the location it occurred and the rotation of the affected vehicles. For example, an intersection on the peripherals is less important than an intersection on the center even for the same intersection area factor. This phenomenon is numerically added to the equation using an importance factor as calculated below:

$$f_a = \frac{Ae_{ab}}{(Ae_a + Ae_b - Ae_{ab})}$$

$$f_i = \frac{ol_{ab}}{(I_a + I_b - ol_{ab})}$$

$$f = f_a f_i$$

L_n	$\triangleq$	Car_n length
W_n	$\triangleq$	Car_n width
Ae_n	=	$\pi(R_n + L_n)(R_n + W_n)$
E_n	$\triangleq$	$\frac{x^2}{(R_n + L_n)^2} + \frac{y^2}{(R_n + W_n)^2} = 0$
Ae_{ab}	=	$E_a \cap E_b$
d_{ab}	$\triangleq$	Distance between the centers of Car_a & Car_b
$\vec{D_{ab}}$	$\triangleq$	Vector formed by the centers of Car_a & Car_b
α_a	$\triangleq$	Angle between $\vec{D_{ab}}$ & $\vec{D_a}$
α_b	$\triangleq$	Angle between $\vec{D_{ab}}$ & $\vec{D_b}$
I_a	=	$(R_a + L_a)\sin\alpha_a + (R_a + W_a)\cos\alpha_a$
I_b	=	$(R_b + L_b)\sin\alpha_b + (R_b + W_b)\cos\alpha_b$
ol_{ab}	=	$max(0, min(I_a/2, I_{ab} + I_b/2) - max(I_a, I_{ab} - I_b/2))$



(a) Legacy Version (b) CaTch Version

Figure 4.9: Local check: sudden appearance

4.1.1.2.9 Sudden appearance check

The sudden appearance flag is triggered when a vehicle appears inside a minimum range T_{min}. Accordingly, $A_{T_{min}}$, is the area of the circle formed by T_{min}. The idea is that a new vehicle detected in this range T_{min} is not entering the edge of our reception range and is thus suddenly appearing. In order to integrate the error range, CaTch uses the inverse of the method used for the range plausibility (see 4.1.1.2.1). The system finds the impossible positions by calculating the intersection of $A_{T_{min}}$ with the confidence range of each vehicle. Therefore, the plausibility factor is calculated as follows:

$$f = \frac{(R_r + R_s - a_r - a_s)}{(R_r + R_s)}$$

T_{min}	$\triangleq$	Minimum acceptable range for sudden appearance
$A_{T_{min}}$	=	$\frac{\pi T_{min}^2}{4}$
a_r	=	$A_{T_{min}} \cap R_r$
a_s	=	$A_{T_{min}} \cap R_s$

4.1.2 Misbehavior fusion applications

CaTch generates results that are contentious instead of binary, thus contains more information. Providing more information should translate into at least similar or better detection results, depending on the detection application.

In order to evaluate the impact of CaTch results, we run two local misbehavior detection applications:

- **Simple App** The simple app is threshold based and is described in section 3.3.4.2. This application consists of a simple threshold. Using this application, all messages with uncertainty factor f less than 0.5 for any detector are reported. This application does not take advantage of the additional information provided by CaTch and theoretically both binary and contentious versions of the detectors should perform similarly.
- **Advanced App** This application is machine learning based and is described in section 3.3.4.2. Using data from the simulation output, we trained a neural network based on Multilayer Perceptrons (MLPs). The choice

of MLPs was done because it was the simplest to train and hyper-tune at the time. The output values of the detectors are placed in an array and used as the MLPs features. The training is done for both versions of the detectors (i.e., with and without CaTch), then tested using new simulation data. The simulation scenarios of this data is detailed in section 4.1.3. Theoretically, this application should perform better than the threshold based app as it uses the additional information provided by CaTch.

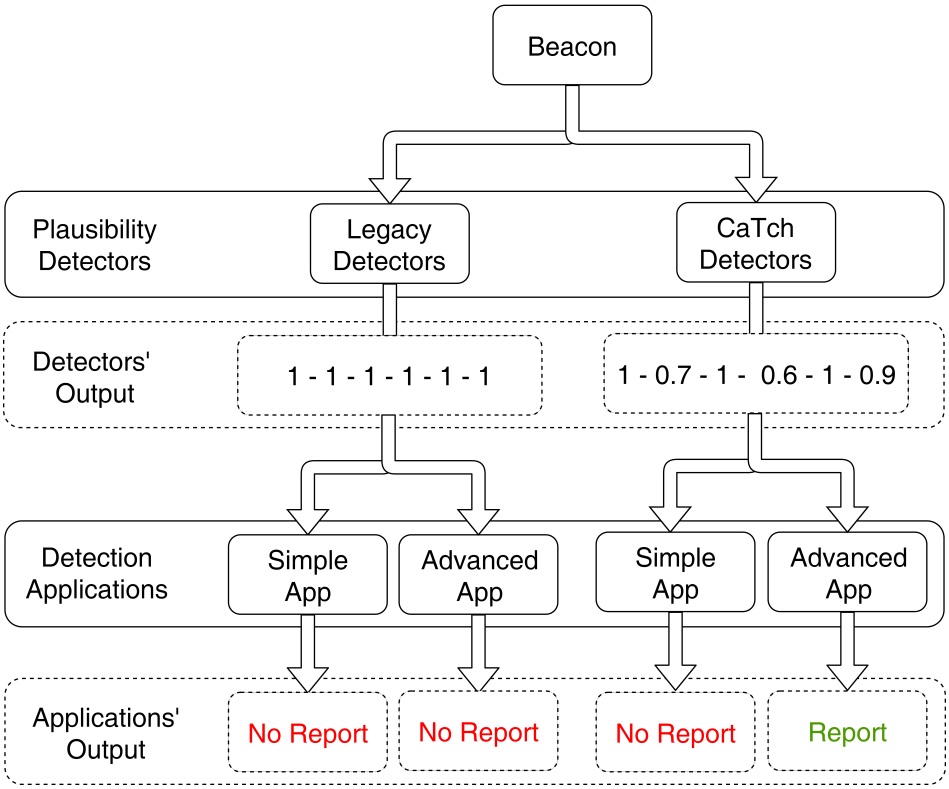


Figure 4.10: CaTch example: illustrating its advantage

To illustrate how CaTch impacts misbehavior detection, we present the simple example depicted in figure 4.10.

A beacon message is received. Both detectors check the beacon data (in this example, an arbitrary value of 6 checks is used). The legacy detector outputs binary values only. Here, all the 6 checks output the value '1'. CaTch detector outputs real values between 0 and 1 (uncertainty factor f). For checks 1, 3 and 5, CaTch outputs the value '1' like the legacy detector. However, for the remaining checks, CaTch does not attribute categorical values (i.e. 0 or 1) as the legacy one and rather, provides floating values.

The detection applications then use these results to decide whether or not a misbehavior report has to be sent. In this example, as the legacy detector generates only '1' values for each check, none of the detection applications trigger the emission of a misbehavior report (remember that the value '1' means the check passed whereas the value '0' means the check failed). However, using the CaTch results, the advanced application triggers a misbehavior report whereas the simple application does not as no CaTch checks are below the threshold value of 0.5.

4.1.3 Evaluation

In this section, we show the impact of the measurements uncertainty on the detection results. We evaluate the performance of the legacy basic checks and we compare them with CaTch using multiple detection applications.

4.1.3.1 Experimental setup

Both versions of the detectors are implemented in the F^2MD framework (see section 3.3.4.1). For the network, we used the test bench scenario of Paris-Saclay (see section 3.3.2). A network that combines a suburban-like grid and some organic network properties. Furthermore, all the vehicles introduced in the simulation are running all of the detectors in both the legacy and the CaTch version. Every time a vehicle receives a message, it is checked for implausibility. The results of these checks are then analyzed by a local misbehavior fusion application to determine whether or not to report the subject node.

4.1.3.2 Considered attacks

We chose to evaluate two types of misbehavior previously described in section 3.3.3: *Constant Position Offset* and *Traffic Sybil Attack*. The attacker density was set to 10%. In addition, given the nature of the simulator, measurements were not initially included. In order to simulate measurements uncertainty, we implemented and used the sensor error model described in section 4.3.2.2.

4.1.3.3 Results

Table 4.2 shows the results of the previously described simulated scenarios. The evaluation metrics used here are explained in section 3.3.8. The scenarios consist of detecting both the Position Offset faulty behavior and the Sybil attack using the simple threshold app and the more advanced Machine Learning app. We run both versions of the detectors simultaneously within each scenario to avoid distorting the results due to randomness. We kept the simulations running until the point of heuristic equilibrium, where the cumulative evaluation metrics stabilized.

The first observation we make is that the detection results depend greatly on the type of misbehavior. The detection results of the *Constant Offset* scenario are better than that of the *Sybil Attack*. Therefore, in our scenarios, a *Sybil Attack* is much harder to detect than a *Constant Offset*. This is an expected result since that the former have much more plausible features than the latter.

This leads us to our next point: because the claimed positions are far from being plausible, the MLP app did not find any use for CaTch's uncertainty factor for the detection of faulty nodes. Instead the MLP model used CaTch here to better characterize a genuine node. Consequently, the amount of False Positives decreases which lead to a higher Precision (+21.0%). However, this comes at the cost of some of the True Positives and consequently a lower Recall (-3.8%). We notice that all the other evaluation metrics fluctuate accordingly. This constitutes a trade-

Table 4.2: Simulation Results

Scenario / Fusion App	Detectors	Evaluation Metrics							
		Recall	**Prec**	**Accu**	**F_1s**	**BM**	**MK**	**MCC**	**K**
Threshold	**Legacy**	0.8822	0.8879	0.9768	0.8850	0.8696	0.8746	0.8721	0.7515
	CaTch	0.8798	0.9148	0.9795	0.8970	0.8706	0.9013	0.8858	0.7809
		Δ-0.3%	Δ3.0%	Δ0.3%	Δ1.4%	Δ0.1%	Δ3.1%	Δ1.6%	Δ3.9%
		Recall	**Prec**	**Accu**	**F_1s**	**BM**	**MK**	**MCC**	**K**
Machine Learning	**Legacy**	0.8806	0.8207	0.9684	0.8496	0.8589	0.8072	0.8326	0.6630
	CaTch	0.8469	0.9931	0.9839	0.9142	0.8462	0.9761	0.9088	0.8285
		Δ-3.8%	Δ21.0%	Δ1.6%	Δ7.6%	Δ-1.5%	Δ20.9%	Δ9.2%	Δ25.0%

(a) Constant Offset Scenario

Scenario / Fusion App	Detectors	Evaluation Metrics							
		Recall	**Prec**	**Accu**	**F_1s**	**BM**	**MK**	**MCC**	**K**
Threshold	**Legacy**	0.6475	0.9288	0.9186	0.7631	0.6349	0.8457	0.7328	0.5608
	CaTch	0.6511	0.9419	0.9213	0.7700	0.6409	0.8598	0.7424	0.5746
		Δ0.6%	Δ1.4%	Δ0.3%	Δ0.9%	Δ0.9%	Δ1.7%	Δ1.3%	Δ2.5%
		Recall	**Prec**	**Accu**	**F_1s**	**BM**	**MK**	**MCC**	**K**
Machine Learning	**Legacy**	0.6483	0.9961	0.9286	0.7854	0.6476	0.9145	0.7696	0.6130
	CaTch	0.7903	0.9749	0.9536	0.8729	0.7852	0.9244	0.8519	0.7368
		Δ21.9%	Δ-2.1%	Δ2.7%	Δ11.1%	Δ21.2%	Δ1.1%	Δ10.7%	Δ20.2%

(b) Sybil Attack Scenario

off between Recall and Precision and we don't find using CaTch in this scenario definitely beneficial. For the case of faulty nodes detection, the Legacy detectors could be sufficient.

However, this is not the case for the *Sybil Attack* scenario. Since the messages are more plausible, and the implausibilities are more elusive, the MLP app finds a major advantage by using CaTch's additional information. The extra edge given by CaTch's uncertainty factor enables the MLP app to increase the Recall (+21.9%) with a small loss in terms of precision (-2.1%). Accordingly, The F_1Score, the BM and the MCC all increase by more than 10%. Therefore, we find that using CaTch in this scenario definitely increases the quality of the detection. Given these points we conclude that the main benefit of CaTch is to detect attacks that are subtler where the attacker is intelligent and tries to remain within the plausible range. In fact, using CaTch we can train a system to extract a kind of a "fingerprint" of an attack, which is not possible to elaborate when using the binary detectors.

Nevertheless, this improvement comes at the cost of higher processing time. In fact, the binary detectors are 3 times faster than their CaTch counterparts as shown in Table 4.3.

Table 4.3: Mean processing time of the Binary and CaTch detectors

Detector Version	Binary	CaTch
Time (μs)	34.561	110.994

4.1.3.4 Discussion

It is worth noting that the model we chose for the machine learning App is not optimal. It was implemented this way to give a more comprehensive and fairer comparison and to remain consistent with the illustrative example. However, by adding only a notion of node history to the machine learning model, using only the last few messages of a node instead of only one, we were able to significantly increase the detection quality. In particular, the false positives ceases being much of an issue. Additionally, we found that the fingerprinting of attacks with CaTch over multiple sequential messages could be much more elaborated and complex. This is discussed in more detail in section 4.2.

Additionally, one could assume that an attacker could manipulate the CaTch uncertainty factor to avoid detection. This is in fact possible and is a scenario that should be taken into account when designing the detection application. For example, an application should consider a plausible maximum value for the confidence range depending on the environment. However, CaTch is supposed to be replacing the legacy detectors. Additionally, it is worth noting that it is much easier for an attacker to just manipulate the sensor value to remain within the legacy detectors limits than to manipulate the value and the confidence range to do the same for CaTch.

4.1.4 Summary

In this work, we focused on embedded misbehavior detection mechanisms in C–ITS. More precisely, we evaluate the impact of physical measure uncertainty on the plausibility detectors performance. We notice that such uncertainty information is included in the standard V2X messages. We theorize that utilizing it may lead to improving the misbehavior detection process.

To benefit from this available information, we propose CaTch, an enhanced version of plausibility detectors that takes into account the physical measure uncertainty. We showed that CaTch provides for each plausibility detector an uncertainty factor that indicates the levels of implausibility. We showed through extensive simulations that CaTch performs better or in the worst case similar to the current misbehavior detectors. In particular, the uncertainty information can be used by intelligent misbehavior detection applications to improve the detection quality in the decision making process.

4.2 Evaluation of local detection mechanisms

After introducing a new set of local plausibility and consistency checks, we direct our interest to the next step in the local misbehavior detection process: the fusion application. The results of plausibility checks are analyzed by the local fusion application to classify the received V2X message as misbehaving or genuine. Our goal is to evaluate different approaches for local fusion applications. Although many researchers have published fusion mechanisms, the results are often difficult to compare since the tests are done on different data and with different implementations.

To this end, we use our F^{2}MD simulator described in chapter 3. We re-implement the different fusion mechanisms and test then while running the same scenario. Our comparative results show a clear trade-off between the accuracy of the detection mechanisms and the calculation latency.

4.2.1 Attacker model

A misbehaving entity in C–ITS is any ITS–S sending inaccurate or fake V2X messages. Misbehaving entities could be divided into two categories: Faulty and Attackers. A Faulty behavior is any inaccurate V2X message data coming from a broken vehicle sensor. An attack is an intentional modification of the V2X message data. The implemented set of possible misbehavior types is inspired from the literature [49] [85] (see section 3.3.3). Please note that every new attacker sets the attack parameters randomly within a certain range. This is done to render the detection more difficult specifically for the Machine Learning based solutions.

4.2.2 Detection mechanisms

In this part we extract the base logic behind some related works described in section 2.6. We also detail our implementation of the extracted detection logics. We put forward solutions based on a deterministic approach and machine learning based mechanisms.

4.2.2.1 Deterministic approaches

The deterministic fusion mechanisms are described in detail in section 3.3.4.2.

Threshold Based: a purely data-centric baseline solution. If any detector fails, the message is misbehaving.

Non-Cooperative Trust Based (N-CTB): a node-centric trust evaluation using data-centric mechanisms. This is a similar approach to the logic used by Schmidt et al. in [62] and Bißmeyer et al. in [63].

Cooperative Trust Based (CTB): a solution with a form of information sharing between the ITS–Ss. This approach is similar to the one used in Leinmüller et al. [76] and Kerrache et al. in [77].

4.2.2.2 Machine learning based

The goal of this solution is to train a machine learning algorithm to detect if a V2X message is misbehaving. Many algorithms exist for this purpose, however we revert to testing *SVM* proposed by Boser et al. in [108], *XGBoost* proposed by Chen et al. in [109], *MLP* proposed by Van Der Malsburg et al. in [110] and *LSTM* as proposed by Hochreiter et al. in [111]. We detail below the models and parameters trained and tested in this study. All the hyper-parameters of the proposed model are tuned using a grid search based on 5-fold cross validation.

Common Features: For every received V2X message a set of features is created. These features are important indications used by the tested ML algorithm to evaluate the plausibility of a message. We propose two features sets suitable for different ML algorithms.

- **Checks Feature Set:** The local detection checks done on V2X messages described in section 3.3.4.1. Note that the CaTch version of the checks are used here.
- **Kinematic Feature Set:** The $Position$, $Speed$, $Accel$, $Heading$ and $Time$ of the last beacon. The $\Delta Position$, $\Delta Speed$, $\Delta Acceleration$, $\Delta Heading$ and $\Delta Time$ between the last 2 beacons.

XGBoost: *eXtreme Gradient Boosting (XGBoost)* is a relatively new algorithm and currently arguably the most performing of the tree-based models. The model is given a set of V2X messages with the *Checks Feature Set*. The messages are given independently of each other. This entails an assumption that no time dependency exists between the data. All messages are treated as independent entities similarly to the case of the *Threshold based* solution. Consequently, some valuable information is lost from the base data due to this assumption. However, this model is useful to evaluate and better understand the treated data.

SVM: As proposed by So et al. in [87] we use SVM as our baseline ML solution. This model is also trained with the *Checks Feature Set*. Multiple implementations exist for the SVM classification. The default SVM implementation (C-Support Vector Classification (SVC)), is not designed for large data sets. The SVC training times exhibit quadratic growth with the increase of the number of samples. Therefore, we are able to train SVC with only 10% of our original training data-set. Alternatively, we tested the Linear Support Vector Classification (LinearSVC), a similar implementation that could scale better with a large numbers of samples. However, LinearSVC performed significantly worse than SVC even when trained on the full data-set.

MLP: *MLP* is feedforward backpropagation Artificial Neural Network (ANN). It is the algorithm used by Singh et al. in [90]. We tested two implementations for this model. The first implementation (MLP-T1) makes use of the same features *Checks Feature Set* as the previous models. The proposed MLP-T1 model has 1 Dense layer with 18

nodes. The second implementation (MLP-T10) takes as input the previous 10 time-steps. The feature set consists of the Minimum and the Average of the *Checks Feature Set*. The proposed MLP-T10 model has 1 Dense layer with 36 nodes.

LSTM: *LSTM* is also an algorithm of choice used by Singh et al. in [90]. Moreover, is a well suited algorithm for our problem due the temporal based relation between the successive V2X messages. LSTM is part of the Recurrent Neural Network (RNN) family of ML algorithms specifically designed to treat time dependent data. Therefore, the LSTM was additionally given the *Kinematic Feature Set* as input. The proposed model contains a single bidirectional LSTM layer with 20 nodes. A dropout and batch-normalization were added to combat over-fitting.

For more technical details, the implementation of these models is open-source and shared on GitHub [26].

4.2.3 Evaluation results

4.2.3.1 Simulation settings and scenarios

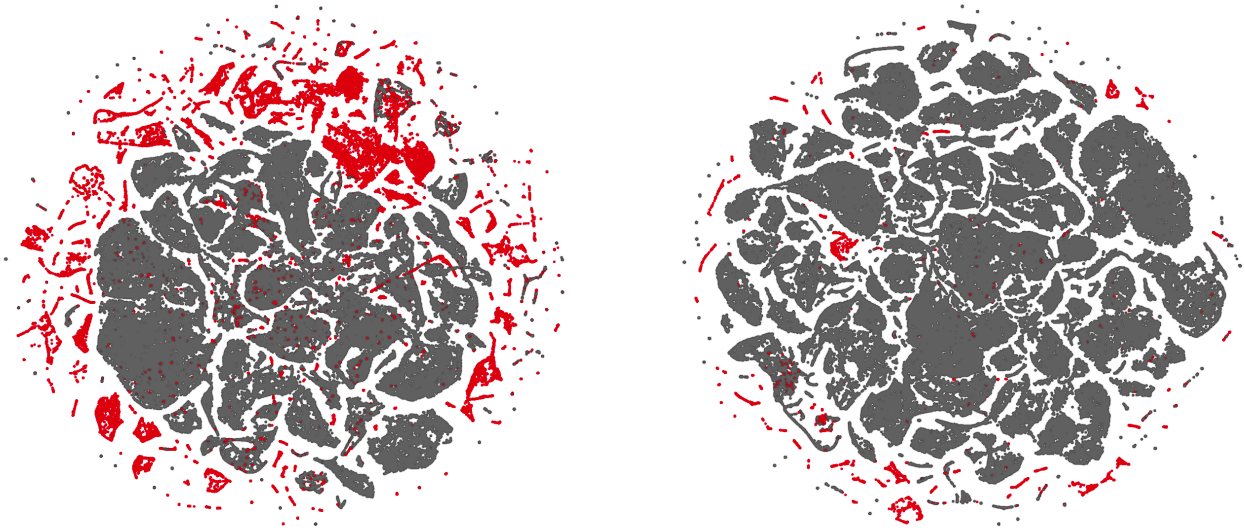
We used F^2MD as a base to implement the previously described solutions (see section 3). In order to correctly evaluate the ML algorithms, we used a different scenario for the training and the testing. The small LuST scenario is used for training. The Paris-Saclay scenario is used to generate the test set. Both scenarios include a mix of the attacks described in section 3.3.3. The attacker rate is set at 25% for the train scenario and 5% for the test scenario. This scenario is described in detail in section 3.3.2. For further details, the raw data, the source code and all the configuration details of the scenarios are published on GitHub [26].

Provided our relatively large data-set, high performance computing is needed for the prepossessing and training of our models. The *SVM* and *XGBoost* training is done on CPU server with a 176 core Intel(R) Xeon(R) CPU E7-8880 v4 @ 2.20GHz and 2TBs of RAM. The *MLP* and *LSTM* training is done on a GPU server with a couple of NVIDIA Tesla P100s. All the algorithms are tested on a local workstation with an 8 core Intel(R) Xeon(R) W-2123 CPU @ 3.60GHz and 32GBs of RAM.

4.2.3.2 Data evaluation

Figure 4.11 shows a t-Distributed Stochastic Neighbor Embedding (t-SNE) visualization of the train and test data-sets. t-SNE is a dimensional reduction technique used to reduce multiple features into a two dimensional space [112]. We observe that the two classes are not perfectly separated into clusters. Some of the genuine and misbehaving data points are mixed. Therefore, we suspect that a linear model is not suitable for this classification. A non-linear kernel or a deep-learning model might perform better in our scenario.

Figure 4.12 shows the feature importance by weight as calculated by the XGBoost module. We can see that not all the features are equally important. A relative correlation is apparent between the complexity of the calculated



(a) Train Data (b) Test Data

Figure 4.11: Local detection: t-SNE: Genuine (Gray) and Misbehaving (Red)

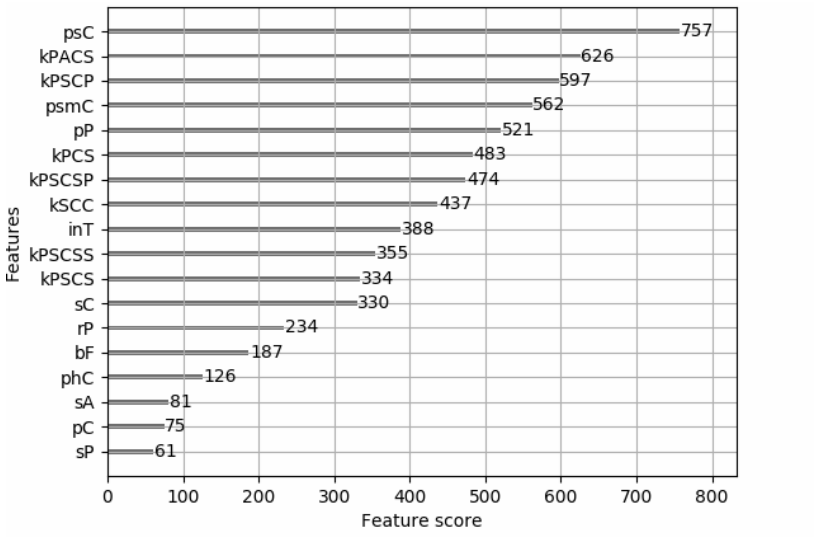
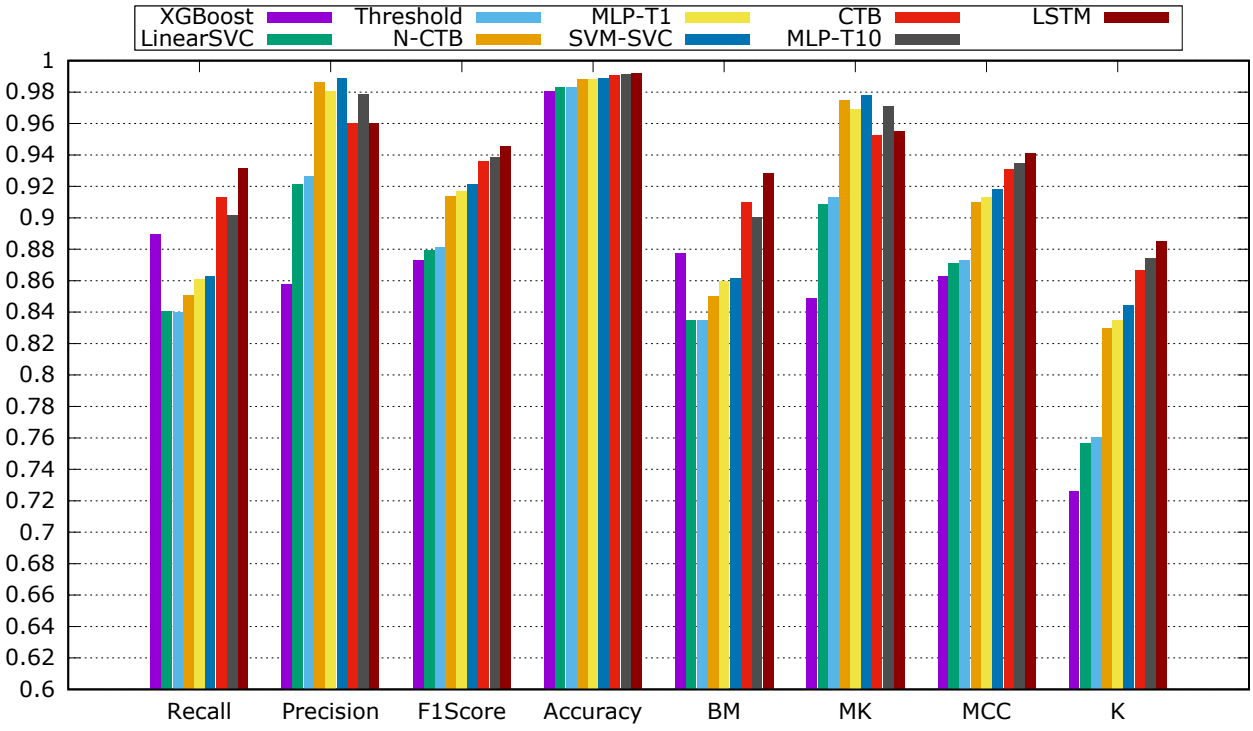


Figure 4.12: Local detection: XGBoost feature importance

check and the importance with respect to the XGBoost classification. Specifically, the Kalman-Filter based check is especially important. All the Kalman Filter extracted checks rank high on the feature importance scale. Future studies treating ML in local misbehavior detection should consider adding it to their feature sets. Ultimately, it is as important to consider the checks calculated as the ML algorithms used for the classification.

4.2.3.3 Results analysis



Detection Solution	Evaluation Metrics								MPT (μs)
	Recall	Precis	F_1s	Acc	BM	MK	MCC	K	C++($*$) Python($\circ$)
Threshold	0.8401	0.9262	0.8811	0.9827	0.8346	0.9131	0.8730	0.7604	0.00010 ($*$)
N-CTB	0.8509	0.9864	0.9137	0.9880	0.8500	0.9745	0.9101	0.8300	0.00019 ($*$)
CTB	0.9131	0.9599	0.9359	0.9904	0.9100	0.9527	0.9311	0.8664	0.00023 ($*$)
XGBoost	0.8892	0.8578	0.8732	0.9808	0.8774	0.8489	0.8630	0.7262	745.01 ($\circ$)
LinearSVC	0.8406	0.9216	0.8792	0.9828	0.8349	0.9088	0.8711	0.7564	7089.6 ($\circ$)
SVM-SVC	0.8626	0.9887	0.9213	0.9891	0.8618	0.9778	0.9180	0.8441	115.14 ($\circ$)
MLP-T1	0.8611	0.9804	0.9169	0.9884	0.8598	0.9694	0.9129	0.8348	1358.6 ($\circ$)
MLP-T10	0.9018	0.9787	0.9387	0.9912	0.9002	0.9708	0.9349	0.8746	0.176 ($*$) 1562.6 ($\circ$)
LSTM	0.9312	0.9603	0.9455	0.9920	0.9281	0.9547	0.9413	0.8852	0.614 ($*$) 8298.9 ($\circ$)

Figure 4.13: Local detection: evaluation metrics by tested fusion application

Figure 4.13 shows the evaluation metric results of the models classification of the test data-set. The considered evaluation metrics are: *Recall*, *Precision*, *F_1score*, *Accuracy*, *Bookmaker Informedness (BM)*, *Markedness (MK)*, *Matthews Correlation Coefficient (MCC)* and *Cohen's kappa* (κ). The evaluation metrics are detailed in section 3.3.8. The Mean Processing Time (MPT) is also measured for every considered detection application.

First thing we notice is that all the detection mechanisms are within a small accuracy range. In fact, all the

detection mechanisms score more that 98% accuracy. This is due to the mechanisms relatively high precision and the unbalanced test data-set of 5% attacker rate. Consequently, accuracy is not a suitable detection metric for our use case. For comparison purposes we rely on *Cohen's kappa*, *MCC* or the F_1*score* which all have the same ranking for the considered mechanisms.

Second thing we notice is the LinearSVC performed significantly worse than the SVC with respect to all the evaluation metrics. In fact, LinearSVC even performed nearly identically to the simple *Threshold* application. This result is in line with our previous analysis of the t-SNE plot in section 4.2.3.2. To emphasize, a simple *LinearRegression* is also tested with similarly performing results.

Moving on to the MPT, we have two platforms of execution. The deterministic models are executed in C++ within the simulation. The ML-based solutions are executed on Keras in Python. Keras is not optimized for single predictions, instead it performs much better with batch predictions, which is not the case in out model. Consequently, the Python and the C++ executions are unsuitable for processing time comparison. To this end, we re-implement the *MLP-T10* and the *LSTM* models in C++ using the previously trained weights. Nevertheless, even with the C++ optimization, the deterministic models calculates around 800 times faster than their ML-based counterparts. However, the ML-based solutions do not entirely outperform the deterministic solutions. We notice three clusters within the results. The metrics of the *Threshold* solution is comparable to the *LinearSVC* and the *XGBoost*. The *N-CTB* is comparable to the *MLP-T1* and the *SVC*. The *CTB* is closer to the *MLP-T10* and the *LSTM*. Accordingly, there is no positive correlation between the processing time and the detection quality.

On the other hand, some solutions have their own drawbacks. The *CTB* application relies on the authenticity of the neighboring vehicles to determine the level of trust. Therefore, it is vulnerable to Sybil attacks. Additionally, the system could completely fall apart in sub-environments where a definite honest majority of vehicles is not assured. In contrast, all ML-based solutions are venerable to adversarial attacks. Moreover, a large and reliable training set is required for the models to function adequately. Therefore, the ML-based solutions could not protect against zero-day vulnerability, i.e. in the early stages of deployment we do not have enough data to train a ML-based detection system. Furthermore, the deploy-ability and certification of these ML-based solution for an embedded implementation is relatively complex. Finally, the detection of new types of previously unknown attacks might require the re-training of the model.

We believe that a robust set of well calibrated detectors coupled with a non-cooperative deterministic application, like the *N-CTB*, could be the more suitable solution for this stage of local detection. It has a fast processing time and easy deploy-ability. It is not vulnerable to Sybil or adversarial attacks. It is agnostic to new types of attacks. And it requires no training data so it could be implemented immediately with the first deployment. Nevertheless, this result is not conclusive as the local detection is not an independent system. Even though, the global MA should be designed to withstand a number of False Positive reports and a number of missed reports. The effect of the local detection quality on the global MA should also be evaluated for a more rigorous analysis.

4.2.4 Summary

In this work, we evaluate different local fusion solutions. To achieve this, we extract the detection logic from published studies. Then we describe our implementation of the different extracted solutions. We show through testing results that some Machine Learning solutions outperforms the deterministic algorithms but only by a small margin. We put in question the need for Machine Learning solutions in this use case. We argue that *Non-Cooperative Trust Based* could be a suitable solution for this application.

4.3 VeReMi dataset extension

In the previous section, we re-implemented all the fusion models that we wanted to compare. Nevertheless, there is an easier method for researchers to produce verifiable and comparable results: using a reference dataset. To this end, the original VeReMi was created as the first public dataset for vehicular misbehavior detection [81]. VeReMi has proven to be very useful for researchers in this domain, being applied in multiple studies [87] [89] [92] [88]. However, the VeReMi dataset still has room for improvement, especially considering the small number of attacks and the lacking physical error model.

In this work, we upgrade VeReMi by treating these issues. Accordingly, we devise and implement a realistic sensor error model on the vehicle's physical layer. Moreover, we implement a larger more complex set of attacks. The new attacks enable the manipulation of the message frequency and the digital certificates as well as the manipulation of the message contents. Finally, we describe and implement a set of local plausibility detectors and a simple fusion detection mechanism. The dataset is then tested against this misbehavior fusion mechanism and the results are provided as a benchmark for future researchers.

4.3.1 Related works

Multiple studies used the original VeReMi dataset to test and validate their various misbehavior detection mechanisms.

Steven et al. [87] analyzed the dataset with the K-Nearest Neighbors (K-NN) and Support Vector Machine (SVM) Machine Learning (ML) classification algorithm with positive results. In their experiment, they only made minor adjustments to the labeling of the data in order to help with the classification, but the attacker types and messages were left unmodified. In their discussion, they pointed to a small weakness of VeReMi. The vehicles performing an *EventualStop* are constantly labeled as attackers, though they may behave normally for a certain time before the start of their malicious behavior. They suggest that the vehicle should be labeled as an attacker only when the attack is ongoing. This suggestion is retained and this issue is now resolved in this new VeReMi version. Attacker nodes are only labeled as such when actively performing a certain attack, otherwise their behavior is labeled as normal.

Singh et al. [89] used VeReMi to perform feature engineering for a ML misbehavior detection solution. They tested various feature selections and scored different results in accuracy. They performed their experiments with Logistic Regression and an SVM. Their best feature set was achieved with an SVM and contained the position, the speed and the difference between the position and speed of the sender and receiver. In the current version of VeReMi, the acceleration and heading are added as new features to the dataset. This could enable more feature combinations and a better feature selection in future studies.

Gyawali et al. [92] also used the dataset for an ML application using a Feed Forward Neural Network (FFNN) and an SVM. They calculated their solution's detection metrics including the accuracy, precision, recall and F_1 Score.

They then compared these metrics to the ones included in the original version of VeReMi. This type of comparison is still possible in the latest version of VeReMi as these detection metrics were calculated for each newly added scenario.

4.3.2 Dataset

4.3.2.1 Simulation platform

In order to generate our dataset, we make use of F^{2}MD. We use the vehicle traces provided by the LuST scenario created by Codeca et al. [96]. For our dataset, we use the subsection of the LuST network with a size $1.61km^2$ and a peak density of 67.4 Veh/km^2 described in section 3.3.2.

4.3.2.2 Sensor error models

In this version of VeReMi, we aim to render the provided data more realistic and in line with real world field tests. Accordingly, we add sensor error models to the four main data fields: Position, Velocity, Acceleration and Heading.

4.3.2.2.1 Position error

Several positioning systems exist with different levels of precision. These levels also differ by region, i.e. the GPS precision is limited to about 3 to 5 m in open sky environments and up to 20 m in urban areas [63]. However, we consider an internal correction system on-board every vehicle [113]. Consequently, the position error model is as follows:

$$\mathcal{E}_0^P = U([-5,5])$$
$$\mu = \frac{\mathcal{E}_0^P + \mathcal{E}_{t-1}^P}{2} \qquad \sigma = 0.03\mathcal{E}_0^P$$
$$\mathcal{E}_t^P = \mathcal{N}(\mu, \sigma^2)$$
$$P_t^{\mathcal{E}} = P_t + \mathcal{E}_t^P$$

P_t	≙	Real Position at time t
$P_t^{\mathcal{E}}$	≙	Broadcasted Position at time t
$\mathcal{E}_t^P$	≙	Position Error at time t
U	≙	Uniform Distribution
$\mathcal{N}$	≙	Normal Distribution

4.3.2.2.2 Velocity error

The majority of vehicles estimate velocity from the wheel spin with an average error around $0.05m/s$ [114]. The error is also proportional to the velocity. As a result, the speed error model is as follows:

$$\mu = 0 \qquad \sigma = 0.00016$$
$$\mathcal{E}_0^V = \mathcal{N}(\mu, \sigma^2)$$
$$V_t^{\mathcal{E}} = V_t + V_t * \mathcal{E}_0^V$$

V_t	≙	Real Velocity at time t
$V_t^{\mathcal{E}}$	≙	Broadcasted Velocity at time t
$\mathcal{E}_t^V$	≙	Velocity Error at time t

4.3.2.2.3 Acceleration error

In our model, the acceleration error is inferred from the velocity error. Therefore, the acceleration error model is as follows:

$$A_t^{\mathcal{E}} = A_t + \frac{\mathcal{E}_t^V - \mathcal{E}_{t-1}^V}{\delta t}$$

A_t	$\triangleq$	Real Acceleration at time t
$A_t^{\mathcal{E}}$	$\triangleq$	Broadcasted Acceleration at time t

4.3.2.2.4 Heading error

The vehicle heading could be calculated using a magnetic compass or inferred from successive positions. The accuracy of a magnetic compass is dependent on the device quality. A study by Hölzl et al. found that the probability of having an error below 20° is around 85% for most mobile devices [115]. Whereas Deelertpaiboon et al. successfully equipped a vehicle with a magnetic compass with a 0.1° accuracy [116]. Additionally, accuracy of the heading derived from successive positions is dependent on the vehicle velocity. At a high velocity the position heading is more probable to have higher accuracy than the compass heading, whereas the opposite is true at lower speeds or when a vehicle is stationary. For our solution, we propose the following heading error model:

$$\mathcal{E}_0^H = U([-20, 20])$$
$$\mathcal{E}_t^H = \mathcal{E}_0^H * e^{-0.1 * V_t}$$
$$H_t^{\mathcal{E}} = H_t + \mathcal{E}_t^H$$

H_t	$\triangleq$	Real Heading at time t
$H_t^{\mathcal{E}}$	$\triangleq$	Broadcasted Heading at time t

4.3.2.3 Misbehavior models

In this study, we also aim to expand the VeReMi attacks library with a set of new attacks aggregated from the models used in the literature [49]. We make the distinction between malfunctions and attacks. The former constitutes non-malicious behaviors that results from a malfunctioning OBU or vehicle sensors while the latter is malicious behavior of vehicles intentionally sending wrong information. The misbehavior models are defined in section 3.3.3.

4.3.2.4 Generated datasets

Table 4.4 shows a brief description of the newly generated datasets parameters. The data is encoded in JSON following the same format as the original VeReMi datset:

```
{"type":ℤ[0,20],
 "rcvTime":ℝ[0,+∞],
 "sendTime":ℝ[0,+∞],
```

```
"sender":ℤ[0,+∞],
"senderPseudo":ℤ[0,+∞],
"messageID":ℤ[0,+∞],
"pos":[ℝ[−∞,+∞],ℝ[−∞,+∞],ℝ[−∞,+∞]],
"pos_noise":[ℝ[0,+∞],ℝ[0,+∞],ℝ[0,+∞]],
"spd":[ℝ[−∞,+∞],ℝ[−∞,+∞],ℝ[−∞,+∞]],
"spd_noise":[ℝ[0,+∞],ℝ[0,+∞],ℝ[0,+∞]],
"acl":[ℝ[−∞,+∞],ℝ[−∞,+∞],ℝ[−∞,+∞]],
"acl_noise":[ℝ[0,+∞],ℝ[0,+∞],ℝ[0,+∞]],
"hed":[ℝ[−∞,+∞],ℝ[−∞,+∞],ℝ[−∞,+∞]],
"hed_noise":[ℝ[0,+∞],ℝ[0,+∞],ℝ[0,+∞]]}
```

In the updated VeReMi dataset, two subsets are created for each type of misbehavior described in section 4.3.2.3. One subset in rush hour time (7h-9h) and another in low traffic time (14h-16h) (see Fig. 3.4). Every subset includes a file for the ground truth and a list of files containing the received message data. Additionally, one test-bench subset is created with a mix of all the previously described misbehavior attacks spanned on the whole simulation day (0h-24h). All the subsets are a result of the simulation of the Luxembourg network described in section 4.3.2.1. The misbehavior attacker penetration rate is set at 30% for all the simulations. All the 39 resulting datasets are published and could be found on our Cloud Drive [117]. Additionally, replicating this dataset or creating new scenarios is possible using F^{2}MD.

Table 4.4: datasets information per described scenario

		Dataset Id		
		Attack_0709	*Attack_1415*	*MixAll_0024*
Scenario	Time	07h-09h	14h-16h	00h-24h
	Density	37.03 V/km^2	16.36 V/km^2	23.29 V/km^2
Attacker	Vehicles	1,220	505	7,399
	Messages	924,251	249,612	7,505,418
Genuine	Vehicles	2,846	1,179	17,264
	Messages	2,221,825	569,723	11,951,021
Average Size	Plain	1.92 GBs	0.59 GBs	0.91 GBs
	Gzipped	0.40 GBs	0.12 GBs	0.19 GBs
Total Size	Plain	40.51 GBs	11.92 GBs	10.90 GBs
	Gzipped	8.41 GBs	2.42 GBs	2.25 GBs

Table 4.5: Testing results

Id	Results			
	Accuracy	*Precision*	*Recall*	F_1 *Score*
DoSRandom_1416	0.9994	0.9993	0.9996	0.9995
DoSRandom_0709	0.9994	0.9994	0.9995	0.9994
RandomPos_1416	0.9991	0.9981	0.9989	0.9985
RandomPos_0709	0.999	0.9979	0.9987	0.9983
ConstPos_0709	0.9958	0.9979	0.9878	0.9928
ConstPos_1416	0.9954	0.998	0.9869	0.9924
DoS_1416	0.9876	0.9993	0.9788	0.9889
DoSRandomSybil_0709	0.9898	0.999	0.9784	0.9886
DoSRandomSybil_1416	0.989	0.9991	0.9773	0.9881
DoSDisruptive_1416	0.9862	0.9857	0.9896	0.9876
DoSDisruptive_0709	0.9861	0.9864	0.9887	0.9876
DoS_0709	0.9859	0.9993	0.9752	0.9871
RandomPosOffset_1416	0.9877	0.9979	0.9614	0.9794
RandomPosOffset_0709	0.9865	0.9973	0.9566	0.9765
Disruptive_1416	0.9827	0.9805	0.9622	0.9713
Disruptive_0709	0.9829	0.9777	0.9638	0.9707
RandomSpeed_1416	0.981	0.998	0.9394	0.9678
RandomSpeed_0709	0.9787	0.998	0.9294	0.9625
ConstPosOffset_0709	0.9665	0.9979	0.8879	0.9397
ConstPosOffset_1416	0.9606	0.9979	0.8726	0.9311
DelayedMessages_0709	0.9512	0.9971	0.8362	0.9096
ConstSpeed_1416	0.9441	0.9974	0.8187	0.8992
MixAll_0024	0.9293	0.9912	0.8228	0.8992
DataReplay_0709	0.9393	0.9372	0.8503	0.8916
DelayedMessages_1416	0.9402	0.998	0.8052	0.8913
ConstSpeed_0709	0.939	0.9976	0.7942	0.8843
DataReplay_1416	0.9307	0.9318	0.8333	0.8798
RandomSpeedOffset_1416	0.8928	0.9972	0.65	0.787
RandomSpeedOffset_0709	0.895	0.997	0.6444	0.7828
TrafficSybil_0709	0.8204	0.9973	0.5902	0.7415
TrafficSybil_1416	0.8001	0.9972	0.5842	0.7367
EventualStop_1416	0.8827	0.9952	0.5301	0.6918
DoSDisruptiveSybil_1416	0.7787	0.988	0.5318	0.6914
EventualStop_0709	0.8856	0.9942	0.5163	0.6796
DoSDisruptiveSybil_0709	0.7699	0.9822	0.5014	0.6639
ConstSpeedOffset_0709	0.8263	0.9953	0.4107	0.5814
ConstSpeedOffset_1416	0.8157	0.9957	0.3969	0.5676
DataReplaySybil_1416	0.7948	0.892	0.3705	0.5235
DataReplaySybil_0709	0.8011	0.92	0.3527	0.5099

4.3.3 Results

In this section we run a simple detection algorithm on the previously described dataset. The detection is based on a set of plausibility and consistency checks on the received message data. These checks are done on the Position, Speed, Heading and Acceleration fields. The tests include: (1) Absolute plausibility, (2) Temporal consistency, (3) Relative consistency of a field with respect to another, (4) Consistency with respect to a Kalman Filter, (5) Overlap of two vehicles, (6) Beacon frequency compliance, (7) Sudden appearance plausibility, (8) Transmission range plausibility. For a more detailed description of the used checks please refer to section 3.3.4. Additionally, the implementation is open source and available on GitHub [26].

The plausibility checks are calculated then passed through a threshold mechanism. If the threshold is reached for a certain check, the vehicle message is considered misbehaving. The output of the detection mechanism is partitioned into four groups: True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (PN) and evaluated as described in section 3.3.8.

Table 4.5 shows the results of our basic detection algorithm for each subset of our dataset. The first thing we notice is that the detection quality is largely dependent on the type of misbehavior. The $F_1 score$ almost doubles when comparing between the least and the most detected attack. We also notice that this is mainly due to a lower *Recall* than a lower *Precision*. This means that our detection solution has a tendency to reduce the False Positives at the expense of some missed detections. We also notice that the time of day and consequently the general vehicle density does not greatly affect the detection quality. For each attack, both the peak time scenario (07h - 09h) and the low density scenario (14h - 16h) are within the same detection range. This effect could be specific to our benchmarking detection solution. Future research with more advanced solutions, especially cooperative detection schemes, could be more revealing in this regard. Last thing we notice is that the *Precision* is generally lower for the attacks that contains replaying other vehicle's data. This means these attacks are successfully tricking the detection system into flagging the target genuine vehicles as misbehaving. Future detection solutions should consider the effects of this new attack vector and find mechanisms that are resistant to this category of disruptive nodes.

4.3.4 Summary

In this paper we provide an extension to the VeReMi dataset for misbehavior detection in VANETs. This extension includes a new set of more elaborate attacks, a realistic physical error model and a larger collection of data. This study also includes the detection results of a simple misbehavior detection mechanisms as an initial benchmark. Our dataset will enable other researchers to further improve their detection mechanisms, create new mechanisms for the newly provided attack vectors and compare their results to our benchmark.

4.3.5 Conclusion

This section concludes our work on the local misbehavior detection. In the first section, we present our work on the use of the confidence range in the misbehavior detectors. This work lead us to conclude that the confidence range is important for the detection process and local misbehavior studies should take this data field into consideration in the future. In the second section, we analyse the results of a comparison between different local fusion mechanisms. This comparison shows that future work on the evaluation of fusion detectors should take into consideration the response time as well as the detection accuracy. Finally, our work on the VeReMi dataset shows the importance of a comparable evaluation of different mechanisms. The VeReMi dataset also shows the differences in the detection rate of various attacks. These attacks on the ITS system should be better anticipated by researchers in order to more effectively defend against issues that might be encountered during deployment.

Chapter 5

Misbehavior Reporting

In this chapter, we focus on the reporting process of Misbehavior Detection. In section 5.1, we propose a misbehavior report message format that enables an entity to report a detected misbehaving entity. We explain first the functional requirements of a misbehavior reporting mechanism. Then, we detail the data information that are integrated in the reports in order to provide reliable evidences to the misbehavior authority. In section 5.2, we create a set of misbehavior reports based on the format and according to the protocol we propose. We publish the resulting dataset for the research community. It can be used to evaluate the global component of misbehavior detection systems.

5.1 Misbehavior reporting protocol

Misbehavior detection is a set of mechanisms that rely on monitoring C–ITS communications to detect and report potentially misbehaving entities. The system is based on four steps: Local detection, reporting, Global detection and reaction (see section 2.4.3). Although many works on misbehavior detection exist in the literature, to the best of our knowledge, most of them focus only on the first step. In this section, we focus mainly on the second step. We believe that the reporting process is as important as the local detection process because it allows the MA to collect massive amounts of information about potential a misbehavior in the C–ITS system. Consequently, this leads the MA to build a centralized view of the misbehavior situations and to generate reliable misbehavior detection results. The choice of the data integrated to the misbehavior report is a key point that may impact the centralized misbehavior detection process in the MA.

Actually, only few works define the needed data of the Misbehavior Report (MBR) [118] [119]. These works agreed on the fact that evidences should be included in MBRs as a proof of what is reported. Not including this evidence would make forging a report a trivially easy action. However, none of the studies discuss and specify what actually should be these proofs. In this work, we propose a MBR, message format and detail relevant information

that should be included in it. Also, for each detected misbehavior type we propose the corresponding proofs to be included in the MBR as well as a related confidence level. The latter is an indication that enables to differentiate non-forged proofs and self-forged proofs (i.e. if a proof could be forged by the reporting entity).

5.1.1 Misbehavior reporting scenario and requirements

A typical misbehavior reporting scenario occurs when a genuine vehicle detects a suspicious behavior generated by another vehicle sending fake beacons or warnings on the vehicular network. The genuine vehicle reports this misbehavior to the MA located in the back-end security system (see Figure 2.5).

The detection is based on a set of plausibility and consistency checks shown in Table 5.3. These set of checks are performed by a vehicle when receiving a V2X message such as a CAM or DENM. When a vehicle detects a misbehavior, it generates a MBR and sends it to the MA. Notice that the reporting is not a real time process. The report is sent to the MA when a connectivity is available via the cellular network or directly through the ITS-G5 network. The MA should proceed extensive data analysis to investigate whether a misbehavior has occurred or not in the network. Thus, a vehicle does not wait for a decision response about the reported node from the MA. Instead, it should be able to take appropriate decision locally such as blocking packet reception from the suspicious node.

The misbehavior reporting process should fit to the following requirements:

- Privacy protection: The MA should not be able to link the short term and the long term identity of the reported and the reporter entities. The reporter uses its pseudonym to communicate with the MA.
- Efficiency and minimum resource consumption: The MBRs should not overload the communication channel. The reporting process should avoid sending repetitive and redundant information about the same misbehavior.
- Reliability and proof-based: The reporter should integrate the required proofs of the misbehavior: using the input data from the reporter, the MA should be able to re-compute the same misbehavior checks and get the same reported results.
- Flexibility: The MBR should be extensible in order to integrate new misbehavior checks and new data proofs if needed in the future.

5.1.2 Proposed misbehavior reporting approach

5.1.2.1 Misbehavior message

The proposed report format is provided in Annex 9.1 in ASN.1. This format includes several key features detailed in this section:

- Reducing overhead by relating messages

- Verifying the identities with a pseudonym certificate
- Specifying the type of misbehavior
- Specifying the evidence required by misbehavior type

Reducing overhead by relating messages: In our system, the ITS entity should refrain from reporting a misbehaving station that is continuously misbehaving. Instead, the station should send an initial report then wait whilst collecting evidences. After a certain period of time the entity sends a new report that includes the *relatedReportContainer*. This container specifies the ID of the initial report and the number of omitted reports along with the collected evidences. However, if in the meantime the reporter changes its pseudonym, the report should not include the initial report ID. This protocol would indeed prevent the linkability of the reporter pseudonyms by the MA thus ensuring the reporter privacy.

Verifying the identities with a pseudonym certificate: For a report to be valid, the report should be signed with the pseudonym certificate issued by the PKI. This signature is not visible in the report ASN.1. It is included in the security header that encapsulates this message. Additionally, the report format requires at least one valid pseudonym certificate of the reported entity in the *reportedMessageContainer* to be valid.

Specifying the type of misbehavior: The detection type is specified in the *misbehaviorTypeContainer*. It could be on a security or semantic level. Security layer are done on a message as soon as it is received. If the message passes the security checks, it goes to the facilities layer where the semantic checks are applied. In this analysis, we focus mainly on the CAM message, the base message for C–ITS services. However similar approaches for the misbehavior evidence can be applied for other type of messages [68] [69].

In case of a fail on the security level, an *OCTET STRING* should specify the error code (Table 5.1). Every bit set to one infers a failed security test. This variable should include bits for all the security tests specified in the ETSI Technical Specifications [44] and [120]. In case of a fail on the semantic level, the error code would depend on the type of the message included in *reportedMessageContainer*. In the case of a CAM, the fail is linked to one or more data fields as shown in Table 5.3. Therefore, the *OCTET STRING* should point to the relevant data fields (Table 5.2). The error code of the *semanticDetectionReferenceCAM* is coupled with a detection level. Several types of checks can be performed on the CAM, with different types of data necessary to execute them. This is why we propose the following approach: we define 4 level of checks which corresponds to the levels of data we need to perform a check. The levels are defined as follows:

- *Level 1:* Implausibility within a single message.
- *Level 2:* Inconsistencies between successive messages.

- *Level 3:* Inconsistencies with the local environment.
- *Level 4:* Inconsistencies with respect to on-board sensors.

Table 5.1: securityDetectionErrorCode Description

Octet ID	**Bit ID**	**Security Reference**
Verification of the certificate:		
0	0	The certificate is an AT
0	1	The parent certificate is known
0	2	The parent certificate is an AA
0	3	Certificate validity period
...	...	...
Verification of the CAM security profile:		
1	0	signer_info, generation_time and its_aid are not duplicated
1	1	Ascending order of hearder fields
1	2	No Forbiden header fields for CAM
1	3	The payload is present and its length is not nul
...	...	...

Table 5.2: semanticDetectionErrorCodeCAM Description

Octet ID	**Bit ID**	**Data Field**
0	0	ReferencePosition
0	1	Heading
0	2	Speed
0	3	DriveDirection
0	4	VehicleLength
0	5	VehicleWidth
0	6	LongitudinalAcceleration
0	7	Curvature
1	0	YawRate
...	...	...

Specifying the evidence required by misbehavior type: Table 5.3 provide a set of checks for the data fields of the CAM message classed by detection level. The table also includes the required evidence to recreate the misbehavior checks defined by detection level. This evidence is the same data used by the ITS–S to initially detect the misbehavior. This approach allows us to determine what evidence should be included in the *evidenceContainer* based on the error code and the detection level. Here are the data required for the execution of these checks for each level:

- **Level 1:** The CAM of the reported vehicle.
- **Level 2:** A history of 2 or more successive CAMs of the reported vehicle.
- **Level 3:** Level 2 evidence with additional environmental data (eg. LDM).

Table 5.3: Misbehavior Detectors For Cooperative Awareness Messages (CAMs)

CAM Data	Detection Level			
	Level 1	**Level 2**	**Level 3**	**Level 4**
Reference Position	· Data Unavailable · Confidence Too Large	· Position Change (PC) Too Large · PC $Inc\Phi^*$ with Speed · PC $Inc\Phi$ with Heading	· Position not on a Road · Position overlap with other Vehicles	· Position $Inc\Phi$ with Relative Position (Lidar Radar) · Position $Inc\Phi$ with Maximum Plausible Range
Heading	· Data Unavailable · Confidence Too Large	· Heading Change (HC) Too Large · HC $Inc\Phi$ with Speed · HC $Inc\Phi$ with YawRate	· Heading $Inc\Phi$ with Road Heading	· Heading $Inc\Phi$ with Relative Heading
Speed	· Data Unavailable · Confidence Too Large · Speed Vaue Too High	· Speed Change (SC) Too Large · SC $Inc\Phi$ with Acceleration	· Speed $Inc\Phi$ with Road Plausible Speed	· Speed $Inc\Phi$ with Relative Speed
Drive Direction	· Data Unavailable	· Direction $Inc\Phi$ with PC & Heading · Direction $Inc\Phi$ with Speed	· Direction $Inc\Phi$ with Road Way	· Direction $Inc\Phi$ with Perceived Direction
Vehicle Length/Width	· Data Unavailable	· Length/Width Change	-	· Vehicle Length and Width $Inc\Phi$ with Perceived Dimentions
Longitudinal Acceleration	· Data Unavailable · Confidence Too Large · Acc Value Too High	· Acceleration Change Too Large	-	· Acceleration $Inc\Phi$ with Relative Acceleration
Curvature	· Data Unavailable · Confidence Too Large · Curve Radius Too Small	· Curvature Change (CC) Too Large · CC $Inc\Phi$ with Speed · CC $Inc\Phi$ with HC · CC $Inc\Phi$ with YawRate	· Curvature $Inc\Phi$ with Road Shape	· Curvature $Inc\Phi$ with Relative Curvature
YawRate	· Data Unavailable · Confidence Too Large · YawRate Value Too High	· YawRate Change (YC) Too Large · YC $Inc\Phi$ with Speed · YC $Inc\Phi$ with Curvature	-	· YawRate $Inc\Phi$ with Perceived YawRate
Evidence Required	· One Reported CAM (At least Including a full certificate)	· Multiple reported CAMs (At least one Including a full certificate)	· One Reported CAM (With a full certificate) · CAMs of neighbors (With a full certificate each) · Map of the Area (Already available for the MA)	· One Reported CAM (With a full certificate) · Sender Sensor Information
Evidence Reliability	Total	Total	Partial	Minimal

*$Inc\Phi$: inconsistency symbol.

- **Level 4:** Level 2 evidence with additional local sensor data which could be encapsulated in a Collective Perception Message (CPM) [35].

For each of these levels, we associate a value of trust. Indeed, levels 1 and 2 are reliable because the proofs consist only of the received CAMs, which are signed by the sender. Whereas levels 3 and 4 require environmental or internal data to the reporter vehicle that it could have forged himself in order to report an innocent vehicle. Confidence is therefore lower in an MBR of level 3 or 4. Nevertheless, we consider that it is more useful to send a MBR to the MA than not to send any at all, even with unverifiable evidence. It will be up to the MA to identify, with the global vision it has via the MBRs received from other vehicles, whether the MBRs should be taken into account or not.

5.1.2.2 Misbehavior report protocol

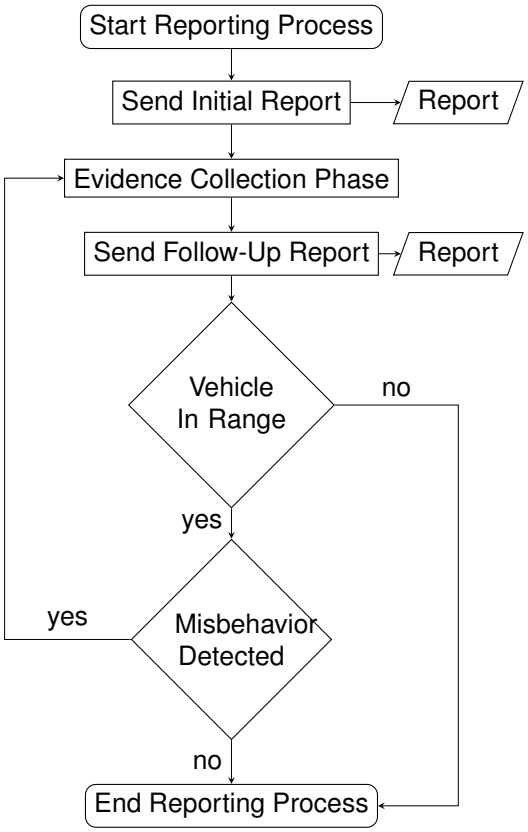


Figure 5.1: Reporting protocol: flowchart description

The misbehavior reporting protocol is designed to include the necessary information for the back-end detection while simultaneously reducing network overhead (see Fig. 5.1). Basically speaking, a vehicle performs the following actions upon detection of a malicious node:

i. Send an initial report including the necessary information described in Section 5.1.2.1.

ii. Collect evidence for a predefined time period.

iii. Send a follow-up report with all the evidence collected in the previous step.

iv. According to the current status, if:

   (a) A new anomalous behavior is detected while collecting evidence: go back to step ii.

   (b) The reported vehicle is out of range: End reporting process

   (c) No misbehavior is detected: End reporting process

5.1.3 Conclusion

In this section, we proposed a detailed misbehavior reporting protocol which provides a set of misbehavior proofs to the central misbehavior authority. This allows the misbehavior authority to reproduce the reported misbehavior detection results and to combine them with other received reports. We defined precisely the report format in ASN.1 and describe the functionalities of each field of the message. We based this analysis on the CAM message, but we plan on extending this work for all the types of messages.

5.2 Misbehavior reports dataset

After introducing a misbehavior reporting format, we wanted to facilitate the use of this format by researchers. For this reason, we created and published the DAtaset of REports (DARE). DARE consists of reports of misbehaving entities collected from individual ITS–S performing simple plausibility checks. It includes multiple scenarios and parameters that could help researchers evaluate their solutions for different use cases. The dataset was also used in our work on global detection, where we propose solutions for the MA architecture (see section 6.1 and section 6.2). The free publication of this dataset in open source format will enable researchers to better reproduce and evaluate our global detection systems as well as improve and adequately compare future proposals.

5.2.1 Related works

Numerous misbehavior detection systems are available in the literature. However, relative effectiveness of a certain solution compared to other is seldom correctly evaluated. To this end, researchers require an adequate dataset for comparable evaluations. Multiple types of datasets exist in the literature as described in [86]. They can be divided into the following two categories:

- Real-world datasets: these datasets collected using digital video cameras or deployment projects are particularly valuable due to the unprecedented level of detail and accuracy. Their limitation however is the difficulty to capture cases of misbehavior detection. E.g. it is difficult legally to deploy real misbehaving entities on the road.
- Core simulation datasets: these datasets rely on software implementations of mathematical models that replicate fundamental driver behavior logic. For example, SUMO and VEINS are well developed simulation software often used to test vehicular solutions.

Gozálvez et al. presents a field test campaign as part of the iTETRIS European research project [121]. Their work is an example of a deployment project dataset. The project aims at testing the quality of IEEE 802.11p Vehicle to Infrastructure communications. Their dataset includes 22 different RSU broadcast messages to a vehicle moving in an urban environment. Additionally, it contains the local positioning information and the Received Signal Strength Information (RSSI) of the received messages for different positions of the deployed vehicles. However, this dataset includes only normal behavior without any attacker data, which is restrictive for misbehavior detection evaluation. In our proposed dataset, we have normal and misbehaving entities, which can be used to evaluate misbehavior schemes.

Van der Heijden et al. propose a VeReMi for local misbehavior detection validation [81]. Their work is an example of a simulation dataset. They used VEINS, a co-simulation of SUMO and OMNET++. The dataset consists of message logs for every vehicle in the simulation (vehicle position and speed and the message RSSI). The number

of vehicles, the number of attackers, as well as the attacker rates, and many other parameters are also provided. VeReMi was specific for local misbehavior detection component and is not as useful for studies targeting the global detection component.

Accordingly, we create a new dataset tailored for testing the global detection component. The dataset includes multiple scenarios, attacker percentage and vehicle densities. This work can help other researchers working on this topic to test their global misbehavior detection solutions and compare their contributions to ours.

5.2.2 System model

5.2.2.1 Simulation scenarios

In order to create this dataset, we use our F^{2}MD simulator. We use the networks provided in F^{2}MD described in section 3.3.2. Specifically, we use the large $6.51km^2$ LuST scenario as the main base for the dataset, but we also include a set of the $1.11km^2$ Paris-Saclay scenarios as a test bench.

5.2.2.2 Attacker model

In this study we consider the attacker as insider, i.e. the attacker is an authenticated vehicle with a set of valid certificates. We also consider that the attacker has full unrestricted access on his previously acquired certificates and could change it at will. Our dataset includes the types of misbehavior described in section 3.3.3.

5.2.2.3 Local detection

Local misbehavior detection is based on simple and fast checks on the data that are fused within a detection application. The local checks consist mostly of verifying the plausibility and consistency of V2X message data. In this study, 18 selected checks are executed simultaneously on every message. These checks are described in section 3.3.4.1 and span levels 1, 2 and 3 described in section 5.1.2.1.

The local detection application is a layer used to evaluate the previously calculated checks and determine if a report to the global MA is required. For the generation of this dataset we use a non-cooperative trust based approach. This approach is described in section 3.3.4.2.

5.2.3 Proposed dataset description

5.2.3.1 Dataset format

Our proposed dataset includes the initial reports and follow-up reports as described in section 5.1.2. Both types of reports are encoded in JSON, a lightweight data-interchange format. The format has the following description:

```
{"Report":{
  "Metadata":{
    "senderId":ℤ[0,+∞],
    "reportedId":ℤ[0,+∞],
    "generationTime":ℝ[0,+∞],
    "senderRealId":ℤ[0,+∞],
    "reportedRealId":ℤ[0,+∞],
    "attackType":"String"
  },
  "Messages":[
    {
      "CreationTime":ℝ[0,+∞],
      "Pos":[ℝ[−∞,+∞],ℝ[−∞,+∞],ℝ[−∞,+∞]],
      "Speed":[ℝ[−∞,+∞],ℝ[−∞,+∞],ℝ[−∞,+∞]],
      "Accel":[ℝ[−∞,+∞],ℝ[−∞,+∞],ℝ[−∞,+∞]],
      ...
    },
    ...
  ]
  "Checks":[
    {
      "rangePlausibility":ℝ[−∞,+1],
      "posPlausibility":ℝ[−∞,+1],
      "posConsistency":ℝ[−∞,+1],
      "speedPlausibility":ℝ[−∞,+1],
      ...
    },
    ...
  ]
}}
```

5.2.3.2 Dataset categories

Annex 9.2 shows all the subsets categories available in our dataset. In this dataset the evidence collection phase of the reporting protocol is set to 10 seconds. We mainly altered four variables: the vehicle traces, time of day, attacker rate and attacker type. The vehicle traces change between the Luxembourg and the Paris-Saclay network. We generally use the Paris network as our Test-Bench. The changes in the time of day entails a change in the general vehicle density as shown in Figure 3.4d. The attacker rate is stated for every scenario. The attack launcher is designed to inject a new attacker when the rate drops below the set value. All the listed datasets could be downloaded individually from our Cloud Drive [122].

5.2.3.3 Local detection results

Table 5.4: Dataset scenarios local detection

Id	Local Detection Metrics		
	Accuracy	F_1 *Score*	*C's Kappa*
Lust-0024-25S	0.91379	0.86995	0.65324
Lust-0611-15M	0.98191	0.93798	0.86432
Lust-0611-15A	0.90524	0.82466	0.55098
Lust-1115-25A	0.89421	0.86547	0.61923
Lust-1115-05M	0.99183	0.91278	0.82463
Lust-1521-25M	0.97060	0.93797	0.85216
Lust-1521-05A	0.96135	0.79448	0.54616
Paris-0024-05S	0.98840	0.91767	0.83622

Table 5.4 shows the local detection metrics for some scenarios. The list of all the local detection results with more detection metrics is included with the dataset. The detection quality is based on the detection checks and applications described in section 5.2.2.3. To evaluate the detection quality, we used the *Accuracy*, F_1 *Score* and *Cohen's kappa*. These detection metrics are described in section 3.3.8.

These results show that the attacker rate and type alteration affect the local detection quality. Consequently, it also affects the number and quality of collected misbehavior reports. We strongly believe that there is a relation between these variables and the global detection results. Accordingly, these metrics should be considered when analyzing results extracted from this dataset.

5.2.3.4 Dataset parser

We provide a specific parser designed to read and store this type of data. This parser is provided in order to facilitate the treatment and study of the proposed datasets. This parser is implemented in python to enable easy machine

learning applications. The data is sorted automatically by reported pseudonym. A simple threshold application example is provided along with evaluation metrics and graph plotting mechanisms. This parser can also be found in open source format on our GitHub [26]

5.2.4 Conclusion

Global Misbehavior detection in C–ITS is still a developing topic. The demand for robust and concrete detection solution is rising especially in ITS standardization working groups of the ETSI and IEEE. We provide a large dataset of misbehavior reports for comparative evaluation of global detection solutions. This dataset is aimed at facilitating and catalyzing the currently in demand work on the global side. Now that we have defined the report, in the next chapter we work on the design of the MA which receives and processes these MBRs.

Chapter 6

Global Misbehavior Detection

Global misbehavior detection in C–ITS is carried out by a central entity called the Misbehavior Authority (MA). The global detection is based on the Misbehavior Reports (MBRs) sent by Vehicle's OBUs and by RSUs. By analyzing these reports, the MA is able to compute various misbehavior detection information. In section 6.1, we propose and evaluate different Machine Learning (ML) based solutions for the internal detection process of the MA. In section 6.2, we extend the ML-based MA with a mechanism to detect Sybil attacks.

6.1 Machine learning based misbehavior authority

The first step in the global misbehavior detection is the collection of reports coming from the local vehicles. The MA receives a large quantity of data in these reports. This data need to be analyzed to detect different patterns of misbehavior. Due to the large amount of expected data, we believe that the MA will benefit from using Artificial Intelligence (AI) solutions such as Machine Learning (ML) to perform this task.

In our model, the goal of the MA is to classify the reported ITS–S as:

1. Misbehaving (and what kind of attack it does),
2. Faulty (e.g. the ITS–S has a broken sensor),
3. Genuine (false positive).

We cast this problem as a multi-class classification problem. We are given a series of observations $(x^1, x^2, ..x^n)$ and the task is to train a classifier that generates predictions $\hat{y}$ of the true labels y. In our context, the data we are dealing with is sequential. Within the same ITS–S, all the data sent to the MA are time-dependent. To elaborate, data of more recent MBRs depends on data of previously received MBRs. Therefore, a predictive system that can learn and model these types of dependencies is highly recommended.

In this work, we propose and evaluate different ML-based solutions for the internal detection process of the MA. We use the F^{2}MD simulator (see section 3) and show through extensive simulation and several detection metrics the ability of these solutions to precisely identify different misbehavior types.

6.1.1 Simulation settings and scenarios

In order to evaluate any machine learning model, we first need a set of data. We use the DARE Dataset described in section 5.2. We use different simulation scenarios provided in F^{2}MD for the training part and testing part of our algorithms (see section 3.3.2). We use the 24h LuST scenario vehicle traces [96] for the training of our models (see entry Lust-0024-25S in Annex 9.2). As for the testing, we used the Paris Saclay network (see entry Paris-0024-05S in Annex 9.2). The choice of this test scenario has a purpose of having a significantly different train and test set. We use a mix of all the attacks introduced in section 3.3.3, except the Sybil attacks which are treated in section 6.2.

6.1.2 Model Development

In this section we describe the step by step process we followed in the development of our machine learning model. During this development we followed an iterative process. In this process, we define three key elements of our model: the features, the algorithm and the hyper-parameters. We first start by defining and improving the quality of the ML features.

6.1.2.1 Feature selection

A feature is any attribute that can be used to characterize the data. They are individual independent variables that serve as inputs to a ML system. The quality and quantity of the features can highly affect the results we are trying to achieve. Thus, selecting a good set of features is of paramount importance. In this step, we use the gradient boosting trees algorithm eXtreme Gradient Boosting (XGBoost) proposed by Chen et al. in [109]. Our choice of XGBoost is based on two main reasons:

1. **Kaggle performance:** Kaggle is a Google owned web platform for organizing data science competitions [123]. On this platform, companies offer problems in data science and offer a price to the best performing system. XGBoost is consistently amongst the best performing algorithms in various use cases.

2. **CPU intensive learning:** During this thesis we had access to a High Performance Computing (HPC) unit with four Intel(R) Xeon(R) CPU E7-8880 v4 @ 2.20GHz resulting in 88 CPU cores and 176 CPU threads. Since XGBoost learning is done on CPU threads, our setup enables us to quickly run multiple instances of the algorithm with different hyper-parameters. Using model selection tools like the grid search by scikit learn [99] and the hyperopt python library [124] we are able to easily hyper-optimize and cross-validate our algorithm.

This parameter hyper-optimization would enable us to more confidently compare different models and attribute the difference in performance to the input features.

Before selecting any features, we first start by calculating a weighted random predictor on the test data. This predictor helps us determine that a certain result is caused by a trained model and not simply attributed to chance. In other words, the result of this predictor will enable us to attribute any higher accuracy classification to the selected features instead of the unbalanced nature of the test data set. Table 6.1 shows the prediction evaluation metrics averaged on 100 runs.

Table 6.1: Prediction results for the weighted random predictor

Evaluation Metrics	*Precision*	*Recall*	F_1 *score*	*Accuracy*
Macro average	0.0624	0.0624	0.0624	0.0680
Weighted average	0.0680	0.0680	0.0680	

6.1.2.1.1 Feature set: single check

The first and most obvious feature set we tested are the plausibility checks done on the messages previously developed for the local detection (see section 3.3.4.1). We wanted to quantify the usefulness of this information in determining the type of misbehavior. We extracted the plausibility checks from each report and used them as individual features to the XGBoost model. Table 6.2 shows the results of the hyper-optimized and 5 fold cross validated model compared to a weighted random predictor. These results show a large and clear improvement with respect to the weighted random model. This result validates our hypothesis that the checks could be used to identify the type of misbehavior.

Table 6.2: Prediction results for the single check feature set

Evaluation Metrics	*Precision*	*Recall*	F_1 *score*	*Accuracy*
Macro average	0.8360	0.8298	0.8249	0.8334
Weighted average	0.8511	0.8334	0.8379	

To make our results easily reproducible, we include the following XGBoost hyper-optimized parameters: (a) Learning rate: *0.5*, (b) Maximum depth of a tree: *9*, (c) Learning objective: *binary:logistic*, (d) Minimum sum of instance weight needed in a child: *0*, (e) Subsample ratio of the training instances: *0.9*, (f) Subsample ratio of columns when constructing each tree: *1.0*, (g) All other parameters were set to their default value. The explanation behind each value could be found in [109].

Table 6.3: Prediction results for the checks average feature set

Evaluation Metrics	*Precision*	*Recall*	F_1 *score*	*Accuracy*
Macro average	0.9757	0.9687	0.9714	0.9715
Weighted average	0.9731	0.9715	0.9716	

6.1.2.1.2 Feature set: Checks average

The previous test shows that a single independent set of checks does contain information that is useful to identify a type of misbehavior. However, while treating the data one useful part of the information was lost: the relation between reports. Multiple reports are received from multiple ITS–Ss signaling the same misbehaving station. The plausibility checks in these reports could be treated together to give a more accurate description of the misbehavior. To fuse the checks in these related reports, we simply opt to calculate these four functions: (1) the mean value, (2) the minimum value, (3) the maximum value, (4) the standard deviation. Using the merged features as input, we re-trained a new XGBoost model. Table 6.3 shows the results of the hyper-optimized and 5 fold cross validated model. This result shows that the relation between reports contains valuable information in the detection of the misbehavior type.

To make our results easily reproducible, we include the following XGBoost hyper-optimized parameters: (a) Learning rate: *0.1*, (b) Maximum depth of a tree: *4*, (c) Learning objective: *binary:logistic*, (d) Minimum sum of instance weight needed in a child: *5*, (e) Subsample ratio of the training instances: *1.0*, (f) Subsample ratio of columns when constructing each tree: *0.7*, (g) All other parameters were set to their default value.

6.1.2.1.3 Feature set: Beacon data

Previously, we focused on the features based on the plausibility checks done on the data. However, the reports also include the raw data of the beacons such as: (1) the position, (2) the position confidence, (3) the velocity, (4) the velocity confidence, (5) the acceleration, (6) the acceleration confidence, (7) the heading, (8) the heading confidence, (9) the dimensions. Using this data, we calculate the absolute difference between the values of two consecutive beacons emitted by the same ITS–S. Finally, we combine the raw data and the absolute difference and we calculate (1) the mean value, (2) the minimum value, (3) the maximum value, (4) the standard deviation. We use the output of these function as input features for a new XGBoost model. Table 6.4 shows the results of the hyper-optimized and 5 fold cross validated model. These results show that the information contained in the raw data that is untreated with the plausibility checks is still valuable with respect to detection of the misbehavior type.

To make our results easily reproducible, we include the following XGBoost hyper-optimized parameters: (a) Learning rate: *0.3*, (b) Maximum depth of a tree: *3*, (c) Learning objective: *binary:logistic*, (d) Minimum sum of instance weight needed in a child: *0*, (e) Subsample ratio of the training instances: *1.0*, (f) Subsample ratio of columns when

Table 6.4: Prediction results for the beacon data feature set

Evaluation Metrics	*Precision*	*Recall*	F_1 *score*	*Accuracy*
Macro average	0.9314	0.8998	0.8926	0.9043
Weighted average	0.9203	0.9043	0.8920	

constructing each tree: *0.8*, (g) All other parameters were set to their default value.

6.1.2.1.4 Feature set: Combined data

Previously, we treated the data and the plausibility checks independently. In combining both these feature sets, we should provide the XGBoost algorithm with more information to predict the misbehavior type. Table 6.5 shows the results of the hyper-optimized and 5 fold cross validated model. The results are slightly better than both the previous models. We consider this set of features as a baseline for the algorithm optimization.

Table 6.5: Prediction results for the combined data feature set

Evaluation Metrics	*Precision*	*Recall*	F_1 *score*	*Accuracy*
Macro average	0.9860	0.9786	0.9819	0.9814
Weighted average	0.9818	0.9814	0.9812	

To make our results easily reproducible, we include the following XGBoost hyper-optimized parameters: (a) Learning rate: *0.5*, (b) Maximum depth of a tree: *1*, (c) Learning objective: *binary:logistic*, (d) Minimum sum of instance weight needed in a child: *0*, (e) Subsample ratio of the training instances: *0.8*, (f) Subsample ratio of columns when constructing each tree: *0.8*, (g) All other parameters were set to their default value.

6.1.2.2 Algorithm selection

After selecting the relevant features towards detecting the misbehavior type, we now focus on finding the optimal algorithm to take advantage of these features. Until now, we have used XGBoost for all of our training and predictions. Even though XGBoost is amongst the best classifiers in the literature, the performance of a classification algorithm is use case dependent. Therefore, for our use case, we evaluate three additional ML algorithms: Random Forests (RF), Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM).

6.1.2.2.1 Random Forests (RF):

RF and XGBoost are both tree-based models. The difference between the two lies in the way the trees are constructed: the order and the combination methods of different branches. Table 6.6 shows the results of the hyper-optimized and 5 fold cross validated RF model. The RF model results are slightly worse than the XGBoost ones.

This is expected and in line with other use cases of the state of the art where XGBoost generally has better performance than RFs.

Table 6.6: Prediction results for the RF

Evaluation Metrics	*Precision*	*Recall*	F_1 *score*	*Accuracy*
Macro average	0.9843	0.9548	0.9616	0.9601
Weighted average	0.9645	0.9601	0.9542	

To make our results easily reproducible, we include the following RF hyper-optimized parameters: (a) The maximum depth of the tree: *12*, (b) The minimum number of samples required to be at a leaf node: *2*, (c) The minimum number of samples required to split an internal node: *5*, (d) All other parameters were set to their default value.

6.1.2.2.2 Multi-Layer Perceptron (MLP):

MLP is a type of artificial neural network organized in several layers within which information flows only in one direction: from the input layer to the output layer. Therefore, it is a feed-forward network. Each layer is made up of a variable number of neurons, the neurons of the last layer being the outputs of the global system. Table 6.7 shows the results of the hyper-optimized and 5 fold cross validated MLP model. The MLP performs slightly worse than the XGBoost model. This could be due to the relatively low size of the data compared to a real world use case. Deep learning would be more beneficial while using a larger sample of collected data. The results could also be due to the heterogeneous nature of the input features.

Table 6.7: Prediction results for the MLP

Evaluation Metrics	*Precision*	*Recall*	F_1 *score*	*Accuracy*
Macro average	0.9536	0.9615	0.9559	0.9676
Weighted average	0.9706	0.9676	0.9684	

6.1.2.2.3 Long Short-Term Memory (LSTM):

A LSTM is a type of Recurrent Neural Network (RNN) with short-term and long term memory. An RNN is a network of artificial neurons with recurrent connections. A recurrent neural network is made up of interconnected neurons interacting non-linearly and for which there is at least one cycle in the structure. The units are connected by weighted synapses. The output of a neuron is a non-linear combination of its inputs.

The LSTM model is able to take as input a time series of observation. This means this algorithm could handle the series of features, from beacon data to plausibility checks, without the need to combine them with an average. This means it could represent and benefit from the relation between successive observation to increase the detection

quality. Table 6.8 shows the results of the hyper-optimized and 5 fold cross validated LSTM model. The LSTM model has a similar performance to the XGBoost. However, according to the literature, we suspect that this model would scale better with larger data samples.

Table 6.8: Prediction results for the LSTM

Evaluation Metrics	*Precision*	*Recall*	F_1 *score*	*Accuracy*
Macro average	0.9760	0.9788	0.9771	0.9816
Weighted average	0.9823	0.9816	0.9817	

6.1.3 Summary

In this work, we focus on global misbehavior detection in C–ITS. Specifically, we explore ML solutions for global misbehavior type classification. To achieve this goal, we extract features from the local ITS–Ss detector checks and engineer additional features from the raw beacon data. We evaluate the effects of these features on the detection quality. Last, we propose and test various ML algorithms and analyze their results. These tests are supposed to serve as an indication or a starting point for implementing an ML based MA. However, a better algorithm selection should be done on real data collected from a large-scale deployment.

6.2 Misbehavior authority for Sybil attack detection

This work extends the solution proposed in section 6.1. Here, we focus on Sybil attacks. A Sybil attack takes place when an ITS–S takes advantage of its available pool of pseudonyms and uses them simultaneously to disturb the system: it periodically broadcasts V2X messages and signs them with different pseudonyms. The pseudonyms used for the Sybil attack are valid which complicates the MA detection. We propose a misbehavior detection process at the MA level which is able to identify and detect both Sybil and other types of attacks. It is based on advanced ML algorithms. In addition, we evaluate our solution by integrating it in both ETSI and IEEE C–ITS standard architecture.

6.2.1 The Sybil attack

In the current C–ITS system, each vehicle should use a single pseudonym certificate for a certain time period to sign its generated V2X message. However, it is important to ensure the ability of vehicles to continuously send V2X messages even without connection to the PKI. Therefore, it is necessary that the vehicle possesses a pool of several valid pseudonyms simultaneously. The European Commission published a Security Policy where it recommends the use of a maximum pool of 100 valid pseudonym certificates [125]. Even with this pool of pseudonyms, vehicles should not use more than one pseudonym certificate during a certain period of time to sign their messages. However, a misbehaving vehicle may intentionally use multiple valid pseudonym certificates at the same time, which results in a Sybil attack.

In section 3.3.3.2 we propose different types of Sybil attacks in the V2X network. In this section, we expand this definition with a more in-depth description:

1. S1 Traffic Congestion Sybil: As shown in figure 6.1, the attacker uses valid pseudonyms to simulate multiple ghost vehicles. Vehicles within the communication range of the malicious vehicle receive the fake messages and conclude that a congestion occurs on the road. The attacker intelligently calculates the kinematic data for the ghost vehicles such that the fake messages have plausible and coherent contents.

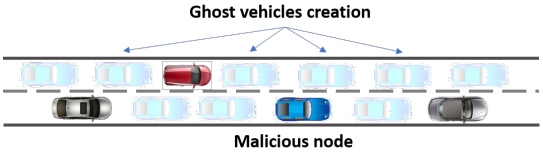


Figure 6.1: Sybil attack: Traffic congestion

2. S2 Data replay Sybil: This attack consists on reporting legitimate vehicle as malicious. The attacker chooses a victim vehicle and creates messages containing positions broadcasted by the victim vehicle. As shown in figure 6.2, the attacker sends at time t=1 a message containing the same position (X1) as the victim vehicle.

One of the hardest challenge of the detection system is to know which node is the real one (the victim) and which one is the ghost one. In this case, there is a good probability that the victim vehicle is reported as attacker.

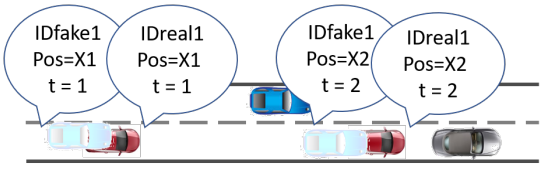


Figure 6.2: Sybil attack: Data replay

3. S3 Dos Random Sybil: As shown in figure 6.3, the attacker creates messages with random data (e.g., the position is not on the road). The attacker uses a different pseudonym for every sent message. The motivation behind such attack could be to overwhelm the misbehavior detection algorithms of neighboring ITS–S.

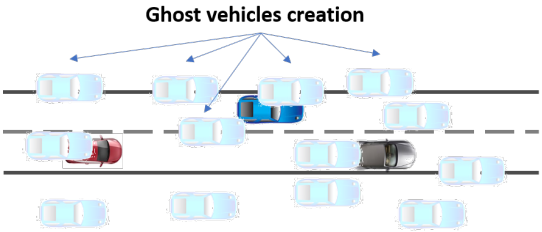


Figure 6.3: Sybil attack: DoS Random

4. S4 Dos Disruptive Sybil: This attack is a combination between S3 and S2. As shown in figure 6.4, the attacker uses a different pseudonym for each message but does not fill them with random data. Instead, the transmitted data is based on the ones received from the neighboring vehicles. The difference between S4 and S2 is that S4 does not follow one victim, the attacker is trying to disturb the system with sudden appearance of vehicles. For example, the attacker sends at time t=1 a message containing a position (pos=X1), and at time t=2 a message containing another position (pos=X2) which is the position of another vehicle. The motivation of the attacker could be the degradation of the safety system quality thus decreasing the reliability of the exchanged information.

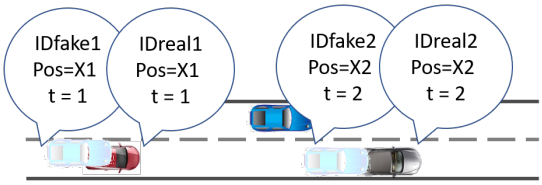


Figure 6.4: Sybil attack: DoS disruptive

6.2.2 Misbehavior authority investigation process

We propose a MA system architecture (see figure 6.5). The proposed architecture consists of three main phases: *General Misbehavior Type Detection*, *Pseudonym Linkage* and *Sybil Type Detection*. The MA system takes a MBR as input and returns the predicted attack type as output. In the first phase we start by detecting misbehavior types related to one single pseudonym identity. This phase is explored in detail in section 6.1. This detection is effective against misbehavior types that are non-Sybil, since all reports related to the attack will also be linked by the attacker's pseudonym. However, this detection fails against attacks that makes use of multiple pseudonyms. To address this problem, we propose the pseudonym linking schemes. In the second phase, we attempt to link the pseudonyms related to the same physical reported ITS–S. If no link is found the process is complete and the misbehavior type is returned. If a link is found, then a Sybil attack is suspected and the linked pseudonyms are candidates for Sybil attack type detection in phase three. In this third phase, the linked pseudonyms are treated as one and the evidences collected from all the reports of linked pseudonyms is used in a specific Sybil type detection process. The predicted Sybil misbehavior type is returned and the process is complete.

6.2.2.1 Phase 1: General misbehavior type detection

The goal of this phase is to detect as accurately as possible the type of misbehavior related to one pseudonym. This evaluation of this phase is described in more detail in section 6.1. It is accomplished using the following steps:

1. Pre-processing:

    - Database Storage: We start by adding the reports to a spatial database. This enable us to do fast and efficient geographic queries.
    - Filter Similarities: We aggregate similar data from multiple reports (e.g. several ITS–S detecting the same implausibility and sending the same evidences). We found this filtering to significantly improves the detection speed and quality.
    - Feature Generation: We extract key information from the collected evidence in our database. These information, also called features, are then used by the ML-based detection algorithm to determine the type of misbehavior. The quality of the extracted features is crucial to the robustness and accuracy of the detection. We evaluated different sets of features through cross validation on our relatively large data-set and verified a set of 30 features:
        - The local detection checks done on V2X messages described in section 3.3.4.1.
        - The $Position$, $Speed$, $Acceleration$, $Heading$ and $Time$ of the last beacon.
        - The $\Delta Position$, $\Delta Speed$, $\Delta Acceleration$, $\Delta Heading$ and $\Delta Time$ between the last 2 beacons.
        - The $\Delta Time$ between the last 2 received reports.

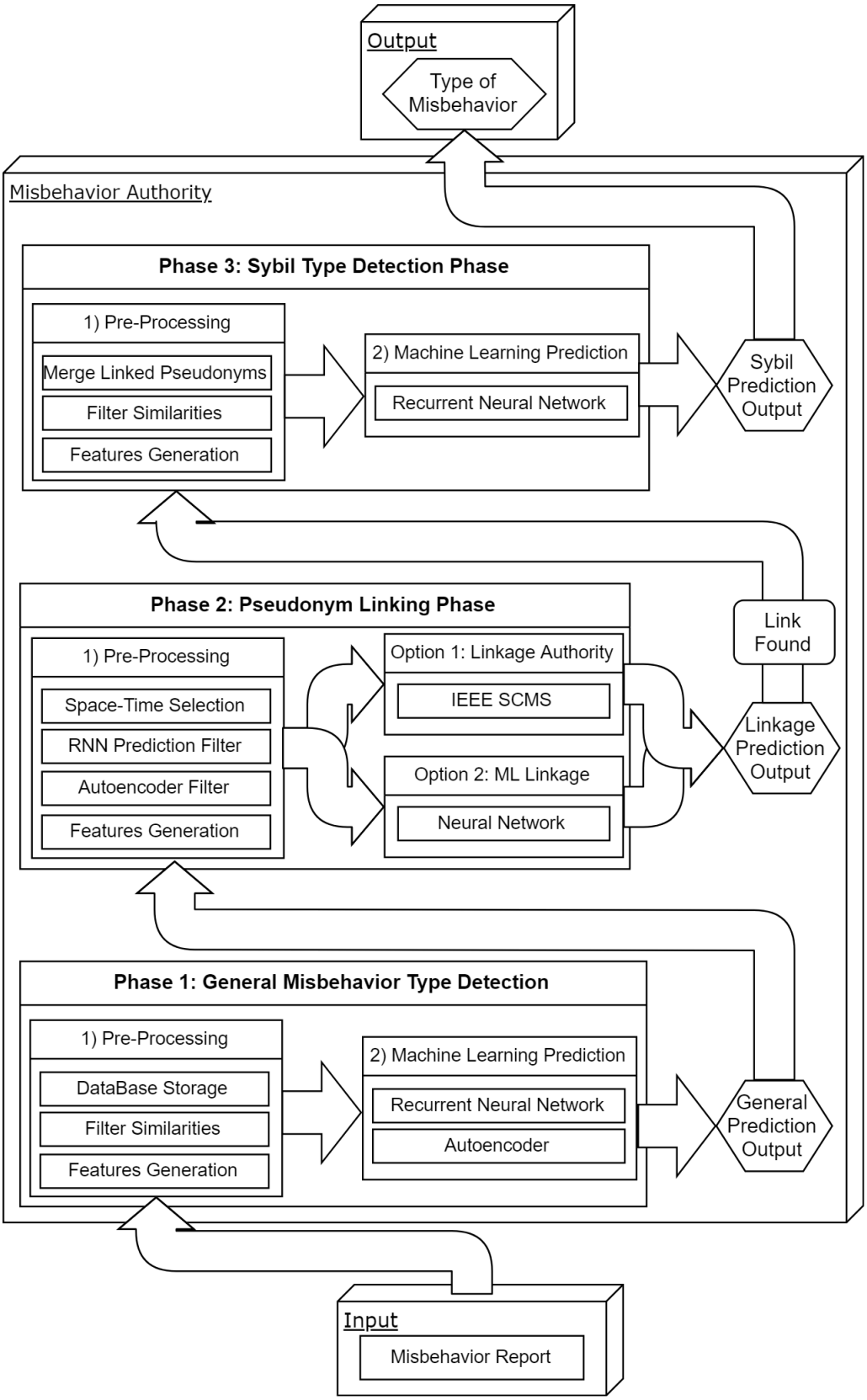


Figure 6.5: Sybil attack: global detection system architecture

- The number of time this evidence has been received (e.g. the number of filtered reports data).

2. Prediction:

   - Autoencoder: An autoencoder is a ML tool used to reduce the dimensions of data. We use it compress the previously created features. Although this step may not be important for the detection of a non-Sybil attack, the result is useful for the pseudonym linkage Phase.
   - RNN: We provide the previously calculated and compressed features by the auto-encoder to an RNN. The choice of an RNN was made due to the temporal relation between the received reports. We tested different simple models and determined that the LSTM has a good performance in our use case. Hence, for our testing purposes we use an LSTM. However, additional experimenting is needed to explore the efficiency of different models, thus more complex and elaborate models could be proposed in the future.

6.2.2.2 Phase 2: Pseudonym linking

The goal in this phase is to link the pseudonyms coming from the same vehicle as accurately as possible. However, in order to be compliant with both US and European C-ITS Systems, we explore two options for pseudonym linking.

1. Pre Processing:

   - Space-Time selection: In this step we use the spatial database to recall all the reports within a range and time of the reporter node. We propose this for processing efficiency reason, e.g. it prevents having to test all the previously received pseudonyms and limits the detection to the target region.
   - Prediction Filter: We use the output prediction of the RNN to filter reports with diverging predictions. If the RNN detects the same type of attack for two different pseudonyms in the same region and type, we consider them candidates for the linking test. Otherwise, the pseudonyms are discarded.
   - Autoencoder Distance Filter: We use the output prediction of the autoencoder to filter reports with diverging compressed features. We calculate distances between the compressed features of the recalled and the current pseudonym. We exclude the pseudonyms with compressed features far from the current one.
   - Linkage Features Generation: Similarly to the first feature generation step, we need to extract the relevant information form the selected pair of pseudonyms. These features are used by the ML algorithm to determine if the reported pseudonyms are linked or not. Therefore, from each pair of reports we extract and similarly validate the following set of features:
     - The difference between all the previously calculated features of the two latest received report of each pseudonym.
     - The Euclidean distances between the reporter ITS–S position and broadcasted position of the reported pseudonym for both selected pseudonyms.

- The Euclidean distances between the reporter ITS–S position of one pseudonym and the broadcasted position of the other reported pseudonym.
- The absolute difference between the two latest RNN predictions of the selected pseudonyms.

2. Linking:

- Linkage Authority (Option 1): The US architecture supports a Linkage Authority (LA). The LA is able to cooperate with the PKI to link several pseudonyms that belong to the same vehicle. This enables us to do straightforward linking between the selected pseudonyms. No ML-based prediction is needed.
- ML-based linking (Option 2): The European architecture lacks a LA. To cope with this issue we propose using a ML-based solution. The goal of this solution is to determine, using the previously calculated features, if two reported pseudonyms are generated by the same physical ITS–S. For testing purposes we use an MLP, which is the classical type of neural networks. However, more rigorous experimenting is needed to propose more complex solutions.

6.2.2.3 Phase 3: Sybil type detection

This algorithm is activated if a link is found in the previous phase. The goal is to detect the type of Sybil attack related to the number of linked pseudonyms in the previous phase.

1. Sybil Algorithm Pre-Processing:

- Merge Multiple Linked Pseudonyms: In this step we prepare a new database entry where we merge the evidence data of the multiple linked pseudonyms.
- Filter Similarities: Similarly to the previous filter, we aggregate similar data from the new database entry. This also improves the prediction performance.
- Sybil Features Generation: We extract from the new and filtered database entry the key detection information. These features are the indications used by the ML algorithm to determine the type of Sybil attack. We create the same features used by the general algorithm described in the first phase. Additionally, we add two specific feature to the Sybil type detection:
  - The number of linked pseudonyms.
  - The number of reports in the new database entry.

2. Prediction:

- Recurrent Neural Network: Similarly to the general prediction algorithm, we provide the previously calculated features to an RNN. We also use the LSTM for testing purposes. The Model and the ML algorithms and hyper-parameters should be investigated further.

Finally, the output of the MA algorithm will be the *Sybil Attack Type* if a pseudonym link is found and the *General Misbehavior Type* otherwise.

6.2.3 Simulation settings and scenarios

In order to evaluate our proposed solution, we use the F^{2}MD framework (see section 3). We use the LuST scenario for the vehicle traces proposed in F^{2}MD (see section 3.3.2). We use different sections of the scenario for the training part and testing part of our ML algorithms. We use the Large LuST scenario for training and the Small one for testing. Both scenarios have an attacker rate of 5%. We use the reporting protocol proposed in section 5.1 to transmit the reports. This results in *5,209,072* MBRs for training and *294,160* MBRs for testing.

In both scenarios we implement the attacks described in section 6.2.1. Additionally, we include a set of other types of misbehavior in order to increase the complexity of the classification. These types proposed and described in F^{2}MD (see section 3.3.3): *Fixed Position Offset*, *Random Position Offset*, *Fixed Speed*, *Fixed Speed Offset* and *Random Speed Offset*.

6.2.4 Results and analysis

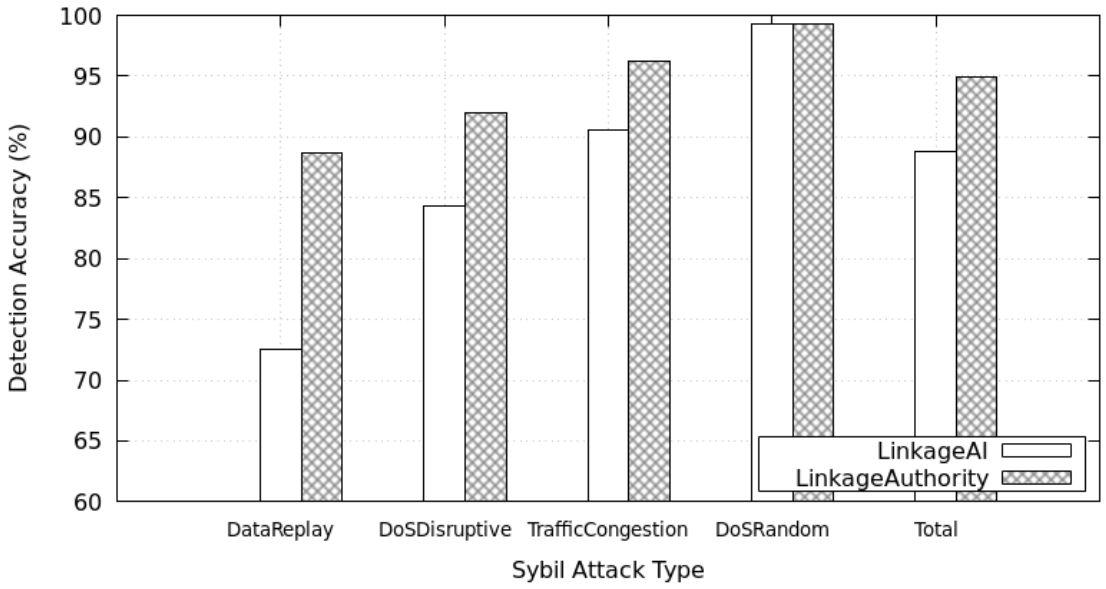


Figure 6.6: Sybil attack: detection accuracy by type of linkage

Figure 6.6 shows the results of detection accuracy of the Sybil attacks by linkage type. The detection accuracy is the ratio of the true classified reported vehicles over all the reported vehicles. The first result we notice is that the total detection for Sybil attacks types using a LA is at 94.97%, whereas it's only at 88.83% using the Linkage AI model. This is an expected result as the AI prediction is uncertain compared to the absolute information provided by the LA. We also notice that the detection accuracy difference between the two linkage types is proportional to

the general detection accuracy for each type of Sybil attack. This is due to the prediction output of the first phase. Attacks that are difficult to classify, are less likely to be linked by the AI Linkage. Especially since the classification output of the first phase is used as an input feature for the AI model. This problem however is not present using the LA.

Table 6.9: AI Linkage Evaluation Results

$Precision$	$Recall$	$Accuracy$	$Fallout$	$Specificity$	F_1Score
95,6%	89,6%	96,4%	1,3%	98.7%	92.5%

Table 6.9 shows the Evaluation Results of the AI Linkage Mechanism. The evaluation metrics are detailed in our previous publication [14]. As this system is replacing the LA, a high confidence in a perceived linkage is needed before it is considered. This shows clearly in the results as the $Precision$ is significantly higher than the $Recall$. Consequently, the lower $Recall$ (with respect to the LA) results in the lower detection accuracy perceived in Figure 6.6.

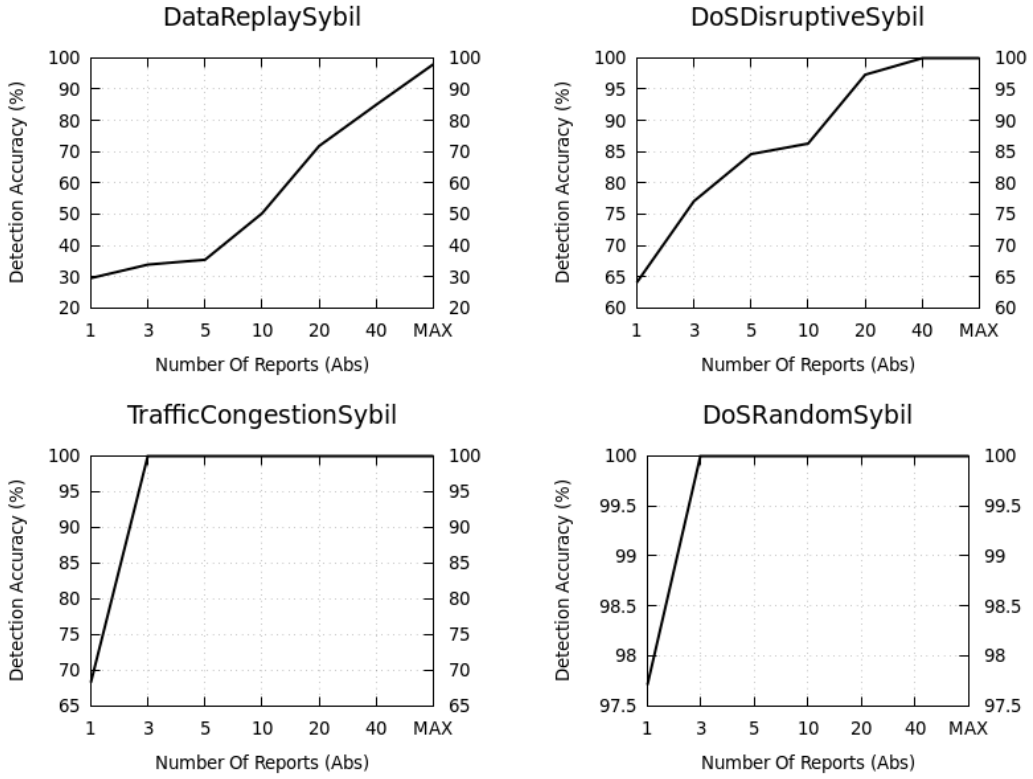


Figure 6.7: Sybil attack: detection accuracy by number of received reports

Figure 6.7 shows the detection accuracy of the attacks by the number of the received reports. In other words, it shows the number of reports needed for an accurate detection.

First, we notice that the detection accuracy for the *Data Replay Sybil* and *Dos Disruptive Sybil* attacks require more reports to converge than for the *Traffic Congestion Sybil* and *Dos Random Sybil*. The reasoning for that is both

the former attacks cause the local vehicles to simultaneously report other genuine vehicles alongside the attacker. These false positive reports add a significant amount of noise to the data. Therefore, more data is required to sort the genuine pseudonyms from the attacker pseudonyms. We also notice that the *Data Replay Sybil* attack requires more information than the *Dos Disruptive Sybil* to converge. This is a consequence of the former intelligently generating a realistic path instead of just replaying data incoherently.

Additionally, we notice that *Traffic Congestion Sybil* has a relatively low detection rate with one report. However, even though the attacker tries to intelligently remain within the plausible range, the detection then quickly converges. This is due to the lack of the simultaneously falsely reported genuine vehicles. The information is clean from false positives thus multiple reports are analyzed much more efficiently.

Finally, the *Dos Random Sybil* attack does not cause false positives neither is it within the plausible ranges. As a result, it is easily detected even with evidence from only one report.

6.2.5 Conclusion

Sybil attacks are a dangerous threat that can significantly deteriorate the C–ITS system quality and lead to catastrophic road accidents. In this work, we propose a global misbehavior detection mechanism for C–ITS. More precisely, we proposed an MA architecture specifically robust against Sybil attacks. This is achieved using machine learning analysis on the Misbehavior Report and pseudonym linking on the global level. We propose an implicit Machine Learning (ML) based linking or direct linking using the IEEE Linkage Authority. We show through extensive simulations that overall detection rate for various types of Sybil attacks is relatively high.

Chapter 7

Conclusion

7.1 Summary

In this thesis, we start by studying the current state of misbehavior detection in the ITS system. This study shows how misbehavior detection integrates into the C–ITS model. In chapter 2, we explore the related works already published on this subject. We then extract the detection mechanisms used in these works and we evaluate their feasibly with respect to the current standardized C–ITS.

In chapter 3, we describe our simulation framework for misbehavior detection in vehicular networks. A framework that includes a V2X communication network along with pre-implemented attacks as well as different detection mechanisms and evaluation metrics. Our goal with this framework is to enable the research community to easily develop, test, and compare detection algorithms.

In chapter 4, we focus on local misbehavior detection. First, we propose to integrate the confidence range of sensors in the detection checks calculation. The confidence range field is included in the standard V2X messages. Our results show that taking into account this field enabled a more reliable and fine-tuned detection of implausibility and inconsistencies on the V2X network. Then, we show a comparison of different local detection fusion applications. We extract mechanisms used in the literature and re-implement them over the same simulation scenario. These mechanisms include machine learning, cooperative and trust based mechanisms. The results include the detection quality and the processing time of each detection mechanism. Finally, as our last contribution on local misbehavior detection, we propose some improvements to VeReMi, a popular dataset for misbehavior detection. We update the dataset with a realistic sensor error model and implement a new set of mechanisms of misbehavior. We also provide different scenarios on different networks with various configurations. We publish this dataset in open source format for the research community.

In chapter 5, target the misbehavior reporting process. First, we propose a misbehavior reporting protocol. We define the information to be added in the misbehavior report. This information ensures that the MA is able

to verify the sender and the reported identities. The MA is also able to verify to a certain extent the occurring misbehavior through included evidence. We also propose a reporting protocol that reduces the network overhead. Then, we create a new misbehavior REports DAtaset called DARE. This dataset includes reports in the format and the protocol that we previously created. It includes different detection and attacker configurations as well as various network scenarios. We publish this dataset to encourage further research on global misbehavior detection.

In chapter 6, we research the global misbehavior detection. We start by evaluating the use of machine learning in the misbehavior authority detection process. We define different features needed in the detection process. We then test various machine learning and deep learning algorithms. Finally, we propose a deep learning model adapted for the use case of global misbehavior detection at the level of the misbehavior authority. Last, we improve the detection system of the misbehavior authority against Sybil attacks. We define a three steps detection process where we enable the linkage of different pseudonyms as well as the detection of specific types of Sybil attacks.

7.2 Perspectives

In this thesis, we explored multiple solutions in the various links of the misbehavior detection chain: local detection, misbehavior reporting and global detection. In our final analysis, we discovered multiple gaps that could be addressed in the future.

First, we think that current works focus mostly on the general aspects of the V2X safety messages like the position or event forging. Instead, studies should focus on safety application specific misbehavior detection. This misbehavior could have more precise attacker intentions with direct implications. This type of detection could be more useful if deployed alongside applications to ensure their correct functionality. This type of misbehavior detection was more difficult to achieve in the past since V2X applications were not yet well defined. However, with the current ongoing work on the specifications of various safety applications, we are seeing new studies that target specific applications. For instance, Allig et al. explore misbehavior within Collective Perception in [126] and Van der Heijden et al. and Iorio et al. within the context of Cooperative Adaptive Cruise Control (CACC) in [127] [128].

Another aspect of misbehavior detection that requires more study is the Misbehavior Authority. The MA detection process is not based on the V2X messages but on the received misbehavior reports. This gives the MA a limited view on the reality depending of the quantity and quality of the received reports. The MA then has to analyze the situation using this limited view and issue the correct reaction to protect the system. We think this is a complicated issue and more studies and resources should be devoted to finding specific detection and reaction mechanisms to adequately protect the C–ITS system.

Finally, the misbehavior reaction is the most ill-defined link in the misbehavior detection chain. The only reaction mechanism that is currently discussed is the revocation of a misbehaving user certificates. However, a vehicle that detects a misbehavior in its vicinity, after reporting it to the MA, should be able to take measures to protect

its safety mechanisms. A real time immediate reaction against certain known attacks should be defined. We think that a coherent misbehavior reaction strategy across local and global systems, in real-time and for the long-term, is needed for effective C–ITS attack mitigation system.

7.3 Lessons learned

During the course of this research, I gained a great deal of knowledge about a plethora of new domains. In this section, I share the three lessons that I think could benefit the scientific community the most.

The first lesson that I learned is about the scientific contributions in the midst of a moving technological medium. I come from an engineering background, so when I started researching the ITS domain, I immediately started reading the most up to date standards and specifications. I envisioned any solution created should fit within the confines of the ITS spec sheets. However, after participating in the process of creating technical requirements and directives, I now see things differently. Scientific contributions are not chained to the current technical specifications; they should be one step ahead. These contributions are driving force behind the new standards.

The second lesson is about the use of machine learning technologies. Before I started this thesis, I had already been exposed to artificial intelligence mechanisms during various previous projects. During my research into the related works, I found a number of studies applying machine learning to the misbehavior detection problem. This lead me to further develop my machine learning skills and to apply it to various parts of the misbehavior detection process. However, I also discovered that much of the related studies into machine learning, especially in misbehavior detection, suffered from a common issue. The detection algorithms resulting from the machine learning training were not always adequately evaluated. For instance, while using a strongly time-dependent data like the beacon messages, one should not use a random split for the train and test data. A random split in this case would lead to over-fitting. This is a common issue for studies using the VeReMi dataset. The lesson here is that it is important to be prudent while using a complicated technology, I personally was lucky enough to receive help in my studies form a machine learning expert.

Finally, the third lesson is specific to the cyber-security domain. While reviewing the related works I discovered that the proposed misbehavior attacks were somewhat weak and hasn't been updated in a long time. This is due to the fact that the C–ITS solutions are not yet deployed and therefore the attacks were mostly created by researchers. I too dedicated a strong effort into coming up with new types of attacks in order to test my detection mechanisms. However, up until the deployment of this system, we cannot predict which attacks or even classes of attacks we missed. However, I noticed during my work that often the most valuable insights into new attack mechanisms comes from new researchers into the domain or from complete outsiders. Therefore, in order to minimize our chances of missing preventable attacks, I would recommend dedicating white hat researchers into finding new and optimized C–ITS attacks. Specifically, attacks targeting the implementations of C–ITS safety applications.

Chapter 8

List of Publications

Journal articles:

- Joseph Kamel, Mohammad Raashid Ansari, Jonathan Petit, Arnaud Kaiser, Ines Ben Jemaa, and Pascal Urien. Simulation framework for misbehavior detection in vehicular networks. *IEEE Transactions on Vehicular Technology*, 69(6):6631–6643, 2020

Conference papers:

- Joseph Kamel, Arnaud Kaiser, Ines Ben Jemaa, Pierpaolo Cincilla, and Pascal URIEN. Feasibility Study of Misbehavior Detection Mechanisms in Cooperative Intelligent Transport Systems (C-ITS). In *2018 IEEE 87th Vehicular Technology Conference: VTC2018-Spring*, Porto, Portugal, June 2018

- Joseph Kamel, Arnaud Kaiser, Ines Ben Jemaa, Pierpaolo Cincilla, and Pascal Urien. CaTch: a confidence range tolerant misbehavior detection approach. In *2019 IEEE Wireless Communications and Networking Conference (WCNC) (IEEE WCNC 2019)*, Marrakech, Morocco, April 2019

- Joseph Kamel, Ines Ben Jemaa, Arnaud Kaiser, Loic Cantat, and Pascal Urien. Misbehavior detection in C-ITS: a comparative approach of local detection mechanisms. In *2019 IEEE Vehicular Networking Conference (VNC) (IEEE VNC 2019)*, Los Angeles, USA, December 2019

- Joseph Kamel, Michael Wolf, Rens Wouter van der Heijden, Arnaud Kaiser, Pascal Urien, and Frank Kargl. VeReMi extension: A dataset for comparable evaluation of misbehavior detection in VANETs. In *2020 IEEE International Conference on Communications (ICC): SAC Internet of Things Track (IEEE ICC'20 - SAC-06 IoT Track)*, Dublin, Ireland, June 2020

- Joseph Kamel, Ines Ben Jemaa, Arnaud Kaiser, and Pascal Urien. Misbehavior Reporting Protocol for C-ITS. In *2018 IEEE Vehicular Networking Conference (VNC)*, Taipei, Taiwan, December 2018

- Issam Mahmoudi, Joseph Kamel, Ines Ben-Jemaa, Arnaud Kaiser, and Pascal Urien. Towards a Reliable Machine Learning Based Global Misbehavior Detection in C-ITS: Model Evaluation Approach. In *Vehicular Ad-hoc Networks for Smart Cities, Springer Singapore*, pages 73–86, Singapore, 2020
- Joseph Kamel, Farah Haidar, Ines Ben Jemaa, Arnaud Kaiser, Brigitte Lonc, and Pascal Urien. A Misbehavior Authority System for Sybil Attack Detection in C-ITS. In *The IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference – IEEE UEMCON 2019*, New York, United States, October 2019
- Farah Haidar, Joseph Kamel, Ines Ben Jemaa, Arnaud Kaiser, Brigitte Lonc, and Pascal Urien. DARE: A Reports Dataset for Global Misbehavior Authority Evaluation in C-ITS. In *The 2020 IEEE 91st Vehicular Technology Conference – IEEE VTC2020-Spring*, Antwerp, Belgium, May 2020

Chapter 9

Annex

9.1 Annex 1: Misbehavior Report in ASN.1 Format

```
Its-Report DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

IMPORTS
TimestampIts, StationType, ReferencePosition, Heading, Speed,
DriveDirection, VehicleLength, VehicleWidth, Curvature,
LongitudinalAcceleration, CurvatureCalculationMode,
YawRate, PerceivedObjectContainer,
SensorInformationContainer FROM ITS-Container {
    itu-t (0) identified-organization (4) etsi (0) itsDomain (5)
    wg1 (1) ts (102894) cdd (2) version (1)
}
EtsiTs103097Data, EtsiTs103097Certificate FROM EtsiTs103097Module {
    itu-t(0) identified-organization(4) etsi(0) itsDomain(5)
    wg5(5) ts(103097) v1(0)
}

Version FROM EtsiTs102941BaseTypes {
    itu-t(0) identified-organization(4) etsi(0) itsDomain(5)
    wg5(5) ts(102941) baseTypes(3) version2(2)
};

-- The root data frame for report messages
MisbehaviorReport::= SEQUENCE {
    version Version,
    reportMetadataContainer ReportMetadataContainer,
    reportContainer ReportContainer
}

ReportMetadataContainer::= SEQUENCE {
    reportID IA5String,
    generationTime TimestampIts,
    relatedReportContainer RelatedReportContainer OPTIONAL
}

RelatedReportContainer::= SEQUENCE {
    relatedReportID IA5String,
    omitedReportsNumber OmitedReportsNumber
}
```

```
ReportContainer::= SEQUENCE {
    reportedMessageContainer ReportedMessageContainer,
    misbehaviorTypeContainer MisbehaviorTypeContainer,
    evidenceContainer EvidenceContainer OPTIONAL
}

ReportedMessageContainer::= CHOICE {
    certificateIncludedContainer CertificateIncludedContainer,
    certificateAddedContainer CertificateAddedContainer
}

CertificateIncludedContainer::= SEQUENCE{
    reportedMessage EtsiTs103097Data
}

CertificateAddedContainer::= SEQUENCE{
    reportedMessage EtsiTs103097Data,
    reportedCertificate EtsiTs103097Certificate
}

MisbehaviorTypeContainer::= CHOICE {
    securityDetection SecurityDetection,
    semanticDetection SemanticDetection
    ...
}

SecurityDetection::= SEQUENCE {
    securityDetectionErrorCode OCTET STRING (SIZE (0..4)),
    ...
}

SemanticDetection::= CHOICE {
    semanticDetectionReferenceCAM DetectionReferenceCAM,
    semanticDetectionReferenceDENM DetectionReferenceDENM,
    semanticDetectionReferenceCPM DetectionReferenceCPM,
    semanticDetectionReferenceSPAT DetectionReferenceSPAT,
    semanticDetectionReferenceMAP DetectionReferenceMAP,
    ...
}

DetectionReferenceCAM::= SEQUENCE{
    detectionLevelCAM DetectionLevel,
    semanticDetectionErrorCodeCAM OCTET STRING (SIZE (0..2))
}

EvidenceContainer::= SEQUENCE {
    reportedMessageContainer MessageEvidenceContainer OPTIONAL,
    neighbourMessageContainer MessageEvidenceContainer OPTIONAL,
    senderInfoContainer SenderInfoContainer OPTIONAL,
    senderSensorContainer SenderSensorContainer OPTIONAL
}

MessageEvidenceContainer::= SEQUENCE OF EtsiTs103097Data

SenderInfoContainer::= SEQUENCE {
    stationType StationType,
    referencePosition ReferencePosition,
    heading Heading,
    speed Speed,
    driveDirection DriveDirection,
    vehicleLength VehicleLength,
    vehicleWidth VehicleWidth,
    longitudinalAcceleration LongitudinalAcceleration,
    curvature Curvature,
    yawRate YawRate
    ...
}
```

```
SenderSensorContainer::= SEQUENCE OF SenderSensorChoice

SenderSensorChoice::= CHOICE{
    fieldofViewContainer FieldofViewContainer,
    perceivedObjectContainer PerceivedObjectContainer
}

DetectionLevel ::= INTEGER { level(1) } (1..4)
OmitedReportsNumber ::= INTEGER { oneReport(1) } (0..1024)

END
```

9.2 Annex 2: DARE dataset specifications by scenario

Id	Scenario		Attacker			Genuine		Reports		File Size	
	Net	*Time*	*Rate*	*Type*	*Vehicles*	*Messages*	*Vehicles*	*Messages*	*Initial*	*Follow-up*	*Plain*	*Zipped*
Lust-0024-25S	Lust	0h-24h	25%	Shuffle-All	20,318	109,823,057	61,607	223,214,564	8,862,467	19,972,105	167.1GBs	14.41GBs
Lust-0611-25M	Lust	6h-11h	25%	Malfunctions	6,556	27,677,626	19,845	82,911,912	566,002	2,482,071	29.71GBs	2.983GBs
Lust-0611-25A	Lust	6h-11h	25%	Attacks	6,560	52,930,127	19,845	85,646,078	5,828,229	11,339,048	91.24GBs	7.537GBs
Lust-0611-15M	Lust	6h-11h	15%	Malfunctions	3,937	16,815,969	22,491	93,773,569	391,395	1,714,996	20.69GBs	2.072GBs
Lust-0611-15A	Lust	6h-11h	15%	Attacks	3,933	32,977,691	22,491	95,319,109	3,957,115	7,802,816	64.14GBs	5.368GBs
Lust-0611-05M	Lust	6h-11h	5%	Malfunctions	1,314	5,640,072	25,137	104,949,466	154,812	661,708	7.849GBs	0.795GBs
Lust-0611-05A	Lust	6h-11h	5%	Attacks	1,307	11,231,920	25,137	105,539,193	1,424,943	2,773,928	23.41GBs	1.973GBs
Lust-1115-25M	Lust	11h-15h	25%	Malfunctions	3,757	9,311,363	11,353	27,619,797	201,393	859,738	10.13GBs	1.006GBs
Lust-1115-25A	Lust	11h-15h	25%	Attacks	3,748	18,298,605	11,353	28,558,086	2,049,910	3,966,993	30.673GBs	2.471GBs
Lust-1115-15M	Lust	11h-15h	15%	Malfunctions	2,255	5,547,090	1,760	31,384,070	138,351	582,879	6.885GBs	0.679GBs
Lust-1115-15A	Lust	11h-15h	15%	Attacks	2,250	10,883,990	12,867	31,936,083	,1325,323	2,561,363	20.23GBs	1.649GBs
Lust-1115-05M	Lust	11h-15h	5%	Malfunctions	751	1,788,546	14,381	35,142,614	52,958	215,489	2.562GBs	0.247GBs
Lust-1115-05A	Lust	11h-15h	5%	Attacks	751	3,548,232	14,381	35,333,220	420,467	854,075	6.966GBs	0.581GBs
Lust-1521-25M	Lust	15h-21h	25%	Malfunctions	7,776	34,550,797	23,475	104,635,169	685,534	3,144,205	37.35GBs	3.773GBs
Lust-1521-25A	Lust	15h-21h	25%	Attacks	7,766	68,001,064	23,482	108,001,498	7,132,715	15,683,151	117.4GBs	9.486GBs
Lust-1521-15M	Lust	15h-21h	15%	Malfunctions	4,651	20,918,031	26,605	118,267,935	472,852	2,164,576	25.65GBs	2.577GBs
Lust-1521-15A	Lust	15h-21h	15%	Attacks	4,661	40,517,058	26,612	120,265,804	4,487,167	9,837,161	79.67GBs	6.669GBs
Lust-1521-05M	Lust	15h-21h	5%	Malfunctions	1,549	6,930,299	29,735	132,255,667	183,613	820,235	9.844GBs	0.969GBs
Lust-1521-05A	Lust	15h-21h	5%	Attacks	1,550	13,183,858	29,735	132,950,041	1,550,561	3,517,203	28.72GBs	2.403GBs
Paris-0024-05S	Paris	0h-24h	5%	Shuffle-All	619	635,311	11,914	7,830,985	60,788	123,413	1.031GBs	0.091GBs

Bibliography

[1] Directorate-General for Mobility and Transport (DGMT). Annual Accident Report 2018. *European Commission (EC)*, pages 1–85, December 2018.

[2] National Highway Traffic Safety Administration (NHTSA). Summary of Motor Vehicle Crashes. *Department of Transportation (DOT)*, pages 1–8, September 2018.

[3] Samantha H. Haus, Rini Sherony, and Hampton C. Gabler. Estimated benefit of automated emergency braking systems for vehicle–pedestrian crashes in the united states. *Traffic Injury Prevention*, 20(sup1):S171–S176, 2019. doi: 10.1080/15389588.2019.1602729.

[4] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental security analysis of a modern automobile. In *2010 IEEE Symposium on Security and Privacy*, pages 447–462, May 2010. doi: 10.1109/SP.2010.34.

[5] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, page 6, USA, 2011. USENIX Association.

[6] Charlie Miller and Chris Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015:91, 2015.

[7] European Telecommunications Standards Institute (ETSI). ETSI EN 302 637-2 V1.4.0: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service. *ETSI WG5 Technical Specification*, pages 1–45, August 2018.

[8] European Telecommunications Standards Institute (ETSI). ETSI EN 302 637-3 V1.3.1: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service. *ETSI WG5 Technical Specification*, pages 1–74, April 2019.

[9] Society of Automobile Engineers (SAE). Dedicated short range communications (DSRC) message set dictionary™, sae j2735. *SAE International*, 2016.

[10] IEEE. IEEE standard for wireless access in vehicular environments–security services for applications and management messages - amendment 1. *IEEE Std 1609.2a-2017 (Amendment to IEEE Std 1609.2-2016)*, pages 1–123, Oct 2017. doi: 10.1109/IEEESTD.2017.8065169.

[11] European Telecommunications Standards Institute (ETSI). ETSI TS 102 940 v1.3.1: Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management. *ITS-WG5*, pages 1–42, April 2018.

[12] Jonathan Petit, Florian Schaub, Michael Feiri, and Frank Kargl. Pseudonym schemes in vehicular networks: A survey. *IEEE Communications Surveys Tutorials*, 17(1):228–255, Firstquarter 2015. ISSN 1553-877X. doi: 10.1109/COMST.2014.2345420.

[13] Joseph Kamel, Arnaud Kaiser, Ines Ben Jemaa, Pierpaolo Cincilla, and Pascal URIEN. Feasibility Study of Misbehavior Detection Mechanisms in Cooperative Intelligent Transport Systems (C-ITS). In *2018 IEEE 87th Vehicular Technology Conference: VTC2018-Spring*, Porto, Portugal, June 2018.

[14] Joseph Kamel, Arnaud Kaiser, Ines Ben Jemaa, Pierpaolo Cincilla, and Pascal Urien. CaTch: a confidence range tolerant misbehavior detection approach. In *2019 IEEE Wireless Communications and Networking Conference (WCNC) (IEEE WCNC 2019)*, Marrakech, Morocco, April 2019.

[15] Joseph Kamel, Ines Ben Jemaa, Arnaud Kaiser, Loic Cantat, and Pascal Urien. Misbehavior detection in C-ITS: a comparative approach of local detection mechanisms. In *2019 IEEE Vehicular Networking Conference (VNC) (IEEE VNC 2019)*, Los Angeles, USA, December 2019.

[16] Framework For Misbehavior Detection (F^2MD). F^2MD website, 2010. URL `https://www.irt-systemx.fr/f2md`.

[17] Joseph Kamel, Michael Wolf, Rens Wouter van der Heijden, Arnaud Kaiser, Pascal Urien, and Frank Kargl. VeReMi extension: A dataset for comparable evaluation of misbehavior detection in VANETs. In *2020 IEEE International Conference on Communications (ICC): SAC Internet of Things Track (IEEE ICC'20 - SAC-06 IoT Track)*, Dublin, Ireland, June 2020.

[18] Joseph Kamel, Ines Ben Jemaa, Arnaud Kaiser, and Pascal Urien. Misbehavior Reporting Protocol for C-ITS. In *2018 IEEE Vehicular Networking Conference (VNC)*, Taipei, Taiwan, December 2018.

[19] Farah Haidar, Joseph Kamel, Ines Ben Jemaa, Arnaud Kaiser, Brigitte Lonc, and Pascal Urien. DARE: A Reports Dataset for Global Misbehavior Authority Evaluation in C-ITS. In *The 2020 IEEE 91st Vehicular Technology Conference – IEEE VTC2020-Spring*, Antwerp, Belgium, May 2020.

[20] Issam Mahmoudi, Joseph Kamel, Ines Ben-Jemaa, Arnaud Kaiser, and Pascal Urien. Towards a Reliable Machine Learning Based Global Misbehavior Detection in C-ITS: Model Evaluation Approach. In *Vehicular Ad-hoc Networks for Smart Cities, Springer Singapore*, pages 73–86, Singapore, 2020.

[21] Joseph Kamel, Farah Haidar, Ines Ben Jemaa, Arnaud Kaiser, Brigitte Lonc, and Pascal Urien. A Misbehavior Authority System for Sybil Attack Detection in C-ITS. In *The IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference – IEEE UEMCON 2019*, New York, United States, October 2019.

[22] Joseph Kamel, Mohammad Raashid Ansari, Jonathan Petit, Arnaud Kaiser, Ines Ben Jemaa, and Pascal Urien. Simulation framework for misbehavior detection in vehicular networks. *IEEE Transactions on Vehicular Technology*, 69(6):6631–6643, 2020.

[23] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, Jan 2011. ISSN 1536-1233. doi: 10.1109/TMC.2010.133.

[24] András Varga. The omnet++ discrete event simulation system. In *In ESM'01*, 2001.

[25] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.

[26] Joseph Kamel. Github repository: Framework for misbehavior detection (f^{2}md), 2020. URL `https://github.com/josephkamel/f2md`.

[27] European Telecommunications Standards Institute (ETSI). ETSI TR 103 460 v0.0.10: Intelligent Transport Systems (ITS); Security; Pre-standardisation Study on Misbehavior Detection. *ITS-WG5*, pages 1–27, July 2019.

[28] IEEE. Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 6: Wireless access in vehicular environments. *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)*, pages 1–51, July 2010. ISSN null. doi: 10.1109/IEEESTD.2010.5514475.

[29] European Telecommunications Standards Institute (ETSI). ETSI EN 302 571 V2.1.1: Intelligent Transport Systems (ITS); Radiocommunications equipment operating in the 5 855 MHz to 5 925 MHz frequency band;

Harmonised Standard covering the essential requirements of article 3.2 of Directive 2014/53/EU. *ETSI WG5 Technical Specification*, pages 1–49, February 2017.

[30] IEEE. Ieee standard for wireless access in vehicular environments (wave) - identifier allocations. *IEEE Std 1609.12-2016 (Revision of IEEE Std 1609.12-2012)*, pages 1–21, March 2016. ISSN null. doi: 10.1109/IEEESTD.2016.7428792.

[31] 3GPP. Technical Specification Group Services and System Aspects; Release 14 Description; Summary of Rel-14 Work Items (Release 14) . Technical Report (TR) 21.914, 3rd Generation Partnership Project (3GPP), 05 2018. Version 14.0.0.

[32] European Telecommunications Standards Institute (ETSI). ETSI TS 101 539-1 V1.1.1: Intelligent Transport Systems (ITS); V2X Applications; Part 1: Road Hazard Signalling (RHS) application requirements specification. *ETSI WG1 Technical Specification*, pages 1–38, August 2013.

[33] European Telecommunications Standards Institute (ETSI). ETSI TS 101 539-2 V1.1.1: Intelligent Transport Systems (ITS); V2X Applications; Part 2: Intersection Collision Risk Warning (ICRW) application requirements specification. *ETSI WG1 Technical Specification*, pages 1–30, June 2018.

[34] European Telecommunications Standards Institute (ETSI). ETSI TS 101 539-3 V1.1.1: Intelligent Transport Systems (ITS); V2X Applications; Part 3: Longitudinal Collision Risk Warning (LCRW) application requirements specification . *ETSI WG1 Technical Specification*, pages 1–29, November 2013.

[35] European Telecommunications Standards Institute (ETSI). ETSI EN 302 562 V2.1.1: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications,Analysis of the Collective Perception Service (CPS), Release 2. *ETSI WG1 Technical Specification*, pages 1–119, December 2019.

[36] European Telecommunications Standards Institute (ETSI). ETSI EN 302 324 V0.0.14: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications,Specification of the Collective Perception Service. *ETSI WG1 Technical Specification*, pages 1–153, October 2019.

[37] European Telecommunications Standards Institute (ETSI). ETSI EN 302 571 V2.1.1: Intelligent Transport Systems (ITS); Cooperative Adaptive Cruise Control (CACC); Pre-standardization study . *ETSI WG1 Technical Specification*, pages 1–45, June 2019.

[38] European Telecommunications Standards Institute (ETSI). ETSI TS 103 298 V0.0.4: Intelligent Transport Systems (ITS); Platooning; Pre-standardization study. *ETSI WG1 Technical Specification*, pages 1–26, January 2019.

[39] European Telecommunications Standards Institute (ETSI). ETSI EN 302 579 V0.0.6: Intelligent Transport Systems (ITS); Pre-Standardization Study on payment applications in ETSI ITS-G5. *ETSI WG1 Technical Specification*, pages 1–47, January 2020.

[40] European Telecommunications Standards Institute (ETSI). ETSI EN 302 665 v1.1.1: Intelligent Transport Systems (ITS); Communications Architecture. *ISO TC204*, pages 1–44, September 2010.

[41] ISO. Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model. Standard, International Organization for Standardization, Geneva, CH, November 1994.

[42] European Telecommunications Standards Institute (ETSI). ETSI EN 102 636 - All Parts: Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking;. *ITS-WG3*, 2010.

[43] European Telecommunications Standards Institute (ETSI). ETSI TS 102 941 v1.2.1: Intelligent Transport Systems (ITS); Security; Trust and Privacy Management. *ITS-WG5*, pages 1–71, May 2018.

[44] European Telecommunications Standards Institute (ETSI). ETSI TS 103 097 v1.3.1: Intelligent Transport Systems (ITS); Security; Security header and certificate formats. *ITS-WG5*, pages 1–23, October 2017.

[45] Farah Haidar, Arnaud Kaiser, and Brigitte Lonc. On the performance evaluation of vehicular PKI protocol for V2X communications security. In *2017 IEEE 86th Vehicular Technology Conference (VTC2017-Fall)*, Toronto, Canada, September 2017.

[46] Farah Haidar, Arnaud Kaiser, Brigitte Lonc, and Pascal URIEN. C-ITS PKI protocol: Performance Evaluation in a Real Environment. In *WONS 2019 15 th Wireless On-demand Network systems and Services Conference*, Wengen, Switzerland, January 2019.

[47] Pierpaolo Cincilla, Omar Hicham, and Benoit Charles. Vehicular PKI scalability-consistency trade-offs in large scale distributed scenarios. In *Vehicular Networking Conference (VNC)*, Columbus, United States, October 2016. doi: $10.1109/\mathrm{VNC}.2016.7835970$.

[48] Philippe Golle, Dan Greene, and Jessica Staddon. Detecting and correcting malicious data in vanets. *Proceedings of the 1st ACM Workshop on Vehicular Ad Hoc Networks (VANET 2004), Philadelphia, PA, USA, October*, 2004.

[49] Rens Wouter van der Heijden, Stefan Dietzel, Tim Leinmüller, and Frank Kargl. Survey on misbehavior detection in cooperative intelligent transportation systems. *IEEE Communications Surveys Tutorials*, 21(1): 779–811, Firstquarter 2019. ISSN 1553-877X. doi: $10.1109/\mathrm{COMST}.2018.2873088$.

[50] John R. Douceur. The sybil attack. In *Peer-to-Peer Systems*, pages 251–260, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45748-0.

[51] Ali Akbar Pouyan and Mahdiyeh Alimohammadi. Sybil Attack Detection in Vehicular Networks. In *Computer Science and Information Technology 2.4*, pages 197 – 202, 2014. doi: 10.13189/csit.2014.020403.

[52] Chen Chen, Xin Wang, Weili Han, and Binyu Zang. A robust detection of the sybil attack in urban vanets. In *2009 29th IEEE International Conference on Distributed Computing Systems Workshops*, pages 270–276, June 2009. doi: 10.1109/ICDCSW.2009.48.

[53] Soyoung Park, Baber Aslam, Damla Turgut, and Cliff C. Zou. Defense against sybil attack in vehicular ad hoc network based on roadside unit support. In *MILCOM 2009 - 2009 IEEE Military Communications Conference*, pages 1–7, Oct 2009. doi: 10.1109/MILCOM.2009.5379844.

[54] Shan Chang, Yong Qi, Hongzi Zhu, Jizhong Zhao, and Xuemin Shen. Footprint: Detecting sybil attacks in urban vehicular networks. *IEEE Transactions on Parallel and Distributed Systems*, 23(6):1103–1114, June 2012. ISSN 2161-9883. doi: 10.1109/TPDS.2011.263.

[55] Yong Hao, Jin Tang, and Yu Cheng. Cooperative sybil attack detection for position based applications in privacy preserved vanets. In *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, pages 1–5, Dec 2011. doi: 10.1109/GLOCOM.2011.6134242.

[56] Bin Xiao, Bo Yu, and Chuanshan Gao. Detection and localization of sybil nodes in vanets. In *DIWANS 2006*, pages 1–8, 2006.

[57] Tong Zhou, Romit Roy Choudhury, Peng Ning, and Krishnendu Chakrabarty. Privacy-preserving detection of sybil attacks in vehicular ad hoc networks. In *2007 Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking Services (MobiQuitous)*, pages 1–8, Aug 2007. doi: 10.1109/MOBIQ.2007.4451013.

[58] ByungKwan Lee, Eun-Hee Jeong, and Ina Jung. A DTSA (Detection Technique against a Sybil Attack) Protocol using SKC (Session Key based Certificate) on VANET. *International Journal of Security and Its Applications*, 7:1–10, 2013.

[59] Adnan Vora and Mikhail Nesterenko. Secure location verification using radio broadcast. *IEEE Transactions on Dependable and Secure Computing*, 3(4):377–385, Oct 2006. ISSN 2160-9209. doi: 10.1109/TDSC.2006.57.

[60] Jean-Pierre Hubaux, Srdjan Capkun, and Jun Luo. The security and privacy of smart vehicles. *IEEE Security Privacy*, 2(3):49–55, May 2004. ISSN 1558-4046. doi: 10.1109/MSP.2004.26.

[61] Mingshun Sun, Ming Li, and Ryan Gerdes. A data trust framework for vanets enabling false data detection and secure vehicle tracking. In *2017 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9, Oct 2017. doi: 10.1109/CNS.2017.8228654.

[62] Robert K Schmidt, Tim Leinmüller, Elmar Schoch, Albert Held, and Günter Schäfer. Vehicle behavior analysis to enhance security in vanets. In *V2VCOM 2008*, pages 1–8, 2008.

[63] Norbert Bißmeyer, Christian Stresing, and Kpatcha M. Bayarou. Intrusion detection in vanets through verification of vehicle movement data. In *2010 IEEE Vehicular Networking Conference*, pages 166–173, Dec 2010. doi: 10.1109/VNC.2010.5698232.

[64] Tim Leinmüller, Elmar Schoch, Frank Kargl, and Christian Maihöfer. Decentralized position verification in geographic ad hoc routing. *Security and communication networks*, 3(4):289–302, 7 2010. ISSN 1939-0114. doi: 10.1002/sec.56. 10.1002/sec.56.

[65] Rens W. van der Heijden, Ala'a Al-Momani, Frank Kargl, and Osama M. F. Abu-Sharkh. Enhanced position verification for vanets using subjective logic. In *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, pages 1–7, Sep. 2016. doi: 10.1109/VTCFall.2016.7881000.

[66] Audun Jøsang. A logic for uncertain probabilities. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 9(3): 279–311, June 2001. ISSN 0218-4885. doi: 10.1142/S0218488501000831. URL https://doi.org/10.1142/S0218488501000831.

[67] Kamran Zaidi, Milos B. Milojevic, Veselin Rakocevic, Arumugam Nallanathan, and Muttukrishnan Rajarajan. Host-based intrusion detection for vanets: A statistical approach to rogue node detection. *IEEE Transactions on Vehicular Technology*, 65(8):6703–6714, Aug 2016. ISSN 1939-9359. doi: 10.1109/TVT.2015.2480244.

[68] Sushmita Ruj, Marcos Antonio Cavenaghi, Zhen Huang, Amiya Nayak, and Ivan Stojmenovic. On data-centric misbehavior detection in vanets. In *2011 IEEE Vehicular Technology Conference (VTC Fall)*, pages 1–5, Sep. 2011. doi: 10.1109/VETECF.2011.6093096.

[69] Mainak Ghosh, Anitha Varghese, Arobinda Gupta, Arzad A Kherani, and Skanda N Muthaiah. Detecting misbehaviors in vanet with integrated root-cause analysis. *Ad Hoc Networks*, 8(7):778–790, 2010.

[70] Zhen Cao, Jiejun Kong, Uichin Lee, M. Gerla, and Zhong Chen. Proof-of-relevance: Filtering false data via authentic consensus in vehicle ad-hoc networks. In *IEEE INFOCOM Workshops 2008*, pages 1–6, April 2008. doi: 10.1109/INFOCOM.2008.4544650.

[71] Abhinav Kamra, Vishal Misra, Jon Feldman, and Dan Rubenstein. Growth codes: Maximizing sensor network data persistence. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, page 255–266, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933085. doi: 10.1145/1159913.1159943. URL https://doi.org/10.1145/1159913.1159943.

[72] Hsu-Chun Hsiao, Ahren Studer, Rituik Dubey, Elaine Shi, and Adrian Perrig. Efficient and secure threshold-based event validation for vanets. In *Proceedings of the Fourth ACM Conference on Wireless Network Security*, WiSec '11, page 163–174, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306928. doi: 10.1145/1998412.1998440. URL https://doi.org/10.1145/1998412.1998440.

[73] Tiffany Hyun-Jin Kim, Ahren Studer, Rituik Dubey, Xin Zhang, Adrian Perrig, Fan Bai, Bhargav Bellur, and Aravind Iyer. Vanet alert endorsement using multi-source filters. In *Proceedings of the Seventh ACM International Workshop on VehiculAr InterNETworking*, VANET '10, page 51–60, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450301459. doi: 10.1145/1860058.1860067. URL https://doi.org/10.1145/1860058.1860067.

[74] Maxim Raya, Panagiotis Papadimitratos, Imad Aad, Daniel Jungels, and Jean pierre Hubaux. Eviction of misbehaving and faulty nodes in vehicular networks. *IEEE Journal on Selected Areas in Communications*, 25 (8):1557–1568, Oct 2007. ISSN 1558-0008. doi: 10.1109/JSAC.2007.071006.

[75] Xuejun Zhuo, Jianguo Hao, Duo Liu, and Yiqi Dai. Removal of misbehaving insiders in anonymous vanets. In *Proceedings of the 12th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '09, page 106–115, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605586168. doi: 10.1145/1641804.1641824. URL https://doi.org/10.1145/1641804.1641824.

[76] Tim Leinmüller, Robert K. Schmidt, and Albert Held. Cooperative position verification - defending against roadside attackers 2.0. In *Proceedings of 17th ITS World Congress*, 2010.

[77] Chaker Abdelaziz Kerrache, Nasreddine Lagraa, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni. T-vnets: A novel trust architecture for vehicular networks using the standardized messaging services of etsi its. *Computer Communications*, 93:68 – 83, 2016. ISSN 0140-3664. doi: https://doi.org/10.1016/j.comcom.2016.05.013. URL http://www.sciencedirect.com/science/article/pii/S0140366416302213. Multi-radio, Multi-technology, Multi-system Vehicular Communications.

[78] Maxim Raya, Panos Papadimitratos, Virgil D. Gligor, and Jean-Pierre Hubaux. On data-centric trust establishment in ephemeral ad hoc networks. In *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, pages 1238–1246, April 2008. doi: 10.1109/INFOCOM.2008.180.

[79] Attila Jaeger, Norbert Bißmeyer, Hagen Stübing, and Sorin A. Huss. A novel framework for efficient mobility data verification in vehicular ad-hoc networks. *International Journal of Intelligent Transportation Systems Research*, 10(1):11–21, Jan 2012.

[80] Qiang Xu, Rong Zheng, Walid Saad, and Zhu Han. Device fingerprinting in wireless networks: Challenges and opportunities. *IEEE Communications Surveys Tutorials*, 18(1):94–104, Firstquarter 2016. ISSN 2373-745X. doi: 10.1109/COMST.2015.2476338.

[81] Rens W. van der Heijden, Thomas Lukaseder, and Frank Kargl. Veremi: A dataset for comparable evaluation of misbehavior detection in vanets. In Raheem Beyah, Bing Chang, Yingjiu Li, and Sencun Zhu, editors, *Security and Privacy in Communication Networks*, pages 318–337, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01701-9.

[82] Jyoti Grover, Nitesh Kumar Prajapati, Vijay Laxmi, and Manoj Singh Gaur. Machine learning approach for multiple misbehavior detection in vanet. In Ajith Abraham, Jaime Lloret Mauri, John F. Buford, Junichi Suzuki, and Sabu M. Thampi, editors, *Advances in Computing and Communications*, pages 644–653, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-22720-2.

[83] Jyoti Grover, Vijay Laxmi, and Manoj Singh Gaur. Misbehavior detection based on ensemble learning in vanet. In P. Santhi Thilagam, Alwyn Roshan Pais, K. Chandrasekaran, and N. Balakrishnan, editors, *Advanced Computing, Networking and Security*, pages 602–611, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-29280-4.

[84] Fuad A. Ghaleb, Anazida Zainal, Murad A. Rassam, and Fathey Mohammed. An effective misbehavior detection model using artificial neural network for vehicular ad hoc network applications. In *2017 IEEE Conference on Application, Information and Network Security (AINS)*, pages 13–18, Nov 2017. doi: 10.1109/AINS.2017.8270417.

[85] Jyoti Grover, Manoj Singh Gaur, and Vijay Laxmi. Position forging attacks in vehicular ad hoc networks: Implementation, impact and detection. In *2011 7th International Wireless Communications and Mobile Computing Conference*, pages 701–706, July 2011. doi: 10.1109/IWCMC.2011.5982632.

[86] Federal Highway Administration (FHWA). Next Generation Simulation (NGSIM) Vehicle Trajctories Dataset, 2006. URL `https://catalog.data.gov/dataset/next-generation-simulation-ngsim-vehicle-trajectories`.

[87] Steven So, Prinkle Sharma, and Jonathan Petit. Integrating plausibility checks and machine learning for misbehavior detection in vanet. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 564–571, Dec 2018. doi: 10.1109/ICMLA.2018.00091.

[88] Steven So, Jonathan Petit, and David 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, WiSec '19, page 84–93, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367264. doi: 10.1145/3317549.3323406. URL https://doi.org/10.1145/3317549.3323406.

[89] Pranav Kumar Singh, Shivam Gupta, Ritveeka Vashistha, Sunit Kumar Nandi, and Sukumar Nandi. Machine learning based approach to detect position falsification attack in vanets. In Sukumar Nandi, Devesh Jinwala, Virendra Singh, Vijay Laxmi, Manoj Singh Gaur, and Parvez Faruki, editors, *Security and Privacy*, pages 166–178, Singapore, 2019. Springer Singapore. ISBN 978-981-13-7561-3.

[90] Pranav Kumar Singh, Manish Kumar Dash, Paritosh Mittal, Sunit Kumar Nandi, and Sukumar Nandi. Misbehavior detection in c-its using deep learning approach. In Ajith Abraham, Aswani Kumar Cherukuri, Patricia Melin, and Niketa Gandhi, editors, *Intelligent Systems Design and Applications*, pages 641–652, Cham, 2020. Springer International Publishing. ISBN 978-3-030-16657-1.

[91] Chunhua Zhang, Kangqiang Chen, Xin Zeng, and Xiaoping Xue. Misbehavior detection based on support vector machine and dempster-shafer theory of evidence in vanets. *IEEE Access*, 6:59860–59870, 2018. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2875678.

[92] Sohan Gyawali and Yi Qian. Misbehavior detection using machine learning in vehicular communication networks. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2019. doi: 10.1109/ICC.2019.8761300.

[93] European Parliament. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union*, L119:1–88, May 2016.

[94] William Whyte, André Weimerskirch, Virendra Kumar, and Thorsten Hehn. A security credential management system for v2v communications. In *2013 IEEE Vehicular Networking Conference*, pages 1–8, Dec 2013. doi: 10.1109/VNC.2013.6737583.

[95] European Telecommunications Standards Institute (ETSI). ETSI TR 103 415 V1.1.1: Intelligent Transport Systems (ITS); Security; Pre-standardisation study on pseudonym change management . *ETSI WG5 Technical Specification*, pages 1–32, April 2018.

[96] Lara Codeca, Raphael Frank, and Thomas Engel. Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research. In *2015 IEEE Vehicular Networking Conference (VNC)*, pages 1–8, Dec 2015. doi: 10.1109/VNC.2015.7385539.

[97] Jonathan Petit and Raashid Ansari. V2X Validation Tool. https://bitbucket.org/onboardsecurity/dsrcvt, BlackHat 2018.

[98] Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35, 1960. doi: 10.1115/1.3662552.

[99] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[100] François Chollet et al. Keras. `https://keras.io`, 2015.

[101] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[102] Gael Varoquaux. joblib documentation. `https://media.readthedocs.org/pdf/joblib/latest/joblib.pdf`, 2018.

[103] Travis E. Oliphant. A guide to numpy. `http://www.numpy.org/`, 2006.

[104] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3), May 2011. ISSN 2157-6904. doi: 10.1145/1961189.1961199. URL `https://doi.org/10.1145/1961189.1961199`.

[105] CAR 2 CAR. C2C-CC Basic System Profile V1.1.0. *Communication Consortium*, pages 1–69, December 2015.

[106] Axel Wegener, MichałPiórkowski, Maxim Raya, Horst Hellbrück, Stefan Fischer, and Jean-Pierre Hubaux. Traci: An interface for coupling road traffic and network simulators. In *Proceedings of the 11th Communications and Networking Simulation Symposium*, CNS '08, pages 155–163, New York, NY, USA, 2008. ACM. ISBN 1-56555-318-7.

[107] Society of Automobile Engineers (SAE). On-board system requirements for v2v safety communications, sae j2945/1. *SAE International*, 2016.

[108] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92,

pages 144–152, New York, NY, USA, 1992. ACM. ISBN 0-89791-497-X. doi: 10.1145/130385.130401. URL http://doi.acm.org/10.1145/130385.130401.

[109] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL http://doi.acm.org/10.1145/2939672.2939785.

[110] Christoph Van Der Malsburg. Frank rosenblatt: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. In Günther Palm and Ad Aertsen, editors, *Brain Theory*, pages 245–248, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg. ISBN 978-3-642-70911-1.

[111] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[112] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 2008.

[113] James A. DiLellio, Edward Carolipio, Jya-Syin Wu Chien, and Kamran Ghassaei. Global positioning system accuracy enhancement, 2011.

[114] Kazuyuki Kobayashi, Ka C. Cheok, and Kazuji Watanabe. Estimation of absolute vehicle speed using fuzzy logic rule-based kalman filter. In *Proceedings of 1995 American Control Conference - ACC'95*, volume 5, pages 3086–3090 vol.5, June 1995. doi: 10.1109/ACC.1995.532084.

[115] Michael Hölzl, Roland Neumeier, and Gerald Ostermayer. Analysis of compass sensor accuracy on several mobile devices in an industrial environment. In Roberto Moreno-Díaz, Franz Pichler, and Alexis Quesada-Arencibia, editors, *Computer Aided Systems Theory - EUROCAST 2013*, pages 381–389, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-53862-9.

[116] Chirdpong Deelertpaiboon and Manukid Parnichkun. Fusion of gps, compass, and camera for localization of an intelligent vehicle. *International Journal of Advanced Robotic Systems*, 5(4):46, 2008. doi: 10.5772/6231.

[117] Joseph Kamel. Veremi extension: A dataset for comparable evaluation of misbehavior detection in vanets, 2020. URL https://github.com/josephkamel/VeReMi-Dataset.

[118] Crash Avoidance Metrics Partners (CAMP) LLC. Ee requirements and specifications supporting scms software release 1.2.2. *SCMS CV Pilots Documentation*, 2016.

[119] Norbert Bißmeyer. *Misbehavior Detection and Attacker Identification in Vehicular Ad-hoc Networks*. PhD thesis, Technische Universität, Darmstadt, December 2014.

[120] European Telecommunications Standards Institute (ETSI). ETSI TS 103 096-2 V1.4.1: Intelligent Transport Systems (ITS); Testing; Conformance test specifications for ITS Security; Part 2: Test Suite Structure and Test Purposes (TSS & TP). *ITS-WG5*, pages 1–52, August 2018.

[121] Javier Gozalvez, Miguel Sepulcre, and Ramon Bauza. Ieee 802.11p vehicle to infrastructure communications in urban environments. *IEEE Communications Magazine*, 50(5):176–183, May 2012. ISSN 1558-1896. doi: 10.1109/MCOM.2012.6194400.

[122] Joseph Kamel. Dare: A reports dataset for global misbehavior authority evaluation in c-its, 2020. URL `https://github.com/josephkamel/DARE-Dataset`.

[123] Anthony Goldbloom. Your machine learning and data science community. *Kaggle by Google*, 2010.

[124] James Bergstra, Dan Yamins, and David D. Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in Science Conference (SciPy 2013)*, 2013.

[125] European Commission (EC). Security Policy & Governance Framework for Deployment and Operation of European Cooperative Intelligent Transport Systems (C-ITS). *Cooperative, connected and automated mobility (CCAM)*, pages 1–36, December 2017.

[126] Christoph Allig, Tim Leinmüller, Prachi Mittal, and Gerd Wanielik. Trustworthiness estimation of entities within collective perception. In *2019 IEEE Vehicular Networking Conference (VNC) (IEEE VNC 2019)*, Los Angeles, USA, December 2019.

[127] Rens van der Heijden, Thomas Lukaseder, and Frank Kargl. Analyzing attacks on cooperative adaptive cruise control (cacc). In *2017 IEEE Vehicular Networking Conference (VNC)*, pages 45–52, Nov 2017. doi: 10.1109/VNC.2017.8275598.

[128] Marco Iorio, Fulvio Risso, Riccardo Sisto, Alberto Buttiglieri, and Massimo Reineri. Detecting injection attacks on cooperative adaptive cruise control. In *2019 IEEE Vehicular Networking Conference (VNC) (IEEE VNC 2019)*, Los Angeles, USA, December 2019.

INSTITUT POLYTECHNIQUE DE PARIS

Titre: Détection d'anomalies comportementales pour les systèmes de transport intelligents et coopératifs (STI-C)

Mots clés: Cybersécurité, STI-C, V2X, Apprentissage Automatique, Détection d'Anomalies

Résumé: Les systèmes de transport intelligents coopératifs (STI-C) est une technologie qui changera notre expérience de conduite. Dans ce système, les véhicules coopèrent en échangeant des messages de communication Vehicle-to-X (V2X) sur le réseau véhiculaire. Les applications de sécurité routière utilisent les données de ces messages pour détecter et éviter à temps les situations dangereuses. Par conséquent, il est crucial que les données des messages V2X soient sécurisées et précises. Dans le système STI-C actuel, les messages sont signés avec des clés digitales pour garantir leur authenticité. Cependant, l'authentification ne garantit pas l'exactitude des données. Un véhicule authentifié pourrait avoir un capteur défectueux et donc envoyer des informations inexactes. Un attaquant pourrait également obtenir des clés légitimes en piratant l'unité embarquée de son véhicule et donc transmettre des messages malveillants signés. La détection des mauvais comportements dans les STI-C est un sujet de recherche visant à garantir l'exactitude des messages V2X échangés. Il consiste à surveiller la sémantique des données des messages échangés pour détecter et identifier des entités à comportement suspect. Le processus de détection est divisé en plusieurs étapes. La détection locale consiste à effectuer d'abord des vérifications de plausibilité et de cohérence sur les messages V2X reçus. Les résultats de ces vérifications sont ensuite fusionnés à l'aide d'une application de fusion locale. L'application est capable d'identifier diverses anomalies V2X. Si une anomalie est détectée, le véhicule collectera les preuves nécessaires et créera un rapport de mauvais comportement. Ce rapport est ensuite envoyé à une autorité cloud de mauvais comportement. Cette autorité a pour objectif d'assurer le bon fonctionnement du système C-ITS et d'atténuer les effets des attaques. Elle recueillera d'abord les rapports des véhicules, puis enquêtera sur l'événement et décidera de la réaction appropriée. Dans cette thèse, nous évaluons et contribuons aux différents composants du processus de détection des comportements malveillants : la détection locale, le reporting et la détection globale.

Title: Misbehavior Detection for Cooperative Intelligent Transport Systems (C-ITS)

Keywords: Cybersecurity, C–ITS, V2X, Machine Learning, Anomaly Detection

Abstract: Cooperative Intelligent Transport Systems (C-ITS) is an upcoming technology that will change our driving experience in the near future. In such systems, vehicles cooperate by exchanging Vehicle-to-X communication (V2X) messages over the vehicular network. Safety applications use the data in these messages to detect and avoid dangerous situations on time. Therefore, it is crucial that the data in V2X messages is secure and accurate. In the current C-ITS system, the messages are signed with digital keys to ensure authenticity. However, authentication does not ensure the correctness of the data. A genuine vehicle could have a faulty sensor and therefore send inaccurate information. An attacker could also obtain legitimate keys by hacking into the on-board unit of his vehicle and therefore transmit signed malicious messages. Misbehavior Detection in C-ITS is an active research topic aimed at ensuring the correctness of the exchanged V2X messages. It consists of monitoring data semantics of the exchanged messages to detect and identify potential misbehaving entities. The detection process is divided into multiple steps. Local detection consists of first performing plausibility and consistency checks on the received V2X messages. The results of these checks are then fused using a local detection application. The application is able to identify various V2X anomalies. If an anomaly is detected, the vehicle will collect the needed evidence and create a misbehavior report. This report is then sent to a cloud based misbehavior authority. This authority has a goal of ensuring the correct operation of the C-ITS system and mitigating the effects of attacks. It will first collect the misbehavior reports from vehicles and would then investigate the event and decide on the suitable reaction. In this thesis, we evaluate and contribute to the local, reporting and global steps of the misbehavior detection process.

Institut Polytechnique de Paris 91120 Palaiseau, France