Physics Joins AI: A Real-Time Hybrid Misbehavior Detection Framework for Vehicular Networks

Lorenzo Ghiro[a,c], Cristina Pezzoni[c], Marco Franceschini[b], Renato Lo Cigno[a,c]

[a] *Department of Information Engineering, University of Brescia, Italy,*
[b] *CHECK24 GmbH, München, Germany,*
[c] *CNIT – Consorzio Nazionale Interuniversitario per le Telecomunicazioni, UdR of Brescia, Italy,*

Abstract

Trusting autonomous, connected vehicles is necessary to build Cooperative Driving applications and enhance Smart Mobility. A necessary step is full confidence in communications beyond identification, pseudonyms and certificates: Messages must be unquestionably correct to enable cooperation, even if errors are due to errors of trusted entities. Extreme reliability requires multiple evaluation tools, as independent as they can be, to fuse estimation into a dependable decision. This work proposes to combine two methods of evaluating messages, one based on Artificial Intelligence (AI) analysis and one on physical coherence of message content, achieving extremely good performance both on the VeReMi dataset, and in run-time execution in a highway scenario simulated with Plexe. Moreover, the paper proposes a simple protocol demonstrating how to safely dismantle a platoon of Cooperative Driving vehicles returning to autonomous (or human) driving when misbehaving messages are received.

Keywords: Misbehavior Detection Systems, Cooperative Driving, Vehicular Networks, Smart Mobility, Trust and Reliability

1. Introduction

Vehicular networks are built upon a range of communication technologies including Cellular V2X (C-V2X) and IEEE-based Direct Short Range Communications (DSRC). These technologies support Cooperative Driving (CD) applications, which are central to the development of Smart Mobility. Such applications rely on the timely and reliable exchange of information between Connected Vehicles (CVs) to enable coordinated actions on the road.

While the communication layer in vehicular networks has been extensively studied, this alone is not sufficient to ensure the safe and effective deployment of CD applications. In fact, CD requires more than just the correct delivery of messages, it also demands messages to be semantically accurate and trustworthy. When this is not the case, robust emergency protocols must be in place to prevent potentially dangerous situations. Wrong and untrustworthy messages arise from *misbehaviors*, which can result from unintentional faults, such as sensor errors due to GPS drift or radar malfunction, or by intentional actions, where an attacker forges and transmits incorrect messages to disrupt the coordination among vehicles. In both cases, the consequences can compromise the safety and efficiency of the entire system.

To mitigate such risks, Misbehavior Detection Systems (MDSs) are required to analyze messages and identify anomalies that deviate from expected patterns. Two key contributions in this area are the VeReMi [1, 2] dataset and the F^{2}MD framework [3]. VeReMi provides a large collection of vehicular messages affected by different types of misbehavior in urban scenarios, making it a widely used resource for training MDSs. F^{2}MD provides an extension to Veins[1] supporting the modeling of misbehaviors and the evaluation of candidate detection methods, including both heuristic and machine learning approaches.

Despite these efforts, several open questions remain. Existing studies do not evaluate the perfor-

[1] Veins `https://veins.car2x.org/` is an open source framework to jointly simulate vehicular traffic and communications [4].

mance of MDS in the context of full CD applications like platooning. There is also no systematic investigation into the benefits of hybrid detection systems that combine AI models with heuristics based on physical models. Moreover, previous evaluations have mostly been limited to off-line datasets and do not explore the impact of misbehaviors on CD at run time.

This paper addresses these gaps by introducing a complete framework for detecting misbehavior during platooning operations, extending the paper presented at WONS 2025 [5]. The main contributions of this work are:

- A new hybrid MDS model that fuses two evaluation components, one AI based and one based on physics consideration we call RULE based;
- The separate analysis of AI and RULE based components as well as their fusion strategy and performance;
- The run-time evaluation of the MDSs;
- A protocol to safely dismantle a Cooperative Driving platoon in presence of misbehaviors.

2. Background and Related Work

Several surveys have been conducted on the security of vehicular networks [6, 7, 8, 9]. These works broadly categorize threats into external and internal attacks.

External attacks, such as jamming and denial-of-service (DoS), are typically countered through the use of authentication and authorization credentials [3, 10, 11]. Internal misbehaviors, instead, are more problematic. They originate from vehicles that are correctly authenticated but transmit false data, either intentionally to induce critical CD systems to make unsafe decisions, or randomly due to malfunctions. Regardless of their intentional or unintentional origin, false data are a danger and demand the development of Misbehavior Detection Systems (MDSs) that go beyond identification, authentication, and authorization to analyze the semantics of the messages.

2.1. MDSs in V2X

MDSs can be broadly divided into two categories: Rule-based and AI-based systems. Rule-based systems verify the plausibility of received messages against physical and kinematic laws, along with system and logical constraints. AI-driven ones, on the other hand, use structures learned from labeled datasets to assess message correctness. Reinforcement learning [12] adds continuity to the learning process, possibly leading to better results in dynamic environments. Some authors also consider federated learning [13, 14, 15]. Although the approach is promising, the need for real-time packet analysis in dynamic environments renders the federation of learned knowledge particularly difficult to realize in practice. Cooperation is often considered (see for instance [16]) as a means to inform Misbehavior Authorities for authorization revocation processes.

Another common distinction lies in the deployment location of the MDS: on-board, edge, or cloud. This work focuses on on-board MDSs, as they are inherently better suited for low-latency safety applications such as platooning and CD. In fact, designing real-time edge or cloud-based MDS architectures is inherently challenging, since the strict latency requirements of message-by-message analysis make remote processing a less viable option.

In the following we recap the main works (or surveys summarizing the state of the art) for AI and RULE based methods, with focus on methods suitable for on-board implementation.

2.1.1. AI-based Methods

In the wake of AI and Machine Learning interest, AI-based MDSs have received great attention and surveys like [17, 18, 19, 20, 21, 22] offer a comprehensive overview of the proposed approaches. These include supervised learning and unsupervised models [23, 24], as well as advanced architectures like Recurrent Neural Networks [12], Long Short-Term Memory (LSTM) [25, 26, 27] and Federated learning [13].

Each technique offers trade-offs: Supervised models require labeled data and offer usually higher accuracy, still, high accuracy is often achieved overfitting training data and damaging the transfer of knowledge to driving scenarios different from the one where labeled data were collected. Conversely, unsupervised ones can adapt more easily to new data distributions but may lack precision.

Deep learning methods, especially LSTM-based models, are popular due to their ability to handle sequential time-series data such as streams of Cooperative Awareness Messages (CAMs); however, they require large and diverse training datasets and, even

when such datasets are available, transfer learning to new scenarios remains challenging.

In general, AI-based model performance is evaluated only offline on simulation-based datasets, while the performance of such systems at run-time is not explored.

2.1.2. Rule-based Methods

Rule-based approaches evaluate message plausibility by comparing data with model-driven predictions. Examples of plausibility checks can be found in [1, 28].

Plausibility checks for CAMs (we use this term for any message for simplicity) should first of all be based on the standardized message generation rules, as those reported in [29]. However, implementing the full set of generation rules is a complex task, also because these rules are still evolving. A preliminary effort in this direction is presented in [30]. Moreover, CAMs in widely used datasets are typically not generated according to these standards, but rather emitted at a fixed rate (e.g., 1 Hz in the case of VeReMi), disregarding the actual generation conditions defined in the specifications.

Examples of RULE methods can be found in [31, 32, 33, 34]. Particularly interesting is [33] that combines RULE and AI (Deep Learning) as we do in this paper, but in the context of an edge (Road Side Unit (RSU)) implementation. Also notable are the ideas reported in [32], where RULEs are expressed in terms of spatio-temporal coherence evaluated with appropriate filtering techniques.

2.2. The VeReMi dataset

Most of the MDS for Vehicular Networks (VNs) discussed in Section 2.1.1 require the training of a neural network based on a dateset where messages are labeled either as *genuine* or *malicious*: The only well known open dataset of this kind is the VeReMi dataset [1, 2], which contains tens of millions of messages collected during simulations of different kind of misbehaviors in VNs involving thousands of vehicles, traveling for several hours in an urban scenario. We exploit the VeReMi dataset to design an MDS crafted to tackle the selection of attacks reported in Tab. 1. The key simulation assumptions that characterize the VeReMi dataset are:

- They are based on the well-known Luxembourg SUMO Traffic (LuST) scenario [35];
- Simulations span over a 24h time-horizon, but messages are collected only during 2 main time intervals, i.e., 7AM-9AM (rush hour) and 2PM-4PM (medium density traffic);
- In all simulations a fraction of vehicles, namely, the 30% of them, are malicious/malfunctioning, while the remaining 70% always generate genuine messages;
- Messages (genuine or not) are standard CAMs or similar messages sent at 1 Hz.

This last hypothesis has unfortunately a major impact on performance evaluation, as CAMs are not generated following the standard procedures [29], and the low transmission frequency of 1 Hz introduces substantial communication delays. As a result, previously received information quickly becomes outdated, making accurate model predictions particularly challenging.

In real-world deployments, CAMs and other cooperative driving messages are primarily triggered by situational changes rather than the periodic 1 s deadline. Constructing a more realistic dataset for MDS evaluation represents a relevant path for future research; nevertheless, since the proposed framework does not rely on inter-message timing as an explicit feature, it remains applicable across varying generation rates, as demonstrated by its deployment in platoon scenarios where CAMs are generated at 10 Hz.

3. The Hybrid MDS

The Hybrid MDS architecture we propose, shown in Fig. 1, consists of two main components: the AI-MDS and the RULE-based MDS, detailed in Section 4 and Section 5. This section focuses on the overall structure and the additional components that complete the MDS.

The Hybrid MDS is designed to be installed on board of each CV; its duty is to raise a warning when received CAMs are believed to be generated by a malfunction or an ongoing attack. To this purpose, the first component is the *CAM Stream Processor* (Section 3.1), that in our architecture is responsible to group together CAMs coming from the same transmitter and to preprocess them to simplify the task for the subsequent AI-MDS and RULE-MDS blocks. The two MDS are fed by the Stream Processor in parallel and they independently emit a verdict $\Lambda = \{0, 1\}$, with 0 indicating genuine

Misb. Family	Misb.	Label	Description
Position	Constant	C_{pos}	Transmit the same GNSS coordinates over time
	Random	R_{pos}	Transmit random GNSS coordinates
	RndOffset	O_{pos}	Transmit true GNSS coordinates shifted with a random offset
	Eventual Stop	$EvSt$	As for Constant, but speed is also set to 0. This attack also falls in the Speed-based category
Speed	Random	R_{spd}	Transmit random speed values
	RndOffset	O_{spd}	Transmit true speed of the vehicle shifted with a random offset
Information Replication	Data Replay	D_{rep}	Transmit information previously received from a specific target neighbor
	Disruptive	Dis	Transmit information previously received from a random target neighbor

Table 1: Misbehaviors selected from the VeReMi dataset used for training the proposed MDS. The last two misbehaviors represent explicit malicious attacks, while the remaining ones may originate either from unintentional faults or from deliberate malicious actions.

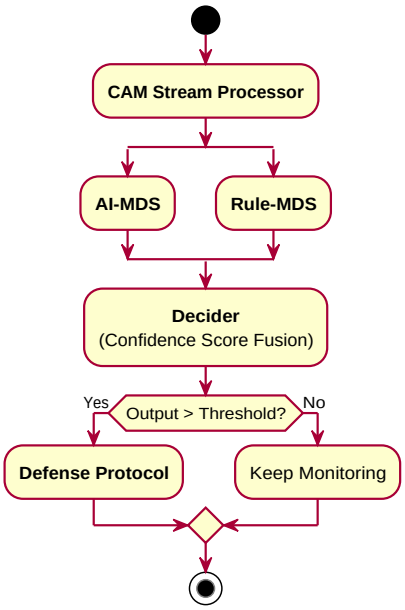

Figure 1: Hybrid MDS architecture: a DECIDER combines the outputs of the AI and RULE components by weighting them according to their respective confidence levels.

messages and 1 defective ones, associated with confidence level $\nu \in [0, 1]$ that can be interpreted as the probability that the verdict is correct. Unlike model-driven approaches such as RULE-MDS, which are not trained on labeled data, the AI-MDS is capable of associating each verdict with a misbehavior label, as illustrated in Tab. 1.

Verdicts are combined in the DECIDER, described in Section 3.2. If its output exceeds a given threshold, then the DECIDER triggers a warning that can activate countermeasures. In Section 7.1 we propose and evaluate an emergency protocol to dismantle a platoon if the misbehavior of a fellow platooning vehicle is detected.

3.1. CAM Stream Processor

The CAM Stream Processor performs two main tasks: (i) grouping messages, and (ii) normalizing them. The first task is essential for both the RULE-MDS and the AI-MDS. The second, however, is specific to the AI-MDS, as normalization enhances generalization during learning. In contrast, the RULE-MDS relies on the semantics of individual CAM fields to build predictions and assess plausibility without the need for normalization.

The CAM Stream Processor has an additional task related to the AI-MDS training: CAM *labeling*. This step is necessary because of the structure of the extended VeReMi dataset [2], which organizes data into folders according to the simulated misbehavior scenario. While this structure provides a high-level indication of the type of attack, it does not include message-level labels. As a result, individual CAMs must be labeled explicitly before they can be used to train a neural network. To address this, the Stream Processor infers the correct label for each CAM by exploiting the dataset structure where each folder contains:

1. **Log files** with the messages received by each vehicle, potentially including incorrect data if the sender was misbehaving.

Feature	Description
Label	A number in the range $[0-8]$. 0 if the message is classified as "genuine", 1 to 8 refer to the misbehaviors presented in Tab. 1.
sendTime	Timestamp added by the sender vehicle.
sender	Id of the sender vehicle.
receiver	Id of the receiver vehicle.
pos(x,y)	X,Y GNSS coordinates of the sender. All recorded positions are in the municipal area of the city of Luxembourg.
spd(x,y)	X,Y speed components of the sender.
acl(x,y)	X,Y acceleration components of the sender.
hed(x,y)	X,Y heading components of the sender, computed by an on-board compass.

Table 2: Features of classified messages. Some features originally available in the VeReMi dataset are not used in this work, so they are not reported in this table. The heading feature is used by the Rule-MDS but not by the AI-MDS; AI-MDS uses only the absolute value |acl(x,y)|.

2. **Ground truth** files with the actual data sent by each vehicle, recorded directly at the transmitter side.

The CAM Stream Processor labels each message by comparing the logs with the ground truth: if the version of the message found in the logs (i.e., at the receiver side) matches the corresponding ground truth, the message is labeled as "genuine". Otherwise, the message is labeled according to the misbehavior type defined by the corresponding VeReMi folder. The resulting dataset of classified messages is structured as shown in Tab. 2.

3.1.1. Message Grouping

The traces of messages exchanged between each transmitter–receiver pair constitute CAM streams, reflecting how messages would be received by vehicles in real-world scenarios. For MDS purposes, these streams are modeled as time-ordered sequences. To evaluate the k-th CAM, an MDS considers the most recent w messages, including the k-th itself. In this work, we set $w = 2$ for the Rule-MDS and $w = 5$ for the AI-MDS; these values are chosen to maintain low computational complexity while still achieving good detection performance. In the case of the AI-MDS, if the $w = 5$ messages do not fall within a reasonable temporal window, the k-th message is labeled as *undecidable*, and the window slides on. While in real deployments this temporal window would need to be relatively short, for consistency with existing literature –where all messages are considered valid regardless of timing– we set it to ∞ in our VeReMi-based analysis.

In AI literature, and specifically for LSTM, message grouping has been performed with both sliding and jumping windows, with varying window sizes from a few units up to hundreds of them. For example, in [36] a sliding window of size 10 is used. A larger window size favors the learning of long-horizon latent features, however, it increases the classification delay hampering driving safety. In [5], we adopted *jumping* window approach with $w = 5$, reducing this way the computational complexity as classification was performed only once every 5 messages. However, this introduces a delay in decision-making, which may be detrimental for CD applications that are highly sensitive to latency. For this reason, in the present work we shift to a sliding window approach, enabling timely per-message evaluation. The sliding window approach computational requirements are small enough to make it suitable for on-board implementation. As for the Rule-MDS, its computational overhead is negligible and does not pose any performance concerns.

3.1.2. Message Normalization for AI-MDS

During the offline training, it is necessary to eliminate the bias related to positions, all in the municipality of Luxembourg. To de-correlate the position from locality and allow generalization, we fix the first message of each window as reference for the rest of the group. Then we compute the feature-wise differences between the remaining 4 messages and the reference one. Tab. 3 illustrates the result of this operation. This "progressive difference table" captures well the regularity of CAMs in time, allowing a neural network to learn the plausibility of messages.

3.2. Decider

AI-MDS and Rule-MDS verdicts Λ_{A} and Λ_{R} associated with their confidence levels ν_{A} and ν_{R} are combined by the Decider with Eq. (1).

$$\begin{aligned} s_d &= \nu_{\text{R}} \cdot (2\Lambda_{\text{R}} - 1) + \nu_{\text{A}} \cdot (2\Lambda_{\text{A}} - 1) \\ \Lambda_{\text{D}} &= \text{sign}[s_d] \end{aligned} \tag{1}$$

where sign[ ] is a function that returns 1 (misbehaved CAM) if the argument is ≥ 0 and 0 (genuine CAM) otherwise. This decision rule is designed to favor agreement between the AI-MDS and Rule-MDS, but when their outputs differ, it selects the verdict

	ΔT [s]	Δ(x,y) [m]	Δspd(x,y) [m/s]	Δacl [m/s^2]
$m_1 - m_0$	1	$(0.5, 0)$	$(1, 0)$	0
$m_2 - m_0$	2	$(2, 0)$	$(2, 0)$	0
$m_3 - m_0$	3	$(4.5, 0)$	$(3, 0)$	0
$m_4 - m_0$	4	$(8, 0)$	$(4, 0)$	0

Table 3: Example of the differential features for $w = 5$ for a vehicle that constantly accelerates 1 m/s^2 driving on a straight road sending CAMs at 1 Hz. The vehicle departs (at time of message m_0) from position $x, y = (0\,\text{m}, 0\,\text{m})$ and with initial speed $\vec{v}_0 = (0\,\text{m/s}, 0\,\text{m/s})$. The x coordinate is always aligned with the vehicle direction, thus the y coordinate is always 0 both for position and speed.

with the greater confidence. We call this simple decision method Confidence-based Score Fusion (CSF). In case Λ_{R} or Λ_{A} are undecided (see Sections 3.1.1 and 5 for the explanation why this can happen), the decision relies on the one available verdict, i.e., we assume a confidence 0 from the MDS that has not classified the message.

3.3. Scalability

The computational complexity of the MDS is dominated by the AI-MDS component, since the RULE-MDS performs only a small number of arithmetic operations per message. As detailed in Section 4, the neural network architecture is lightweight, and training is performed entirely offline; consequently, on-board inference is the only processing step required at runtime. Scalability is therefore determined by the number of concurrent message streams a vehicle must monitor, which is bounded by the number of vehicles the own vehicle is interested into. Even in highly dense traffic conditions, this number is unlikely to exceed a few tens of vehicles: a worst-case urban scenario such as a large three-lane roundabout with a 50 m radius accommodates at most 50–60 vehicles, and even accounting for vehicles on the approach and exit roads, the total rarely exceeds 100. A comparable upper bound holds for highway scenarios, where considering 7 lanes with 10–12 vehicles of interest per lane yields fewer than 100 vehicles.

In summary, the proposed MDS requires a computing platform capable to maintain up to 100 per-vehicle message queues, and to process up to 1000 messages per second given that safety messages are standardized to a maximum generation rate of 10 Hz.

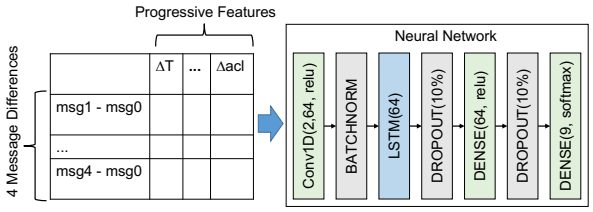

Figure 2: Architecture of the neural network proposed to enhance the one published in [5] and format of the VeReMi processed data provided as input.

This throughput is well within the capabilities of commodity processors.

4. AI-MDS

The AI-MDS works on the "message progressive differences" exemplified in Tab. 3 on groups of $w = 5$ messages, but since we use a sliding window decisions are taken message per message.

Fig. 2 illustrates the architecture of the optimized neural network we devised starting from the one presented in [5]. The novel architecture has ≈ 39k parameters, while the previous one ≈ 311k, thus the model complexity is reduced by a factor 8. A smaller model improves the training speed, and most of all has a smaller computing and memory footprint for the on-board implementation, even if a modern CV can be considered an unconstrained computing unit from the perspective of a simple CAM classification task.

The initial CONV1D layer performs BATCH-NORMALIZATION. This front-end layer captures local regularities in the input, and also denoises and compresses it. The presence of this layer (compared to the architecture proposed in [5]) allows reducing the size of the expensive LSTM layer from 256 to 64 neurons without loss of accuracy.

The LSTM architecture is designed to capture temporal correlations between consecutive messages, with the final 9-neurons softmax layer that outputs the logit( ) over the classification space $\{0, 1, \ldots, 8\}$, where 0 indicates genuine messages while labels greater than zero indicate one of the misbehavior reported in Tab. 1.

To estimate the confidence level ν_{A}, we introduce two DROPOUT filters –one after the LSTM and another after the subsequent dense layer– each with a drop rate of 10%. This means that, during each forward-pass, the output of each neuron in the preceding layer is randomly suppressed with probability

0.1. This technique introduces two benefits: first, it reduces the risk of overfitting during training, and more importantly, DropOut enables multiple stochastic forward passes through the network at inference time, allowing confidence estimation via the standard Monte Carlo method [37].

Although DropOut increases the computational load due to repeated network evaluations, it is essential to enable reliability assessment, crucial to the operation of the Hybrid MDS, where the Decider fuses verdicts based on confidence scores (Eq. (1)). In the results presented in Sections 6 and 8, we perform 10 stochastic forward passes and compute the average logit vector $\bar{L}(\,)$ across repetitions. The predicted class is then selected by identifying the index i corresponding to the highest average component $\bar{L}(i)$. Accordingly, the verdict Λ_{A} is set to 1 (indicating misbehavior) for all $i > 0$, and to 0 otherwise. The confidence level c_l is computed using the standard T-Student technique, taking $\bar{L}(i)$ as the point estimate and defining a confidence interval of $\pm 10\%\, \bar{L}(i)$, with saturation at the bounds 0 and 1. Finally, the confidence ν_{A} is computed as:

$$\nu_{\text{A}} = \bar{L}(i) \cdot c_l \tag{2}$$

This formulation ensures that the final confidence reflects both the predicted class probability and its variability across the multiple network evaluations.

Clearly ν_{A} tends to 1 when repeated forward-passes are highly consistent, with most of the softmax probability mass concentrated on the same label consistently. Conversely, ν_{A} rapidly drops toward 0 when the logit( ) of different runs are inconsistent or simply scattered across different labels.

5. Rule-MDS

The Rule-MDS operates on pairs of consecutive CAMs and assesses the plausibility of the most recent message based on physical and kinematic extrapolations from the previous one.

Let M_p and M_c be the previous and current CAMs transmitted by the same vehicle v_{tx} and received by the ego vehicle. For Rule-MDS purposes the CAMs are the time stamped (t) collection of vectorial representation of position $\vec{p}$, speed $\vec{v}$, and acceleration $\vec{a}$, as well as associated communication level information θ, such as the signal strength, signal to noise ratio, and other useful information to characterize the frame:

$$M_p = (\vec{p_p}, \vec{v_p}, \vec{a_p}, t_p, \theta_p); \quad M_c = (\vec{p_c}, \vec{v_c}, \vec{a_c}, t_c, \theta_c)$$

The time interval between the previous and the current CAM is $\Delta_t = t_c - t_p$.

The acceleration cannot be predicted, as it is the consequence of driving conditions and decisions. Thus we can only check that the acceleration variation, the jerk $\jmath = ||\vec{a_p} - \vec{a_c}||/\Delta_t$ in m/s^3, never exceed plausible values. If $\jmath$ is plausible, we use the previous $\vec{a_p}$ to predict $\vec{v_c}$ and $\vec{p_c}$ from M_p values:

$$\begin{cases} \hat{v} = \vec{v_p} + \vec{a_p}\Delta_t \\ \hat{p} = \vec{p_p} + \vec{v_p}\Delta_t + \frac{1}{2}\vec{a_p}{\Delta_t}^2 \end{cases} \tag{3}$$

Finally, we compute the distance, or error, between the estimates and the M_c values:

$$\begin{cases} \eta_v = ||\hat{v} - \vec{v_c}|| \\ \eta_p = ||\hat{p} - \vec{p_c}|| \end{cases} \tag{4}$$

The jerk $\jmath$, and errors η_v and η_p are used to assign metric-specific scores ($s_\jmath$, s_v, s_p), that in turn will define the confidence ν_{R} of the Rule-MDS related verdict. The scores are computed with a simple trapezoid function (see Fig. 3) that assigns score 1 (fully plausible) if the error (or jerk) is below a lower bound LB; a linearly decreasing score between LB and an upper bound UB, and 0 (completely non plausible) outside the trapezoid base. The values selected for LB and UB for the three metrics are reported in Tab. 4, with position errors that are related to the expected displacement $\Delta_p = \dfrac{\vec{v_p} + \hat{v}}{2}\Delta_t$. Jerk values are set considering comfort in standard driving conditions (LB) and physical limits of vehicles (UB) [38]. The other limits simply derive

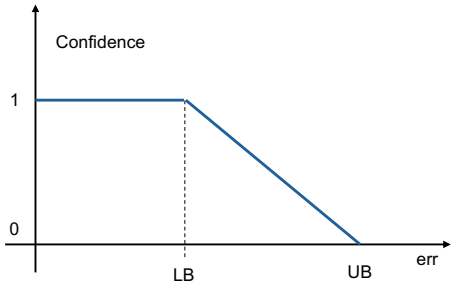

Figure 3: Plausibility function for $\jmath$, η_v, and η_p (err) modeled as a trapezium.

	LB	UB
$\jmath$	$8\,\mathrm{m/s^3}$	$20\,\mathrm{m/s^3}$
η_v	$10\%\,\hat{v}$	$25\%\,\hat{v}$
η_p	$20\%\,\Delta_p$	$30\%\,\Delta_p$

Table 4: Bounds LB and UB for $\jmath$, η_v, and η_p.

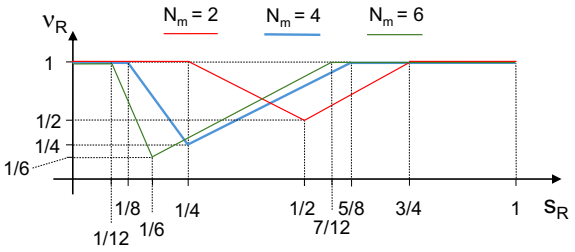

Figure 4: Piecewise linear function used to compute ν_{R} as a function of N_m and s_{R}. ν_{R} is high (1) for s_{R} well apart from the critical threshold $1/N_m$ and drops to the its minimum for scores getting closer to $1/N_m$.

from reasonable approximations and safety considerations: UBs are set at values that define dangerous situations for CD, and LBs to values compatible with sensors errors and uncertainties.

Additional coherence checks should be done at the transmission and electromagnetic (EM) level on a set of parameters we called θ. Unfortunately the VeReMi dataset does not include any feature related to communications, which makes impossible to propose consistency and plausibility checks. These checks should start from the characteristics of the communication standard adopted and appropriate propagation models, and also include the distance between the transmitter and the ego vehicle as well as the relative speed to estimate, for instance, the Doppler spread of the signal. Given the impossibility to include these checks with a realistic detail, we include in RULE-MDS two extremes: i) no EM checks at all, and ii) a trivial unit-radius transmission range approximation.

For the second case, we note that the simulations used to construct the VeReMi dataset were conducted using 802.11p technology with a transmission power of 100 mW (20dBm). According to the 802.11p standard, the outdoor communication range with a transmission power of 20dBm is 200 m. We thus consider any message originating from a vehicle located beyond a distance of 220 m –as computed from the CAMs content– to be non-plausible. This threshold corresponds to a 10% margin above the estimated maximum communication range. In other words for this test we set LB = 200 m and UB = 220 m. We refer to this check as the ElecroMagnetic Coherence (EMC) test, to reflect its intended purpose. While we acknowledge that this is a coarse approximation, it is introduced solely to evaluate whether such a constraint has any impact on the overall results. This type of test was originally introduced in [28] as one of the features, where it was referred to as the Accepted Range Threshold (ART) check.

Summing up, the RULE-MDS evaluates N_m physical metrics related to kinematics, EM propagation, and possibly other model-based characteristics, each of these metrics is assigned a score $s \in [0, 1]$ with 0 indicating perfect adherence to the model and 1 complete disagreement. Based on these metrics the RULE-MDS should emit the verdict Λ_{R} and its confidence level ν_{R}. We proceed as follows. First we compute the average score s_{R} as

$$s_{\mathrm{R}} = \frac{1}{N_m} \sum_{i=1}^{N_m} s_i \tag{5}$$

where s_i are the scores of the single metrics, e.g, $\jmath$, η_v, and η_p. Then the verdict is assigned:

$$\Lambda_{\mathrm{R}} = \mathrm{sign}\left[s_{\mathrm{R}} - \frac{1}{N_m} \right] \tag{6}$$

and ν_{R} is computed with the piecewise linear function in Fig. 4, which is a function of the number of metrics N_m.

In practice, the RULE-MDS classifies a CAM as misbehaved either when at least one plausibility score equals 1, or when the sum of multiple sub-threshold scores reaches a total weight equivalent to a single score of 1. The confidence ν_{R}, however, is set to 1 for $\Lambda_{\mathrm{R}} = 0$ only if s_{R} is smaller than $\frac{1}{2N_m}$, and for $\Lambda_{\mathrm{R}} = 1$ only if s_{R} is larger than $\frac{1}{N_m} + \frac{1}{2}(1 - \frac{1}{N_m})$.

In the VeReMi dataset, due to the way CAMs are generated without respecting the standard, around 10% of $(M_p,\ M_c)$ pairs exhibit data implying an "unbelievable" vehicle behavior, e.g., full 180° turns or acceleration and speed modulo sign change. This usually happens for large Δ_t likely due to message collisions or losses. As a result, model-based prediction is not applicable, and the RULE-MDS returns "undecidable" as Λ_{R} verdict.

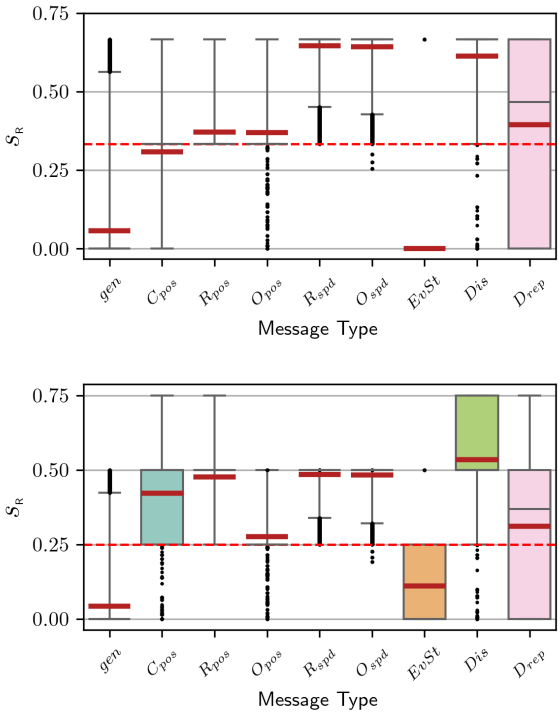

Figure 5: Boxplots of s_R for all message types computed without EMC ($N_m = 3$) in the upper plot and with the approximated EMC ($N_m = 4$) in the lower plot.

Prior to discussing the results on VeReMi (Section 6) and in a real-time CD scenario (Section 8), we examine the influence of the EMC-based rules, though they are applied in a simplified form as noted above. Fig. 5 shows s_R obtained without EMC (upper plot) and with EMC (lower plot) separated for each message type. It is evident that kinematic models alone are insufficient to detect misbehaviors that involve only incorrect position information, especially in cases such as Constant Position and Eventual Stop, which are inherently similar. In the Eventual Stop scenario, all kinematic parameters appear plausible, effectively mimicking the behavior of a parked vehicle (note that the VeReMi dataset does not include actual parked vehicles). However, the vehicle is in fact moving, and its growing distance from the ego vehicle becomes implausible. As a result, even the approximated EMC test is able to detect the anomaly. In a real implementation of an EMC-based model, received packets should be consistent with the observed variation in inter-vehicle distance, making this type of test both robust and reliable. For this reason, from now on, all results

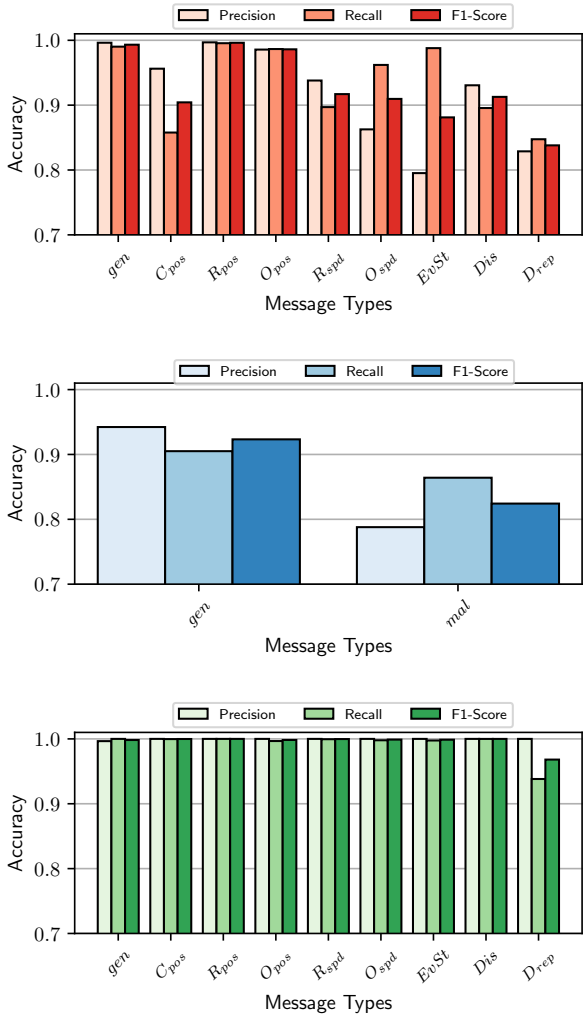

Figure 6: Precision, Recall and F1-Score metrics evaluated on VeReMi with AI-MDS only (top plot), Rule-MDS only (middle plot) and Hybrid-MDS (lower plot).

are shown for $N_m = 4$ including EMC.

6. Offline MDSs Performance

Fig. 6 report the performance in terms of Precision, Recall and F1-Score for the AI-MDS, the Rule-MDS and the Hybrid-MDS evaluated on the VeReMi dataset. The AI-MDS distinguishes all types of messages, as they are labeled in VeReMi, while the Rule-MDS instead performs a binary classification.

The first clear observation is that the Hybrid-MDS achieves significantly better performance than the AI-MDS or Rule-MDS used individually. A

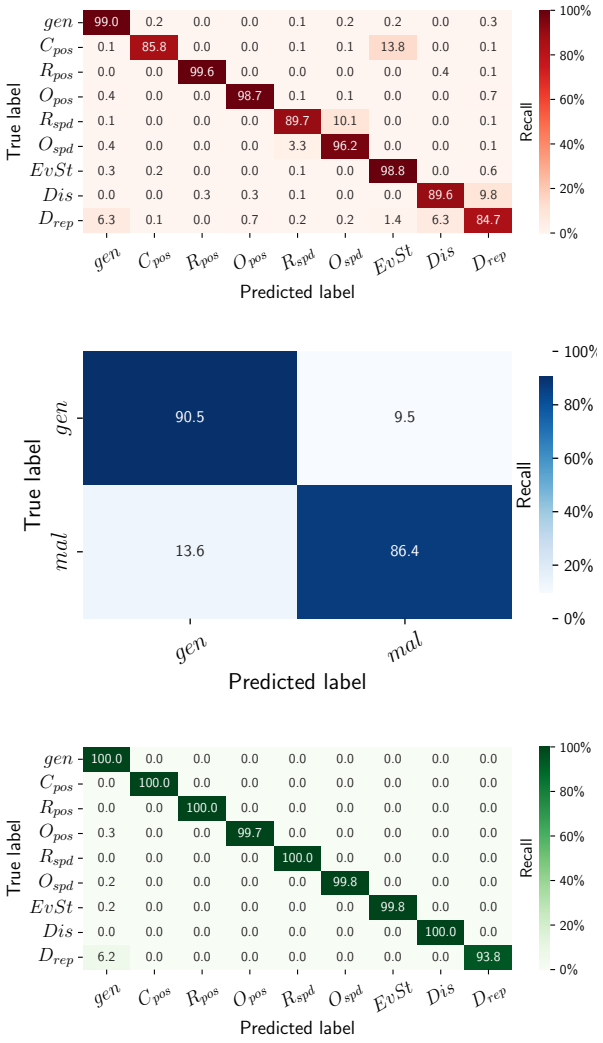

Figure 7: Confusion matrices evaluated on VeReMi with AI-MDS only (top plot), Rule-MDS only (middle plot) and Hybrid-MDS (lower plot).

joint analysis of Fig. 6 and the lower plot of Fig. 5 highlights that the Data Reply misbehavior poses the greatest challenge for the detection of incorrect messages. In this scenario, the messages are structurally valid but are simply replayed with a delay; without additional checks, many of them may therefore appear legitimate. This type of misbehavior is also close to what could be considered an intentional attack. Other techniques might be required to detect it effectively. A deeper inspection of real CAMs and their encapsulating frames could reveal useful features for AI-MDS and inspire additional rules for Rule-MDS, thus improving detection capabilities.

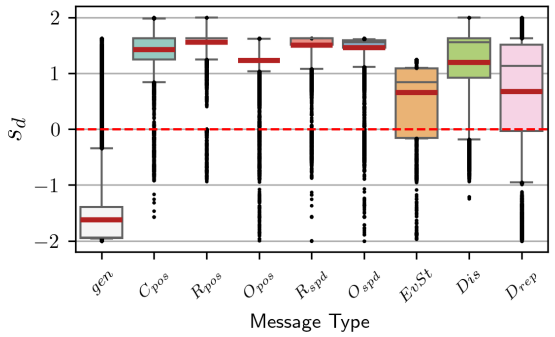

Figure 8: Boxplots of s_d for all message types.

The confusion matrices in Fig. 7 provide a more detailed view of the system's performance. Again, the top plot corresponds to the AI-MDS, the middle to the Rule-MDS, and the bottom to the Hybrid-MDS. First of all, the AI-MDS alone already achieves a good classification precision in general, since most of the diagonal values exceeds 88%, with few exceptions, namely, C_{pos} and D_{rep}, with C_{pos} that tends to be confused with $EvSt$ ($\approx$ on the 14% of the C_{pos} samples). The similar nature of the 2 misbehaviors explains this evidence: In fact, with both Constant Position and the Eventual Stop, misbehaving messages are characterized by the "frozen" Global Navigation Satellite System (GNSS) coordinates, with $EvSt$ messages further characterized by zeros as speed and acceleration values, still, the common position patterns mildly confuses the classifier. Some D_{rep} (Data Replay) messages are instead falsely reported as genuine: This can happen especially for those messages that are replayed after a small casual delay, small enough to make the replayed message extremely similar to the genuine message they were copied from. The Rule-MDS appears less accurate in its classifications, primarily due to the high number of outliers in the computed plausibility scores (see Fig. 5). Notably, the Hybrid-MDS achieves near-perfect classification performance, with the only significant exception being the D_{rep} messages, consistent with the challenges previously discussed.

Fig. 8 offers a final insight into the causes of the residual classification errors, which, although minimal, are still present. The Hybrid-MDS correctly classifies the vast majority of messages, but the presence of clear outliers explains the remaining errors. Particularly puzzling are the outliers among genuine messages. This can be attributed to the way the

VeReMi dataset is built: CAMs are sent at a fixed rate of 1 Hz, without following the standard generation rules. As a result, a one-second interval may be too long for model-based prediction –especially when relying on sample-and-hold assumptions– affecting both the AI-MDS and RULE-MDS. Additionally, in SUMO simulations actions are atomic: for example, a 90° turn may occur within a single simulation step of 0.1 s. Thus, a vehicle that turns immediately after sending a CAM may appear at a position significantly different from what any model could plausibly predict, an artifact unlikely to occur in datasets built according to the standard CAM generation rules.

Considering only studies with comparable methodologies [2, 3, 28, 30, 39, 40, 41], the HYBRID-MDS demonstrates consistently superior performance over existing literature approaches. The remarkable works [3, 39] exhibit indeed nearly optimal detection accuracy in line with our HYBRID-MDS, however, the proposed HYBRID-MDS architecture is lighter and simpler, its Rule based component offers an easier tuning interface and, above all, in Section 8 we prove its ability to be successfully transferred to novel, runtime scenarios different from the offline analysis of VeReMi.

7. Emergency Platoon Disassemble

One of the main motivations and contributions of this work is to assess whether the results obtained in offline settings are confirmed in an online implementation, and to evaluate the extent to which this improves CD safety. A particularly relevant aspect, already partially explored in [5], is the deployment of the AI-MDS in a completely different scenario. Although it was designed with the specific goal of learning generalizable rather than scenario-specific features, only experimental validation can confirm its effectiveness. The RULE-MDS, by contrast, is intrinsically more robust, and is expected to perform even better in an online setting where beacons are generated at 10 Hz, typical frequency for CD applications. For this analysis, we select highway platooning as a representative and practically relevant scenario.

7.1. Emergency Disassemble Protocol

Conceptually, disassembling a platoon is straightforward: when a member detects a misbehavior, it notifies the other vehicles that it is safer to revert to independent autonomous driving, as illustrated in Fig. 9. In practice, however, the underlying dynamics are significantly more complex and may lead to unstable behaviors or even collisions if not properly handled, as discussed in [42]. The key challenge is to guarantee that all vehicles safely increase their inter-vehicle distance before transitioning from Cooperative Adaptive Cruise Control (CACC) to Adaptive Cruise Control (ACC), or to any other autonomous controller that does not rely on inter-vehicle communication. To this end, we adopt the Gap Control Algorithm proposed in [42], which enables a smooth and coordinated increase of spacing among platoon members prior to the transition. The detailed design and analysis of this mechanism are provided in [42] and are not repeated here for brevity.

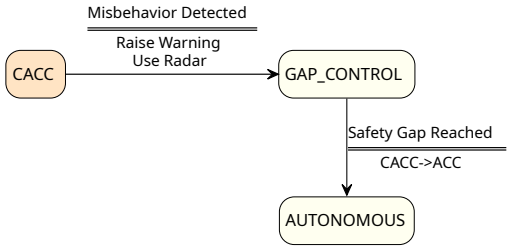

Figure 9: Finite State Machine describing the transition from a CACC driving regime to an AUTONOMOUS one upon detection of any misbehavior.

Algorithm 1 outlines the vehicle logic underlying an emergency disassembly procedure triggered in response to a warning raised by the MDS within the platoon. During the initialization routine, each vehicle allocates the *neighMap* data structure to collect received CAMs. A garbage collection routine, not shown for brevity in Algorithm 1, is regularly invoked and ensures that too aged CAMs (received more than 10s ago) are not kept in memory.

The *onCAM* procedure is invoked whenever a vehicle receives a CAM message m from a neighbor v. If v is a new neighbor, a 5-sized message queue is allocated for storing the beacons sent by v. New CAMs are enqueued per sender, replacing the oldest entry if needed. The algorithm collects and evaluates messages from all surrounding vehicles, not just those belonging to its own platoon. Clearly, the driving reaction will be different depending on the misbehaving vehicle.

Fig. 10 shows the effect of the platoon disassembling protocol activated when a misbehavior is detected in a platoon of 4 vehicles. The leader is the vehicle v_0 (not shown in Fig. 10) while v_1, v_2, v_3

Algorithm 1 Driving algorithm informed by the message classifier.

```
1: procedure INIT()
2:     neighMap ← {}
3: procedure onCAM(msg=m, neigh=v)
4:     if v ∉ neighMap then
5:         neighMap ← new msgQueue()
6:     neighMap[v].push(m)
7:     Λ_D ← Hybrid-MDS.evaluate(neighMap[v])
8:     if Λ_D AND v is a platoon member then
9:         EmergencyProtocol()
```

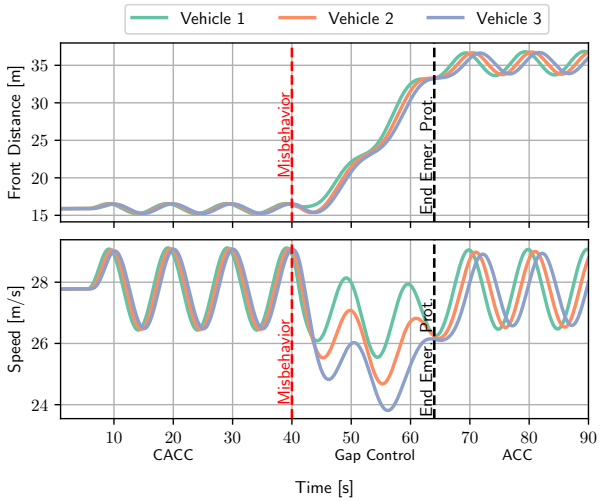

Figure 10: Evolution over time of the front-distance of the vehicles before and after the detection of a misbehavior. Before $t = 40\,\text{s}$ vehicles adopts the Ploeg CACC with a time-headway of 0.5 s; the leader follows a sinusoidal speed pattern with period 10 s and an average speed of 100 km/h $\approx$ 27.77 m/s, implying a front-distance of $\approx$ 15 m. At $t = 40\,\text{s}$ the leader $v0$ starts sending Constant Position CAMs, the followers detect it and enter the GAP_CONTROL mode, slowing down to enlarge the front-distance until they reach the target one for an ACC with a 1.2 m/s time headway ($\approx$ 35 m). This happens at $t \approx 63\,\text{s}$ when they switch to the AUTONOMOUS mode.

are the platoon followers. To introduce stress into the scenario, the leader follows a sinusoidal speed profile, as shown in the caption. A constant-speed profile would not sufficiently reveal the risks and limitations of autonomous or cooperative driving systems. The followers correctly keep a distance corresponding to a 0.5 s time-headway until a misbehavior is introduced in the simulation, triggering the transition delimited by dashed vertical lines in the plot. During this phase the front distance gently increases for all vehicles until they switch to autonomous driving based on local sensors with a front-distance of $\approx$ 35 m. This means that, under emergency, the protocol is able to guarantee the safety of all vehicles.

8. Run-Time Evaluation

We implemented the framework proposed in this work in Plexe [42, 43], the Cooperative Driving (CD) framework that extends Veins [4], allowing the realistic simulation of platoons.[2] In particular, through the use of Python Bindings, we have connected the C++ CD Application on board of each Plexe vehicle with the MDS implemented with PyTorch. We have also customized the Driving Application that now fully supports the collection of beacons and the computation of their progressive differences so to feed the MDS at run-time. Furthermore, we implemented the Emergency Protocol as a Plexe module, to be used when the MDS signals a misbehavior. All the misbehaviors reported in Tab. 1 have been implemented by altering the CAMs generated by the Beaconing service already available in Plexe. Finally, we configured the Plexe Sinusoidal Scenario[3] to mimic the misbehaviors selected by the experimenter at random start times, and analyze the effect in a stressful situation. Simulations are run for platoons in isolation and for platoons surrounded by other vehicles that generate CAMs and drive close to the platoon average speed, so that during the simulation the relative position of vehicles change continuously, as the platoon follows a sinusoidal speed profile. Tab. 5 reports the main parameters of the simulation campaign.

The main objectives of the experiments are two:

1. Evaluate the ability of the framework to reduce accidents;
2. Evaluate the accuracy and responsiveness of the MDS at run-time.

It is worth emphasizing that this scenario, involving highway platoons, differs significantly from the LuST scenario. As such, these experiments also serve to evaluate the generalization capabilities of the proposed framework, particularly of the AI-MDS,

[2] The code and documentation for the framework proposed in this manuscript are available online at https://ans.unibs.it/software/plexemds.

[3] Documented online: https://plexe.car2x.org/tutorial/#sinusoidal-scenario

	Parameter	Value
Scenario	Road Type	3-Lane Highway
	Duration	120 s
	Misb. Start Time	Uniform[15,30]s
	Default CACC	PLOEG
	PLOEG Headway	0.5 s
	Autonomous Controller	ACC
	ACC Headway	1.2 s
	Beaconing Frequency	10 Hz
	Platoon Size	4
	Leader Speed	100 km/h
	Speed Oscillation Amplitude & Freq	5 km/h, 0.1 Hz
	Background vehicles when present	9
	Repetitions per Experiment	100
Comm.	L2-technology	dual radio 802.11p
	Tx power	100 mW
	Broadcast MCS	3 Mbit/s
	Unicast MCS	12 Mbit/s
	Rx sensitivity	−94 dBm

Table 5: Parameters characterizing vehicles and communications in the simulation experiments.

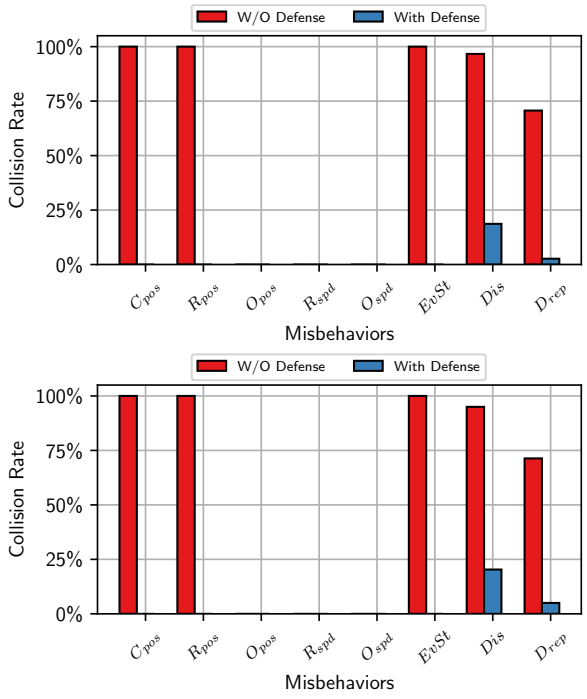

Figure 11: Barchart comparing the collision rate in simulations where the emergency protocol was either enabled (blue) or disabled (red). The upper plot is for platoons in isolation, the bottom one with background traffic.

which is trained exclusively on the VeReMi dataset and has never been exposed to highway or platoon-specific data.

To evaluate the protection offered by the Hybrid-MDS, we compute the collision rate, i.e., the fraction of simulations resulting in a collision, across scenarios with and without the Hybrid-MDS and disassembly protocol. Fig. 11 shows the comparison in terms of collision rate. The red bars refer to simulations without Hybrid-MDS, where platooning vehicles end up using wrong information in the platoon control law, and the blue ones to those with Hybrid-MDS. The upper plot refer to platoons in isolation and the bottom one with the background traffic. When background traffic is present, any vehicle can be the misbehaving one, either in the platoon or not, but the protocol is activated only if the misbehaving vehicle is in the platoon. The first observation is that there is no evident impact of the background traffic, which is good news and a novel result compared to [5], where instead more results for different platoons sizes in isolation were reported.

Some misbehaviors are naturally less dangerous than others, for example, even turning off all defense mechanisms the O_{pos}, R_{spd} or O_{spd} misbehaviors never lead to any collision, but this may change using different CACC algorithms or in heterogeneous conditions [44]. The defense mechanism turns out to be instead perfectly able to always detect and defuse the C_{pos}, R_{pos} and $EvSt$ misbehaviors, reducing the collision rate from 100% to 0%. The impact of such errors on platooning is critical, as the longitudinal control in Ploeg relies heavily on the ego vehicle's distance from the preceding vehicle. This strong dependency explains the observed 100% collision rate.

The Dis and D_{rep} misbehaviors exhibit a different pattern. Even without any defense mechanism, not all simulations result in collisions. However, in some cases, the Hybrid-MDS fails to detect the misbehavior leading to collisions, though at a significantly lower rate. These findings are consistent with the observations in [2], where D_{rep} was shown to be particularly effective in evading detection systems. As previously noted, this is most likely due to the impossibility of implementing EMC checks against the VeReMi dataset, and we did not implement them in the online setup to enable a fair comparison with the offline MDS.

We now shift the attention towards another key performance metric for an MDS evaluated at runtime, i.e., its reaction time. In fact, it is not enough for an MDS to be accurate, as a slow reaction time may in any case lead to dangerous situations, up to fatal collisions.

Fig. 12 shows the distribution of reaction times

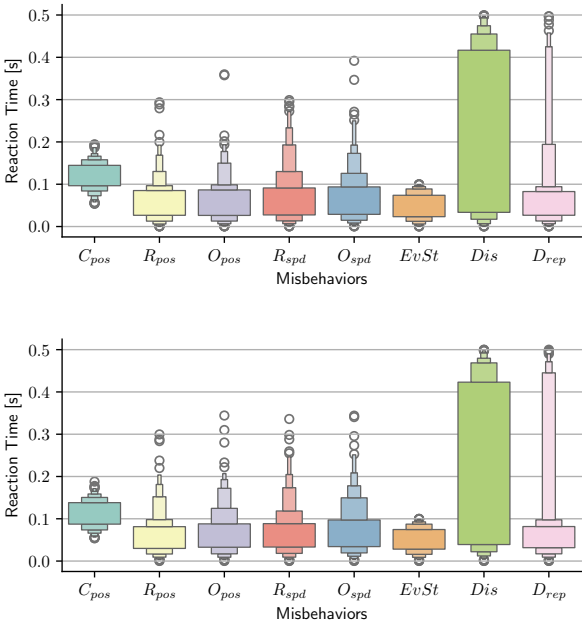

Figure 12: Discrete Violin Plots of the reaction-times observed in the experiments. The upper plot is for platoons in isolation, the bottom one with background traffic.

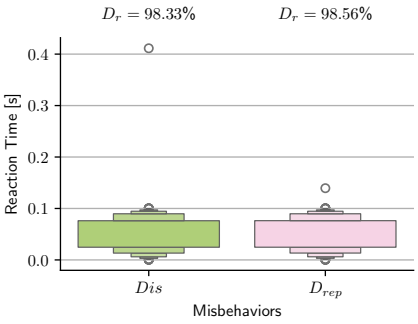

Figure 13: Discrete Violin Plots of the detection time when misbehaved vehicles are not in the platoon for the Dis and D_{rep} cases. The D_r reported on top of violin indicates the detection ratio, i.e., the fraction of simulations where the misbehavior was detected, thus providing a valid value of reaction time to populate the distribution.

across all simulated platoons affected by some misbehavior, represented as discrete violin plots. In the vast majority of cases, misbehavior is detected upon reception of the first incorrect CAM, which is a highly encouraging result. Although a few outliers require up to 4 or 5 CAMs to trigger detection, the reaction time remains within safe bounds, as CAMs are generated at 10 Hz in platooning scenarios, thus enabling timely countermeasures. A noteworthy observation is that C_{pos} misbehaviors often require two incorrect messages before being detected. This is likely because the first message still lies within the trapezoidal plausibility range defined for the RULE-MDS (recall Fig. 3), while the second, possibly in combination with AI-MDS, triggers the detection. A similar delay is observed for Dis and D_{rep} misbehaviors, which are primarily detected by the AI-MDS, but only after a complete window of $w = 5$ incorrect CAMs has been received. Lastly, it may appear counterintuitive that the reaction time is often shorter than a single CAM interval; however, this is consistent with the fact that the onset of misbehavior occurs at a random time, uniformly distributed between two consecutive CAMs.

Finally, Fig. 13 reports the detection time when the misbehaving vehicle is not in the platoon, but only for Dis and D_{rep}, which are the cases when messages could (they never do, but they could) interfere with the platooning application. The numbers on the top (D_r) refer to the detection ratio, and the Violins to the detection delay. Also in this case results are very good, and even if these misbehaviors do not interfere with platoon cooperation, it is important that they are swiftly detected, for instance for reporting to an Authority for further actions.

9. Discussion and Conclusions

This work presents a substantial extension of the misbehavior detection framework for vehicular networks previously introduced in [5]. The advancement goes beyond a refinement of the AI model, introducing a novel hybrid approach that combines machine learning with rule-based physical validation. Alongside the neural detector, we have introduced a Rule-based component capable of assessing the physical and kinematic consistency of received messages using deterministic models. These two complementary perspectives —statistical inference on one side, and semantic plausibility on the other— are integrated through a confidence score fusion mechanism that balances their respective contributions in a principled way.

Beyond conceptual integration, the work also brings significant architectural improvements. The original neural network has been redesigned to significantly reduce its complexity (from over 300k to approximately 39k parameters), while preserv-

ing classification accuracy. This makes the system more efficient and well-suited for real-time execution on in-vehicle hardware. The message preprocessing pipeline has also been revised, shifting from a jumping window strategy to a sliding window approach, enabling faster and more continuous decision-making, essential for time-sensitive Cooperative Driving applications. In experimental evaluations, the system demonstrated not only strong offline performance on the VeReMi dataset, but also high effectiveness in real-time simulated scenarios, reducing the average reaction time from 0.5 s to approximately 0.1 s.

These results confirm that a hybrid approach can overcome the limitations of individual techniques, achieving a favorable balance between accuracy, responsiveness, and robustness. Nevertheless, several challenges remain to be addressed in order to enable industrial-grade deployment. While synthetic datasets such as VeReMi remain essential tools for initial training and evaluation, they must be enriched to include additional metrics related to the physical communication layer –such as received and transmission power– which are currently missing but crucial for accurately tuning rule-based modules and enhancing the inputs of AI-based ones.

Looking forward, the most pressing challenge is the realistic calibration of synthetic datasets, which can only be achieved by incorporating real-world data. In this regard, collaboration with vehicle manufacturers will be key. It is desirable that, in accordance with ETSI standards governing CAM generation frequency, automotive OEMs provide access to real message traces. Such data would enable more accurate tuning and validation of detection models, bridging the gap between simulation and real-world deployment. Only through this step can experimental solutions evolve into reliable systems ready for large-scale adoption in future autonomous and cooperative driving scenarios.

Real network traces would further enable the characterization of channel-induced impairments, such as message losses and delay jitter, currently absent from simulation-based datasets. In addition, the availability of physical-layer measurements — such as received signal strength or channel quality indicators— would open the way to richer feature sets for MDSs, potentially leading to more accurate and robust detection systems.

Acknowledgments

This work was partially supported at the University of Brescia by the European Union and Italian Ministry for Universities and Research through the National Recovery and Resilience Plan (NRRP), project “Sustainable Mobility Center (MOST)”, 2022-2026, CUP D83C22000690001, Spoke N^o 7, “CCAM, Connected networks and Smart Infrastructures,” program “Security and Rights in CyberSpace (SERICS),” (PE00000014), Spoke 7, Project SCAR, Cascade Call “SCAR: a Privacy enHAnced SEcurity framework (SCARPHASE),” CUP C89J24000580008, and project “PROSDS" Spoke 4, Structural Project SUPER part of program RESTART (PE00000001), CUP C89J24000270004 .

References

[1] R. W. Van Der Heijden, T. Lukaseder, F. Kargl, VeReMi: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs, in: 14th EAI Int. Conf. on Security and Privacy in Communication Networks (SecureComm), Singapore, Singapore, 2018, pp. 318–337.

[2] J. Kamel, M. Wolf, R. W. Van Der Hei, A. Kaiser, P. Urien, F. Kargl, VeReMi Extension: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs, in: IEEE Int. Conf. on Communications (ICC), Dublin, Ireland, 2020, pp. 1–6.

[3] J. Kamel, M. R. Ansari, J. Petit, A. Kaiser, I. B. Jemaa, P. Urien, Simulation Framework for Misbehavior Detection in Vehicular Networks, IEEE Trans. on Vehicular Technology 69 (6) (2020) 6631–6643.

[4] C. Sommer, R. German, F. Dressler, Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis, IEEE Trans. on Mobile Computing 10 (1) (2011) 3–15.

[5] M. Franceschini, L. Ghiro, R. Lo Cigno, An On-Line Misbehavior Detection Framework for Vehicular Networks, in: 20th IEEE Wireless On-Demand Network Syst. and Services Conf. (WONS), Hintertux, Austria, 2025, pp. 1–8.

[6] R. W. van der Heijden, S. Dietzel, T. Leinmüller, F. Kargl, Survey on Misbehavior Detection in Cooperative Intelligent Transportation Syst., IEEE Communications Surveys & Tutorials 21 (1) (2019) 779–811.

[7] X. Sun, F. R. Yu, P. Zhang, A Survey on Cyber-Security of Connected and Autonomous Vehicles (CAVs), IEEE Trans. on Intelligent Transportation Syst. 23 (7) (2022) 6240–6259.

[8] T. Zaidi, F. Syed, An Overview: Various Attacks in VANET, in: 4th IEEE Int. Conf. on Computing Communication and Automation (ICCCA), Greater Noida, India, 2018, pp. 1–6.

[9] S. Sharma, A. Kaul, A survey on Intrusion Detection Syst. and Honeypot based proactive security mechanisms in VANETs and VANET Cloud, Elsevier Vehicular Communications 12 (2018) 138–164.

[10] S. Khan, F. Luo, Z. Zhang, M. A. Rahim, M. Ahmad, K. Wu, Survey on Issues and Recent Advances in Vehic-

ular Public-Key Infrastructure (VPKI), IEEE Communications Surveys & Tutorials 24 (3) (2022) 1574–1601.
[11] S. S. Manvi, S. Tangade, A survey on authentication schemes in VANETs for secured communication, Elsevier Vehicular Communications 9 (2017) 19–30.
[12] R. Sedar, C. Kalalas, F. Vázquez-Gallego, J. Alonso-Zarate, Reinforcement Learning Based Misbehavior Detection in Vehicular Networks, in: IEEE Int. Conf. on Communications (ICC), Seoul, Korea, 2022, pp. 3550–3555.
[13] A. Uprety, D. B. Rawat, J. Li, Privacy Preserving Misbehavior Detection in IoV Using Federated Machine Learning, in: 18th IEEE Annual Consumer Communications & Networking Conf. (CCNC), Las Vegas, NV, USA, 2021, pp. 1–6.
[14] H. Yakan, I. Fajjari, N. Aitsaadi, C. Adjih, Federated Learning for V2X Misbehavior Detection System in 5G Edge Networks, in: Int. ACM Conf. on Modeling Analysis and Simulation of Wireless and Mobile Syst. (MSWiM), Montreal, Quebec, Canada, 2023, p. 155–163.
[15] E. Mármol Campos, J. L. Hernandez-Ramos, A. González Vidal, G. Baldini, A. Skarmeta, Misbehavior detection in intelligent transportation systems based on federated learning, Elsevier Internet of Things 25 (2024) 101127.
[16] R. Sultana, J. Grover, M. Tripathi, Cooperative approach for data-centric and node-centric misbehavior detection in VANET, Elsevier Vehicular Communications 50 (2024) 100855.
[17] M. Almehdhar, A. Albaseer, M. A. Khan, M. Abdallah, H. Menouar, S. Al-Kuwari, A. Al-Fuqaha, Deep Learning in the Fast Lane: A Survey on Advanced Intrusion Detection Syst. for Intelligent Vehicle Networks, IEEE Open J. of Vehicular Technology 5 (2024) 869–906.
[18] A. Boualouache, T. Engel, A Survey on Machine Learning-Based Misbehavior Detection Syst. for 5G and Beyond Vehicular Networks, IEEE Communications Surveys & Tutorials 25 (2) (2023) 1128–1172.
[19] M. L. Bouchouia, H. Labiod, O. Jelassi, J.-P. Monteuuis, W. B. Jaballah, J. Petit, Z. Zhang, A survey on misbehavior detection for connected and autonomous vehicles, Elsevier Vehicular Communications 41 (2023) 100586.
[20] H. Bangui, B. Buhnova, Recent advances in machine-learning driven intrusion detection in transportation: Survey, Elsevier Procedia Computer Science 184 (2021) 877–886.
[21] A. Talpur, M. Gurusamy, Machine Learning for Security in Vehicular Networks: A Comprehensive Survey, IEEE Communications Surveys & Tutorials 24 (1) (2022) 346–379.
[22] O. Y. Al-Jarrah, C. Maple, M. Dianati, D. Oxtoby, A. Mouzakitis, Intrusion Detection Syst. for Intra-Vehicle Networks: A Review, IEEE Access 7 (2019) 21266–21289.
[23] M. Matousek, M. Yassin, A. Al-Momani, R. van der Heijden, F. Kargl, Robust Detection of Anomalous Driving Behavior, in: IEEE 87th Vehicular Technology Conf. (VTC Spring), Porto, Portugal, 2018, pp. 1–5.
[24] M. Matousek, M. EL-Zohairy, A. Al-Momani, F. Kargl, C. Bösch, Detecting Anomalous Driving Behavior using Neural Networks, in: IEEE Intelligent Vehicles Symposium, Paris, France, 2019, pp. 2229–2235.
[25] X. Liu, Misbehavior Detection based on Deep Learning for VANETs, in: IEEE Int. Conf. on Networks, Communications and Information Technology (CNCIT), Beijing, China, 2022, pp. 122–128.
[26] T. Alladi, V. Kohli, V. Chamola, F. R. Yu, A deep learning based misbehavior classification scheme for intrusion detection in cooperative intelligent transportation systems, Elsevier Digital Communications and Networks 9 (5) (2023) 1113–1122.
[27] J. Kamel, I. B. Jemaa, A. Kaiser, L. Cantat, P. Urien, Misbehavior Detection in C-ITS: A comparative approach of local detection mechanisms, in: IEEE Vehicular Networking Conf. (VNC), Los Angeles, CA, USA, 2019, pp. 1–8.
[28] J. Kamel, A. Kaiser, I. ben Jemaa, P. Cincilla, P. Urien, CaTch: A Confidence Range Tolerant Misbehavior Detection Approach, in: IEEE Wireless Communications and Networking Conf. (WCNC), Marrakech, Morocco, 2019, pp. 1–8.
[29] ETSI, Intelligent Transport Syst. (ITS); Vehicular Communications; Basic Set of Applications; Cooperative Awareness, Tech. Rep. TS 103 900 V2.1.1, ETSI (Nov. 2023).
[30] A. Willecke, B. Kulke, L. C. Wolf, A4MD: Artery for Misbehavior Detection, Reporting, and Reaction in the ETSI C-ITS, in: IEEE Vehicular Networking Conf. (VNC), Istanbul, TR, 2023, pp. 33–40.
[31] S. So, P. Sharma, J. Petit, Integrating Plausibility Checks and Machine Learning for Misbehavior Detection in VANET, in: 17th IEEE Int. Conf. on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 2018, pp. 564–571.
[32] F. A. Ghaleb, M. Aizaini Maarof, A. Zainal, M. A. Rassam, F. Saeed, M. Alsaedi, Context-aware data-centric misbehaviour detection scheme for vehicular ad hoc networks using sequential analysis of the temporal and spatial correlation of the consistency between the cooperative awareness messages, Elsevier Vehicular Communications 20 (2019) 100186.
[33] S. Park, D. Kim, S. Lee, Enhancing V2X Security Through Combined Rule-Based and DL-Based Local Misbehavior Detection in Roadside Units, IEEE Open J. of Intelligent Transportation Syst. 5 (2024) 656–668.
[34] R. Sultana, J. Grover, M. Tripathi, P. Sharma, LA-DETECTS: Local and Adaptive Data-Centric Misbehavior Detection Framework for Vehicular Technology Security, IEEE Open J. of Vehicular Technology 6 (2024) 145–169.
[35] L. Codecá, R. Frank, S. Faye, T. Engel, Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation, IEEE Intelligent Transportation Syst. Magazine 9 (2) (2017) 52–63.
[36] H.-Y. Hsu, N.-H. Cheng, C.-W. Tsai, A deep learning-based integrated algorithm for misbehavior detection system in VANETs, in: ACM Int. Conf. on Intelligent Computing and its Emerging Applications, Jinan, China, 2021, pp. 53–58.
[37] Y. Gal, Z. Ghahramani, Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning, in: 33rd PMLR Int. Conf. on Machine Learning, New York, NY, USA, 2016, pp. 1050–1059.
[38] K. N. de Winkel, T. Irmak, R. Happee, B. Shyrokau, Standards for passenger comfort in automated vehicles: Acceleration and jerk, Elsevier Applied Ergonomics 106 (2023) 103881.
[39] S. C. Sangapu, K. S. N. Prasad, R. J. Kannan, T. M. Chen, M. Sathiyanarayanan, Impact of class imbalance in VeReMi dataset for misbehavior detection in

autonomous vehicles, Springer Soft Computing (2023) 1–11.

[40] R. Drenyovszki, Z. C. Johanyák, Development of an attack detection module for vehicular ad-hoc networks, in: IEEE 18th Int. Symposium on Applied Computational Intelligence and Informatics (SACI), Timisoara, Romania, 2024, pp. 255–60.

[41] O. Slama, M. Tarhouni, S. Zidi, B. Alaya, One Versus All Binary Tree Method to Classify Misbehaviors in Imbalanced VeReMi Dataset, IEEE Access 11 (2023) 135944–135958.

[42] M. Segata, R. Lo Cigno, T. Hardes, J. Heinovski, M. Schettler, B. Bloessl, C. Sommer, F. Dressler, Multi-Technology Cooperative Driving: An Analysis Based on PLEXE, IEEE Trans. on Mobile Comp. 22 (2023) 4792–4806.

[43] M. Segata, S. Joerer, B. Bloessl, C. Sommer, F. Dressler, R. Lo Cigno, PLEXE: A Platooning Extension for Veins, in: IEEE VNC 2014, Paderborn, Germany, 2014, pp. 53–60.

[44] M. Segata, L. Ghiro, R. Lo Cigno, On the Progressive Introduction of Heterogeneous CACC Capabilities, in: 13th IEEE Vehicular Networking Conf. (VNC), Ulm, Germany, 2021, pp. 1–8.