An On-Line Misbehavior Detection Framework for Vehicular Networks

Marco Franceschini*, Lorenzo Ghiro†, Renato Lo Cigno†
*DII – University of Brescia, Italy
†DII – University of Brescia and CNIT, Italy

***Abstract*—Cooperative Driving applications, such as platooning, pose stringent requirements in terms of safety and reliability. Multiple communication technologies can be used in parallel to enhance the reliability of Vehicular Networks (VNs), but this is not enough. Sensors malfunctions, malicious attacks, and other possible risks –collectively called *misbehaviors*–, threaten the safety of passengers, urging the development of Misbehavior Detection Systems (MDSs). A full safety framework for VNs should include one such MDS and be complemented by an emergency protocol, to be activated when a misbehavior is detected. The literature offers standalone MDSs whose detection accuracy is evaluated offline against simulation traces but, to the best of our knowledge, only few are the MDSs that work also online, and none of them is coupled with an emergency protocol. In this paper, we develop a novel framework which includes an online MDS and an emergency protocol. The detector is based on a Recurrent Neural Network (RNN) trained over *VeReMi*, a dataset that tracks messages exchanged over VNs under various simulated misbehaviors. The system accurately classifies malicious messages and distinguishes them from genuine ones offline. We further show how a good level of accuracy is preserved while executing the MDS online in a scenario different from the one used to generate the VeReMi dataset, i.e., during simulations of platoons afflicted by multiple attacks or sensor errors. We conclude showing that, online, the MDS quickly detects failures and timely activates the emergency protocol to dismantle a misbehaving platoon, ultimately contributing to the safety of passengers.**

I. Introduction

The exchange of information in Vehicle to Everything (V2X) systems is crucial for the implementation of Cooperative Perception (CP) and Cooperative Driving (CD) systems, the pillars of Smart Mobility Environments (SMEs). CP refers to systems where a group of distributed agents with sensing capability exchange information to build a common knowledge of the environment they operate in. CD indicates instead the ability of a group of road vehicles to act in coordination to achieve a common goal as, for example, avoid accidents, protect Vulnerable Road Users (VRUs), or efficiently cross an intersection. One specific CD application is *platooning*, where a group of vehicles drive together exploiting a distributed control system so that vehicles can drive at a reduced distance improving battery life, infrastructure usage, and safety of passengers.

The general safety of SMEs is jeopardized not only by deliberate cyber-attacks, but also by any misbehavior or anomaly in the exchange of information. Such anomalies stress the algorithms governing CP causing wrong perceptions that may lead to dangerous CD decisions and actions. For instance, the authors of [1] consider the loss of information (missing messages) in a multi-technology communication system as an anomaly to properly inform a platooning application on what is the most safe behavior, selecting between different control laws and finally dismantling the platoon if the information is not reliable enough to allow cooperation.

Anomaly and misbehavior are used interchangeably in this paper, although they may have slightly different meaning in specific contexts. Our goal is the analysis of any exception —from cyber-attacks to Global Navigation Satellite System (GNSS) malfunction, but also jamming, or packet losses due to interference and so forth— that may hamper the coordination required to achieve CD. We then seek for a general, simple, and robust system able to identify all such possible anomalies.

We remark that, in the context of SME, the detection of anomalies must be done on-line, or more appropriately at run-time, with small computational complexity and without offloading the task to a data-center. In fact, offloading the task would require large transfers of information, thus high delays, unacceptable for most CD systems. The use of decentralized edge computing [2] can be considered, but it is outside the scope of this work, which instead concentrates on the local detection of anomalies and on the countermeasures to be taken accordingly. Given the complexity of the topic, we focus on run-time misbehavior detection within a platooning application, and on the actions and protocols implemented to safely dismantle the platoon, returning the control of vehicles to either a human driver or an autonomous driving system.

The anomaly detection system must be trained, and this is traditionally done off-line. Unfortunately, datasets describing anomalies and misbehavior are very few: We identified the VeReMi dataset [3], [4] as a suitable starting point for our work, also because it is a good match with the simulation environment we use for run-time experiments. The contributions of this paper[1] are thus as follows:

1) We propose a simple, yet effective Machine Learning (ML) framework based on Long Short-Term Memory (LSTM) to identify all communication behaviors that deviate from normalcy, discussing its advantages and limitations, and train it on the VeReMi dataset;
2) We implement a Misbehavior Detection System (MDS) based on the aforementioned LSTM suitable for a

[1]Additional details can be found in [5].

run-time execution on board of single vehicles while these communicate with neighboring cars and/or the infrastructure;
3) We validate the MDS showing that its use prevents practically all accidents that would affect a platoon of vehicles if these last are unable to identify the simulated misbehaviors.

We focus on a simple platoon application [6] as an example of CD where wrong communications may lead to disaster.

II. Background and Related Work

Various surveys cover the topics of attacks to Vehicular Networks (VNs) [7]–[9], and some focus specifically on ML based MDSs for VNs [10]–[14]. A first, full overview on the possible attacks in a VN is given in [9], [12], where the authors stress the differences between internal and external attacks. Sun et al. [12], beyond exploring cyber-attacks against VNs, include a discussion on the misbehaviors arising from the malicious and potentially remote tampering of sensors installed on vehicles. Considering the internal defense against attacks, Van der Heijden et al. [10] introduce and analyze various misbehavior detection mechanisms, based on 4 major metrics, namely: Behavioral, Trust, Consistency, and Plausibility. Those detection mechanisms have limitations in defending against strategies that mimic normal behaviors, this is why it is necessary to devise an MDS that exploits hybrid strategies, including Artificial Intelligence (AI) techniques, to provide a wide, potentially full, coverage of all possible anomalies and misbehaviors. Boualouache et al. [14] highlight and discuss classes of ML methods used in VNs to implement MDSs, namely, Supervised, Unsupervised, Reinforcement, Deep and Transfer Learning techniques, while the survey from Talpur et al. [11] stress the links between the chosen ML technology and the target VN application.

Another comparison between different ML techniques is made in [15], [16], where the researchers contrast unsupervised and semi-supervised ML-based algorithms against Neural Network (NN) approaches on the same simulations set, showing that NNs tend to have better performances compared to other techniques like k-Nearest Neighbor (k-NN), Support Vector Machine (SVM) and Isolation Forest. The most common NN approach involves using Recurrent Neural Networks (RNNs) [17], a well-suited model for managing ordered data sequences, specifically when the ordering variable is time, thanks to their architecture that allows maintaining the memory of previous inputs. LSTM networks [18] are arguably the most popular used variant, specifically designed to handle long-term dependencies. Many works provide a misbehavior detector based on LSTM [19]–[22], showing how to deal with input data, conducting effective data manipulation, such as creating sliding or jumping windows of message sequences to be provided as NN input. This approach represents the simplest, yet effective way to deal with misbehavior detection problems in VNs and this is why we consider LSTMs as the standard solution for a detection system. The literature related to MDSs offers also more complex solutions where multiple ML models are combined. In [23] Sedar et al. present a combined architecture between LSTM and Reinforcement Learning (RL) techniques, while in [24] Uprety et al. study a Federated Learning (FL) approach, aggregating different models within a centralized server, and providing a new local detector.

We remark that none of these works attempt to implement a real-time MDS nor test it within a CD application. Furthermore, albeit all the proposed techniques have very good performance from a purely detection point of view, there is no proof that they would work just as well in avoiding the collapse of a CD system. Indeed, in all these works the MDS is not coupled with a protocol suitable to manage the detected misbehavior and do not consider how the dynamics of traffic may change the properties of the message sequences analyzed by the MDS.

III. The VeReMi dataset

We do not consider, in this work, jamming or other explicit attacks to the network itself. In fact, jamming is easy to detect and defuse at the radio level, e.g., declaring the wireless network unavailable. Similarly, we do not consider Spoofing, Sybil or other attacks based on the violation of authentication mechanisms, as they are normally countered by the use of per-message digital signatures and a centralized, classical Public Key Infrastructure (PKI). We therefore consider vehicles as authenticated and in general trusted as digital entities, still, we expect that such vehicles can simply start "*misbehaving*" because of faults or because of any action, malicious or not, that alters the semantic of the messages sent. The word "*misbehavior*" is thus used to indicate the transmission of authentic messages that, albeit digitally signed, contain wrong information because the sender vehicle is either maliciously injecting wrong data or it is equipped with malfunctioning sensors. The MDS should detect this kind of messages and, if detected, the defense protocol described in Sect. V-A should be activated to preserve vehicles safety.

Most of the MDS for VNs discussed in Sect. II require the training of a neural network based on a dateset of messages labeled either as *genuine* or *malicious*: The only well known open dataset of this kind is the VeReMi dataset [3], [4], 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. I. The simulations performed to populate the VeReMi dataset share this common facts:

- They are based on the well-known Luxembourg SUMO Traffic (LuST) scenario [25].
- 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).

Attack Family	Attack	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 also speed is set to 0. This attack falls also 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 I: Misbehaviors selected from the VeReMi dataset used for training the proposed MDS.

Feature	Description
Label	A number in the range $[0-8]$. 0 if the message is classified as "genuine", 1 to 8 refer to the attacks/misbehaviors presented in Tab. I
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 vehicle. NB: All recorded positions fall in the municipal area of the city of Luxembourg
spd(x,y)	X,Y speed components of the sender vehicle
acl(x,y)	As above but for the acceleration

Table II: 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.

- In all simulations a fraction of vehicles, namely, the 30% of them, are malicious/malfunctioning, while the remaining 70% always generate genuine messages; genuine messages are standard Cooperative Awareness Messages (CAMs) or similar messages sent at 1 Hz.

A. Message Extraction, Manipulation and Classification

This work exploits the 2nd extended version of the VeReMi dataset [4] where different folders are used to group data according to the simulated misbehavior. Each folder contains:

1) The **log files** where each vehicle dumped the received messages. These logs may report "*wrong*" information if the logging vehicle was receiving data from a misbehaving sender.
2) The **ground truth**, i.e., the true data that should have been transmitted, thus recorded at transmitter side.

Logged messages are not explicitly tagged as genuine or not, so, to perform the supervised training of the model, we add message labels comparing the logs with the ground truth. We thus recreate the trace of messages exchanged between each pair of transmitter and receiver, tagging a message as "genuine" if its version found in the logs (receiver side) matches with the ground truth, otherwise we apply the label of the misbehavior as defined by the VeReMi folder. The resulting dataset of classified messages is structured as shown in Tab. II.

	ΔT [s]	$\Delta(x,y)$ [m]	Δspd(x,y) [m/s]	Δacl [m/s^2]
$m_1 - m_0$	1	$(-1.5, 0)$	$(1, 0)$	1
$m_2 - m_0$	2	$(-4.0, 0)$	$(2, 0)$	1
$m_3 - m_0$	3	$(-7.5, 0)$	$(3, 0)$	1
$m_4 - m_0$	4	$(-12.0, 0)$	$(4, 0)$	1

Table III: Example of the differential features for a subgroup of 4 messages for a vehicle that accelerates $1\,\mathrm{m/s^2}$ driving on a straight road sending standard CAMs at 1 Hz. The x coordinate is always aligned with the vehicle direction, thus the y coordinate is always 0 both for position and speed.

B. Feature Engineering and Embedding

To improve the generalization ability and responsiveness of the MDS, we pre-pend three steps of data manipulation before feeding the LSTM. The goal is to group messages, select the appropriate features and normalize the data, and finally to relabel message groups to help their classification.

1) **Message grouping**: It is of the utmost importance to treat messages as part of time-ordered sequences so to enable, during the training, the learning of the key temporal-correlations between consecutive messages that may reveal the symptoms of an attack. In the literature message grouping has been performed with sliding or jumping windows, with varying window sizes from a few units up to hundreds of them. For example, in [26] a sliding window of size 10 is used. A larger window size favors the learning of long-horizon latent features, however, it would increase the classification delays of the corresponding online MDS, hampering driving safety. In this work we choose a jumping window with window size 5, to empower an online MDS with low memory overhead (only a 5 positions buffer is required), reduced computational complexity (classification is performand once every 5 messages), and low-latency decisions. In particular, considered that most of CD applications rely on 10 Hz beaconing, the MDS takes decisions every 0.5 s, a time-interval sufficiently small to activate the defense mechanism.
2) **Feature Selection, Normalization and Scaling**: Further manipulations on the message windows are necessary. First, it is necessary to eliminate the bias given by the fact that all recorded positions are in Luxembourg, thus in each window we fix the first message position 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. III illustrates the result of this operation. We stress that this "progressive difference tables" captures well the cinematic evolution of the sender vehicle over time, allowing a neural network to artificially learn the plausibility and consistency checks that are proposed as deterministic filters in previous works [3].
3) **Group relabeling**: The MDS is designed to work on groups of four "message progressive differences"

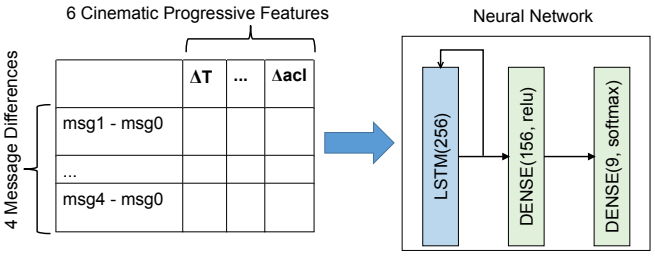

Figure 1: Architecture of the neural network at the core of the MDS and format of the VeReMi processed data provided as input.

built to highlight the vehicle cinematic evolution as shown in Tab. III. Such groups must be labeled to support a supervised learning: We chose to keep the label of the last message, i.e., the label of m_4 out of the $\{m_0, m_1, \ldots, m_4\}$ original messages, as label for the whole group. The most frequent label could also be chosen, but we prefer to chose the label of m_4 (i.e, the very last message of the group) because in VeReMi misbehaviors are *persistent*, namely, a vehicle that starts misbehaving then it keeps misbehaving till the end of the simulation. Choosing the label of the last message helps therefore the classifier to detect also attacks that starts right at the end of a sampling window, i.e., leads to the raise of a warning also when the detector samples a 5-message window where only the last message is not genuine.

IV. MDS Architecture and Validation

Fig. 1 illustrates the architecture of the neural network empowering the proposed MDS. It comprises a first LSTM layer to capture temporal correlations between consecutive messages and a final 9-neurons softmax layer to output labels in the classification space $\{0, 1, \ldots, 9\}$, where 0 indicates genuine messages while labels greater than zero indicate one of the misbehavior reported in Tab. I. The middle layer is dense with 156 neurons and ReLu activation.

Prior to training the model, the dataset of message progressive differences needs to be balanced. In fact, in VeReMi only the 30% of vehicles is misbehaving, therefore the dataset over-represents genuine messages. We keep in the balanced dataset the same amount of progressive differences for each category of misbehaviors, and a double amount of genuine ones. This way, we enforce a weak bias towards genuine messages, which lowers the number of attacks false-positives, leading to a system where emergency maneuvers are not activated if not strictly needed. We apply a standard-scaler to the balanced dataset and then we split it in 77%–23% portions for training and validation respectively. We start the offline training with batch-size equal to 64 and 100 epochs, stopping the training when accuracy improvements are not greater than 0.001.

A. Offline Accuracy Analysis

The system achieves an overall offline accuracy of 95.22% in validation. Fig. 2 highlights the differences in terms of Precision, Recall and F1-Score observed among the various

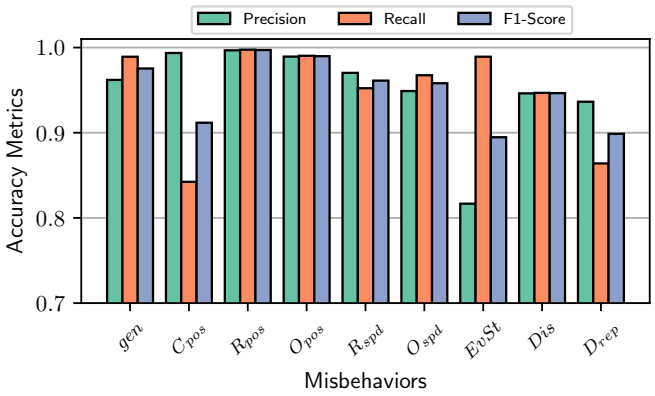

Figure 2: Precision, Recall and F1-Score metrics evaluated on all the misbehavior categories.

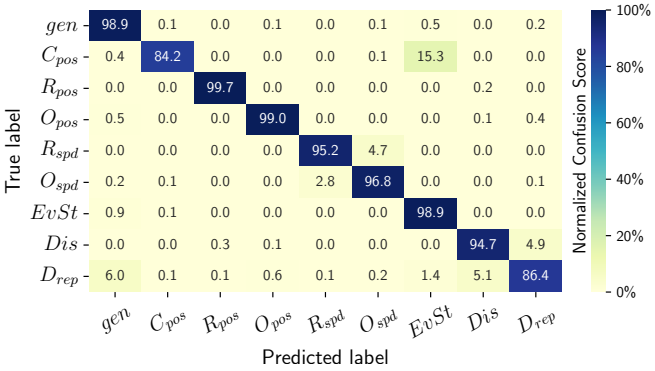

Figure 3: Offline Confusion matrix in percentage, normalized by row.

message classes, while the confusion matrix shown in Fig. 3 helps interpreting these differences. It is immediately clear that the performance is influenced by the misbehavior type.

Fig. 2 highlights that the MDS worst performance is on $EvSt$ misbehavior and the best one for R_{pos}, immediately followed by O_{pos}. The other misbehaviors fall in between. Clearly, explaining the detailed reason of each mistake (or mistake rate) is not possible, but the very good performance on R_{pos} and O_{pos} is good news, as these are the most common misbehaviors in GNSS systems also due to positioning system signal dilution and pollution.

Even if detailed explanation is not possible, the confusion matrix in Fig. 3 gives some hints on the reasons of some errors. First of all, it indicates a good classification precision in general, since most of the diagonal values exceeds 94%, with few exceptions, namely, C_{pos} and $EvSt$, with the first that tends to be confused with the second (it happens on the ≈15% of the C_{pos} samples). The similar nature of the 2 misbehaviors explains this evidence: In fact, with both the *Constant Position* (C_{pos}) and the *Eventual Stop* ($EvSt$), misbehaving messages are characterized by the "frozen" GNSS coordinates, with $EvSt$ messages further characterized by zeros as speed and acceleration values, still, the common position patterns mildly confuses the classifier. Despite these flaws limited to specific misbehaviors, the most important result is the great precision with regards to genuine messages, because in most of the use cases the classification problem

may be reduced to the binary choice between *genuine* and *misbehavior*. From this perspective, the confusion matrix confirms that the classifier can accurately support the decision of raising emergency flags.

V. Run-Time MDS and Emergency Protocol

The neural network described in Sect. IV, once trained over data extracted from the VeReMi dataset and pre-processed as described in Sect. III, produces as output a trained model, i.e., the message classifier. However, classifying messages from a dataset is only an exercise, while the final goal is to correctly identify misbehaviors while driving and appropriately react to them for the safety of vehicles passengers.

As mentioned earlier we focus on a simple platooning application, where each vehicle is equipped with the designed MDS to support the driving strategy reported in Algorithm 1.

Algorithm 1 Driving algorithm informed by the message classifier.

```
1: procedure INIT()
2:     misbehaviors ← load_misbehavior_labels()
3:     CLF ← load_trained_classifier()
4:     neighMap ← {}
5: procedure ONBEACON(msg=m, neigh=v)
6:     if v ∉ neighMap then
7:         neighMap ← new msgBuffer[5]
8:     neighMap[v].append(m)
9:     if neighMap[v].size() == 5 then
10:        label ← clf.evaluate(neighMap[v])
11:        neighMap[v].clear()
12:        CLF.DECISION(label)
13: procedure CLF.DECISION(label=lbl)
14:    if lbl ∈ misbehaviors then
15:        EMERGENCYPROTOCOL()
```

During the initialization routine, each vehicle loads the trained classifier. The ONBEACON procedure is invoked whenever a vehicle receives a beacon message m from a neighbor v. If v is a new neighbor, a 5-sized message buffer is allocated for storing the beacons sent by v (lines 6-7). New beacons are added to the buffer associated to the specific sender (line 8). The classifier requires 5 messages to emit a verdict, so it is invoked when the buffer becomes full (line 9), hence, a label is generated (line 10) and a decision should be taken. Decisions upon labels are straightforward: whenever the classifier detects a misbehavior the emergency protocol is activated (lines 13-15), otherwise the vehicle continues with the platooning cooperative driving. The algorithm collects and evaluates messages from any vehicle, not only from the vehicles in its own platoon. Clearly, the driving reaction will be different depending on the misbehaving vehicle. If after a proper timeout the MDS does not collect at least 5 messages from the same vehicle, the related buffer is discarded.

A. Platoon Dismantle Protocol for Emergencies

The Finite State Machine drawn in Fig. 4 shows the life-cycle of vehicles that activate the emergency protocol. Once a misbehavior is detected, vehicles enter in a GAP_CONTROL state. In this state they run the Gap Control Algorithm [1]

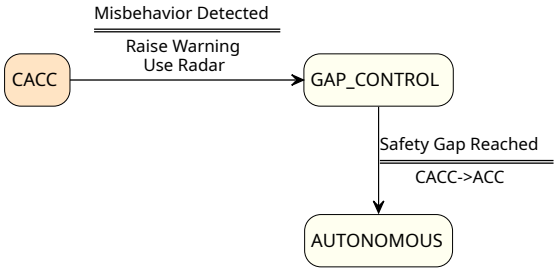

Figure 4: Finite State Machine describing the transition from a CACC driving regime to an AUTONOMOUS one upon detection of any misbehavior.

whose details are omitted here. In a nutshell, the key countermeasures taken by vehicles are:

1) Send an emergency warning in broadcast. This way the first vehicle that detects a misbehavior quickly alerts nearby vehicles. Warnings are repeated by receiving vehicles;
2) All vehicles start increasing the distance from the vehicle in front (GAP_CONTROL);
3) When the distance is safe, switch to an Adaptive Cruise Control (ACC)-based autonomous driving exclusively based local sensors.

Vehicles part of a well-behaving platoon drive at distances that are smaller than those kept by autonomous vehicles that cannot rely on any form of cooperation. The temporary GAP_CONTROL regime is thus necessary to increase the safety gaps before completing the switch to autonomous driving.

Fig. 5 shows the effect of the platoon dismantling protocol activated when a misbehavior is detected in a platoon of 4 vehicles. The leader is the vehicle v_0 (not shown in Fig. 5) while v_1, v_2, v_3 are the platoon followers. The leader follows a sinusoidal speed pattern as described in the Caption. The followers correctly keep a distance corresponding to a $0.5\,\mathrm{s}$ 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 $\approx 35\,\mathrm{m}$. This means that, under emergency, the protocol is able to guarantee the safety of all vehicles, at the (obvious) cost of loosing the advantages of platooning, so a good MDS should report all true misbehaviors, but should not indulge too much on the side of caution to avoid wasting the advantages.

VI. Experimental Evaluation of the On-Line MDS

We implemented Algorithm 1 and the Emergency Protocol detailed in Sect. V-A within Plexe [1], the well-known CD framework which extends Veins [27], 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

[2]The code will be made publicly available on the Plexe website when appropriate.

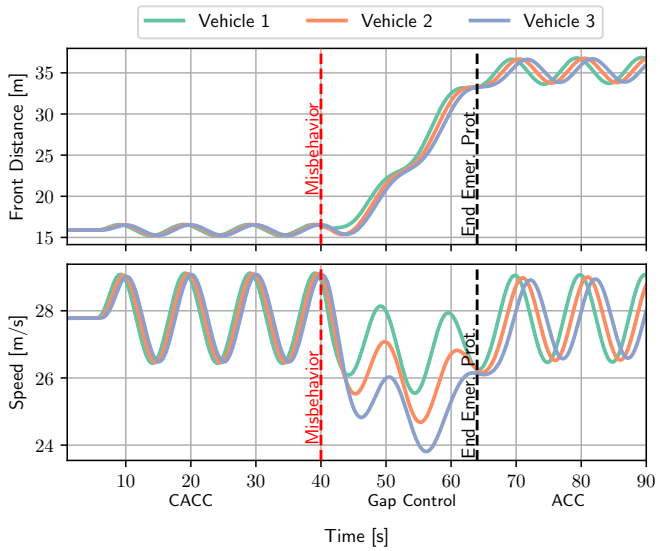

Figure 5: Evolution over time of the front-distance of the vehicles before and after the detection of a misbehavior. Before $t = 40\,\mathrm{s}$ vehicles adopts the PLOEG Cooperative Adaptive Cruise Control (CACC) with a time-headway of $0.5\,\mathrm{s}$; the leader follows a sinusoidal speed pattern with period $10\,\mathrm{s}$ and an average speed of $100\,\mathrm{km/h} \approx 27.77\,\mathrm{m/s}$, implying a front-distance of $\approx 15\,\mathrm{m}$. At $t = 40\,\mathrm{s}$ the leader $v0$ starts sending constPos 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 $1.2\,\mathrm{m/s}$ time headway ($\approx 35\,\mathrm{m}$). This happens at $t \approx 63\,\mathrm{s}$ that is when they switch to the AUTONOMOUS mode.

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 further PLEXE module, to be used when the MDS signals a misbehavior. All the misbehaviors reported in Tab. I have been implemented by altering the CAMs generated by the Beaconing service already available in PLEXE. Finally, we configured the PLEXE Sinusoidal Scenario[3] so to mimic the misbehaviors selected by the experimenter at random start times. Tab. IV reports the main parameters of the simulation campaign.

A. Experiments Definition and Goals

We crafted the experiments with two main objectives:

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

The experiments start with platoons of various size that drive on a highway at steady-state following the leader. The platoon leader keeps changing his speed, slowly accelerating and then decelerating, to mimic the usual small variations of a common driving pattern and also to create a situation where accidents can happen with higher probability due to continuous variations. A random platoon member starts misbehaving at a random time instant: The other vehicles should in principle detect the misbehavior and activate the emergency protocol. With Platoon Size equal to 4 and 8 the misbehaving vehicle is selected at random, but the last platoon member is never chosen, this because a PLOEG controlled vehicle listens only to messages sent by its predecessor, so wrong messages transmitted by the last platoon member would not be processed nor classified by any vehicle. With Platoon Size equal to 16 the misbehaving platoon member is chosen at random between the leader ($v0$) or the 8th platoon member ($v7$), to avoid exploring a too large combinatorial space. Overall, we run $(3+7+2) \times 2 \times 8 \times 100 = 19200$ different simulations, where $(3 + 7 + 2)$ is the number of potential misbehaving vehicles for the 3 tested values of platoon size, $\times 2$ for simulations either with or without enabling the emergency protocol, $\times 8$ represents the 8 simulated misbehaviors and finally $\times 100$ is the number of repetitions for pure statistical purposes.

The second goal stated is to assess the ability to avoid accidents, so we count and compare the number of car collisions over simulations where the emergency protocol is respectively enabled or disabled. However, this means that we test the online prediction accuracy of our MDS on a large number of "genuine" messages, i.e., those generated from the start of the simulation till the start of the misbehavior, but on a smaller number of misbehaving ones, as we stop predictions as soon as one message is classified as malicious.

We finally highlight how our testing scenario, involving platoons on a highway, is very different from the Luxembourg SUMO Traffic (LuST) scenario [25] used for training, so with our experiments we also want to test the generalization ability of our framework and its limits in transferring knowledge from an urban to an highway scenario.

B. Accuracy Evaluation

Tab. V presents the Confusion Matrix for the online MDS restricted to the binary *Genuine* VS *Misbehaving* classification problem. Rows report the true message type

	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	4,8,16
	Leader Speed	100 km/h
	Speed Oscillation Amplitude & Freq	5 km/h, 0.1 Hz
	Repetitions per Experiment	100
Comm.	L2-technology	dual radio 802.11p
	Tx power	500 mW
	Broadcast MCS	3 Mbit/s
	Unicast MCS	12 Mbit/s
	Rx sensitivity	−94 dBm

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

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

	Genuine	Misbehavior
Genuine	1.0000	0.0000
Misbehavior	0.0197	0.9803

Table V: Confusion Matrix for the online MDS restricted to the binary *Genuine* VS *Misbehaving* classification problem.

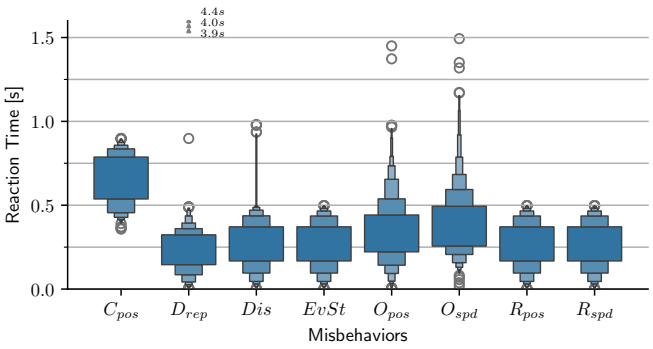

Figure 6: Discrete Violin Plots of the reaction-times observed over all the experiments. Separate Violins are plotted for each category of misbehavior. Three outliers for the D_{rep} attack are reported out of scale.

and columns indicate the estimated one, so the diagonal values represent correct hit-ratios, while the opposite elements represent miss-ratios. The first row of Tab. V highlights that, on the large number of simulated genuine messages, our MDS never reports false positives, hence, no spurious activation of the emergency protocol is ever triggered. We claim, accordingly, that our MDS is always able to preserve the driving advantages of platoons and never introduces needless overheads. The second row reveals instead that a minority of wrong messages (the 1.97%) remains undetected, which means platoons are protected in $\approx 98\%$ but not all of the cases. We further observe that all cases of undetected misbehaviors are due to the same class, namely, due to the *Data Reply* (D_{rep}) attack.

C. Analysis of the Reaction Time

The analysis of Tab. V establishes the perfect suppression of spurious emergencies and a great but not perfect detection accuracy of misbehaving messages. We now shift our interest towards a further key performance metric for an MDS evaluated at run-time, i.e., its reaction time. In fact, it is not enough for an MDS to be accurate, as a slow reaction time may allow the long persistence of misbehaviors leading to dangerous situations, up to fatal collisions.

Fig. 6 reports the distribution of reaction times observed over all the platoons we simulated afflicted by some misbehavior. The majority of misbehaviors is detected in around $0.25\,\mathrm{s}$, which means that the detector is able to catch the anomalies at the very first available sampling window. In fact, the MDS is invoked every 5 messages, which are generated every $0.1\,\mathrm{s}$ so, on average, an attack should starts $0.25\,\mathrm{s}$ before the first possible output of our classifier. However, all boxes extend more towards the upper side of the time-axis, suggesting that misbehaviors are not always detected at the very first sampling opportunity. In the VeReMi dataset

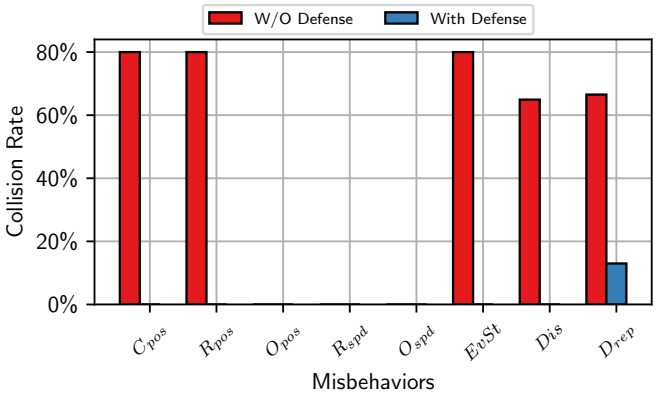

Figure 7: Barchart comparing the collision rate in simulations where the emergency protocol was either enabled (blue) or disabled (red).

vehicles are either misbehaving from the very beginning of time or they do not at all. This means that, while training, the MDS has never observed "partially wrong windows" of messages, i.e., windows that contain a few genuine messages followed by few wrong ones. This fact may biases the ability to detect misbehaviors immediately at run-time, because in our run-time experiments partially wrong windows can actually be sampled. Worth of notice are the outliers, especially visible for the D_{rep} attack, which require further analysis. In the following Sect. VI-D we direct the attention to the study of safety under the just commented distribution of reaction times.

D. Analysis of the MDS Protection Power

The average latency-penalty paid by our classifier seems to be low enough to protect the platoons in the greatest majority of cases; however, the outliers pointed out in Sect. VI-C remain a source of concern. Fig. 7 shows a comparison in terms of collision rate for simulations where the MDS warnings were either used to enable the emergency protocol (blue bars) or when the MDS is not used at all (red bars), thus vehicles use the wrong data. Some misbehaviors are naturally less dangerous than others, for example, even turning off all defense mechanisms the O_{pos}, R_{spd} or O_{spd} misbehaviours never lead to any collision, but this may change using different CACC algorithms or in heterogeneous conditions [28]. The defense mechanism turns out to be instead perfectly able to always detect and defuse the C_{pos}, R_{pos} and O_{pos} attacks, reducing the collision rate from 80% to 0%. Compared to these last three, the Dis and D_{rep} attacks are slightly less dangerous, leading to collision in approximately the 63% of the simulations without defense. The safety protocol fully defuses the Dis attacks and greatly reduces the collisions in case of D_{rep} ones however, unfortunately, the $\approx 15\%$ of D_{rep} attacks remain undetected or the reaction time is too slow and we observe a non zero number of collisions also when the defenses are enabled. We hence confirm the observations reported by Kamel et al [4], who already noticed in their benchmarking study the D_{rep} ability to trick detection systems.

VII. Discussion and Conclusions

This paper introduces a novel safety framework for vehicular networks that includes an AI-based MDSs and an emergency protocol to ensure the safe dismantling of a platoon afflicted by a malicious attack, or malfunctioning sensors or any kind of anomaly. The MDS is trained over the VeReMi dataset and its offline accuracy is assessed. However, compared to most current literature on the subject, we implement the MDS to be used online, on board of simulated vehicles, for run-time classification of messages. We propose a simple emergency protocol to complement the MDS and protect platoons from multiple misbehaviors. We show that the MDS used online is accurate and, above all, extremely responsive, with an average reaction time close to $0.25\,\mathrm{s}$. This good performance allow the timely activation of the emergency protocol which turns to be a crucial security mechanism. In fact, the simulation of platoons where a vehicle starts misbehaving exhibits an overall collision rate of 58%, which is reduced to only 2% once the defense mechanism is enabled.

The MDS is based on a supervised training, which means that the resulting classifier ability to detect novel and previously unseen misbehaviors is limited. Yet, restricting the classification to the binary *Genuine* vs *Misbehaving* problem, the MDS is able to defuse also unseen patterns. No system can be secured against 0-day vulnerabilities, but we think that such a simple MDS is prone to on-line continuous training. For instance one can introduce a reinforcement learning approach to improve the MDS accuracy at run-time, further enhancing the MDS ability of catching novel risks and thus protecting platoons and VNs.

We conclude stressing the necessity to test MDS systems also at run-time in conditions similar to those they may encounter in real traffic, and not only off-line on partitions of the training datasets, which are in general collected ensuring the stationary behavior of the system, thus simplifying the task for 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" and program "Security and Rights in CyberSpace (SERICS)," (PE00000014), Spoke 7, Project SCAR, Cascade Call "SCAR: a Privacy enHAnced SEcurity framework (SCARPHASE)," CUP C89J24000580008.

References

[1] M. Segata et al., "Multi-Technology Cooperative Driving: An Analysis Based on PLEXE," *IEEE Trans. on Mobile Comp. (TMC)*, Feb. 2022.

[2] F. Dressler et al., "V-Edge: Virtual Edge Computing as an Enabler for Novel Microservices and Cooperative Computing," *IEEE Network*, vol. 36, 3 May 2022.

[3] R. W. Van Der Heijden, T. Lukaseder, and F. Kargl, "VeReMi: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs," in *14th International Conference on Security and Privacy in Communication Networks (SecureComm), Springer LNICST 254*, Singapore, Aug. 2018.

[4] J. Kamel et al., "VeReMi Extension: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs," in *IEEE International Conference on Communications (ICC)*, Dublin, Ireland, Jun. 2020.

[5] M. Franceschini, *A neural-network based anomaly detection system and a safety protocol to protect vehicular network*, 2024. arXiv: 2411.07013. [Online]. Available: https://arxiv.org/abs/2411.07013.

[6] R. Lo Cigno and M. Segata, "Cooperative driving: A comprehensive perspective, the role of communications, and its potential development," *Elsevier Computer Communications*, vol. 193, Sep. 2022.

[7] S. Sharma and A. Kaul, "A survey on Intrusion Detection Systems and Honeypot based proactive security mechanisms in VANETs and VANET Cloud," *Vehicular Communications*, vol. 12, 2018.

[8] H. Bangui and B. Buhnova, "Recent advances in machine-learning driven intrusion detection in transportation: Survey," *Procedia Computer Science*, vol. 184, 2021.

[9] T. Zaidi and Syed.Faisal, "An Overview: Various Attacks in VANET," in *4th International Conference on Computing Communication and Automation (ICCCA)*, 2018.

[10] R. W. van der Heijden et al., "Survey on Misbehavior Detection in Cooperative Intelligent Transportation Systems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, 2019.

[11] A. Talpur and M. Gurusamy, "Machine Learning for Security in Vehicular Networks: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, 2022.

[12] X. Sun, F. R. Yu, and P. Zhang, "A Survey on Cyber-Security of Connected and Autonomous Vehicles (CAVs)," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, 2022.

[13] F. Tang et al., "Future Intelligent and Secure Vehicular Network Toward 6G: Machine-Learning Approaches," *Proceedings of the IEEE*, vol. 108, no. 2, 2020.

[14] A. Boualouache and T. Engel, "A Survey on Machine Learning-Based Misbehavior Detection Systems for 5G and Beyond Vehicular Networks," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, 2023.

[15] M. Matousek et al., "Robust Detection of Anomalous Driving Behavior," in *IEEE 87th Vehicular Technology Conference (VTC Spring)*, 2018.

[16] M. Matousek et al., "Detecting Anomalous Driving Behavior using Neural Networks," in *IEEE Intelligent Vehicles Symposium*, 2019.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, 1986.

[18] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation, MIT-Press*, vol. 9, 8 Nov. 1997.

[19] X. Liu, "Misbehavior Detection based on Deep Learning for VANETs," in *International Conference on Networks, Communications and Information Technology (CNCIT)*, 2022.

[20] T. Alladi et al., "A deep learning based misbehavior classification scheme for intrusion detection in cooperative intelligent transportation systems," *Digital Communications and Networks*, vol. 9, no. 5, 2023.

[21] J. Kamel et al., "Simulation Framework for Misbehavior Detection in Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, 2020.

[22] J. Kamel et al., "Misbehavior Detection in C-ITS: A comparative approach of local detection mechanisms," in *IEEE Vehicular Networking Conference (VNC)*, 2019.

[23] R. Sedar et al., "Reinforcement Learning Based Misbehavior Detection in Vehicular Networks," in *IEEE International Conference on Communications (ICC)*, 2022.

[24] A. Uprety, D. B. Rawat, and J. Li, "Privacy Preserving Misbehavior Detection in IoV Using Federated Machine Learning," in *18th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2021.

[25] L. Codecá et al., "Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 2, 2017.

[26] H.-Y. Hsu, N.-H. Cheng, and C.-W. Tsai, "A deep learning-based integrated algorithm for misbehavior detection system in VANETs," in *ACM International Conference on Intelligent Computing and its Emerging Applications*, Jinan, China, Dec. 2021.

[27] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, Jan. 2011.

[28] M. Segata, L. Ghiro, and R. Lo Cigno, "On the Progressive Introduction of Heterogeneous CACC Capabilities," in *13th IEEE Vehicular Networking Conference (VNC)*, Nov. 2021.