Misbehavior Detection in C-ITS: A comparative approach of local detection mechanisms

Joseph Kamel, Ines Ben Jemaa, Arnaud Kaiser, Loic Cantat, Pascal Urien

Misbehavior Detection in C-ITS: A comparative approach of local detection mechanisms

Joseph Kamel
IRT SystemX
Palaiseau, France
joseph.kamel@irt-systemx.fr

Ines Ben Jemaa
IRT SystemX
Palaiseau, France
ines.ben-jemaa@irt-systemx.fr

Arnaud Kaiser
IRT SystemX
Palaiseau, France
arnaud.kaiser@irt-systemx.fr

Loic Cantat
IRT SystemX
Palaiseau, France
loic.cantat@irt-systemx.fr

Pascal Urien
Telecom ParisTech
Paris, France
pascal.urien@telecom-paristech.fr

***Abstract*—MisBehavior Detection (MBD) is an important security mechanism in Cooperative Intelligent Transport Systems (C–ITS). It involves monitoring C–ITS communications to detect potentially misbehaving entities. This monitoring is based on local plausibility and consistency checks done by the Intelligent Transport Systems (ITS) Station (ITS–S) on every received Vehicle–to–Everything (V2X) message. These checks are then analyzed by a local detection mechanisms to estimate the overall plausibility of a message. In this paper we focus on the logic behind different local detection mechanisms. First, we propose different local detection solutions based on logics extracted from the state of the art. Then we present a comparative review of the detection quality and the computation latency of each proposed mechanisms.**

***Index Terms*—Misbehavior Detection, C–ITS, Machine Learning**

I. Introduction

According to the National Highway Traffic Safety Administration (NHTSA) and the Directorate-General for Mobility and Transport (DGMT), more than three million people are injured each year due to traffic accidents in the US and the EU [1] [2]. Cooperative Intelligent Transport Systems (C–ITS) is a technology that aims to reduce traffic accidents and improve road safety in general. The technology is based on the exchange of safety messages between different entities called ITS Stations (ITS–Ss). These Vehicle–to–Everything (V2X) messages could contain information about the vehicle (GPS position, Velocity, Heading, etc.) or various warning about the traffic condition. This information could be critical for the ITS–S to insure the safety of the passengers or surrounding pedestrians. The European C–ITS system standards are published by the European Telecommunications Standards Institute (ETSI) while the US standards are published by the Institute of Electrical and Electronics Engineers (IEEE).

V2X messages must be secure to ensure the reliable operation of C–ITS-based security applications. To this end, the Public Key Infrastructure (PKI) is created and standardized by the IEEE and the ETSI. The PKI issues digital certificates to the ITS–Ss. These certificates are used by theITS–S to sign each transmitted messages thus ensuring their authenticity, integrity and non repudiation. Additionally, the ITS–S are allowed to regularly change their certificates to avoid tracking and protect the vehicles privacy [3]. However, the digital signature could not ensure the accuracy and validity of a message. For instance, a malicious vehicle with a valid certificate could send inaccurate or false data on the C–ITS network. Consequently, a MisBehavior Detection (MBD) system is needed to protect the system and otherwise mitigate the effects of these malicious or otherwise faulty ITS–Ss.

In this paper, we focus our work at the local MBD. This detection is based on plausibility and consistency checks done by the ITS–S on every received V2X message called sensors. The results of these checks are analyzed by a local detection application to classify the received V2X message as misbehaving or genuine. Our goal is to evaluate different approaches for local MBD applications. The different published detection results are often difficult to compare since it's done on different data and with different implementations. To this end we implement the complete MBD process along with the different detection applications in an extension of Vehicles in Network Simulation (VEINS) simulator [4]. Our comparative results show a clear trade-off between the accuracy of the detection mechanisms and the calculation latency.

The remainder of the paper is as follows. Section II presents the state of the art regarding local MBD mechanisms. Section III presents the C-ITS general architecture and details the misbehavior detection concept. Section IV details the tested detection mechanisms. Then, section V presents our evaluations and discuss the obtained results. Finally, section VI concludes this work and gives some future work.

II. Related Work

MisBehavior Detection (MBD) is a well researched topic with studies spanning the last two decades. Van der Heijden et al. published a recent survey on the different MBD studies [5]. In this survey we can extract four families of detection mechanisms: data-centric, node-centric, cooperative and Machine Learning (ML)-based. A study could make use of a combination of mechanisms but every mechanism fits under one of

these families. Data-centric mechanisms that rely purely on the contents of the message to estimate its plausibility. Node-centric mechanisms assigns a trust value to every neighboring ITS–S. Cooperative mechanisms relies on information sharing between ITS–Ss nodes to detect implausibilities. Finally, ML based mechanisms that rely on adequately training ML models to detect anomalies.

Schmidt et al. propose VEhicle Behavior Analysis and Evaluation Scheme (VEBAS) [6]. VEBAS is based on a set of plausibility and consistency checks. These checks are devided into positive rative and negative rating modules. The modules are then combined using Exponentially Weighted Moving Average (EWMA) to evaluate the behavior of a vehicle. Bißmeyer et al. proposes a similar system to VEBAS while adding a plausibility model to check vehicle intersections [7]. This plausibility model has uncertainty calculation that accounts for sensor errors. A trust value is also calculated to support the plausibility module. Using the trust and the plausibility value a detection is then classed as benign, erroneous or unknown.

Leinmüller et al. propose a cooperative solution to defend against roadside attackers [8]. Every vehicle shares the result of its local checks on the network to enhance the collective detection of roadside attackers. Kerrache et al. introduced a novel Trust architecture for Vehicular Networks using the standardized messaging services of ETSI ITS (T-VNets) [9]. T-VNets is a more complex trust evaluation scheme that uses a large sets of detectors and integrate data-centric, event-based, watchdog and Road–Side Unit (RSU)-based trust mechanisms. The scheme also propose sharing the trust values between neighboring vehicles.

Van der Heijden et al. introduced Vehicular Reference Misbehavior Dataset (VeReMi) [10]. It is an MBD dataset created with the VEINS simulator and using the LuST network scenario. VeReMi contains four types of misbehavior: Fixed Position, Fixed Position Offset, Random Position, Random Position Offset and an Eventual stop. So el al. trained and tested multiple ML models using the VeReMi dataset [11]. The solution used plausibility checks as an input feature vector for the ML models. The study aimed to created a baseline ML solution and tested K-Nearest Neighbors (K-NN) and Support Vector Machine (SVM). Both algorithms performed similarly with SVM having a slight edge. Singh et al. proposed a similar solution on the VeReMi dataset [12]. The study tested SVM and Logistic regression with SVM as the better performer. Singh et al. also proposed a deep learning based solution that tested Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM) [13]. LSTM is the better performer although at the cost of more computational time.

III. System Model

A. C–ITS General Architecture

The C–ITS system depend on a set of V2X messages exchanged between On–Board Units (OBUs) and RSUs. These message contain safety information that could be could kinematic beacons (position, speed, heading, etc.) or warnings (road works, hazardous conditions, etc.). The V2X messages

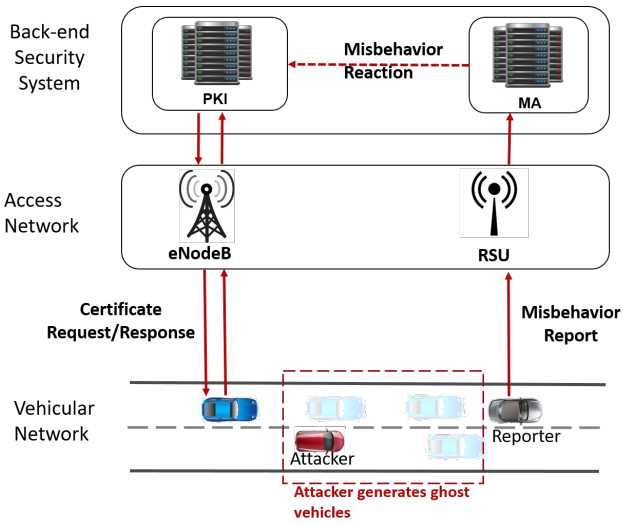


Fig. 1: C–ITS security architecture

are signed using digital certificates to ensure the ITS–S stations identity. The certificates are issued by the PKI. Each ITS–S receives one long term identity and several short term pseudonym identities. These short term identities are disposable certificates periodically changed to limit the track-ability of an ITS–S. The C–ITS security architecture is illustrated in Figure 1. The Figure shows the data exchange for a certificate request between the blue vehicle and the PKI through a cellular network (eNodeB).

A certificate signature ensure the authenticity of the data, i.e. the receiver could verify that the data is not altered during transmission. However, that does not ensure the integrity of the data with respect to the physical truth. An ITS–S with a valid certificate could send erroneous data on the vehicular network. This erroneous data could be a result of a faulty ITS–S node or a malicious attacker. This type of semantic attacks or misbehavior is treated by the Misbehavior Authority (MA). The local ITS–Ss performs MBD checks and send the detection results to the MA under the form of Misbehavior Reports (MBRs). The whole process of MBD is detailed in section III-B.

B. Misbehavior Detection Overview

The MBD process is divided into four steps (see Fig. 2).

a) Local Misbehavior detection: The local MBD is performed by every ITS–S (vehicle's OBUs and RSUs). The goal is to detect potentially misbehaving entities in the local vehicular network. Every received message should pass a set of plausibility and consistency checks (see section III-C). These checks are then analyzed by a local MBD application to decide on the need to send a Misbehavior Report (see section IV).

b) Misbehavior reporting: The reporting process begins as soon as the ITS–S detects an implausibility. The ITS–S then collects the evidence required to prove and recreate a misbehavior on the global level. After collecting enough evidence, the MBR is created and sent to the MA. Figure 2

shows this action performed by the grey vehicle. More details on the reporting protocol are available [14].

c) Global Misbehavior detection: The global MA has a role of collecting and analyzing the received MBRs. Using the MBRs the MA should be able to false positives and genuine reports. Moreover, if a misbehavior is detected, the MA should be able to determine what type of misbehavior. The severity and type of misbehavior determines the suitable reaction required to protect the system and mitigate the misbehavior effect.

d) Misbehavior reaction: Local misbehavior reaction is currently limited to the ITS–S discarding the malicious messages. Global reaction is more developed and starts once the MA is confident in a certain diagnosis. The MA notifies the corresponding authority of the reaction to be carried out. For example, the PKI is in charge of certificate revocations. AS shown in figure 2, if a certain a misbehavior requires certificate revocation, the PKI is informed. The misbehavior reaction is still not yet well defined by the stabilization organisms, thus other authorities may be in charge of misbehavior reaction in the future.

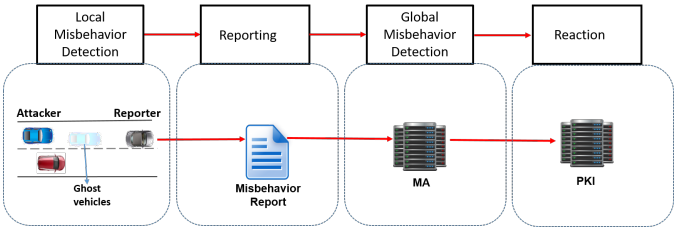


Fig. 2: Misbehavior detection steps

C. Local Detection Checks

The local detection process is based on checks performed by the ITS–Ss. Therefore, these checks should contain relevant and sufficient information for the detection process. In this study, we aggregated and implemented the checks used in the related works [6] [7] [9] [10] [11] [12] . However, the implemented checks does not return a binary value, instead a plausibility factor is calculated as described in [17]. For more details on the implementation of the detectors, all the implementations are open-source on github [18]. Table I shows a summary of all the selected local detectors.

D. Attacker Model

A misbehaving entity in C–ITS is any ITS–S sending inaccurate or fake V2X messages. Misbehaving entities could be divided into two categories: Faulty and Attackers. A Faulty behavior is any inaccurate V2X message data coming from a broken vehicle sensor. An attack is an intentional modification of the V2X message data. The implemented set of possible misbehavior types is inspired from the literature [5] [19] (see Table II). Please note that every new attacker sets the attack parameters randomly within a certain range. This is done to render the detection more difficult specifically for the ML based solution.

IV. Detection Mechanisms

In this part we extract the base logic behind of the state of the art solution described in Section II. We also detail our implementation of the extracted detection logics. We put forward the following solutions: (1) *Threshold Based:* a purely data-centric baseline solution, (2) *Non-Cooperative Trust Based:* a node-centric trust evaluation using data-centric mechanisms, (3) *Cooperative Trust Based:* a solution with a form of information sharing between the ITS–Ss, (4) *Machine Learning Based:* a use of some form of machine learning for misbehavior detection.

A. Threshold Based

This is our simple baseline application. The threshold based app consist of testing the result of every check against a

TABLE I: Description of Local Misbehavior Detection Checks

Range plausibility (rP)	The advertised ITS–S position is inside of the ITS–S maximum reception range
Position plausibility (pP)	The advertised ITS–S position is at a plausible location (e.g. on a road, without overlaps of physical obstacles, etc.)
Speed plausibility (sP)	The advertised ITS–S speed is less than a predefined maximum threshold
Position consistency (pC)	The distance separating two consecutive advertised ITS–S positions is less than a maximum threshold
Speed consistency (sC)	The acceleration separating two consecutive advertised ITS–S speeds is less than a maximum threshold
Position speed consistency (psC)	The distance separating two consecutive advertised ITS–S positions is consistent with the advertised speed
Position heading consistency (phC)	The angle separating two consecutive advertised ITS–S positions is consistent with the advertised heading.
Beacon frequency (bF)	The time separating two consecutive messages from the same ITS–S is compliant with the standards.
Intersection check (inT)	The advertised positions from two different ITS–Ss must not intersect.
Sudden appearance (sA)	A newly advertised ITS–S must not appear with a preset positive speed within a preset close range.
Kalman Filter Tracking	A set of checks derived from Kalman filter tracking [15]. The advertised beacon information must not diverge from the predicted information as proposed in [16]. From this calculation we extract the seven following detectors: - Kalman Position Speed Consistency *(kPSCP, kPSCS, kPSCSP, kPSCSS)*, - Kalman Position Consistency *(kPC)*, - Kalman Position Acceleration Consistency *(kPAC)*, - Kalman Speed Consistency *(kSC)*.

TABLE II: Description of C–ITS Misbehavior Types

Faulty Behaviors	
Fixed Position	The ITS–S broadcasts a faulty fixed position (X,Y)
Fixed Position Offset	The ITS–S broadcasts its real position with a fixed offset (ΔX,ΔY)
Random Position	The ITS–S broadcasts a faulty limited random position ($rand(X_{min} \mapsto X_{max}), rand(Y_{min} \mapsto Y_{max})$)
Random Position Offset	The ITS–S broadcasts its real position with a limited random offset ($\Delta(0 \mapsto X_{max})$,$\Delta(0 \mapsto Y_{max})$
Fixed Speed	The ITS–S broadcasts the same faulty fixed speed (V_x)
Fixed Speed Offset	The ITS–S broadcasts its speed with a fixed offset (ΔV_X)
Random Speed	The ITS–S broadcasts a faulty limited random speed ($\Delta(0 \mapsto V_{max})$)
Random Speed Offset	The ITS–S broadcasts its real speed with a limited random offset ($\Delta(0 \mapsto V_{max})$)
Delayed Messages	The ITS–S broadcasts its information delayed from reality (Δt)
Attacks	
DoS	The ITS–S sends V2X messages at a higher frequency than what is defined in the standard. The frequency increase inflicts an overhead on the broadcasting channel. This may render the channel unusable by other vehicles
DoS Random	The ITS–S performs a *DoS* attack while simultaneously randomizing all the V2X messages data
DoS Random Sybil	The ITS–S performs a *DoS Random* attack while simultaneously changing its pseudonym on each send V2X message
Disruptive	The ITS–S records the data broadcasted by neighbour ITS–Ss. The attacker then proceeds to broadcast V2X messages with data derived from previously received beacons. Given that the falsely broadcated data is initially generated by genuine vehicles, it is plausible on some levels. The intention of this attacker is to flood the network with these type of messages thus deteriorating the quality of the C–ITS
DoS Disruptive	This ITS–S performs a simultaneous *DoS* and *Disruptive* attacks
DoS Disruptive Sybil	This ITS–S performs a *Dos Disruptive* attack while simultaneously changing its pseudonym on each send V2X message
Data Replay	The ITS–S chooses a target and replays its data instantly with a certain minor prediction epsilons added. Consequently, for an observer it would seem that there are two vehicles in the same space-time dimension
Data Replay Sybil	This ITS–S performs a *Data Replay* attack while simultaneously changing its pseudonym while changing the target vehicle. This mechanism the ITS–S avoid detection
Eventual Stop	The ITS–S suddenly starts broadcasting a fixed positions and a null speed thus simulating a sudden stop
Traffic Congestion Sybil	The ITS–S uses the previously acquired and stored pseudonyms simultaneously to generate a set of ghost-vehicles. The ghost vehicles data is calculated somewhat intelligently in a grid like matter while avoiding implausibilities to simulate a realistic traffic congestion

predefined threshold. A message is considered misbehaving if any check fails the test (see algorithm 1).

Algorithm 1: Threshold Based Solution

```
c_x: Check Value, θ: Threshold;
for c_0 ... c_n do
    if c_i < c_min then
        c_min = c_i
    end
end
if c_min < θ then
    Misbehaving
else
    Genuine
end
```

B. Non-Cooperative Trust Based (N-CTB)

The goal of this solution is to evaluate the behavior of the node to determine a level of trust in the received V2X messages from a certain ITS–S. A similar approach to the logic used in [6] and in [7]. The trust is derived from the long-term trust level combined the current calculated plausibility factors. The trust has a negatively exponential relation with the plausibility factor (see equation 1 , figure 3). A message is considered misbehaving if the global trust level falls below a certain value (see algorithm 2).

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

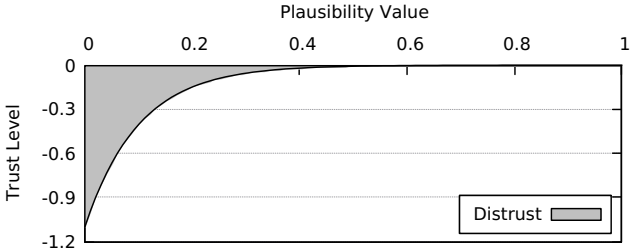


Fig. 3: The exponential variation of the trust level function of the plausibility value

Algorithm 2: Non-Cooperative Trust Based Solution

c_x: Check Value, θ: Threshold, T_L: Long-Term Trust;
for $c_0 \dots c_n$ **do**
  if $c_i < c_{min}$ **then**
    $c_{min} = c_i$
  end
end
$T_S = Trust(c_{min})$
if $T_S > -\varepsilon$ ***and*** $T_L < 0$ **then**
  $T_L = T_L + 0.1$
else
  $T_L = T_L + T_S$
end
if $T_L < \theta$ **then**
  Misbehaving
else
  Genuine
end

C. Cooperative Trust Based (CTB)

The goal of this solution is to cooperatively evaluating the behavior a node to determine a shared level of trust a certain ITS–S. Similarly to the approach used in [8] and in [9]. The trust is calculated identically to the case of *Non-Cooperative Trust Based*. However, the global trust levels are shared between all the ITS–Ss of the network.

D. Machine Learning (ML) Based

The goal of this solution is to train a ML algorithm to detect if a V2X message is misbehaving. Many ML algorithms exist for this purpose, however we revert to testing *SVM* [20], *XGBoost* [21], *MLP* [22] and *LSTM* [23].We detail below the models and parameters trained and tested in this study. All models hyper-parameters of the proposed model are tuned using a grid search based on 5-fold cross validation.

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

- **Checks Feature Set:** The local detection checks done on V2X messages described in section III-C.
- **Kinematic Feature Set:** The $Position$, $Speed$, $Accel$, $Heading$ and $Time$ of the last beacon. The $\Delta Position$, $\Delta Speed$, $\Delta Acceleration$, $\Delta Heading$ and $\Delta Time$ between the last 2 beacons.

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

2) Support Vector Machine (SVM): As proposed in [11] we use SVM as our baseline ML solution. Multiple implementation exist for SVM classification. The model is also trained with the *Checks Feature Set*. However, the default SVM implementation (C-Support Vector Classification (SVC)), is not designed for large data sets. The SVC training times exhibit quadratic growth with the increase of the number of samples. Therefore, we are able to train SVC with only 10% of our original training data-set. Accordingly, we tested the Linear Support Vector Classification (LinearSVC), a similar implementation that could scale better with a large numbers of samples. However, LinearSVC performed significantly worse than SVC even when trained on the full data-set.

3) Multi-Layer Perceptron (MLP): *MLP* is feedforward backpropagation Artificial Neural Network (ANN). It is the algorithm used in [13]. We tested two implementation for this model. The first implementation (MLP-T1) makes use of the same features *Checks Feature Set* as the previous models. The proposed MLP-T1 model has 1 Dense layer with 18 nodes. The second implementation (MLP-T10) takes as input the previous 10 time-steps. The feature set consists of the Minimum and the Average of the *Checks Feature Set*. The proposed MLP-T10 model has 1 Dense layer with 36 nodes.

4) Long Short-Term Memory (LSTM): *LSTM* is also an algorithm of choice used in [13]. Moreover, is a well suited algorithm for our problem due the temporal based relation between the successive V2X messages. LSTM is part of the Recurrent Neural Network (RNN) family of ML algorithms specifically designed to treat time dependent data. Therefore, the LSTM if additionally given the *Kinematic Feature Set* as input. The model proposed contains a single bidirectional LSTM layer with 20 nodes. A dropout of 0.2 and batch normalization is added to combat over-fitting. This is similarly validated through cross validation.

For more technical details, the implementation of these models is open-source and shared on GitHub [18].

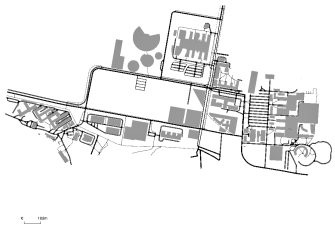
V. Evaluation Results

A. Simulation settings and scenarios

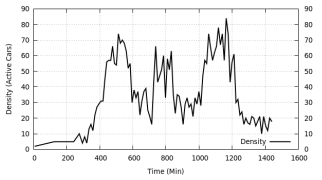
We use the Framework For Misbehavior Detection (F^2MD) to test the previously described solutions [4]. F^2MD is a VEINS module. VEINS is a well known an open source framework for running vehicular network simulations [24]. It is also the simulator of choice for VeReMi [10]. VEINS is based on the Objective Modular Network Testbed in C++ (OMNeT++) [25] for network simulation and the Simulation of Urban MObility (SUMO) [26] for road traffic simulation.

In order to correctly evaluate the ML algorithms, a different scenario is used for the training and the testing (see Fig. 4). The Luxembourg SUMO Traffic (LuST) scenario is used for training [27]. LuST is also the scenario of choice for VeReMi. It is a set of vehicle traces generated with SUMO and validated with real data. The chosen network size is $1.61km^2$ with

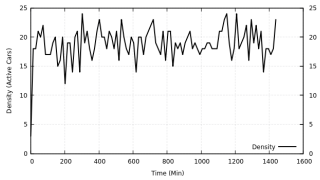
(a) Train Network: Luxembourg



(b) Test Network: Paris Scalay



(c) Train Vehicle Density



(d) Test Vehicle Density

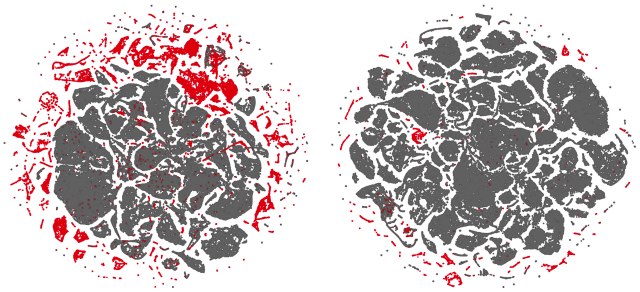
Fig. 4: Simulation Scenarios

a variable density peaking at $67.4 Vehicle/km^2$. In total, the train scenario contains $24,663$ vehicles with $17,097,930$ exchanged V2X messages with an attacker rate of 25%. Alternatively, The testing is done on an area of Paris Scalay with randomly generated vehicle traces. Therefore, the vehicle density is somewhat stable. The test bench network size is $1.11 km^2$ and of density circling around $17.1 Vehicle/km^2$. This selection of scenarios enables a significantly different training and testing data-sets. In total, the test bench contains $12,542$ vehicles with $8,475,371$ exchanged V2X messages with an attacker rate of 5%. For further details, the raw data, the source code and all the configuration details of the scenarios are published on GitHub [18].

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

B. Data Evaluation

Figure 5 shows a t-Distributed Stochastic Neighbor Embedding (t-SNE) visualization of the train and test data-sets. t-SNE is a dimensional reduction technique used to reduce multiple features into a two dimensional space [28]. We observe that the two classes are not perfectly separated into clusters. Some of the genuine and misbehaving data points are mixed. Therefore, we suspect that a linear model is not suitable for this classification. A non-linear kernel or a deep-learning model might perform better in our scenario.



(a) Train Data (b) Test Data

Fig. 5: t-SNE: Genuine (Gray) and Misbehaving (Red)

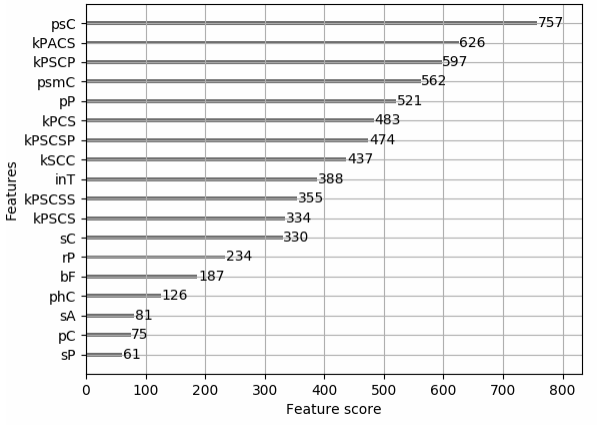


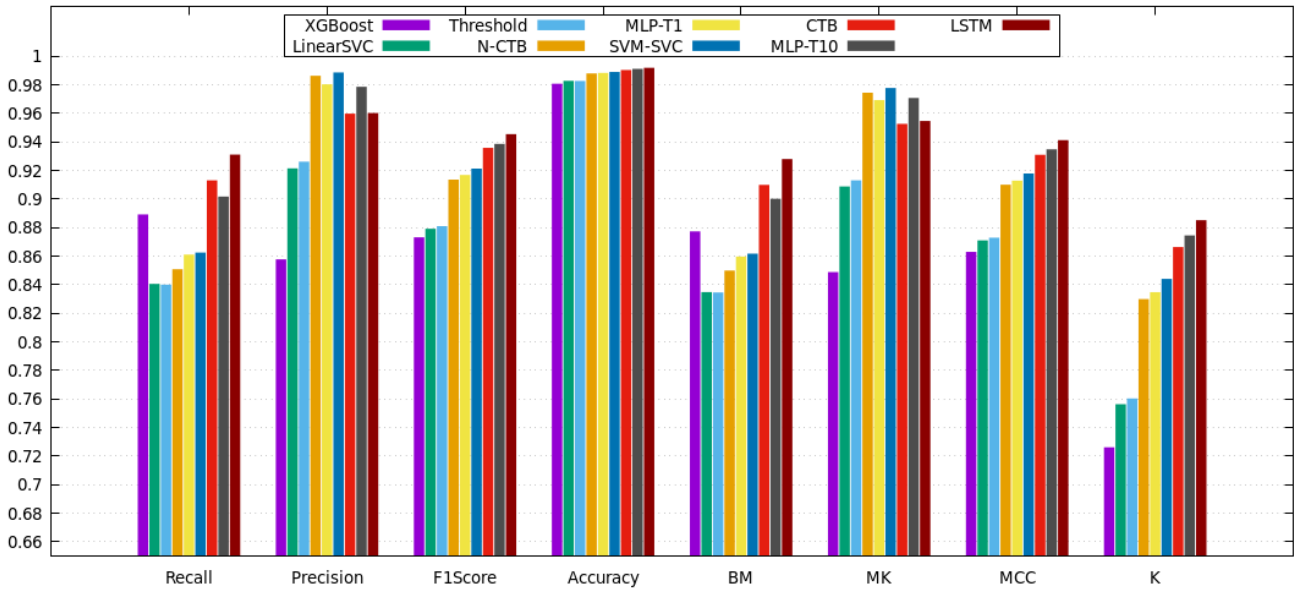
Fig. 6: XGBoost feature importance

Figure 6 shows the feature importance by weight as calculated by the XGBoost module. We can see that not all feature are equally important. A relative correlation is apparent between the complexity of the calculated check and the importance with respect to the XGBoost classification. Specifically, the Kalman-Filter based check is especially important. All the Kalman Filter extracted checks rank high on the feature importance scale. Future studies treating ML in local misbehavior detection should consider adding it to their feature sets. Ultimately, it is as important to consider the checks calculated as the ML algorithms used for the classification.

C. Results Analysis

Figure 7 shows the evaluation metric results of the models classification of the test data-set. The considered evaluation metrics are: *Recall*, *Precision*, $F_1 score$, *Accuracy*, *Bookmaker Informedness (BM)*, *Markedness (MK)*, *Matthews Correlation Coefficient (MCC)* and *Cohen's kappa* (κ). The evaluation metrics are detailed in out previous publication [17]. The Mean Processing Time (MPT) is also measured for every considered detection application.

First thing we notice is that all the detection mechanisms are within a small accuracy range. In fact, all the detection mechanisms score more that 98% accuracy. This is due to the mechanisms relatively high precision and the unbalanced test



Detection Solution	Evaluation Metrics								MPT (μs)
	Recall	Precision	F_1score	Accuracy	BM	MK	MCC	K	C++(∗) Python(○)
Threshold	0.8401	0.9262	0.8811	0.9827	0.8346	0.9131	0.8730	0.7604	0.00010 (∗)
N-CTB	0.8509	0.9864	0.9137	0.9880	0.8500	0.9745	0.9101	0.8300	0.00019 (∗)
CTB	0.9131	0.9599	0.9359	0.9904	0.9100	0.9527	0.9311	0.8664	0.00023 (∗)
XGBoost	0.8892	0.8578	0.8732	0.9808	0.8774	0.8489	0.8630	0.7262	745.01 (○)
LinearSVC	0.8406	0.9216	0.8792	0.9828	0.8349	0.9088	0.8711	0.7564	7089.6 (○)
SVM-SVC	0.8626	0.9887	0.9213	0.9891	0.8618	0.9778	0.9180	0.8441	115.14 (○)
MLP-T1	0.8611	0.9804	0.9169	0.9884	0.8598	0.9694	0.9129	0.8348	1358.6 (○)
MLP-T10	0.9018	0.9787	0.9387	0.9912	0.9002	0.9708	0.9349	0.8746	0.176 (∗) 1562.6 (○)
LSTM	0.9312	0.9603	0.9455	0.9920	0.9281	0.9547	0.9413	0.8852	0.614 (∗) 8298.9 (○)

Fig. 7: Evaluation Metrics by tested detection application

data-set of 5% attacker rate. Consequently, accuracy is not a suitable detection metric for our use case. For comparison purposes we rely on *Cohen's kappa*, *MCC* or the F_1*score* which all have the same ranking for the considered mechanisms.

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

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

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

We believe that a robust set of well calibrated detectors

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

VI. Conclusion and Future Work

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

Future work involves focusing on other components of the MBD system. That includes a more efficient reporting protocol and a robust global detection. Additionally, testing of MBD system within current C–ITS deployment projects is planned.

Acknowledgment

This research work has been carried out in the framework of the Technological Research Institute SystemX, and therefore granted with public funds within the scope of the French Program *Investissements d'avenir*.

References

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

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

[3] J. Petit, F. Schaub, M. Feiri, and F. Kargl, "Pseudonym schemes in vehicular networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 228–255, Firstquarter 2015.

[4] Framework For Misbehavior Detection (F^2MD). (2019) F^2MD website. [Online]. Available: https://www.irt-systemx.fr/f2md

[5] R. W. van der Heijden, S. Dietzel, T. Leinmüller, and F. Kargl, "Survey on misbehavior detection in cooperative intelligent transportation systems," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 779–811, Firstquarter 2019.

[6] R. K. Schmidt, T. Leinmüller, E. Schoch, A. Held, , and G. Schaefer, "Vehicle behavior analysis to enhance security in vanets," in *V2VCOM 2008*, 2008, pp. 1–8.

[7] N. Bißmeyer, C. Stresing, and K. M. Bayarou, "Intrusion detection in vanets through verification of vehicle movement data," in *2010 IEEE Vehicular Networking Conference*, Dec 2010, pp. 166–173.

[8] T. Leinmüller, R. K. Schmidt, and A. Held, "Cooperative position verification - defending against roadside attackers 2.0," in *Proceedings of 17th ITS World Congress*, 2010.

[9] C. A. Kerrache, N. Lagraa, C. T. Calafate, J.-C. Cano, and P. Manzoni, "T-vnets: A novel trust architecture for vehicular networks using the standardized messaging services of etsi its," *Computer Communications*, vol. 93, pp. 68 – 83, 2016, multi-radio, Multi-technology, Multi-system Vehicular Communications. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366416302213

[10] R. W. van der Heijden, T. Lukaseder, and F. Kargl, "Veremi: A dataset for comparable evaluation of misbehavior detection in vanets," in *Security and Privacy in Communication Networks*, R. Beyah, B. Chang, Y. Li, and S. Zhu, Eds. Cham: Springer International Publishing, 2018, pp. 318–337.

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

[12] P. K. Singh, S. Gupta, R. Vashistha, S. K. Nandi, and S. Nandi, "Machine learning based approach to detect position falsification attack in vanets," in *Security and Privacy*, S. Nandi, D. Jinwala, V. Singh, V. Laxmi, M. S. Gaur, and P. Faruki, Eds. Singapore: Springer Singapore, 2019, pp. 166–178.

[13] P. K. Singh, M. K. Dash, P. Mittal, S. K. Nandi, and S. Nandi, "Misbehavior detection in c-its using deep learning approach," in *Intelligent Systems Design and Applications*, A. Abraham, A. K. Cherukuri, P. Melin, and N. Gandhi, Eds. Cham: Springer International Publishing, 2020, pp. 641–652.

[14] J. Kamel, I. Ben Jemaa, A. Kaiser, and P. Urien, "Misbehavior reporting protocol for c-its," in *2018 IEEE Vehicular Networking Conference (VNC)*, Dec 2018, pp. 1–4.

[15] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, p. 35, 1960.

[16] A. Jaeger, N. Bißmeyer, H. Stübing, and S. A. Huss, "A novel framework for efficient mobility data verification in vehicular ad-hoc networks," *International Journal of Intelligent Transportation Systems Research*, vol. 10, no. 1, pp. 11–21, Jan 2012.

[17] J. Kamel, A. Kaiser, I. Ben Jemaa, P. Cincilla, and P. Urien, "CaTch: a confidence range tolerant misbehavior detection approach," in *2019 IEEE Wireless Communications and Networking Conference (WCNC) (IEEE WCNC 2019)*, Marrakech, Morocco, Apr. 2019.

[18] J. Kamel, "Github repository: Framework for misbehavior detection (f^2md)," 2019. [Online]. Available: https://github.com/josephkamel/f2md

[19] J. Grover, M. S. Gaur, and V. Laxmi, "Position forging attacks in vehicular ad hoc networks: Implementation, impact and detection," in *2011 7th International Wireless Communications and Mobile Computing Conference*, July 2011, pp. 701–706.

[20] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92. New York, NY, USA: ACM, 1992, pp. 144–152. [Online]. Available: http://doi.acm.org/10.1145/130385.130401

[21] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

[22] C. Van Der Malsburg, "Frank rosenblatt: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms," in *Brain Theory*, G. Palm and A. Aertsen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 245–248.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] 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, pp. 3–15, Jan 2011.

[25] A. Varga, "The omnet++ discrete event simulation system," in *In ESM'01*, 2001.

[26] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012.

[27] L. Codeca, R. Frank, and T. Engel, "Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research," in *IEEE Vehicular Networking Conference (VNC)*, Dec 2015, pp. 1–8.

[28] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, 2008.