Vehicular Network Anomaly Detection based on 2-Step Deep Learning Framework

Nur Cahyono Kushardianto[a,c,*], Soheyb Ribouh[a,*], Yassin El Hillali[a,*] and Charles Tatkeu[b,*]

[a] *Univ. Polytechnique Hauts-de-France, CNRS, Univ. Lille, ISEN, Centrale Lille, UMR 8520,IEMN,Institut d'Électronique de Microélectronique et de Nanotechnologie, DOAE,Département d'Opto-Acousto-Électronique., Valenciennes, F-59313, France*

[b] *Univ Gustave Eiffel, COSYS-LEOST, Villeneuve d'Ascq, F-59650, France*

[c] *Network and Hardware Technology Laboratory, Politeknik Negeri Batam., Batam, 29461, Indonesia*

ARTICLE INFO

Keywords:
V2V communications
Cyber-attack detection
IDS - Intrusion Detection System
2-Step prediction
Deep learning

ABSTRACT

Intelligent Transportation System (ITS) is one of the newest technologies in the transportation sector that will give hope for better driving safety. Not only in terms of driving safety, but ITS will give also hope for driving comfort. Smart vehicles perchance better versatile to the street circumstances through trade data among vehicles. In case, they can maintain a strategic distance from activity blockage, perilous deterrents, or see activity mishaps prior. The innovation which is meticulously associated with the security of the driver must get extraordinary consideration. V2V-Vehicle-to-Vehicle connection can undermine impedance and indeed attack or anomaly. Many studies have been carried out to address this problem. The primary step is to reinforce the system's capacity to identify anomalies on Vehicular Network. Further, the growing development of machine learning seems to bring hope to support these steps. Within the proposed method, the original of our approach consists in utilizing 2-Step of anomaly detection. This framework is utilizing two classifiers machine learning from two altered preparing data-sets. We appear that the proposed method can make strides essentially attack detection achievement, compared to arrangements depending on a single detection step.

1. Introduction

Intelligent Transport System (ITS) has been established as a leading actor in the next generation of the transportation system by promising more road safety better traffic congestion and comfortable driving [1]. Although ITS was highly supported to be widely implemented by many countries such as the USA, Japan, and parts of Europe, meanwhile these countries have begun to use it in order to overcome traffic congestion problems in crowded cities [2]. ITS utilizes several technologies such as activators, sensory, databases and information, microelectronics, expert system fundamentals, and all of them will interact through the network infrastructure. Vehicles equipped with ITS will become more sophisticated and functional vehicles. This vehicle will be equipped with sensors that can take into account internal and external factors. The devices of artificial intelligence and processors will make vehicles able to process and store information and plan actions. Vehicles will be able to make their own decisions, learn on their own and take appropriate actions [3]. Vehicle in Intelligent Transport System will be equipped with advanced network infrastructure which ensures the interaction between internal and external elements running safe and efficient. To change its state, a smart vehicle is using an actuator and various signals that affected its environment, so it can adapt with an external command of the human being or even autonomously [3].

*Corresponding author:
anung@polibatam.ac.id (N.C. Kushardianto); soheyb.ribouh@uphf.fr (S. Ribouh); yassin.elhillali@uphf.fr (Y.E. Hillali); charles.tatkeu@univ-eiffel.fr (C. Tatkeu)
ORCID(s):

Due to the sensitivity of the exchanged information in the vehicular network, security and privacy are of paramount importance. There must be a consideration for possible security risks associated with a framework that is meticulously relevant to personal security. As a consequence, all Intelligent Transport Systems must adhere to data integrity, authenticity, confidentiality, and non-repudiation. Any cyber-attacks commit pose a menace to ITS, e.g. Denial of Service attacks where the network is flooded with fake messages until the legitimate users can no longer connect to the network. Therefore, attackers can forge traffic state information, which can put road users in danger. Moreover, the personal information of legitimate users can be vulnerable to theft by these malicious users.

Mitigating cyber-attacks can be a cumbersome process. All possible attacks must be accurately detected. Intrusion Detection System (IDS) perform an essential part in this matter. Any common IDS can detect the attack but with the increasing network structure and its complexity, it becomes difficult for IDS to identify all types of attacks. In addition, cyber-attacks on *Vehicle Network* are increasing and the vehicular network becomes more vulnerable to these attacks [4]. Detection and identification of attacks type is a challenging task in vehicular communication. In this context, Artificial Intelligence (AI) and deep learning have seen a prevalent usage as the main actor toward the development of IDS, where various deep learning based-IDS methods have been introduced, like Artificial Neural Networks (ANN), Genetic Algorithm (GA), Decision trees, and so on. Although the ANN is commonly utilized for IDS, it has limits in terms of training time [5]. As a result, IDS has been implemented using alternative deep learning architectures such as LSTM, GRU, and CNN. These methods show their perfection over the classical rule-based algorithms. To reinstate level 1 security experts, deep learning approaches are used in cyber disclosure systems.

Using the most recent data-set version of *Vehicular Reference Misbehavior (VeReMi)*, we constructed an established deep learning V2V attack detection system in this study. The data-set contains 19 different forms of attacks [6], compared to only 5 in the previous edition [7]. We suggest an innovative strategy in this scheme, which entails employing a two-step prediction method to recognize all forms of attacks simultaneously. This method was developed as a result of earlier research [8]. Deep Belief Network (DBN), Long Short Term Memory (LSTM), Gate Recurrent Units (GRU), Random Forest (RF) machine learning models are all subjected to this two-step prediction technique. We compare the achievement of traditional single-step prediction with our proposed two-step prediction for each model. We measure the recognition timing for each simulation and implement it in real-time simulation scenarios in addition to analyzing prediction/recognition accuracy.

2. Related Works

Han et al in [9] choose survival analysis to identify infiltration on V2V networks. To detect the emergence of suspicious information patterns, they deployed an irregularity disclosure method. Without having to update attack pattern recognition, this approach may detect unknown intrusions. Using a mechanically basic IDS, the data contained in the CAN bus (Controller Area Network) is detected. Although the proposed system does not handle post-attack diagnostics, it can only provide early warning to drivers in case of an attack. This research has focused on three attack scenarios: flooding attack, fuzzy attack, and malfunction attack.

Rasheed et al. [10] pay more attention to attacks that involve the injection of fake data on vehicles on a V2V network which will result in a vehicle mistaking its distance from another vehicle. Based on the LSTM-Generative Adversarial Network, they propose a Deep Neural Network (DNN) attack detection approach. They name it the New Deep Reinforcement Learning algorithm structure, which would result in a safe dynamic system for Autonomous Vehicle (AVh) control. The focus of this system is on superior autonomous vehicle control, which allows it to keep a safe distance from other vehicles while regulating its speed. AVh sensor data and AVh beacon signals are the most significant infrastructure requirements

The SerIoT project is one of the projects that focus on C-ITS communication security. The project aims to protect the main network on IoT devices, provide solutions to detect misbehavior, mitigate them through the creation of alternative routes involving specialized devices such as honeypots, and reduce the impact of automated attacks on autonomous vehicles [11]. In this study Hidalgo et al. evaluated the system using a real vehicle in the Tecnalia Lab, to facilitate experiments and obtain a realistic simulation environment. They show that the system can detect and mitigate misbehavior quite quickly. The detection system used is based on a Graph Neural Network consisting of Multi-Layer Perceptrons and Node Deep Neural Networks. This research takes into account only one type of attack which is the DoS attack. From the experimental results, the system can accurately detect and deliver early warnings of DoS attacks at an average time of 3.27 seconds and a standard deviation of fewer than 3 seconds.

Another study by Zhou et al. focused on security in platoon vehicle communication systems, which are inherently vulnerable to cyber-attacks. In this research, they offer a new detection system using invariant set state using physical properties model and system control strategy [12]. The use of Software-Defined Networking (SDN) in the platoon vehicle communication network is the main approach. The types of attacks that were considered in this research were Message Falsification Attacks (MFAs) and sensor spoofing attacks that were simulated using SUMO and OMNET++ applications. In this work, two detection methods were compared: (i) attributed to the set of constant states from us-(ISWSM) that adopted the Minkwoski sum method, (ii) SMF-based attack detection (IEDCM).

In addition to VEINS (Vehicle in Network Simulation), Framework For Misbehavior Detection (F^2MD) is one of the widely used applications for vehicular networks simulation [13]. Kamel et al designed it to simulate real-time anomaly detection in vehicular networks. The F^2MD architecture consists of 5 main functions level: *input data, local detection, local visual output, report data output, and global detection* [14]. This framework uses the LuST SUMO network as a real traffic scenario and also OMNET++ for simulating involving parameters beacons. F^2MD also has machine learning modeling which consists of an offline state and an online state. The offline state is used at the training model stage, while the online state runs on the HTTP Server by calling the machine learning model classifier resulting from the offline state to be used as a misbehavior detector. The machine learning classifier models used are SVM, MLP, and LSTM. The training task of these machine learning classifiers was achieved using the VeReMI data-set in its initial form [7], which includes 5 types of attacks. In this framework, the machine learning model that provides the best accuracy is LSTM. The focus of this study was on the framework's functionality rather than the performance evaluation of machine learning models.

Gyawali et al suggested an established misbehavior detection framework formulated on a hybrid collaborative machine learning and reputation misbehavior disclosure methods [15]. In their research, they developed a data-set based on practical vehicular network circumstances to test false alerts and position falsification and then evaluate it using various machine learning techniques. K-NN, Logistic Regression Model, Decision Tree Classifier, Bootstrap Aggregation, and Random Forest were the models they used. Bootstrap Aggregation and Random Forest provided the greatest outcomes concerning *Precision*, *Recall*, and F_1*-Score* based on their simulation results. They employ the initial version of the VeReMi in addition to the data-set [7] as a benchmark for location verification systems. They find that their technique was better than the VeReMi data-set for 30% attacker density for Eventual Stop, Random, Random Offset, and Constant Position forms of attacks. The simulation was run using the VEINS 4.7 framework, which includes SUMO with the LuST scenario and OMNET++.

D2H-IDS, is a hybrid attack/misbehavior detection system proposed by Aloqaily et al. [16]. It is based on the use of DBN and Iterative Dichotomiser 3 Decision Tree. The ID3 feature was used to select and attack categorization, while the Deep-Belief function was utilized to reduce data dimensionality. They illustrated the efficiency of their approach through ten simulations based on genuine cyber-security attack situations. In this study, 3 different DBN strategies were compared: DBN1-IDS, which is a standard of DBN, DBN2-IDS, that was developed and adapted from [17], and DBN3-IDS, which is sourced from [5]. The normal vehicle movement data was generated using the NS-3 application and NSL-KDD attack data-set during the preprocessing step. Then The data was passed through DBN IDS and ID3 algorithms using Matlab. The built data-set includes Remote to User, DoS, User to Root, and Probe attacks.

3. System Model

In the current work, we consider a wireless communication Vehicular Network (VN) (Figure 1), where each mobile node (vehicle) is assembled with an On-Board Unit (OBU) to exchange messages with neighbors mobile nodes or with fixed nodes (Road Side Unit (RSU)). The sent messages are in the form of a **Basic Safety Message (BSM)**, which contains information on the vehicle ID, position, speed, acceleration, direction, and so on.

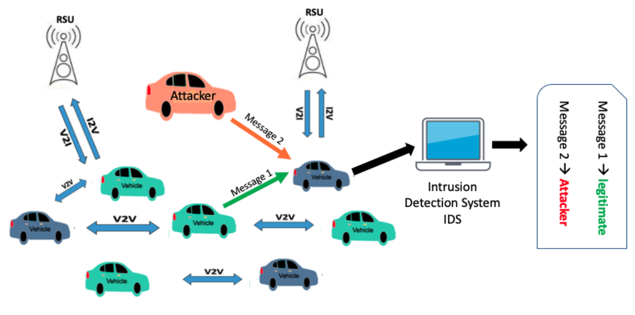

Figure 1: System Model

Regarding the sensitivity of the broadcast messages in the VN, securing the V2X communications is mandatory. So that the surrounding vehicles have to be able to figure out of the received message is legitimate or malicious. In vehicular communications, two types of attackers can be considered: internal attackers, and external attackers such as vehicles or users that do not have credentials in the VN (Figure 2). These Attackers can be defeated by Public Key Infrastructure (PKI) validity. The use of PKI will increase trust between communicating vehicles [18], [19], [20]. Meanwhile, internal attackers are the vehicle or user that already has credentials access to the VN. This type of attackers can be overcome by misbehavior detection, as mention in [14], [21], [22]. In the work present in this paper, we mainly focus on how to identify these internal attacks.

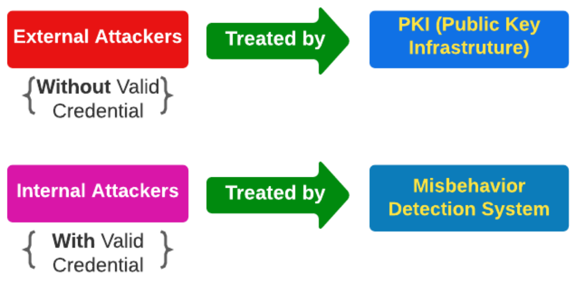

Figure 2: Type of Attacker in VN

3.1. Misbehavior and Attacker Model

Intrusion in vehicle-to-vehicle communication can be classified into misbehavior and attacker. Misbehavior can be caused by equipment damage on the vehicle such as sensors, OBU, etc [23]. So that the message sent by the vehicle becomes incorrect. On the other hand, an attack is the intentional act of an attacker to manipulate the exchanged messages in VN [24]. In our work we use the VeReMi data-set introduced in [6]. In this data-set, anomalies are divided into misbehavior and attacker, as shown in the table 1. In training and detection tasks, misbehavior is considered as an attacker, since the disruption they cause is almost similar.

Table 1: Anomaly List in V2V Communication [8]

Misbehavior	Attacker
• Constant Position	• Data Replay
• Constant Position Offset	• Data Replay Sybil
• Constant Speed	• Disruptive
• Constant Speed Offset	• Denial of Service (DoS)
• Delayed Messages	• DoS Disruptive
• Random Position	• DoS Disruptive Sybil
• Random Position Offset	• DoS Random
• Random Speed	• DoS Random Sybil
• Random Speed Offset	• Eventual Stop
	• Grid Sybil

4. Prediction System

In order to determine if the income messages are from a legitimate vehicle or an attacking vehicle, we have placed two stages classification system (Figure 3). At the first stage, a 0-1 classifier is used in order to identify the vehicle behavior (legitimate/attacker).

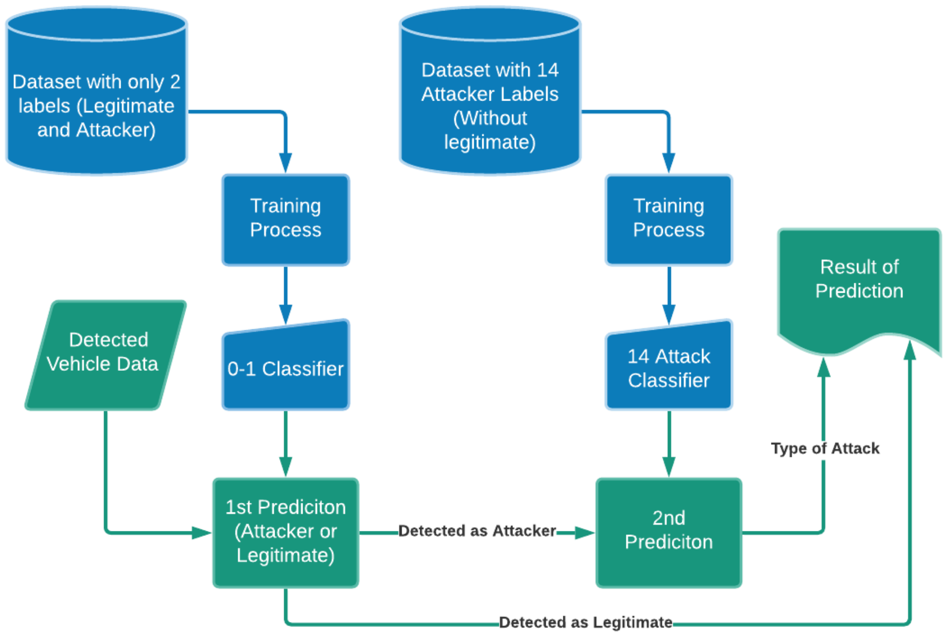

Figure 3: Data pre-processing and Prediction system Scheme [8]

The output result of this classifier is used to trigger a second stage classification sub-system, in case the vehicle is suspected of being an attacker. Thus, the data is fed into the second predictor, which can figure out the type of attack. This second classifier can recognize a variety of VN Attacks.

4.1. Data-set

VeReMi data-sets have been used to train our classification model. The purpose of this data-set is to test misbehavior detection systems for VANETs. The initial data-set consists of several simple attacks. The goal of this data-set release is to serve as a starting point for more complicated attacks as well as a benchmark for comparing detection techniques. Rens van der Heijden of the Institute of Distributed Systems at Ulm University, Germany, mostly compiled the data-set in 2018 [7]. In 2020 Joseph Kamel et al. developed the VeReMi data-set to become a *VeReMi Extension* [6] which is referred also as VeReMi, from now to the rest of paper.

VeReMi is a simulated data-set, generated using F^{2}MD with a subsection of the Luxembourg SUMO Traffic (LuST) network with a size of 1.61 km^2 and a peak density of 67 Veh/km^2 [14].

The VeReMi data-set is organized by attack type and time span. There are 19 types of attacks previously mentioned (table 1). There are two-time spans: rush hour between 07.00 and 09.00 and low traffic between 14.00 and 16.00. Based on this condition, there are a total of 38 data groups. One data set comprises message logs of each simulated vehicle in a one-time span.

VeReMi data-set contains messages logs per vehicle, organized as follows:

- GPS data of the local vehicle (labeled as type=2).
- Basic Safety Messages from other vehicles through Dedicated Short Range Communication (labeled as type=3).
- Messages labeled as type=4 are basically the same as messages type=3 but are collected in the ground truth file.
- Each message contains four primary data fields [8]:
    1. Position
    2. Speed/Velocity
    3. Acceleration
    4. Heading

4.2. Data-set Pre-processing

In our work, we use the messages history where several aggregated messages are used as the input feature of our model. Only two types of data fields are chosen at this point: *position* and *velocity*. As a result, we create a data-set that includes the vehicle's position and velocity progression. The fewer data fields used in the process, it is expected the less time is needed for the training and detection process. The data are organized as shown in the table 2 :

- **id: x**: is the vehicle sender number identification
- **type: y**: indicates the type of the vehicle, and refers to 0, it means that this is a legitimate vehicle (non-attacker). Howerever if value equals to 1, it means that this is an attacker vehicle.

Table 2
Format of a Vehicle Data [8]

id	type	m_0	m_i	m_n
x	y	$pos/0_0$	$pos/0_i$	$pos/0_n$
		$pos/1_0$	$pos/1_i$	$pos/1_n$
		$spd/0_0$	$spd/0_i$	$spd/0_n$
		$spd/1_0$	$spd/1_i$	$spd/1_n$

- m_0, m_i, m_n: refers respectively to 1st message, ith message, and nth message. The value $(n-1)$ indicate the number of the aggregated messages used.
  - **pos**: vehicle position in accord with GPS coordinate x (pos/0) and y (pos/1)
  - **spd**: vehicle speed in accord with speed vector x (spd/0) and speed vector y (spd/1) in meter/second

4.2.1. Parsing Data-set for Training

The raw data-set will first be converted into the same format as table 2, through the steps shown in diagram 4. The parsing process of this data-set begins by creating a data list containing the id of each vehicle and the attack code (see algorithm 1). Then the data-set will be processed by utilizing the attacker's list so that it becomes a data-set that is ready for the training process (see algorithm 2). At the end of this process, we get a labeled data-set.

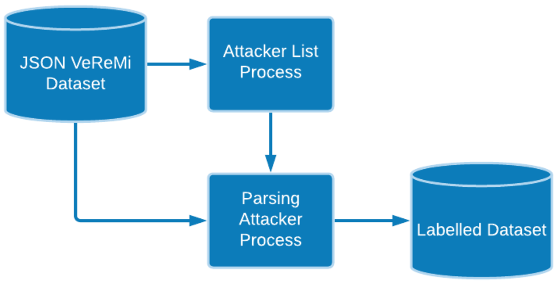

Figure 4: Parsing Data-set

4.2.2. Training Data-set Representation

For data-set training task, we considered different configurations, depending on the group numbers of the aggregated messages used in the input features of the machine learning model. Thus we consider the following cases: 5 aggregated messages (5_{msg}), 10 aggregated messages (10_{msg}), 15 aggregated messages (15_{msg}), 20 aggregated messages (20_{msg}), 25 aggregated messages (25_{msg}), and 30 aggregated messages (30_{msg}) organized as shown in table 2.

After several training processes, it was found that several types of attacks have the same characteristics. Especially the group of Constant Speed, Data Replay, Disruptive, and Dos Random. As a result, if some attacks are gathered together into a small group, it will be easier to identify

Algorithm 1 Attacker List Process

```
 1: labelsAttacks = [1,2,3,4,5,..., 19]
 2: for attackType ∈ labelsAttacks do
 3:     for data ∈ veremidata-set do
 4:         attackInfo ← data
 5:         BSM ← data
 6:     end for
 7:     vehicleList ← join(attackInfo, BSM)
 8:     for vehicle ∈ vehicleList do
 9:         attackerId ← id number of attacker ∈ vehicle
10:     end for
11:     attackerList = openWrite(file)
12:     attackerList.write ← attackerId
13:     attackerList.close()
14: end for
```

Algorithm 2 Parsing Attacker Process

```
 1: function CONVERTDATA
 2:     Pass in: vehicle, nMessage
 3:     attackerData = openRead(attackerList)
 4:     attackerId,labelAttack ← attackerData.read()
 5:     attackerData.close()
 6:     senderVeh ← data vehicle sender ∈ Vehicle
 7:     for $x$ = 0 to length(senderVehicle) do
 8:         if length(senderVehicle[$x$]) ≥ nMessage then
 9:             if senderVehicle[$x$].id ∈ attackerId then
10:                 typeAttack ← labelAttack
11:             else
12:                 typeAttack ← 0
13:             end if
14:             data.attackId ← typeAttack
15:             data.senderId ← senderVeh[$x$].id
16:             data.history.pos ← senderVeh[$x$].position
17:             data.history.spd← senderVeh[$x$].speed
18:         end if
19:     end for
20:     Pass out : data
21: end function
22: labelsAttacks = [1,2,3,4,5,..., 19]
23: nMessage = 5, 10, 15, 20, 25, 30
24: for attackType ∈ labelsAttacks do
25:     vehicleList ← vehicle trace data∈ attackType
26:     for vehicle ∈ vehicleList do
27:         input ← vehicle,nMessage
28:         vehicelBSM = CONVERTDATA ← input
29:     end for
30: end for
31: vehicleBSM = openWrite(file)
32: vehicleBSM.write ← vehicleBSM
33: vehicleBSM.close()
```

them. Regrouping these types of attack groups, resulting in **14 types of attacks**. The new anomaly list that will be implemented in the training process is the following [8]:

1. Constant Position
2. Constant Position Offset
3. *Constant Speed, Constant Speed Offset*
4. *Data Replay, Data Replay Sybil*
5. Delayed Messages
6. *Disruptive, DoS Disruptive, DoS Disruptive Sybil*
7. DoS
8. *DoS Random, DoS Random Sybil*
9. Eventual Stop
10. Grid Sybil
11. Random Position
12. Random Position Offset
13. Random Speed
14. Random Speed Offset

4.3. Machine Learning Prediction System

The selected machine learning models that we propose, has been chosen by referring to previous studies that have discussed the best machine learning methods to be used for cyber-attacks detection on V2V networks. Therefore, we chose to implement our method using the following machine learning model [8]:

1. DBN [25]
2. LSTM [26]
3. GRU [27] [28]
4. RF [29]

The process of determining the hyper-parameters is time-consuming before the actual training process. We use an open-source Bayesian Optimization i.e HyperOpt-Sklearn program [30]. This application is a function from the scikit-learn libraries [31].

4.4. Two-Step Prediction System

A Two-Step (2-step) prediction system involves two different classifiers. Initially, two distinct training methods with two distinct label data-sets are used. The procedure for creating these two data-sets is determined by the method depicted in figure 4. The first data-set consists of two labels vehicle i.e attacker and legitimate. The training task of this data-set produces a classifier we call the "0-1 classifier". The second data-set consists of 14 classes referred to the attack's type aforementioned, without including legitimate vehicles. The second data-set will produce a classifier we refer to as the "14 Attack Classifier".The results of this process will be presented in the next chapter.

5. Performance Analysis

5.1. Accuracy Comparison

The training process has been carried out and gives a classifier for each model. Then an accuracy test is imposed to evaluate the performance of each classifier. First, we look at the single prediction simulation outcomes in table 3 3. Single

Table 3
Accuracy of Single Prediction

	5_{msg}	10_{msg}	15_{msg}	20_{msg}	25_{msg}	30_{msg}
DBN	45.87%	53.77%	59.63%	56.73%	61.10%	60.17%
LSTM	78.53%	79.65%	83.32%	82.62%	85.72%	85.96%
GRU	80.34%	80.06%	83.06%	85.86%	86.28%	**86.42%**
RF	75.91%	77.83%	79.35%	81.62%	82.52%	82.39%

Table 4
Accuracy of 2-Step Prediction

	5_{msg}	10_{msg}	15_{msg}	20_{msg}	25_{msg}	30_{msg}
DBN	64.32%	77.01%	77.27%	73.63%	79.94%	79.26%
LSTM	89.72%	91.05%	92.62%	94.07%	94.56%	**95.88%**
GRU	90.07%	89.97%	92.31%	94.19%	95.13%	95.72%
RF	88.65%	89.03%	91.28%	92.40%	93.89%	93.41%

prediction means that the classifier immediately detects the type of attack that appears and classifies it into an attack-type according to the subsection 4.2.2. In general, there is an accuracy increment when the number of messages is increased for each training and validation process. This also happens in 2-step prediction, see table 4. In a single prediction, the highest accuracy was obtained by GRU architecture, with an accuracy of 86,4% using 30 aggregated messages. In a 2-step prediction configuration, the best accuracy was obtained by LSTM architecture, with an accuracy of 95,9% using also 30 aggregated messages. In this latter case, a result obtained with the GRU model gives an accuracy of only 0.01% less than the LSTM model. We can note that input data that contains more information or features tend to be easier and better detected by the classifier, as seen from the increase in accuracy from 5 messages to 30 messages.

For each model, we compare the results obtained with the single prediction system and with the 2-Step predictions system, where we note that all simulations results show a significant increase in terms of accuracy. The performance disparity between single prediction and 2-step prediction is rather large, with optimal gain accuracy for LSTM and GRU hitting **95%** (see Figures 6 and Figure 7). These two models exhibit similar accuracy, which is understandable given that the GRU algorithm is nearly identical to the LSTM algorithm, except of the kind of gate and memory. Figure 8 shows that the accuracy of RF has increased significantly while approaching that of LSTM and GRU. DBN has the highest accuracy increment, which is equal to 23% for 10 messages, 0.77 from the 2-step prediction minus 0.53 from the single-step prediction (see figure 5). However, overall DBN has the lowest accuracy compared to the three other models (table 3 and table 4).

One thing that is quite interesting to observe is whether the addition of the number of messages has a major effect on the accuracy of attack detection. According to table 3

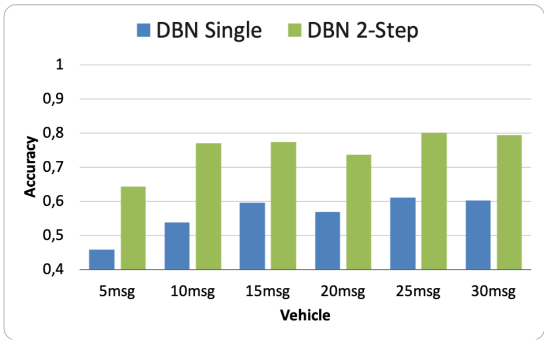

Figure 5: Comparison of Single Prediction and 2-Step Prediction in DBN Model

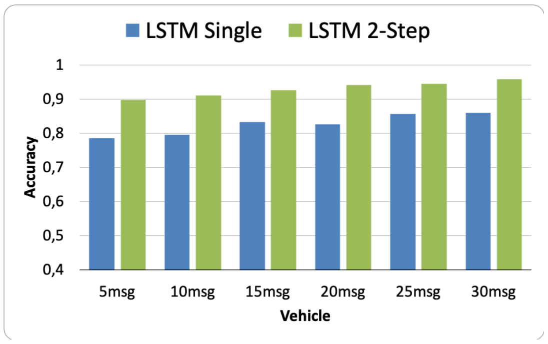

Figure 6: Comparison of Single Prediction and 2-Step Prediction in LSTM Model

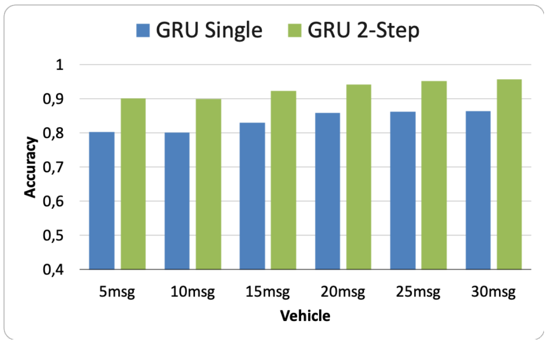

Figure 7: Comparison of Single Prediction and 2-Step Prediction in GRU Model

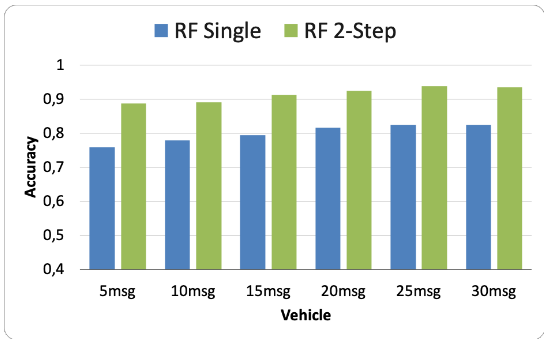

Figure 8: Comparison of Single Prediction and 2-Step Prediction in RF Model

Table 5
The Accuracy Significance of Single and 2-Step Prediciton

	Single Prediction		2-Step Prediction	
	$\bar{x}$	σ	$\bar{x}$	σ
DBN	0.56	0.057	0.752	0.058
LSTM	0.82	0.03	0.930	0.023
GRU	0.83	0.029	0.929	0.025
RF	0.80	0.027	0.914	0.022

and table 4, we calculate the **average accuracy**($\bar{x}$) from 5 messages to 30 messages, then calculate their **standard deviation** (σ) and the overall results are presented in the table 5. The standard deviation value for all models is found to be significantly lower than the average value, indicating that there is no major data deviation between the apparent number of messages. However, the quantity effect of messages has no major influence on detection accuracy, this is true for every model examined. Even though the accuracy of the DBN model is noticeably smaller than the other three models.

5.2. Timing Prediction Comparison

The performance evaluation of misbehavior/attack detection in terms of accuracy is quite important. Furthermore, the detection speed factor is also an important indicator since our proposed system is attended to be used at a crucial time in a vehicular environment. This timing comparison simulation is performed on an Intel Xeon 3.70 GHz processor (16 Cores) workstation,64 GByte of DRAM. In a 2-step detection system, the detection speed is more affected. This is due to the algorithm mechanism of each model being different and also due to the amount of data that can be captured by the detector. In the table 6, 7, 8, and 9, a comparison of the detection speed between single prediction and 2-step prediction is presented. It is shown that LSTM and GRU models require higher detection times. Meanwhile, RF and DBN architectures require less time. The gap of time between the single prediction system and the 2-step prediction system for each model is: for DBN the average gaps time is equal to 0.08 ms. This gap is less for RF model, it is equal to 0.02 ms, while for LSTM and GRU architectures the gap time is more significant, it has an average of 0.63 ms and 0.67 ms respectively.

In general, the 2-step prediction technique will require more time than a single prediction in attacks identification. This is because one input attack data must pass through two classifiers in a 2-step prediction before it can be detected. However, the accuracy of the 2-step prediction system is highly important and more promising in terms of security compared to the single prediction system.

6. Real-time Simulation

6.1. Implementation Setup

The real-time simulation will be run and developed using an established real-time simulation application. At this stage, we modify the F^{2}MD [13] application so that, it can run all

Table 6
Comparison of LSTM Timing Predictions

	Vehicle Timing on Average (ms)	
	Single Prediction	2-Step Prediction
$\mathbf{5}_{msg}$	0.281357	0.729124
$\mathbf{10}_{msg}$	0.367216	0.832994
$\mathbf{15}_{msg}$	0.467365	1.105935
$\mathbf{20}_{msg}$	0.552322	1.278447
$\mathbf{25}_{msg}$	0.654835	1.369581
$\mathbf{30}_{msg}$	0.739149	1.537104

Table 7
Comparison of GRU Timing Predictions

	Vehicle Timing on Average (ms)	
	Single Prediction	2-Step Prediction
$\mathbf{5}_{msg}$	0.255114	0.594953
$\mathbf{10}_{msg}$	0.325892	0.779323
$\mathbf{15}_{msg}$	0.404575	0.959176
$\mathbf{20}_{msg}$	0.494021	1.159909
$\mathbf{25}_{msg}$	0.572481	1.336444
$\mathbf{30}_{msg}$	0.658712	1.917452

Table 8
Comparison of DBN Timing Predictions

	Vehicle Timing on Average (ms)	
	Single Prediction	2-Step Prediction
$\mathbf{5}_{msg}$	0.025007	0.100025
$\mathbf{10}_{msg}$	0.02337	0.105609
$\mathbf{15}_{msg}$	0.025306	0.11346
$\mathbf{20}_{msg}$	0.02614	0.114469
$\mathbf{25}_{msg}$	0.027162	0.120092
$\mathbf{30}_{msg}$	0.028292	0.117776

machine learning model classifiers that are the tasks of the current research works.

The Parameters of real-time simulation are as follows:

- Software environment : OMNET++ v.5.4, SUMO 1.10.0*
- Protocol communication : ITS-G5 (IEEE 802.11p)
- Duration : 2000 second*
- Type Attacker : all of attacker type (random)*
- Scenario : UPHF Map
- Attacker density : 10% and 30%
- Vehicle format data : 5 messages and 30 messages
- ML model : DBN, LSTM, GRU and RF

Table 9
Comparison of RF Timing Predictions

	Vehicle Timing on Average (ms)	
	Single Prediction	2-Step Prediction
$\mathbf{5}_{msg}$	0.035613	0.059768
$\mathbf{10}_{msg}$	0.036318	0.059768
$\mathbf{15}_{msg}$	0.038094	0.058037
$\mathbf{20}_{msg}$	0.038072	0.065021
$\mathbf{25}_{msg}$	0.039105	0.06608
$\mathbf{30}_{msg}$	0.045618	0.061529

In addition to the use of the default settings from F^{2}MD (*), we also made some modifications, including the use of the UPHF map scenario. UPHF stand for Université Polytechnique Hauts-de-France in Valenciennes France see figure 9. By using our campus map, we hope that further research can be more sustainable. Moreover, our vehicle data input is adjusted to the 2-step prediction technique. The use of 5_{msg} and 30_{msg} configurations is intended to see changes from the lowest accuracy to the highest accuracy according to the results of table 4. In real-time simulations, attacker density is defined as the number of attack vehicles divided by the total number of vehicles. The default on F^{2}MD framework is 5%, so for this study, we use 10% and 30% attacker density intending to increase the chances of detecting attack vehicles and also concerning evaluate the impact of raising the number of attackers on each ML model's performance. To adapt our anomalies detection system to the new map scenario, all ML model classifiers are retrained using data generated from the UPHF map scenario.

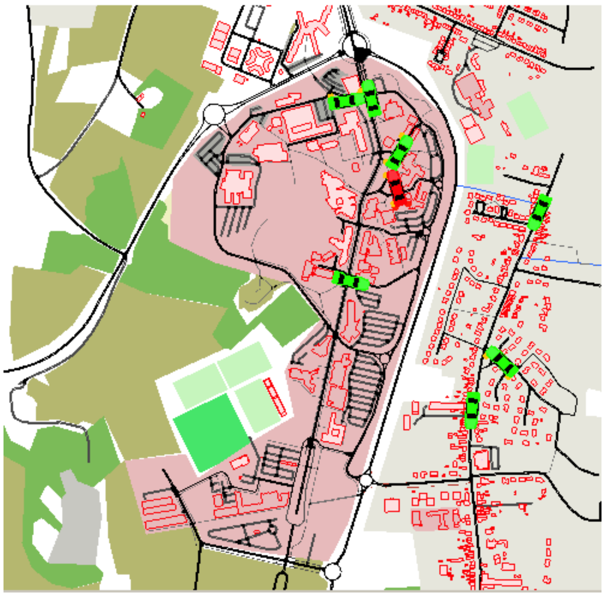
Figure 9: UPHF Map on SUMO GUI

Implementation in a real-time simulation environment has to consider real-time data and training data-sets are in different formats, even though they have the same substance. To be able to retrieve real-time data by the prediction model, an intermediary algorithm is needed (see algorithm 3). This algorithm works by filtering the received BSM and then retrieving the main information such as vehicle ID, vehicle type (legitimate or attacker), vehicle position, and speed. Especially for position and speed data, the format will be changed according to the prediction model that will be used.

Algorithm 3 2-Step Prediction Implementation Algorithm for Real-time Simulation

```
1:  function CONVERTBSM
2:      Pass in : position, speed
3:      pos.array ← position(x,y)
4:      speed.array ← speed(x,y)
5:      Pass out : pos.array, speed.array
6:  end function
7:  bsmLoad ← loads(bsmDataStrem)
8:  clf1 ← load(1st classifier)
9:  clf2 ← load(2nd classifier)
10: idVehicle ← bsmLoad[VehRealID]
11: createTime ← bsmLoad[MsgCreateTime]
12: posBSM ← bsmLoad[Position]
13: speedBSM ← bsmLoad[Speed]
14: pos.array = CONVERTBSM ← posBSM
15: speed.array = CONVERTBSM ← speedBSM
16: tempVehicle ← idVehicle
17: target ← tempVehicle[a]
18: if idVehicle = target then
19:     if CrTime != tempCrTime then
20:         tempCrTime ← createTime
21:         X.Pos ← pos.array
22:         X.Speed ← speed.array
23:         X.Vehicle ← join(X.Pos, X.Speed)
24:         YPrediction1 ← clf1.predcit(X.Vehicle)
25:         if YPredcition == 1 then
26:             YPrediction2 ← clf2.predcit(X.Vehicle)
27:             Attacker ← X.Vehicle
28:         else
29:             Legitimate ← X.Vehicle
30:         end if
31:         a = a + 1
32:     end if
33: end if
```

6.2. Evaluation

The output results of the real-time simulation detection are given in the format of a confusion matrix that includes the following : True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) (see table 10).

$$Recall = \frac{TP}{TP + FN} \tag{1}$$

Table 10
Confusion Matrix of Detection Result

	Attacker (Predict)	**Legitimate (Predict)**
Attacker (Actual)	TP	FN
Legitimate (Actual)	FP	TN

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$F_1 - Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{3}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4}$$

By definition, *Precision* (1) is the ratio between True Positive and the amount of data that is predicted to be positive (Its means "Attacker Detected" = Positive). In other words, precision can be interpreted as *the ability to distinguish between attackers and legitimate ones*. Then as for *Recall* (2), it is a comparison between True Positive and the number of data that are positive or we can call it the *sensitivity to detect attacker*. F_1*-Score* (3) is the harmonic average between *Precision* and *Recall*. The best value for F_1-*Score* is 1.0 and the worst value is 0. In representation, if the F_1-*Score* has a good score, it indicates that our classification model has good *Precision* and *Recall*. While *Accuracy* (4) is the ratio of all data that is correctly detected, whether it is detected as an attacker's vehicle or correctly as a legitimate vehicle.

In this simulation, we only use **2-Step Prediciton** and present two types of tables for each simulation under different attacker densities:

1. Result of *1st prediction* (Table 11 and 13) : These tables show the ability of each ML model in distinguishing the attacker's vehicle from the legitimate vehicle.

2. Result of *2nd prediction* (Table 12 and 14): These tables show the ability of each ML model in distinguishing the types of attacks, including distinguishing them from legitimate vehicles. In other words, the ability to classify attack types is measured in this table.

Reviewing when the attacker density is 10% (table 11) and the number of messages is increased from 5 messages to 30 messages, each ML model has increased sensitivity in detecting the attacker's vehicle. It can be seen from the *Recall* value which has increased for all models, even LSTM, GRU, and RF can detect all attacker vehicles that appear. Even though, there is an issue that the number of attacker vehicles is much less than the legitimate vehicles.

On the other hand, the LSTM *Precision* value does not experience a significant increase. In the case of the 30 msg, the F_1*-Score* value and its *Accuracy* are the lowest compared to other models, this indicates that in real-time implementation, LSTM does not have a good performance in differentiating between attacker vehicle and legitimate vehicle when there is only a small number of attacker vehicle emerging.

In table 12 it will be more concerned with *Recall* value because the classification performance is influenced by changes in the *True Positive* and *False Negative* values. If there is an incorrect classification by the 2nd prediction, it will reduce the TP value and increase the FN value, causing the *Recall* value to decrease.

However, we can see that in the use of 5_{msg}, the ability of each model to classify types of attacks, decreases and looks very low in value. So the use of 5_{msg} to classify low-density attackers is less effective. As for the use of 30_{msg}, the decrease only occurred in DBN.

Considering the addition of the density of attackers to 30% (comparing table 13 and table 11), we see that at 5_{msg}, DBN, LSTM, and GRU experienced an increase in attack detection sensitivity, indicated by an increase in the *Recall* value. Meanwhile, the sensitivity of RF has decreased slightly. The ability to distinguish between attack vehicles and legitimate vehicles is also improved for all models. This is indicated by the increase in the *Precision* value compared to the 10% density condition. This is caused by an increase in the value of TP when there is an expansion in the number of density attackers.

Still, in comparison between table 13 and table 11, accuracy at 5_{msg} also improves for every model except RF as the detection sensitivity decreases. At 30_{msg}, all models experienced a decrease in attack detection sensitivity. At a density of 30%, the number of attacks increases 3 times, and 30_{msg} requires a longer detection time than 5_{msg} resulting in more attacks that can be falsely detected. However, the ability to distinguish between attack vehicles and legitimate vehicles increased significantly, especially for LSTM. it can be seen from the LSTM. *Precision* and F1 Score at a density of 30% which increased significantly both at 5_{msg} and at 30_{msg}, compared to a density of 10%. This makes the LSTM's performance closer to GRU and RF in terms of distinguishing between attacking and legitimate vehicles (see table 13).

To see the performance of the classification of attack types in 30% density attacker, we can review the *Recall* value in the table 14. It can be seen that the classification ability of all models has decreased, meaning that each model has difficulty in distinguishing the type of attack, especially at 5_{msg}. However, at the use of 30_{msg}, only LSTM decreased slightly, while for other models the *Recall* value decreased significantly. This makes the LSTM *Recall* value at 30_{msg} is the highest, this means that LSTM has a better ability in terms of classifying types of attacks than DBN, GRU, and RF. But for total accuracy, GRU and RF are somewhat better than LSTM.

Table 11
Result of 1st Prediction (Real-Time with 10% Density Attacker)

		Recall	Precision	F_1-Score	Accuracy
DBN	5_{msg}	0.3333	0.0811	0.1304	0.6226
	30_{msg}	0.75	1	0.8571	0.9773
LSTM	5_{msg}	0.6667	0.1304	0.2182	0.5943
	30_{msg}	1	0.1818	0.3077	0.5909
GRU	5_{msg}	0.625	0.1042	0.1786	0.578
	30_{msg}	1	0.6667	0.8	0.9545
RF	5_{msg}	0.7778	0.3684	0.5	0.8679
	30_{msg}	1	0.5	0.6667	0.9091

Table 12
Result of 2nd Prediction (Real-Time with 10% Density Attacker)

		Recall	Precision	F_1-Score	Accuracy
DBN	5_{msg}	0.2222	0.0556	0.0889	0.6132
	30_{msg}	0.5	1	0.6667	0.9545
LSTM	5_{msg}	0.3333	0.0698	0.1154	0.566
	30_{msg}	1	0.1818	0.3077	0.5909
GRU	5_{msg}	0.5	0.0851	0.1455	0.5688
	30_{msg}	0.75	0.6	0.6667	0.9318
RF	5_{msg}	0.6667	0.3333	0.4444	0.8585
	30_{msg}	1	0.5	0.6667	0.9091

Table 13
Result of 1st Prediction (Real-Time with 30% Density Attacker)

		Recall	Precision	F_1-Score	Accuracy
DBN	5_{msg}	0.75	0.5455	0.6316	0.6111
	30_{msg}	0.2857	1	0.4444	0.6552
LSTM	5_{msg}	0.8333	0.5738	0.6796	0.6333
	30_{msg}	0.8214	0.7419	0.7797	0.7759
GRU	5_{msg}	0.963	0.5652	0.7123	0.7123
	30_{msg}	0.6842	0.8667	0.7647	0.8261
RF	5_{msg}	0.7241	0.6562	0.6885	0.7432
	30_{msg}	0.7222	0.9286	0.8125	0.8696

7. Discussion

There is a decrease in *Accuracy* in the real-time simulation compared to the simulation in subsection 5.1. This is due to the data created in the real-time simulation is always changing and never the same from time to time,

Table 14
Result of 2nd Prediction (Real-Time with 30% Density Attacker)

		Recall	Precision	F_1-Score	Accuracy
DBN	$\mathbf{5}_{msg}$	0.275	0.3056	0.2895	0.4
	$\mathbf{30}_{msg}$	0.1429	1	0.25	0.5862
LSTM	$\mathbf{5}_{msg}$	0.3095	0.3333	0.321	0.3889
	$\mathbf{30}_{msg}$	0.7857	0.7333	0.7586	0.7586
GRU	$\mathbf{5}_{msg}$	0.4074	0.3548	0.3793	0.5068
	$\mathbf{30}_{msg}$	0.5789	0.8462	0.6875	0.7826
RF	$\mathbf{5}_{msg}$	0.4828	0.56	0.5185	0.6486
	$\mathbf{30}_{msg}$	0.5	0.9	0.6429	0.7826

particularly the vehicle's position and speed data, which are the study's major parameters. Although the changes in the data that occur (every real-time simulation is running) are not very significant, they are quite influential in changing the *Accuracy* of each ML model. Considering that there are quite a lot of different types of attacks that must be distinguished. Detection at once for dozens of types of attacks is also carried out on [6], but here a threshold mechanism is used. The decrease in *Accuracy* also occurs significantly in some types of attacks in this research. The design of the detection model that is more focused on real-time applications needs to be done in more depth.

Some machine learning models have a fairly high level of *Accuracy* while some others have a good detection speed. However, not always a machine learning model has these two advantages. As in this study, a high level of detection *Accuracy* but with a low detection speed is found with LSTM and GRU. However, the opposite happened with DBN. This opens up opportunities for further research, in which a high degree of *Accuracy* machine learning model is desired with a better detection speed.

In addition, further research on the most ideal number of messages to be applied to real-time systems needs to be followed up. So it is hoped that we will get the most optimal model both in terms of speed and attack detection capabilities.

8. Conclusion

The application of 2-Step Prediction on an offline system that involves the availability of data-sets shows good results in classifying several types of attack compared to a single prediction. This indicates that the 2-Step Prediction system can be used to analyze cases of suspected attacks on the Vehicular Network in a certain location.

In *real-time systems* implementation, all ML models have decreased in terms of accuracy, considering the complexity is relatively high. However, by using only the vehicle's position and speed information, 2-step prediction can distinguish between the attacker's vehicle and the legitimate vehicle with an accuracy rate of 87% and 83%, respectively, for Random Forest and GRU. LSTM also has an advantage in the level of sensitivity to distinguish types of attacks compared to other ML models. Although in terms of total accuracy, it is still below GRU and Random Forest. While the lowest is DBN. It's also worth mentioning that the DBN and RF models have the fastest detection speeds, while the LSTM and GRU models have the slowest.

Another conclusion based on the implementation in real-time systems, an increment of messages and density of attackers can affect the detection performance of the machine learning model. The higher the number of messages employed in the detection, the higher the level of accuracy, but on the other hand, the detecting speed will slow down. While the number of attackers increases, the type of attack gets more difficult to classify.

Acknowledgment

This research is supported by Directorate General of Higher Education, Research and Technology (DITJEN DIKTI) of Indonesia, with grand number: T/961/D3.2/KD.02.01/2019. We'd also want to express our gratitude to our colleagues from the Université Polytechnique Hauts-de-France's IEMN/DOAE and the Université Gustave Eiffel's Cosys-Léost Laboratory.

References

[1] S. Ribouh, K. Phan, A. V. Malawade, Y. Elhillali, A. Rivenq, M. A. Al Faruque, Channel state information-based cryptographic key generation for intelligent transportation systems, IEEE Transactions on Intelligent Transportation Systems 22 (12) (2020) 7496–7507. doi:10.1109/TITS.2020.3003577.

[2] W. Feng, Intelligent Transportation Systems for Sustainable Development in Asia and the Pacific. Available at https://www.unescap.org/sites/default/files/ITS.pdf (2014).

[3] I. Malygin, V. Komashinsky, V. V. Tsyganov, International experience and multimodal intelligent transportation system of Russia, in: 2017 Tenth International Conference Management of Large-Scale System Development (MLSD), IEEE, Moscow, 2017, pp. 1–5. doi:10.1109/MLSD.2017.8109658.

[4] Z. El-Rewini, K. Sadatsharan, D. F. Selvaraj, S. J. Plathottam, P. Ranganathan, Cybersecurity challenges in vehicular communications, Vehicular Communications 23 (2020) 100214. doi:10.1016/j.vehcom.2019.100214.

[5] G. Zhao, C. Zhang, L. Zheng, Intrusion Detection Using Deep Belief Network and Probabilistic Neural Network, in: 22017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), IEEE, Guangzhou, China, 2017, pp. 639–642. doi:10.1109/CSE-EUC.2017.119.

[6] 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: ICC 2020 - 2020 IEEE International Conference on Communications (ICC), IEEE, Dublin, Ireland, 2020, pp. 1–6. doi:10.1109/ICC40277.2020.9149132.

[7] R. W. van der Heijden, T. Lukaseder, F. Kargl, VeReMi: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs, arXiv:1804.06701 [cs]ArXiv: 1804.06701 (Apr. 2018).

[8] N. C. Kushardianto, Y. E. Hillali, C. Tatkeu, 2-step prediction for detecting attacker in vehicle to vehicle communication, in: 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), 2021, pp. 1–5. doi:10.1109/VTC2021-Fall52928.2021.9625513.

[9] M. L. Han, B. I. Kwak, H. K. Kim, Anomaly intrusion detection method for vehicular networks based on survival analysis, Vehicular Communications 14 (2018) 52–63. doi:10.1016/j.vehcom.2018.09.004.

[10] I. Rasheed, F. Hu, L. Zhang, Deep reinforcement learning approach for autonomous vehicle systems for maintaining security and safety using LSTM-GAN, Vehicular Communications 26 (2020) 100266. doi:10.1016/j.vehcom.2020.100266.

[11] C. Hidalgo, M. Vaca, M. P. Nowak, P. Frölich, M. Reed, M. Al-Naday, A. Mpatziakas, A. Protogerou, A. Drosou, D. Tzovaras, Detection, control and mitigation system for secure vehicular communication, Vehicular Communications (2021) 100425doi:10.1016/j.vehcom.2021.100425.

[12] M. Zhou, L. Han, H. Lu, C. Fu, Y. Qian, Attack detection based on invariant state set for SDN-enabled vehicle platoon control system, Vehicular Communications (2021) 100417doi:10.1016/j.vehcom.2021.100417.

[13] J. Kamel, F^{2}md github repository, available at https://github.com/josephkamel/F2MD.git [Online] (2019).

[14] J. Kamel, M. R. Ansari, J. Petit, A. Kaiser, I. B. Jemaa, P. Urien, Simulation Framework for Misbehavior Detection in Vehicular Networks, IEEE Transactions on Vehicular Technology 69 (6) (2020) 6631–6643. doi:10.1109/TVT.2020.2984878.

[15] S. Gyawali, Y. Qian, Misbehavior Detection using Machine Learning in Vehicular Communication Networks, in: ICC 2019 - 2019 IEEE International Conference on Communications (ICC), IEEE, Shanghai, China, 2019, pp. 1–6. doi:10.1109/ICC.2019.8761300.

[16] M. Aloqaily, S. Otoum, I. A. Ridhawi, Y. Jararweh, An intrusion detection system for connected vehicles in smart cities, Ad Hoc Networks 90 (2019) 101842. doi:10.1016/j.adhoc.2019.02.001.

[17] B. Wang, S. Sun, S. Zhang, Research on feature selection method of intrusion detection based on deep belief network, in: Proceedings of the 2015 3rd International Conference on Machinery, Materials and Information Technology Applications, Atlantis Press, 2015/11, pp. 556–561. doi:https://doi.org/10.2991/icmmita-15.2015.107.

[18] B. Brecht, D. Therriault, A. Weimerskirch, W. Whyte, V. Kumar, T. Hehn, R. Goudy, A Security Credential Management System for V2X Communications, arXiv:1802.05323 [cs]ArXiv: 1802.05323 (Feb. 2018).

[19] N. Kumar, R. Iqbal, S. Misra, J. J. Rodrigues, An intelligent approach for building a secure decentralized public key infrastructure in vanet, Journal of Computer and System Sciences 81 (6) (2015) 1042–1058, special Issue on Optimisation, Security, Privacy and Trust in E-business Systems. doi:https://doi.org/10.1016/j.jcss.2014.12.016.

[20] J. Alves-Foss, Multi-Protocol Attacks and the Public Key Infrastructure 12.

[21] J. Zacharias, S. Froschle, Misbehavior detection system in VANETs using local traffic density, in: 2018 IEEE Vehicular Networking Conference (VNC), IEEE, Taipei, Taiwan, 2018, pp. 1–4. doi:10.1109/VNC.2018.8628321.

[22] M. Hadded, O. Shagdar, P. Merdrignac, Augmented Perception by V2X Cooperation (PAC-V2X): Security issues and misbehavior detection solutions, in: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), IEEE, Tangier, Morocco, 2019, pp. 907–912. doi:10.1109/IWCMC.2019.8766683.

[23] J. Petit, R. Ansari, V2x validation tool, available at https://bitbucket.org/onboardsecurity/dsrcvt [Online] (2018).

[24] J. Kamel, F. Haidar, I. B. Jemaa, A. Kaiser, B. Lonc, P. Urien, A Misbehavior Authority System for Sybil Attack Detection in C-ITS, in: 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), IEEE, New York City, NY, USA, 2019, pp. 1117–1123. doi:10.1109/UEMCON47517.2019.8993045.

[25] albertbup, A python implementation of deep belief networks built upon numpy and tensorflow with scikit-learn compatibility, available at https://github.com/albertbup/deep-belief-network [Online] (2017).

[26] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780. doi:10.1162/neco.1997.9.8.1735.

[27] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, arXiv:1412.3555 [cs]ArXiv: 1412.3555 (Dec. 2014).

[28] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, arXiv:1406.1078 [cs, stat]ArXiv: 1406.1078 (Sep. 2014).

[29] T. K. Ho, Random decision forests, in: Proceedings of 3rd International Conference on Document Analysis and Recognition, Vol. 1, 1995, pp. 278–282 vol.1. doi:10.1109/ICDAR.1995.598994.

[30] B. Komer, J. Bergstra, C. Eliasmith, Hyperopt-Sklearn: Automatic Hyperparameter Configuration for Scikit-Learn, Austin, Texas, 2014, pp. 32–37. doi:10.25080/Majora-14bd3278-006.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.