Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

SUCCESS-6G: DEVISE

WP4 Deliverable E9

Secure and trustworthy condition monitoring of vehicles' health

Project Title:	SUCCESS-6G: DEVISE
Title of Deliverable:	Secure and trustworthy condition monitoring of vehicles' health
Status-Version:	v1.0
Delivery Date:	08/02/2024
Contributors:	Roshan Sedar, Charalampos Kalalas, Pavol Mulinka (CTTC), Francisco Paredes, Miguel Fornell (IDNEO)
Lead editor:	Roshan Sedar, Charalampos Kalalas (CTTC)
Reviewers:	Charalampos Kalalas (CTTC)
Keywords:	Reinforcement learning-based detection; clustering; PKI; TLS

Document revision history

Version	Date	Description of change
v0.1	12/12/2023	ToC created
v0.2	15/01/2024	Initial content added
v0.3	30/01/2024	Additional content added
v1.0	08/02/2024	Final version uploaded to the website

Disclaimer

Acknowledgment

The research conducted by SUCCESS-6G - TSI-063000-2021-39/40/41 receives funding from the Ministerio de Asuntos Económicos y Transformación Digital and the European Union-NextGenerationEU under the framework of the “Plan de Recuperación, Transformación y Resiliencia” and the “Mecanismo de Recuperación y Resiliencia”.

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

Executive Summary

Secure and trustworthy condition monitoring of vehicles' health is crucial to guarantee the seamless and robust operation of predictive maintenance systems for vehicles. Information exchange between the vehicle and the maintenance control center needs to take place in a secure and robust manner with the aid of advanced security mechanisms able to detect and contain sophisticated attack vectors originated by malicious network entities. This deliverable describes the research activities in the SUCCESS-6G-DEVISE project towards enhancing the security of the monitoring information. In particular, we introduce an attack detection mechanism empowered by reinforcement learning to detect a wide range of attack vectors from unlabelled vehicular data instances. The applicability of the Transport Layer Security (TLS) 1.3 protocol in the communication channels between the on-board unit (OBU) client and the server is also discussed as an additional means of enhancing security, data privacy, and integrity.

Table of Contents

List of Figures

List of Tables

1 Introduction

Despite the multitude of benefits offered by Vehicle-to-Everything (V2X) communication, vulnerabilities and security breaches are not uncommon in vehicular networks. The peculiar characteristics of V2X systems, in conjunction with the increased levels of connectivity and driving autonomy, introduce entirely new security concerns and issues that have not been addressed in a similar context before. As a result, evolving security requirements are expected to be more stringent as services and applications for the automotive sector will be often mission critical. Vehicular services exhibit idiosyncrasies in terms of functionalities and deployment scenarios, with several security threats lurking in. This complex V2X connectivity landscape renders the attack surface sufficiently large with expanded threat vectors, calling for innovative security solutions.

Aiming to address this limitation, the scientific approach of SUCCESS-6G-DEVISE aims at leveraging the advanced capabilities of Artificial Intelligence/Machine Learning (AI/ML) technologies as an effective means to enhance security in V2X connectivity and address vulnerabilities. Secure condition monitoring requires an effective and timely prediction of maliciously abnormal data in order to guarantee robust operation of the predictive maintenance service. Condition monitoring data used for subsequent predictive diagnostics need to be properly secured in untrusted V2X environments such that they do not contain falsified information, while abnormal traffic will be detected and isolated in its entirety. In SUCCESS-6G, we aim to develop defensive mechanisms towards a secure condition monitoring framework for situational awareness of vehicles' health.

2 Secure vehicular condition monitoring with ensemble learning

2.1 Problem statement and motivation

Secure and trustworthy condition monitoring of vehicles' health is crucial to maintain the stability of predictive maintenance systems for vehicles. Message exchange between the vehicle and the maintenance control centre needs to take place in a secure and robust manner in order to properly communicate the identification of defects in vehicles, such as malfunctions of components. However, recent advances in vehicle-to-everything (V2X) connectivity come inadvertently with security vulnerabilities and evolving threat vectors which may destabilize system operation and degrade network performance. Attackers with malicious intents may inject incorrect/erroneous data in the monitoring information communicated to the control/maintenance centres [1]. Such actions often become difficult to detect and contain, since malicious nodes may alter their activity intelligently over time. In this context, real-time detection of abnormal data is essential in order to alleviate the propagation of potential malicious data across edge infrastructure and provide an additional means to guarantee the trustworthiness of exchanged vehicular information.

Aiming to address the complex V2X security landscape, several recent works leverage AI tools for detection of abnormal vehicular information [2], [3], [4]. Authors in [2], [3] focus on identifying position falsification attacks using conventional supervised learning techniques on labelled datasets. However, such schemes may be impractical in real-time V2X scenarios with expanded attack surface, due to limited access to labelled training examples and/or dependence on security threshold values. In [4], a deep neural network architecture is introduced to detect all types of attacks in the open-source VeReMi dataset [5]. Yet, unforeseen changes in V2X traffic, due to either naturally drifting mobility patterns or unprecedented malicious activity, introduce challenges (e.g., model overfitting) to deep-learning-based attack identification methods.

Motivated by these research questions, we have introduced in [6] an edge-based security framework for secure and trustworthy vehicular data monitoring based on ensemble learning. Our approach jointly combines i) an unsupervised learning layer for discovering hidden patterns from unlabelled vehicular traffic traces, and ii) a reinforcement learning (RL) layer for consistently improving malicious data detection over unknown V2X environments without relying on security thresholds. We employ the K-means algorithm to cluster and annotate data instances, and, subsequently, train an RL-based detector to discriminate genuine vehicles from malicious ones. Label provisioning facilitates the generation of reward signals for the detector decisions at each time-step, by comparing with the acquired ground truth information. An in-depth assessment of our learning framework using the VeReMi dataset reveals meaningful insights for the detection performance over various attack types. Compared to benchmark classifiers, our approach exhibits superior or equivalent detection performance in the presence of potentially inaccurate or mislabelled training data. Finally, in an effort to gain perspective on the real-time capabilities of our detection framework, we evaluate the overall latency required for detecting an attack. Detection latencies are shown to comply with edge-related requirements, making our approach suitable for SUCCESS-6G use cases.

2.2 Network scenario and attack model

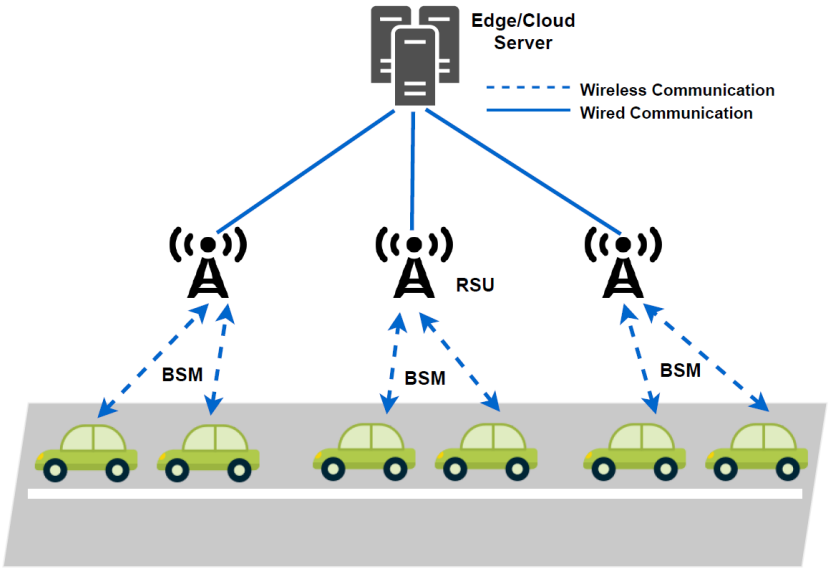


Figure 1: V2X network scenario [6]

Figure 1 illustrates the considered V2X network scenario, where vehicles periodically transmit basic safety messages (BSMs). We assume that vehicular monitoring information is embedded in the BSMs which include the position, speed, acceleration and heading angle of each vehicle, and other relevant information. A roadside unit (RSU) receives BSMs from vehicles located within its coverage, and the edge/cloud server aggregates information from RSUs deployed in a large geographical area. A vehicle transmitting falsified monitoring information embedded in BSMs is considered as an *attacker* (i.e., malicious vehicle). There exist a number of attack types that can potentially undermine V2X security, as prescribed in VeReMi dataset [5].

In what follows, we briefly discuss a set of V2X attacks relevant to our scenario.

Sybil attack: A vehicle may use multiple valid pseudonyms of compromised vehicles to realize an attack while concealing its real identity (ID). For instance, an attacker may generate fake road traffic congestion with a grid of ghost vehicles in a selected geographical region. Valid pseudonymous IDs and BSM frequency may be used for every ghost vehicle.

Data replay attack: A vehicle re-transmits or replays valid BSMs previously received from other vehicles. In this case, the vehicle uses its own ID while replaying the data and tries to exploit the conditions that existed at the time of the original BSM transmission. The attack could also be carried out in Sybil mode by changing the attacker ID.

Denial-of-service (DoS) attack: A vehicle transmits BSMs at a frequency higher than the limit set by the standard. Such high volume of data transmission would result in extensive periods of network congestion and unavailability to serve other legitimate vehicles. DoS attacks may also be launched by setting all BSM fields to random values (i.e., DoS random). Such behaviours can be concealed in a subtle way using compromised vehicles' identities (Sybil mode).

Disruptive attack: The pattern of this attack is similar to data replay, where a vehicle re-transmits previously sent messages by other vehicles. BSMs are selected at random and flood the network with stale data to disrupt genuine information from being propagated. This attack may also be carried out in DoS and Sybil modes.

2.3 Proposed approach

Figure 2 depicts our proposed data-driven detection framework. Our methodology comprises four key steps:

1. Messages (including genuine and attack information) are retrieved from the raw V2X data.
2. Message preprocessing allows the extraction of relevant feature vectors for various attack types.
3. An unsupervised learning module clusters and subsequently annotates data instances per attack type.
4. An RL component leverages the labelled data instances to detect and classify various attack types.

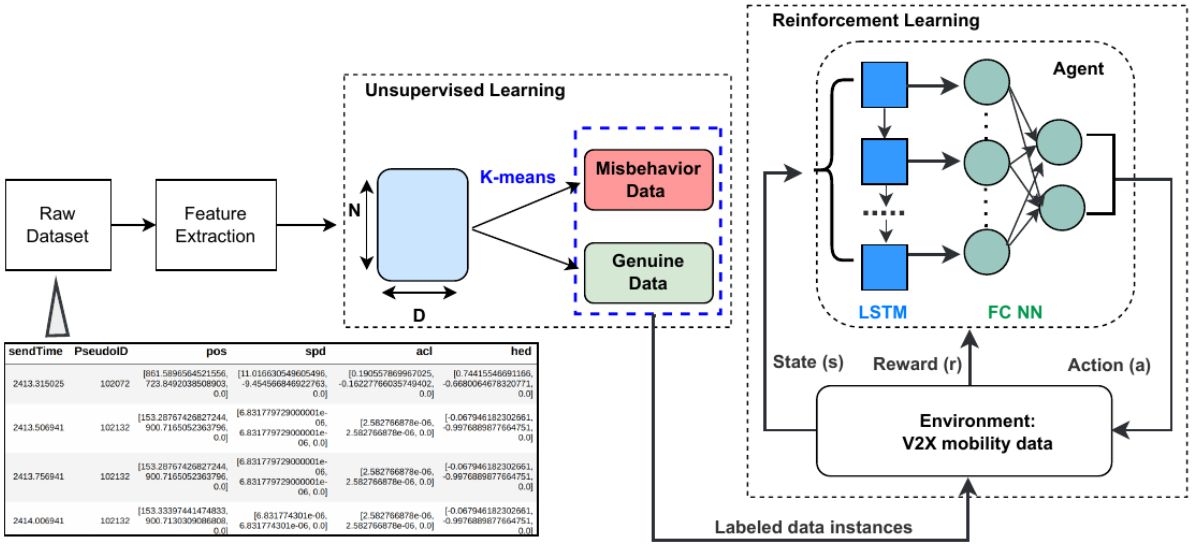


Figure 2: Proposed ensemble learning framework for unsupervised data preprocessing and RL-based abnormal data detection [6].

In what follows, we elaborate on the learning components of our detection framework.

2.3.1 Unsupervised learning for clustering and labeling

Unsupervised learning aims to discover hidden patterns underlying in data without relying on label information. Clustering, an unsupervised learning task, typically relies on the assumption that normal data instances belong to large and dense clusters, while anomalies either belong to small or sparse clusters [7]. This is the case for VeReMi dataset, where the proportion between misbehaving and genuine vehicles for each attack scenario is approximately 30% to 70%, respectively [5]. Each attack scenario contains two BSM types, namely genuine and misbehavior, but cannot clearly categorize data instances into these types without labels. In this work, we leverage the K-means algorithm to cluster unlabeled V2X data instances into genuine and misbehavior groups. K-means is chosen for its simplicity and favorable characteristics, as it attempts to group together data instances that are mutually close in Euclidean space. Advantages such as scaling to large datasets, controlling the number of clusters to extract, and generalizing to clusters of different shapes and sizes, e.g., spherical and elliptical clusters, render K-means well-suited for our study case.

The goal of K-means clustering is to generate ground truth information that is necessary for the subsequent RL-based attack detection and classification. For each cluster, a centroid is defined, which represents the mean of the data instances assigned to the cluster. K-means works iteratively to assign data instances to one of the K clusters based on the given features. In each iteration, the algorithm measures the similarity of data instances by computing their Euclidean distance from the centroid on

the dimension of the feature vector. As such, if a data point belongs to a particular cluster, then it is closer to the considered centroid than any other centroids. Finally, all data instances are assigned to their clusters based on pairwise feature similarity.

At its inception, K-means algorithm requires the number of clusters and initial centroid positions. Since these values are initially unknown, a commonly used method is to resort to random initialization of centroid locations for a range of clusters. The Elbow method [8] is also used to determine the optimum number of K clusters for an attack scenario. Once K is determined, K-means groups data instances for each attack scenario and the algorithm then generates the ground truth information. The inherently imbalanced VeReMi dataset is expected to generate a large cluster size for genuine instances, and a smaller size for misbehaving ones. Thus, upon algorithm convergence, instances belonging to the cluster with the lowest number of samples are labeled as misbehaving (label "1"), while the rest as genuine (label "0").

2.3.2 Reinforcement learning model

Markov decision process (MDP) offers a modelling framework [9] for attack detection with sequential decision-making over V2X data traces. An MDP is defined as a tuple of five elements, i.e., $\mathcal{M} =< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ the set actions, $\mathcal{P}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0,1]$ is the state transition probabilities function, $\mathcal{R}: \mathcal{S} \mapsto \mathbb{R}$ denotes the reward function with a set of possible rewards, and $\gamma \in (0,1)$ denotes the discount factor which reflects the importance of immediate and long-term future rewards. The action of attack detection will change the environment based on the decision of either genuine or malicious behavior at time-step t; subsequently, the next decision at time-step $t+1$ will be influenced by the changing environment at previous time-step t.

As shown in Figure 1, the aggregated V2X data at the edge node (i.e., RSU) constitute a time-series repository of received BSMs with intrinsic temporal and spatial interdependencies. We hereby consider an RL-based attack detector deployed at the edge RSU. The detector (agent) interacts with the V2X environment to learn the optimal detection policy π. Based on the current state s_t, the agent takes an action a_t to maximize its reward r_t. The agent is then rewarded by the environment, and the environment moves to state s_{t+1} following the MDP.

The process described above iterates until an optimal detection policy π is learned. The Q-learning method [10] is adopted to train the RL model to estimate the action-value function $Q(s,a)$. Since it is practically infeasible to use tabular Q-learning with a very large Q-table for V2X state-action space, we utilize a deep learning method for function approximation. In particular, we leverage an artificial neural network (ANN) to approximate the action-value function $Q(s,a)$. In turn, the agent ANN can effectively learn to map input states to Q-values. The ϵ-greedy method is used while training to strike a balance between exploration and exploitation in the agent strategy.

The agent receives the V2X time-series data and prior related decisions as inputs (i.e., state s_t), and generates the new decision made (i.e., action a_t) as output. At each timestep t, the agent actions are selected by the policy π. The agent experience, i.e., $e_t = \langle s_t, a_t, r_t, s_{t+1} \rangle$, stores all the behaviors of the detector. By exploiting experience, the detector is improved to obtain a better estimation of the $Q(s,a)$ function. This process is referred to as experience replay memory, through which deep Q-learning achieves stability [11]. During this process, the agent randomly samples batches from the experienced buffer to learn from. The main objective is to maximize the expected sum of future discounted rewards by learning the optimal detection policy. The discounted reward return is expressed as

$$R_t = \sum_{k=t}^{T} \gamma^{k-t} r_k,$$

where T represents the number of time steps in an episode of training. Q-learning model updates are performed with learning rate α as

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right).$$

The environment controls the training of the agent. After receiving a_t performed by the agent, the environment generates a reward r_t and the next environment state s_{t+1} for the agent. As shown in Figure 2, the environment contains a large population of BSMs with ground truth information generated via the procedure described in Section 2.3.1.

The state contains the sequence of previous actions denoted by $s_{\text{action}} =< a_{t-1}, a_t, \dots, a_{t+n-1} >$, and the current BSM information denoted by $s_{\text{time}} =< X_t, X_{t+1}, \dots, X_{t+n} >$. $X_t \in \mathbb{R}^d$ is a d-dimensional feature vector at time-step t, including information on d different features. According to the state design, the next action taken by the agent depends on the previous actions and the current V2X information. The action space is defined as $\mathcal{A} = \{0,1\}$, where 1 indicates the detection of an attack and 0 represents the genuine behavior. The deterministic detection policy π can be expressed as a mapping, i.e., $\pi: \mathcal{S} \mapsto \mathcal{A}$, from states to actions, where $\pi(s)$ denotes the action that the agent takes at state s.

In a given state s_t, the agent selects the action based on the optimal detection policy given by

$$\pi^* = \arg\max_{a \in \mathcal{A}} Q^*(s, a).$$

The reward r_t helps the agent to explore an environment with different states and learn an effective detection policy. The reward signals are emitted as feedback (i.e., positive/negative) for an a_t taken in s_t. A numerical value for r_t is assigned based on the ground truth information of BSMs. Specifically, a positive reward is given to the agent for correctly detecting an attack, i.e., true positive (TP), or a normal state, i.e., true negative (TN). A negative reward is otherwise provided for incorrect identification of a normal state as an attack, i.e., false positive (FP), or an attack as a normal state, i.e., false negative (FN). The agent is penalized more for FN actions than for FPs since the correct identification of an attack is indispensable to avoid hazardous and life-threatening situations.

2.4 Experiments and results

In this section, we evaluate the effectiveness of our ensemble learning approach for V2X attack detection by performing experiments using the VeReMi dataset [5].

2.4.1 Dataset description and preprocessing

The VeReMi dataset comprises 19 attack variants to simulate different attack types. Two vehicular traffic densities are prescribed for each attack scenario: high-density (37.03 vehicles $/\text{km}^2$) and low-density (16.36 vehicles $/\text{km}^2$). In each scenario, a JSON log file per vehicle is created to record the raw data exchanged (i.e., genuine and attack messages) between neighboring vehicles. BSMs include three-dimensional vectors for the position, speed, acceleration, and heading angle features. During preprocessing, all JSON files with the raw data were converted to CSV format and then concatenated together; this was performed for each attack scenario to compile a CSV file with ordered BSM records using their timestamps. Based on feature analysis, we select six fields, i.e., timestamp, pseudo-ID, position, speed, acceleration, and heading angle, as the most relevant feature set related to attack detection. The Euclidean norm of the position, speed, acceleration, and heading angle vectors is further utilized. As shown in Figure 2, the feature-engineered dataset is then fed to the clustering module to create the ground truth labels for each attack scenario.

Since the V2X edge/cloud server has presumably superior computational power over RSUs, we assume that RL model training is offloaded onto the edge/cloud server. The trained model is then used at RSUs for testing. Detection is performed at RSUs, as the vehicle may not have the complete information in

its range during a short period. In our experiments, the high-density dataset was used to train the RL model under each attack scenario to detect and learn attack patterns more frequently. On the other hand, the low-density dataset was used to test the ability of the RL model to detect attacks when attack patterns are less frequent.

2.4.2 Clustering performance

We evaluate K-means clustering performance by computing the silhouette coefficient [12] for a set of attack scenarios in the VeReMi dataset. In particular, the average silhouette coefficient is computed for each sample using i) the mean intra-cluster distance, i.e., between the sample and all other instances in the same cluster; and ii) the mean inter-cluster distance, i.e., between the sample and all other instances in the next nearest cluster. The silhouette score is bounded between -1 for incorrect clustering and +1 for highly dense clustering, while a score close to 0 indicates overlapping clusters. A performance comparison between K-means and spectral clustering [13] in terms of average silhouette score is depicted in Table 1 for three representative attack scenarios. The spectral clustering algorithm treats data clustering as a graph partitioning problem and offers equivalent simplicity as K-means. It can be observed that K-means outperforms spectral clustering with higher scores for all three attack scenarios.

Table 1: Average silhouette score

Attack scenario	K-means	Spectral
Constant position	0.719	0.206
Random speed	0.719	0.152
Random speed offset	0.718	0.059

2.4.3 Detection performance

To assess the detection performance of our proposed framework, we compute the Accuracy, Precision, Recall, and F1 score metrics, by considering both genuine and attack classes for each attack type in VeReMi. The F1 score provides the harmonic mean between precision and recall; thus, higher F1 values indicate better performance. We hereby differentiate between effectively and moderately detected attacks, as follows.

2.4.3.1 Effectively detected attacks

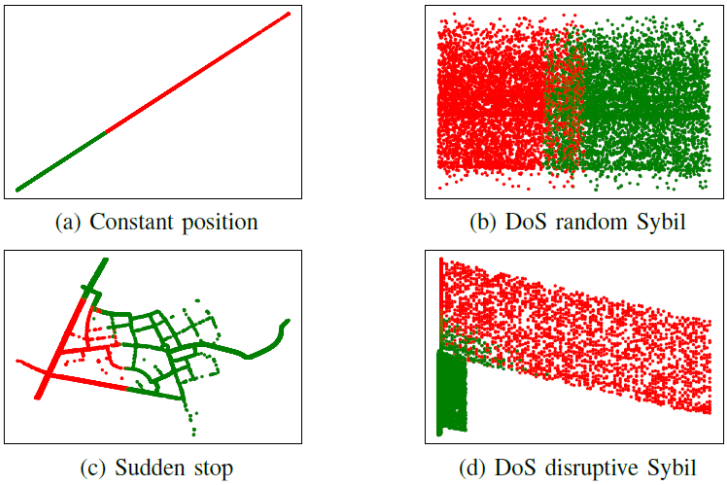
Table 2 depicts the performance of RL-based detection per attack for 19 attack types. Results show that 13 attack types can be effectively detected with over 0.90 F1 score, resulting in high recall and high precision values at the same time. In particular, high F1 values of 0.98 are achieved for attack types 1, 3, 5-8, and 16. Recall values of 1.0 demonstrate that these attack types can be accurately detected with zero FNs. This can be indirectly attributed to the effective clustering performed by K-means, which allows the RL model to be trained with accurate reward signals. In addition, the RL model is penalized more for FNs than FPs, tolerating FPs to an extent that is not excessive. This also contributes to higher recall over precision values.

Table 2: Detection performance per attack type

Type	Attack	Accuracy	Precision	Recall	F1
1	Constant Position	0.9892	0.9648	1.0	0.9820
2	Constant Position Offset	0.9853	0.9512	1.0	0.9750

3	Random Position	0.9915	0.9724	1.0	0.9860
4	Random Position Offset	0.9831	0.9454	1.0	0.9719
5	Constant Speed	0.9918	0.9733	1.0	0.9864
6	Constant Speed Offset	0.9895	0.9661	1.0	0.9874
7	Random Speed	0.9924	0.9751	1.0	0.9874
8	Random Speed Offset	0.9913	0.9716	1.0	0.9856
9	Sudden Stop	0.8038	0.5839	0.7080	0.6400
10	Disruptive	0.9610	0.9868	0.9205	0.9525
11	Data Replay	0.9698	0.9826	0.9461	0.9640
12	Delayed Messages	0.9438	0.8445	1.0	0.9157
13	DoS	0.9539	0.9928	0.8922	0.9398
14	DoS Random	0.6411	0.6338	1.0	0.7759
15	DoS Disruptive	0.6353	0.6306	1.0	0.7735
16	Traffic Congestion Sybil	0.9895	0.9661	1.0	0.9827
17	Data Replay Sybil	0.7527	0.6166	0.9612	0.7512
18	DoS Random Sybil	0.7973	0.9507	0.4845	0.6419
19	DoS Disruptive Sybil	0.6501	0.8608	0.0714	0.1318

Figure 3a shows the outcome of K-means clustering for attack 1, where attack instances are discerned and clustered using the acceleration feature. We observe that resulting clusters are elliptical shaped instead of spherical; such shape improves clustering performance by allowing different widths per dimension. In attack 1 (i.e., constant position), the attacker sends fixed position coordinates that do not accurately correlate in time with the reported kinematic information. The attack can thus be discerned using such mismatch. On the contrary, detection of attack 2 (i.e., constant position offset) is more challenging compared to attack 1, as the attacker sends fixed position coordinates by adding/subtracting a constant offset.



(a) Constant position (b) DoS random Sybil

(c) Sudden stop (d) DoS disruptive Sybil

Figure 3: K-means clustering output (K = 2) with misbehaving data in red [6]

2.4.3.2 Moderately detected attacks

Detection performance in Table 2 reveals that for a set of attacks (e.g., attack types 9, 14-15, and 17-18), RL-based detection performs moderately with F1 scores in the range of 0.64 to 0.78. Out of these attacks, attack 18 yields 0.6419 of F1 score with a lower recall of 0.4845, resulting in increased FNs. In attack 18 (i.e., DoS random Sybil), the attacker executes a typical DoS attack in Sybil mode, setting all BSM fields to random values. As shown in Figure 3b, K-means does not effectively cluster data instances. This, in turn, results in low detection performance of RL-based scheme due to the noisy ground truth labels generated via K-means. Attack 18 corresponds to a high-frequency attack with dense data streams and requires a high-dimensional feature vector to be discerned. It appears that K-means algorithm falls short in clustering such data.

A similar performance trend can be identified for attack 9, where RL-based detection reports an F1 score of 0.64, and a 0.5839 precision value. In attack 9 (i.e., sudden stop), the attacker demonstrates a genuine behavior for a limited period and then stops based on a predefined probability. However, there is no certainty that the attacker is eventually going to stop. Due to such behavior, as observed in the overlapping areas of Figure 3c, K-means is not capable of assigning data instances to the closest cluster centers, resulting in noisy ground truth labeling. Further, the attackers' erratic behaviour over time deceives the detector into incorrectly identifying the genuine state as an attack, resulting in an increased FP rate.

It is to be noted that attack type 19 yields an F1 score of 0.1318 which is the lowest detection performance overall in the VeReMi dataset. Attack type 19 (i.e., DoS disruptive Sybil) is a high-frequency attack with dense data streams. Similar to attack type 18, K-means results in poor clustering performance, as shown in Figure 3d. In turn, RL-based detection ends up in low detection scores. Overall, we can observe that the sensitivity of the K-means algorithm appears to be largely dependent on the attack scenario, e.g., density and attack type, which strongly impacts the detection rate.

2.4.4 Benchmark comparison

In this subsection, the impact of noisy labels on the detection performance of our ensemble learning framework is further explored. Considering label provisioning via the K-means algorithm, we comparatively assess the detection outcome of our approach with respect to two benchmark misbehavior detectors, namely support vector machine (SVM) and multilayer perceptron (MLP) classifiers [14], for four representative attack types, as shown in Table 3. For a fair comparison, the models of both techniques were trained on the same data, using the same feature set as in our RL-based detector. For SVM, a two-class model was trained to classify genuine vehicles from misbehaving ones. The selection of model hyperparameters, such as the regularization parameter for SVM and hidden layer sizes for MLP, impacts the classifier outputs. Thus, we conducted experiments with grid search to find the optimal model for each method, following configurations in [1], [14].

Table 3: Detection performance comparison

Attack type	Approach	Accuracy	Precision	Recall	F1
1	K-means + MLP	0.9902	1.0	0.9669	0.9831
	K-means + SVM	0.9418	1.0	0.8031	0.8908
	K-means + RL	0.9892	0.9648	1.0	0.9820
9	K-means + MLP	0.5412	0.2057	0.3007	0.2443
	K-means + SVM	0.5348	0.2066	0.3122	0.2486
	K-means + RL	0.8038	0.5839	0.7080	0.6400
10	K-means + MLP	0.4604	0.4407	1.0	0.6118
	K-means + SVM	0.9385	0.8868	0.9805	0.9313

	K-means + RL	0.9610	0.9868	0.9205	0.9525
16	K-means + MLP	0.6141	0.6084	0.9781	0.7502
	K-means + SVM	0.6711	0.6582	0.9257	0.7693
	K-means + RL	0.9895	0.9661	1.0	0.9827

It can be observed that the detrimental effect of potentially inaccurate or mislabeled training data limits the detection performance of SVM and MLP detectors. When wrong, imprecise, or inconsistent labels are provided as training inputs by the K-means to SVM and MLP schemes, discrimination between genuine and malicious instances becomes erroneous due to the inaccurate ground truth labeling. Interestingly, RL-based detection is shown to be less sensitive to inaccurate labels and exhibits superior or equivalent detection performance compared to SVM and MLP. The fundamental hallmark of RL constitutes the ability to infer optimal sequential decisions in the interactive V2X environment based on rewards/penalties received as a result of previous actions and experiences. This renders detection more robust to noisy training data since RL is capable of partly rectifying the mismatch of training labels. This can be clearly noticed for the effectively detected attacks 1, 10 and 16. For the moderately detected attack 9, detection performance expectedly registers a decline (as discussed in Section 2.4.3.2), albeit not at prohibitive levels.

2.4.5 Analysis of real-time detection

The real-time performance of the RL-based scheme was also assessed in respect of the time taken to detect attacks. For each attack scenario, the time elapsed for the following three steps is measured separately to approximate the overall latency: (i) environment setup, (ii) loading a trained model, and (iii) detection. The CDF graph in Figure 4 illustrates the overall latency for all 19 attack types, while the average latency measured for steps (i)-(iii) is 19.93 ms, 182.12 ms, and 3.15 ms, respectively. We can observe that an approximate average latency budget of 205 ms is required from setting up the environment for streaming data until detecting attacks. Detection latency is in the order of 3-4 ms, which is considered acceptable for many road safety applications, as periodic beacons are usually broadcast with a frequency of 1-10 Hz.

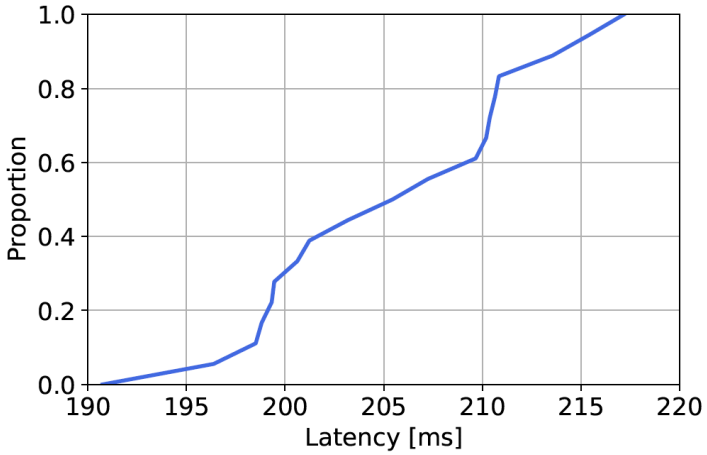


Figure 4: CDF of overall latency for testing datasets [6]

2.5 Summary

To ensure secure and trustworthy condition monitoring of vehicles' health, an ensemble learning framework was introduced for attack detection in vehicular networks. Our approach jointly considered an unsupervised learning module and an RL component to detect various attack types from unlabeled vehicular data instances. While the majority of attack variants can be effectively detected, detection

was curtailed for certain attack types due to the moderate performance of the clustering algorithm and the erratic behavior of attackers. Yet, RL-based attack detection is shown to be more robust to noisy training data compared to its classifier counterparts. In the path forward, we will direct our efforts towards incorporating trust of RSU components into collaborative attack detection, by leveraging the real-time capabilities of our framework.

3 Security mechanisms for in-vehicle data communications

3.1 Communications in the vehicle domain

3.1.1 CAN bus

The Controller Area Network (CAN) bus is a vital component in modern vehicles, serving as a communication network that enables various On-Board Units (OBUs) to exchange data efficiently. However, while CAN bus technology provides numerous benefits for vehicle functionality, it also presents certain security challenges that need to be addressed.

The CAN bus is a robust serial communication protocol that facilitates real-time data exchange between OBUs in vehicles. It enables OBUs responsible for functions like engine control, transmission, brakes, and airbags to communicate seamlessly, enhancing vehicle performance, safety, and efficiency. Despite its widespread adoption, the CAN bus is susceptible to security vulnerabilities due to its inherent design. One significant concern is the lack of built-in security features such as authentication or encryption, delegating all information security through the obfuscation of data.

In technical terms, the CAN protocol is structured around two main layers: the Data Link Layer and the Physical Layer. In the context of high-speed CAN, ISO 11898-1 delineates the responsibilities of the Data Link Layer, which manages logical linking, media access control, and physical coding. Conversely, ISO 11898-2 outlines the functions of the Physical Layer, which include bit encoding/decoding, bit timing, and synchronization.

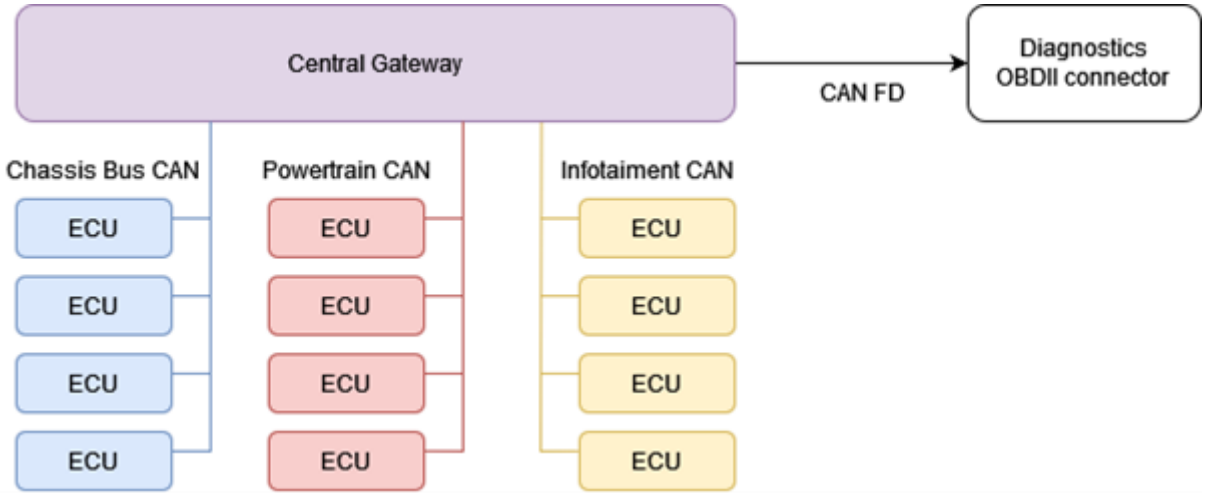


Figure 5: Common in-vehicle CAN bus architecture.

To partly mitigate this security gap, current in-vehicle architectures commonly feature a Central Gateway ECU (as shown in Figure 5) tasked with segregating the most sensitive information, such as brake ECU and engine ECU data, from the rest of the CAN bus domain. When a user accesses the CAN bus through the OBDII interface, they are essentially accessing the data collected by the Gateway ECU from the other ECUs, thus preventing physical access to the most critical ECUs.

3.2 Communications in the public network domain

3.2.1 Cryptography overview

Cryptography serves as the cornerstone of secure communications, using mathematical techniques to protect data from unauthorized access and manipulation. A fundamental aspect is the symmetric and asymmetric encryption, where the first uses a single key for both encryption and decryption process.

Common algorithms, such as Advanced Encryption Standard (AES), exemplify the strength and efficiency of symmetric encryption. Asymmetric encryption, on the other hand, introduces the use of key pairs: a public key for encryption and a private key for decryption (or vice versa). Algorithms like RSA, an established asymmetric encryption method, and Elliptic Curve Cryptography (ECC), another widely used alternative, play a vital role in secure communication using this approach. Hash functions are cryptographic tools that transform input data into fixed-size hash digests. These digests are utilized for integrity verification and creating digital signatures. Examples include SHA-256, which demonstrates the importance of cryptographic hash functions in ensuring data integrity.

Digital signatures contribute to the verification of authenticity and integrity, key mechanisms for security. As shown in Figure 6, created with a private key and verified with the corresponding public key, digital signatures provide a strong means of ensuring the legitimacy of messages or documents. In the aforementioned asymmetric cryptography, there are mechanisms for key exchange, such as Diffie-Hellman, which facilitate the secure exchange of secret keys, forming the basis for establishing secure communication channels. The last but not least security mechanism is Perfect Forward Secrecy (PFS), which further improves security by generating unique session keys for each communication session, even if long-term private keys are compromised.

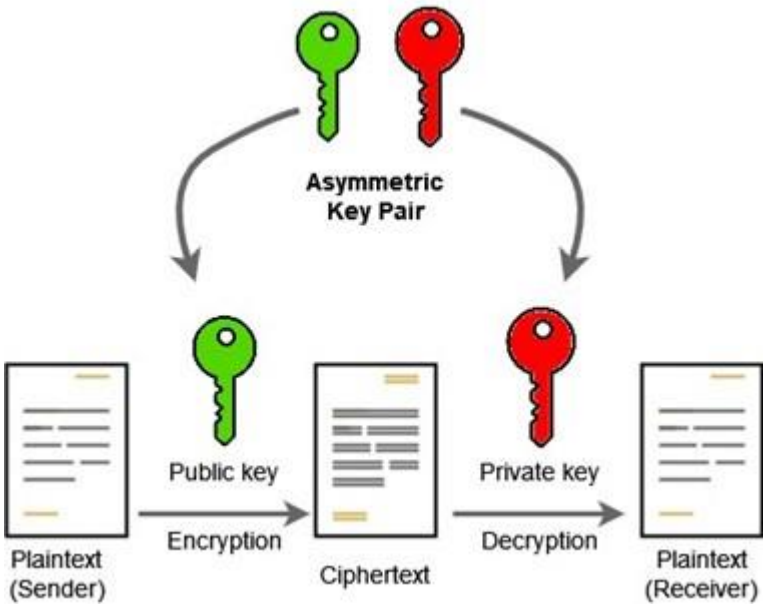


Figure 6: Asymmetric key pair diagram

3.2.2 Public Key Infrastructure (PKI)

A Public Key Infrastructure (PKI) is a fundamental framework in modern cryptography, facilitating secure communication and digital transactions over networks. At its core, PKI comprises a set of hardware, software, policies, and procedures designed to manage the generation, distribution, usage, and revocation of digital certificates and cryptographic keys. The primary components of a PKI include a Certificate Authority (CA), Registration Authority (RA), Certificate Repository, and end entities. The CA acts as a trusted entity responsible for issuing and managing digital certificates, which bind public keys to individuals or entities. The RA assists in the enrollment and validation process, ensuring that certificate requests meet predefined criteria before submission to the CA. The Certificate Repository stores issued certificates and associated public key information, enabling users to access and verify the authenticity of digital identities. End entities, such as users or devices, utilize digital certificates to establish secure communication channels, authenticate identities, and validate data integrity.

3.2.3 TLS 1.3

TLS 1.3 is the latest iteration of the Transport Layer Security (TLS) protocol and has introduced significant enhancements in the security and efficiency of online communications. It stands out for its

advanced implementation of asymmetric cryptography, particularly in optimizing the connection establishment process by reducing message exchanges and efficiently utilizing algorithms such as the aforementioned, Diffie-Hellman. A distinctive feature of TLS 1.3 is its focus on eliminating older versions and insecure protocols, exclusively promoting robust cryptographic methods (cipher suites). Additionally, it prioritizes the concept of PFS, generating unique session keys for each interaction, even in scenarios where long-term private keys may be compromised. This protocol also demonstrates a commitment to reducing latency, contributing to improved performance by streamlining the process of establishing secure connections, thus benefiting the loading speed in applications and other online contexts.

To ensure security in communications within the public network domain, the cryptographic protocol TLS 1.3 will be used in SUCCESS-6G.

3.2.3.1 Handshake process

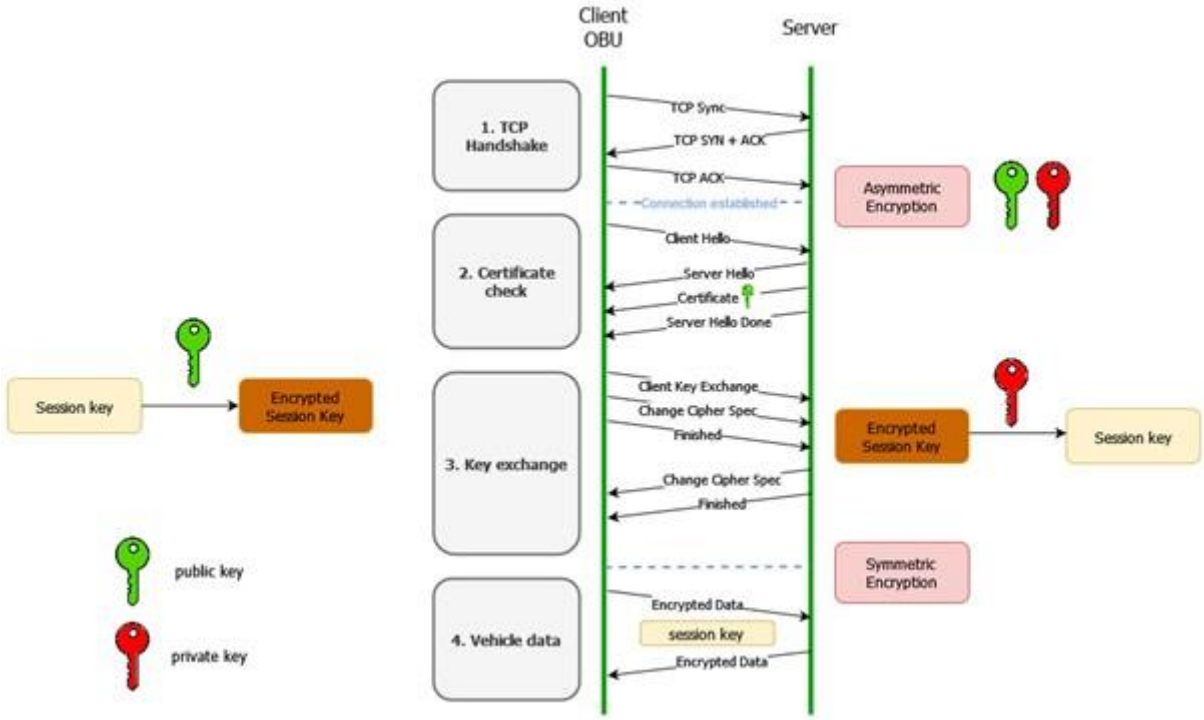


Figure 7: Sequence diagram TLS 1.3 over TCP

As shown in Figure 7, the key steps of the handshake process are as follows:

1. **ClientHello**: The client initiates the handshake by sending a message called ClientHello to the server. This message includes information such as supported TLS versions, cipher suites, and other parameters.
2. **ServerHello**: The server responds with a ServerHello message, selecting the highest TLS version supported by both the client and the server. The server also chooses a cipher suite and sends its digital certificate (if required).
3. **Key Exchange**: The key exchange process varies depending on the chosen cipher suite. In traditional key exchange methods, the server's public key is sent to the client, and a shared pre-master secret is established. In TLS 1.3, key exchange is performed differently to improve efficiency and security, utilizing a process called "Diffie-Hellman Key Exchange" or "Pre-Shared Key" (PSK) modes.
4. **Authentication and Certificate Verification**: If the server provides a digital certificate, the client verifies the certificate's authenticity. This step is crucial for ensuring that the client is communicating with the intended server.
5. **Pre-Master Secret Exchange**: Both the client and the server contribute to generating a pre-master secret. In TLS 1.3, the pre-master secret is derived during the key exchange phase.

6. **Session Key Derivation**: The pre-master secret is used to derive the session keys that will be used for encryption and integrity protection during the secure communication session.

Once the TLS handshake is complete, the client and server can communicate securely using the established session keys. TLS 1.3 introduces improvements to the handshake process, making it more efficient and secure compared to previous versions. Key differences in TLS 1.3 include the elimination of unnecessary round trips and a reduction in the number of messages exchanged during the handshake, resulting in faster and more secure connections.

3.2.3.2 Cipher suites

TLS 1.3 introduces a streamlined set of cipher suites, focusing on modern and secure cryptographic algorithms. The most used cipher suites in TLS 1.3 are designed to provide strong security while minimizing latency, here are some of the commonly used cipher suites in TLS 1.3:

- **TLS_AES_128_GCM_SHA256**: This cipher suite uses the AES-GCM encryption algorithm for confidentiality and SHA-256 for message authentication. It's one of the recommended and widely used cipher suites.
- **TLS_AES_256_GCM_SHA384**: Similar to the previous one, this cipher suite employs the stronger AES-256-GCM encryption algorithm and SHA-384 for message authentication.
- **TLS_CHACHA20_POLY1305_SHA256**: This cipher suite utilizes the ChaCha20-Poly1305 encryption algorithm for confidentiality and SHA-256 for message authentication. It provides an alternative to the AES-based cipher suites.
- **TLS_AES_128_CCM_SHA256**: This cipher suite combines the AES-CCM encryption algorithm with SHA-256 for message authentication. It offers an alternative to GCM and ChaCha20-Poly1305.
- **TLS_AES_128_CCM_8_SHA256**: Similar to the previous one but with shorter authentication tags (8 bytes instead of 16 bytes). This can be useful in situations where a shorter tag is preferred.
- **TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256**: This cipher suite uses the Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) key exchange mechanism with RSA for authentication, AES-128-GCM for encryption, and SHA-256 for message authentication.
- **TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384**: Similar to the previous one but using AES-256-GCM for encryption and SHA-384 for message authentication.

3.2.3.3 Encryption types

TLS uses both symmetric and asymmetric encryption during the communication process. The combination of these two encryption methods is employed to achieve a balance of efficiency and security.

- **TLS Symmetric encryption:** It is used for the bulk of the data transmission. In TLS, a symmetric key is generated for each session, known as the "session key" or "shared secret." This key is used for encrypting and decrypting the actual data being transmitted between the client and the server. Symmetric encryption is more computationally efficient than asymmetric encryption, making it suitable for encrypting the large volume of data exchanged during a session.
- **TLS Asymmetric encryption:** It is utilized for key exchange and authentication during the TLS handshake. During the initial phase of establishing a secure connection, the server presents its digital certificate, which contains its public key. The client can use the server's public key to encrypt a pre-master secret and send it back to the server. The server, possessing the corresponding private key, can then decrypt the pre-master secret. This process enables the

secure exchange of secret information without directly transmitting the symmetric session key.

3.2.3.4 Hash functions

TLS 1.3 uses hash functions primarily for two purposes: the handshake message integrity and the generation of the "Finished" message during the TLS handshake. The hash functions used in TLS 1.3 are part of the HMAC (Hash-based Message Authentication Code) construction. The primary hash functions used in TLS 1.3 are:

- **SHA-256** (Secure Hash Algorithm 256-bit) is a cryptographic hash function that produces a 256-bit hash value. In TLS 1.3, SHA-256 is commonly used in the HMAC construction for message authentication during the handshake.
- **SHA-384** (Secure Hash Algorithm 384-bit) is another cryptographic hash function but produces a longer, 384-bit hash value. In TLS 1.3, SHA-384 is also used in HMAC for message authentication.

These hash functions contribute to the integrity and authenticity of handshake messages exchanged during the TLS handshake process. The use of HMAC ensures that any tampering or modification of handshake messages can be detected by the communicating parties.

4 Conclusions

This deliverable describes the research activities in the SUCCESS-6G-DEVISE project towards enhancing the security of the monitoring information. In particular, we have presented an attack detection mechanism based on ensemble learning, comprising an unsupervised learning module and an RL component, to detect accurately attack vectors from unlabelled vehicular data instances. The applicability of the TLS 1.3 protocol in the communication channels between the OBU client and the server is also discussed as an additional means of enhancing security, data privacy, and integrity.

References

[1] R. Sedar, C. Kalalas, F. Vázquez-Gallego, L. Alonso and J. Alonso-Zarate, "A Comprehensive Survey of V2X Cybersecurity Mechanisms and Future Research Paths," in IEEE Open Journal of the Communications Society, vol. 4, pp. 325-391, 2023, doi: 10.1109/OJCOMS.2023.3239115.

[2] S. So et al., “Integrating plausbility checks and machine learning for misbehavior detection in vanet,” in IEEE ICMLA, 2018, pp. 564–571.

[3] P. Sharma et al., “A machine-learning-based data-centric misbehavior detection model for internet of vehicles,” IEEE Internet of Things Journal, vol. 8, no. 6, pp. 4991–4999, 2021.

[4] T. Alladi et al., “DeepADV: A Deep Neural Network Framework for Anomaly Detection in VANETs,” IEEE Trans. Veh. Technol., p. 11, 2021.

[5] J. Kamel et al., “Veremi extension: A dataset for comparable evaluation of misbehavior detection in vanets,” in 2020 IEEE ICC, 2020, pp. 1–6.

[6] R. Sedar, C. Kalalas, P. Dini, J. Alonso–Zarate and F. Vázquez–Gallego, "Misbehavior Detection in Vehicular Networks: An Ensemble Learning Approach," GLOBECOM 2022 - 2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 2022, pp. 1850-1855.

[7] V. Chandola et al., “Anomaly Detection: A Survey,” ACM Comput. Surv., vol. 41, no. 3, July 2009

[8] T. M. Kodinariya et al., “Review on determining number of cluster in k-means clustering,” International Journal, vol. 1(6), pp. 90–95, 2013.

[9] C. Szepesvari, “Algorithms for reinforcement learning,” Synth. Lect. Artif. Intell. Mach. Learn, vol. 4, no. 1, pp. 1–103, 2010.

[10] C. J. Watkins et al., “Q-learning,” Machine learning, vol. 8, no. 3-4, pp. 279–292, 1992

[11] V. Mnih et al., “Playing atari with deep reinforcement learning,” arXiv preprint arXiv:1312.5602, 2013.

[12] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” J. Comput. Appl. Math., vol. 20, pp. 53–65, 1987

[13] A. Ng et al., “On spectral clustering: Analysis and an algorithm,” Advances in neural information processing systems, vol. 14, 2001.

[14] J. Kamel et al., “Simulation framework for misbehavior detection in vehicular networks,” IEEE Trans. Veh. Technol., vol. 69, no. 6, pp. 6631–6643, 2020.