RCA-IDS: A Novel Real-time Cloud-based Adversarial IDS for Connected Vehicles

Zahra Pooranian*, Mohammad Shojafar†, Pedram Asef‡, Matthew Robinson§, Harry Lees¶, and Mark Longden¶
* Department of Computer Science, University of Reading, UK
z.pooranian@reading.ac.uk
† 5G/6GIC, Institute for Communication Systems (ICS), University of Surrey, Guildford, UK
m.shojafar@surrey.ac.uk
‡ Dept. of Mechanical Engineering, University College London, London, UK
pedram.asef@ucl.ac.uk
§ Dept. of Engineering and Technology, University of Hertfordshire, Hatfield, UK
m.robinson20@herts.ac.uk
¶ R&D Group RL Auto Ltd, Basingstoke, UK
{harry.lees, mark.longden}@rlauto.com

***Abstract*—This paper focuses on the requirement for creating novel frameworks to monitor and identify cyberattacks in Connected Vehicles (CVs). The health of the sensors in CVs becomes crucial when performance predictions and communication-related errors can compromise the resilience of the sensory network. To meet the evolving demands of connected vehicle (CV) systems, Intrusion Detection Systems (IDS) must be regularly updated and tailored as powerful monitoring entities. To equip cloud-tied operators with the ability to comprehend unusual sensor data originating from vehicles at the cloud level, we designed an innovative Real-time Cloud-based Adversarial IDS called RCA-IDS. This system exclusively focuses on detecting and explaining instances of sensor data manipulation caused by poisoning attacks. Two attack mechanisms were created utilizing random-based and silhouette-based clustering methods. Subsequently, two defence mechanisms based on multi-layer neural network-type deep learning were proposed to counter these attacks. The newly introduced RCA-IDS demonstrates a minimum accuracy of 90% in detecting cyberattacks.**

***Index Terms*—Connected Vehicle, Machine Learning, Sensors, Cyberattack, Intrusion Detection Systems (IDS).**

I. Introduction

Implementing vehicle networks using the Intelligent Transportation System (ITS) provides road users more connectivity, safety, and convenience. The use of ITS assists in advancing Connected Vehicles (CVs) and offers traffic management capabilities [1], [2] for system administrators. The growing degree of self-sufficiency in vehicles, the incorporation of numerous wireless communication technologies, and the trend of network virtualization and cloud computing have led to elevated concerns regarding potential cyber-attacks in the achievement of connected and self-driving cars [3], [4]. Connected and self-driving cars are considered a system critical to preserving life. As a result, implementing adequate security measures is crucial to shield CVs from cyber-attacks and ensure the safety of the driver, passengers, other motorists, and the environment. A security mechanism with great potential for safeguarding networks is the intrusion detection system (IDS), which monitors network components' input and output traffic. Several varieties of IDS exist, categorized by their methods of identifying possible intrusions. A kind of IDS known as anomaly-based detection uses a preset model to define normal behaviour and compares incoming traffic against this model. When unusual behaviour is detected, it is classified as an attack. Despite considerable research, traditional IDS solutions face significant obstacles in detecting novel attacks. They cannot provide the required level of detection effectiveness, especially in the context of highly dynamic vehicular networks. Hence, academics and industry have attracted significant attention to using Cloud-based solutions in response to the 5G technology trend. Recent research works [5], [6] have shown that the accuracy of detection systems can be significantly increased by incorporating Machine Learning (ML) capabilities into IDS.

ML algorithms can precisely forecast patterns in data, but some data may originate from untrustworthy and uncertain sources. Attackers can exploit this weakness in *Adversarial Machine Learning (AML)* attacks. A specific type of AML is called a *Poisoning attack* when attackers inject malicious data into the training dataset to decrease the model's accuracy by disrupting the learning process. *Label-flipping* is a form of data poisoning where attackers can control the label assigned to the training samples and manipulate them to reduce the system's performance substantially.

In [7], AML techniques focus on two main aspects: i) Attack Complexity, which aims to simplify the creation of a malicious attack, and ii) Attacker's Knowledge, which pertains to the attacker's understanding of the architecture, algorithm, and training examples, to gain insight into the detector. A type of attack known as a *white-box attack* occurs when the attacker knows training data, features extracted from applications, or the architecture, as described in some approaches such as [8]). Alternatively, the attack is classified as a *black-box attack* [9] when the adversary's understanding is restricted.

The concept of adversarial specificity can take either a *targeted* or *non-targeted* approach. The attacker can fool a classifier in detection systems in a targeted manner by

predicting all adversarial samples as a class. This strategy also increases the likelihood of achieving the desired targeted adversarial outcome. On the other hand, non-targeted attackers can target a class arbitrarily. They achieve this by carrying out various targeted attacks and choosing the one with the minor disturbance or reducing the likelihood of the correct category.

Numerous studies in the literature have concentrated on detecting and addressing the issue of poisoning attacks.For instance, an algorithmic technique assesses how the efficiency of the learning algorithm can be affected by each training sample [10]. Although this approach may be successful in specific instances, it is not universally applicable to a large dataset. Other defensive tactics, such as *outlier detection*, are employed to recognize and eliminate suspicious samples. Nevertheless, this technique's performance is restricted, particularly concerning its accuracy when faced with label-flipping attacks [11]. Another research area concentrates on developing learning tactics that can be utilized to flip labels. There are two main categories of solutions to this problem. The initial category entails acquiring knowledge directly from the flipped labels, while the second category focuses on clean data. In the initial scenario, the label-flipping module identifies accurately labelled data [12], [13]. It then adjusts the label changes to restore the data's terms within the loss function. The efficacy of this approach is notably influenced by the accuracy of cleaning labels and the precision of estimating the flipped samples. The second group of techniques employs an extra collection of malicious data to direct the learning process in handling flipped data [14]. Both categories have common drawbacks despite showing good outcomes. They attempt to correct flipped labels or adjust the weights for data samples, which results in errors for specific data samples.

Motivated by these considerations, this study introduces a Real-time Cloud-based Adversarial IDS (RCA-IDS) architecture that addresses identifying malicious data in Connected Vehicle (CV) systems. This paper illustrates a fleet-based scenario where vehicle sensor data, including tire pressure, temperature, and location, is measured, gathered, and sent to a cloud server via a 5G cellular network. We assume that the attacker's ability is the weakest. The attacker does not have enough information about the loss function, learning algorithm, features, or initial training data. We demonstrate that improved outcomes can be achieved if the system detects and retrains incorrect labels and our suggested semi-supervised training method is employed. Thus, we propose a solution that addresses mislabeled data points and enhances the classification algorithm' accuracy. Our approach requires having correct labels for a small portion of the training set while ignoring returned labels associated with other data. We then train a multi-layer neural network utilizing this selected data semi-supervised. To sum up, this study's key contributions are:

- Our primary focus is on efficiently detecting intrusions across CV systems, and to this end, we introduce RCA-IDS architecture that enables the testing of flipped data.
- We suggest two label-flipping poisoning techniques that target the deep learning-based detection of CVs: one involves randomly flipping labels without knowing which feature is significant. In contrast, the other technique involves silhouette clustering to determine which sample is appropriate to flip its label.
- We propose two defence mechanisms to combat label-flipping attacks in CV systems: the first involves K-means clustering. In contrast, the second involves a semi-supervised DL-based technique that leverages Label Spreading (LS) and Label Propagation (LP) algorithms to predict new labels for the training set.
- We deployed our proposed cyber detection algorithms to the cloud and tested their effectiveness in a pilot trial using live vehicles.
- We perform experiments on a real-world CV systems dataset from RL Capital company with four types of features. The experiments include two attack scenarios and are compared with the non-attack method. We thoroughly review the resulting trade-offs.

This paper is organized as follows. In Section II, the problem definition, architecture, and related components are discussed in detail. Section III outlines the proposed attack model, inspired by Adversarial Machine Learning algorithms (AML), and presents the proposed defence strategies against the raised attacks in Section IV. The real-time verification algorithm presents in Section V. Section VI presents the experiments' results. Finally, Section VII summarises the findings and outlines plans.

II. System model and proposed architecture

Section II-A presents the problem definition. Then, the proposed intrusion detection architecture is introduced in Section II-B. Section II-C explains the proposed classification algorithm used in the paper.

A. Problem definition

Let's examine the datasets in the following manner.

$$\mathcal{D} = \{(x_i, y_i) \in (\mathcal{X}, \mathcal{Y})\}, \quad i = 1, \ldots, n \tag{1}$$

Here, n represents the count of malicious samples. If x_i possesses the j feature, then x_{ij} equals 1. If not, x_{ij} is set to 0, and $\mathcal{X}$ is a subset of a k-dimensional space, where $\mathcal{X}$ consists of elements in the set $\{0,1\}^k$. The samples are labelled with y_i values that belong to the set $\{0,1\}$, and the $\mathcal{D}$ has an undisclosed distribution over $\mathcal{X} \times \mathcal{Y}$. It is assumed that the training set is defined in the following manner.

$$\mathcal{S} = \{(x_k, y_k)\}, \quad k = 1, \ldots, m \tag{2}$$

where $\mathcal{S}$ denots the set of lables. The label-flipping attack aims to discover a set, denoted as P, which consists of samples from $\mathcal{S}$. The attacker intends to minimize the desired target by flipping the labels. For simplicity, we assume that the attacker aims to maximize the loss function, defined as $\mathcal{L}(w, (x_j, y_j))$.

Definition 1: The specific type of poisoning attack is called *Label Flipping Attack (LFA)*, in which the attacker attempts to modify feature labels using specific algorithms, which alters the range of each sample within a cluster.

B. Proposed architecture

Fig. 1 displays our real-time cloud-based adversarial IDS (RCA-IDS) architecture, which is introduced as a solution for detecting malicious attacks within CV systems. In this architecture, each vehicle in the fleet is equipped with a temperature and pressure sensor on every wheel, which measures the internal pressure and temperature. Additionally, the Global Positioning System (GPS) receiver determines the position of each vehicle. A 5G Telematic Unit (TU) is used to connect the location, pressure, and temperature sensors. This TU collects the data and transmits sensor updates to a cloud-based database. To send this data to the database, the TU employs a 5G modem with 4G and satellite feedback in case of any manipulation, ensuring reliable communication in this new setup. Within this sensory system, temperature is recorded in degrees Celsius, pressure is recorded in Pounds per Square Inch (PSI), and location is recorded in degrees of Latitude and Longitude. In the RCA-IDS system, we assume that the vehicles communicate, and the incoming data stream data obtained from CVs is stored in a cloud-based database. Additionally, it is assumed that an attacker could access some CV sensors, allowing them to manipulate data transmitted between vehicles or the cloud. Consequently, the data traffic of each vehicle could potentially contain information from malicious vehicles, which are represented in the figure by the devil symbols. Each vehicle generates a unique feature vector with different labels, whether malicious or benign. The attackers aim to deceive machine learning models and prevent malicious detection by introducing perturbations into the data. Consequently, in this architecture, adversaries can infiltrate the dataset and alter the labels by introducing perturbations to the existing ones. The last element of our framework encompasses our suggested defence algorithms and a machine learning model, forming the detection system. Using our architecture, it is possible to enhance the resilience of the detection system against label-flipping attacks, subsequently enhancing the classification accuracy between malicious and benign entities. The following section details our proposed attack and defence algorithms.

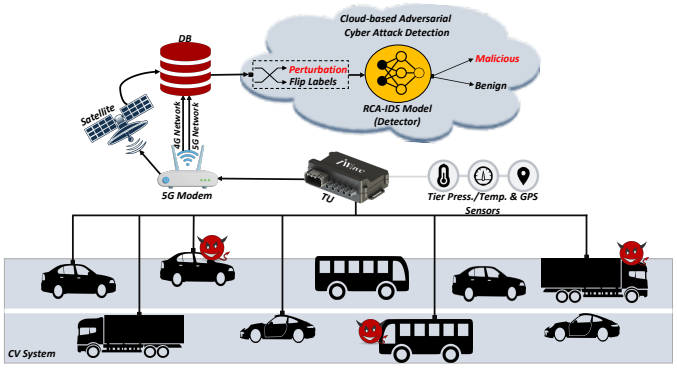


Fig. 1: Architecture overview of Real-time Cloud-based Adversarial IDS (RCA-IDS) in CVs environment; Temp:= Temperature; Press:= Pressure; TU:= Telematic Unit; DB:= Database.

C. Classification algorithm

The paper incorporates a Sequential deep-learning model in order to classify the samples. Fig. 2 illustrates the Sequential architecture proposed for the classification algorithm. The figure shows that the classification algorithm utilizes three sequential layers of dense, each with 32, 16, and 8 units, and activation functions of tanh, relu, and tanh, respectively. After these layers, a dense layer is utilized with a Sigmoid activation function to complete the classification algorithm, resulting in the data being classified.

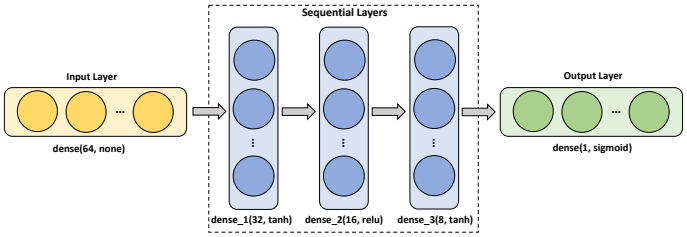


Fig. 2: The architecture of the proposed classification algorithm; (A,B) = (units, activation).

III. Adversarial approaches for intrusion detection system

Our attack approach is targeted, which involves creating misclassified samples to infect a specific vehicle. This section details our attack strategies, which are categorized into two scenarios discussed in subsequent subsections. Table I lists this paper's primary symbols and notations.

TABLE I: Main notations.

Notation	Description
$\mathcal{X}$	Input data, $x \in \{0,1\}^m$, $\mid \mathcal{X} \mid = n$
$\mathcal{Y}$	Benign labels in the classification problem, $y \in \{0,1\}^m$
$\mathcal{X}^*$	Adversarial sample
$\mathcal{Y}^*$	Adversarial labels in target adversarial sample
$\mathcal{M}$	Number of features, $a \in \mathcal{M}$, $\mid \mathcal{M} \mid = m$
F	Machine Learning model, $F : X \rightarrow Y$
$\mathcal{T}$	Percentage of features

An attack plan shows how attackers can compromise the system by considering their goals, understanding, and capabilities. Below, we provide a brief overview of each attack scenario.

Scenario 1: The attacker employs a random approach to modify the labels of malicious applications without knowing which features are most important. This attack is called a *Random-based Label Flipping Attack (RLFA)*.

Scenario 2: The *Silhouette-based Label Flipping Attack (SLFA)* is an attack where the attacker calculates the *Silhouette Value (SV)* and attempts to flip the labels of the training set by adding some perturbation to the existing labels when the SV is negative.

We divide the training and testing datasets into *Malicious* and *Benign* groups based on the class parameters for both scenarios. We execute each algorithm ten times and choose

Algorithm 1 Random-based Label Flipping Attack (RLFA)

Input: $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{T}$
Output: $\mathcal{X}^*$, $\mathcal{Y}^*$

1: Randomly select $\mathcal{T}$ features from $\mathcal{X}$
2: **for** each attribute $a \in \mathcal{T}$ in $\mathcal{X}$ **do**
3: **if** $(\mathcal{X}[a] = 0)$ **then**
4: $\mathcal{X}[a] \leftarrow 1$
5: $\mathcal{X}^* \leftarrow x$
6: **if** $(F(\mathcal{X}^*) = \mathcal{Y}^*)$ **then**
7: break
8: **end if**
9: **end if**
10: **end for**
11: **return** $\mathcal{X}^*$, $\mathcal{Y}^*$

the average values for each parameter. Moreover, we select specific proportions of features from the malicious samples and modify their labels. To be more precise, we opt for a feature that has not been previously chosen and alter its labels. Then, we include these altered malicious samples in the training set and utilize classification techniques to classify the dataset.

A. Attack strategy: Random-based Label Flipping Attack (RLFA)

We use a randomization method to implement LFA. In this approach, the attacker randomly manipulates the malicious applications' labels without knowing which feature is prominent. We call this attack Random-based Label Flipping Attack (RLFA).

Description of the Algorithm 1. The list variable $\mathcal{T}$ in line 1 refers to the features used. To clarify, we select and modify one feature from the list and subsequently repeat this procedure for all the features in the list. We analyze the chosen sample during each loop iteration and focus on the selected feature a from the $\mathcal{T}$ set. We modify instances where the element has a zero value to become one, then save that modified sample in $\mathcal{X}^*$ (lines 2-5 of Algorithm 1). Afterwards, verifying the altered sample and determining whether it has transformed into a benign sample is crucial. To determine whether this condition is met, we depend on evaluating $F(\mathcal{X}^*) = \mathcal{Y}^*$. If this evaluation satisfies the requirement, we can classify $\mathcal{X}^*$ as an adversarial sample generated by Algorithm 1.

B. Attack strategy: Silhouette-based Label Flipping Attack (SLFA)

Silhouette clustering is a method that can interpret and confirm data clusters' coherence, providing a brief graphic representation of object categorization. The Silhouette clustering technique is used to implement LFA, and we call this a *silhouette-based Label Flipping Attack (SLFA)*. This approach introduces a measure known as the *SV*, which quantifies the similarity or cohesion of an object within its cluster compared to other clusters or separations, and it ranges from $[-1, 1]$. In SLFA, we assign a range of values from $[-1, 1]$ to each sample to indicate its membership in the correct cluster. If SV has a negative value, it indicates that the selected sample is a potential candidate for label flipping and firmly belongs to a distinct cluster according to the silhouette algorithm. Therefore, we modify the label of such a sample. This paper adopts a Euclidean distance approach to determine SV. Consider L_i as the label assigned to the i-th sample out of the total n samples in the dataset. Then, we can express it as equation (3).

$$L_i = \begin{cases} (x_i, y_i), & SV > 0 \\ (x_i, |1 - y_i|), & \text{otherwise} \end{cases} \tag{3}$$

Algorithm 2 Silhouette-based Label Flipping Attack (SLFA)

Input: $\mathcal{X}\_trn, \mathcal{Y}\_trn$
Output: $Poisoned\_\mathcal{Y}\_trn$

1: $\mathcal{M}_K \leftarrow$**Generate Model** with two Clusters Using K-means clustering
2: $lbl \leftarrow$**Predict** labels of $\mathcal{X}\_train$ using $\mathcal{M}_K$
3: $SV \leftarrow$ **Compute** Silhouette values using $\mathcal{X}\_trn$ using lbl
4: **for** each $row \in \mathcal{X}\_trn$ **do**
5: **if** $(SV[row] \leq 0)$ **then**
6: $Poisoned\_\mathcal{Y}\_trn[row] = abs(1 - \mathcal{Y}\_trn[row])$
7: **end if**
8: **end for**
9: **return** $Poisoned\_\mathcal{Y}\_trn$

Description of the Algorithm 2. The algorithm presents a new method called SLFA, which involves changing the training sample's label. The foundation of this technique lies in the principles of the K-means clustering algorithm. The algorithm starts in line 1 by building a machine learning model using K-means to classify the training samples ($\mathcal{X}_{trn}$) into *two* clusters, and then in line 2 predicts a new label for each sample. Subsequently, we compute the *Silhouette Value* for both the samples and the predicted labels assigned to them (line 3). As mentioned, values near 1 indicate that the sample is appropriately classified within the cluster, while SVs below 1 and near -1 indicate that the data point is categorized incorrectly. In our approach, we altered the sample's label with SV below zero, meaning we may have picked the samples that could belong to the alternate cluster (lines 4-8).

IV. Defensive strategies against attacks

This section provides explanations of two countermeasures to address the attacks mentioned above. Specifically, we explain K-mean-based Clustering Defence (KCD) and Label-based Clustering Defence (LCD), which are presented in Sections IV-A and IV-B, respectively. We assume that the data is partially labelled. Defence mechanisms analyze the training samples' validation data to identify potential instances of label flipping. Then, new labels are predicted for the data, and replace the original labels with the new predicted labels.

A. Defence strategy: K-mean based Clustering Defence (KCD)

This subsection outlines the proposed techniques to defend the classification model against the adversarial RLFA attack. The defence methodology includes the process of retraining the classification algorithm [15]. The critical distinction between the updated retrained classification and the existing version lies in including the poisoned data within the

training dataset. Classification algorithms work by evaluating the similarity between samples. Classifiers that employ the similarity approach can predict the label of a test set sample by evaluating its similarities with the training set samples' labels and the samples' inter-similarity. This paper utilised the K-mean classification model, and the nearest-neighbour approach was employed for the suggested defence method. Our method is founded on the concept that when examining a sample's neighbourhoods, the distances will have distinct labels and that two close samples will have identical labels. The underlying idea is that samples with similar features and matching labels are unlikely to have been tampered with. However, it is common for training datasets to contain nearby points with differing labels. The KCD technique aims to correct the perturbed training set by utilising the nearest neighbour of a given sample. If we aim to locate the most similar vectors in training set to vector x, which is a part of the test set, y would be considered the most similar element if equation (4) is satisfied.

$$dis(x, y) \leq dis(x, z) \quad \forall z \in train\_set \tag{4}$$

The expression $dis(x, y)$ refers to the distance between x and y, commonly referred to as the *distance*. Various techniques have been discussed previously for computing the distance, including the Hamming distance and Euclidean distance. We will demonstrate that the outcomes of all these approaches are comparable due to the samples' nature, which means there is no ambiguity in selecting a particular method. Accordingly, we can summarize the proposed intrusion detection method as follows:

Description of the Algorithm 3. The KCD algorithm is introduced in this paper to address the label-flipping issue in the training samples, which is based on the K-means clustering approach. In the initial phase, we apply the K-means algorithm on the $\mathcal{X}_{trn}$ samples to generate a model that categorizes them into two clusters and predicts each sample's label (lines 1-4). Subsequently, we compute the Euclidean distances, mean values for the samples, and predicted labels in lines 5-6. Our proposed method involves flipping the labels of the samples with a distance that is lower than the mean value. This way, we can likely choose the appropriate label for the values within the same cluster (lines 7-12).

Algorithm 3 K-mean based Clustering Defence (KCD)

Input: $\mathcal{X}$, $\mathcal{X}^*$, $\mathcal{Y}$, $\mathcal{Y}^*$
Output: $\mathcal{Y}_{Corrected}$

1: $\mathcal{X}_{trn} \leftarrow \mathcal{X} \cup \mathcal{X}^*$
2: $\mathcal{Y}_{trn} \leftarrow \mathcal{Y} \cup \mathcal{Y}^*$
3: $\mathcal{M}_K \leftarrow$ **Generate Model** with two clusters using K-means clustering $(\mathcal{X}_{trn}, \mathcal{Y}_{trn})$
4: $lbl \leftarrow$ **Predict** labels of $\mathcal{X}\_trn$ using $\mathcal{M}_K$
5: $dis \leftarrow$ **Compute** Euclidean Distances using $\mathcal{X}\_trn$ and lbl
6: mean $\leftarrow$ **Compute** Mean value for $\mathcal{X}\_trn$
7: **for** each $row \in \mathcal{X}\_trn$ **do**
8: **if** $(dis[row] \leq mean[row])$ **then**
9: $\mathcal{Y}_{Corrected}[row] = 0$
10: **else**
11: $\mathcal{Y}_{Corrected}[row] = 1$
12: **end if**
13: **end for**
14: **return** $\mathcal{Y}_{Corrected}$

B. Defence strategy: Label-based Clustering Defence (LCD)

In this section, we develop an LCD algorithm that prioritizes Semi-Monitoring Learning (SML) techniques. Hence, validation data is sent as input to SML to predict new labels for each instance and then rank the predicted labels. The primary objective of LCD is to identify examples where the labels are expected to be correct in the perturbed training set. Once we have identified these samples, we must label them to the Semi-Supervised Learning (SSL) algorithm. Establishing a validation set is necessary to supervise the training procedure and select monitor parameters. The LCD approach starts by ranking the samples in each category and saving the labels with the highest scores. A multi-way classification neural network ranks the points if a clean set is unavailable. Therefore, the original training dataset was used to train the ranking system. This defence algorithm consists of training the classifier while clean labels are accessible and separating data with flipped and clean labels. In the initial stage, the defence strategy involves using the Label Propagation (LP) algorithm to allocate labels to data points that do not have labels. Next, the Label Spreading (LS) method is employed in the subsequent phase to reduce the noise that occurs while labelling the samples. The objective of the LCD method is to create a technique that functions like an ensemble learning approach, leveraging propagation models to forecast flipped labels. To achieve this goal, a framework consisting of two stages for reverse label learning is proposed. The subsequent sections explain LP and LS.

- **Label Propagation (LP).** LP is a machine learning algorithm with a semi-supervised approach that assigns labels to unlabeled samples. It operates by assigning labels to a small subset of the dataset and using that to make classifications. Then, it predicts labels for the remaining unlabeled data points. LP can reveal the community organization within complex networks [16]. Unlike other methods, LP has a significantly shorter processing time and does so without the need for prior knowledge about the data points before propagation. However, LP may generate multiple solutions for any given set of samples.
- **Label Spreading (LS).** The LS algorithm belongs to a class of propagation methods that leverage a normalized Laplacian graph and apply soft clamping on an affinity matrix to adjust the labels. Furthermore, it can make the loss function more noise-resistant by reducing its regularization properties [17]. The LS algorithm iteratively operates on an altered data point graph and normalizes the edge weights by calculating the matrix of the normalized Laplacian graph.

In the initial step of the LCD approach, the validation dataset is used to train the LP and LS, and after that, to predict labels for the training samples. During the second phase, all accessible labels, including LP output, LS output, and poisoned labels, are gathered, and select the most frequently

occurring one through voting. The overall LCD algorithm based on semi-supervised learning is presented in *Algorithm 4*.

Description of the Algorithm 4. This algorithm proposes a semi-supervised defence mechanism that relies on Label estimation. In lines 3-5, the LS algorithm is employed to forecast the training data labels through training on the validation data. Likewise, in lines 6-8, the LP, another semi-supervised technique, is utilized to forecast the training data labels. To complete the LCD method, the last stage entails merging the outcomes from the three methods and the poisoned label. This is accomplished through a voting mechanism to determine the label for each training sample.

Algorithm 4 Label-based Clustering Defence (LCD)

Input: $\mathcal{X}\_val, \mathcal{Y}\_val, \mathcal{X}\_trn, Poisoned\_\mathcal{Y}\_trn$
Output: $\mathcal{Y}_{Corrected}$
1: $\mathcal{X} \leftarrow \mathcal{X}\_trn$
2: $\mathcal{Y} \leftarrow \mathcal{Y}\_val$
3: $\mathcal{M}_{ls} \leftarrow$**Generate Model** using LS algorithm
4: **Fit** the $\mathcal{M}_{ls}$ Model on $\mathcal{X}$ and $\mathcal{Y}$
5: $\mathcal{L}_{ls} \leftarrow$ **Predict** labels of $\mathcal{X}\_trn$ using $\mathcal{M}_{ls}$
6: $\mathcal{M}_{lp} \leftarrow$ **Generate Model** using LP algorithm
7: **Fit** the $\mathcal{M}_{lp}$ Model on $\mathcal{X}$ and $\mathcal{Y}$
8: $\mathcal{L}_{lp} \leftarrow$ **Predict** labels of $\mathcal{X}\_trn$ using $\mathcal{M}_{lp}$
9: $\mathcal{Y}_{Corrected} =$ **Voting**$(Poisoned\_\mathcal{Y}\_trn, \mathcal{L}_{ls}, \mathcal{L}_{lp}, \mathcal{L}_{cnn})$
10: **return** $\mathcal{Y}_{Corrected}$

V. Real-time Verification

We integrate the proposed cybersecurity detection models, KCD and LCD, on a cloud-assisted connected vehicle environment (called RCA-IDS) to detect our proposed RLFA and SLFA adversarial attacks. The connected vehicle streams real-time sensor data (benign and malicious samples) to the cloud network. The complete data stream is inserted into the proposed cybersecurity detection algorithms to show the robustness of our proposed algorithms in detecting the poisoned data. The proposed defence ML-based algorithms on the cloud perform analytics on the received data and raise the flag of whether it is malicious or benign. Our system accesses the raw data and the analytics results and displays them side by side to the end user within our user-friendly front-end display. The overall RCA-IDS algorithm is presented in *Algorithm 5*.

Description of the Algorithm 5. This algorithm proposes real-time adversarial detection verification on live vehicles in the pilot trial. The algorithm loads the pre-trained AML models (lines 1-3). These models are KCD or LCD models from our training phases. Then, line 4 is used to predict the labels of the testing data by AML models. If the model predicts the label equal to 1 (0) for each predicted label, it informs the detector module by raising a significant security alarm. It sets the sample status to Malicious (Benign) (lines 5-11). The output of the RCA-IDS algorithm is the alert flag shown on the customer dashboard (line 12).

Algorithm 5 Real-time Cloud-based Adversarial IDS (RCA-IDS)

Input: $\mathcal{X}$, $\mathcal{X}^*$, $\mathcal{Y}$, $\mathcal{Y}^*$
Output: $flag$
1: $\mathcal{X}_{test} \leftarrow \mathcal{X} \cup \mathcal{X}^*$
2: $\mathcal{Y}_{test} \leftarrow \mathcal{Y} \cup \mathcal{Y}^*$
3: $model \leftarrow$ **Load** pre-trained AML (KCD/LCD) model
4: $Labels \leftarrow$ **Predict** labels of $\mathcal{X}\_test$ using $model$
5: **for** each $label \in Labels$ **do**
6: **if** $(label == 1)$ **then**
7: $flag = Malicious$
8: **else**
9: $flag = Benign$
10: **end if**
11: **end for**
12: **return** $flag$

VI. Experimental evaluation

The simulation results of our proposed methods for attack (RLFA and SLFA) and defence (KCD and LCD) are presented in this section.

A. Simulation setup

The following section explains the evaluation metrics, datasets, features, parameters used for classification, and the defence algorithm being compared.

1) Test metrics: In order to conduct a thorough assessment of our attack and defence methods, we utilize the subsequent indices:

- *True Positive (TP):* refers to a situation where a malicious sample is accurately identified as malicious.
- *True Negative (TN):* denotes the count of legitimate applications accurately detected by the classification algorithm.
- *False Positive (FP):* refers to the count of benign applications that have been wrongly classified.
- *False Negative (FN):* refers to the count of malicious files inaccurately classified as normal samples.
- *False Negative Rate (FNR):* can be used to identify samples where the condition is true, but the assessment fails to detect it. This measurement can be computed using the following formula:
$$FNR = \frac{TP}{TP + FP}. \tag{5}$$
- *True Positive Rate (TPR):* This measure reflects the accuracy of predictions for the positive class. It is calculated as the ratio of accurate predictions to all predictions using the following formula:
$$TPR = \frac{TP}{TP + FN}. \tag{6}$$
- *Accuracy:* This value indicates the proportion of accurate predictions made by the algorithm compared to the total number of input examples. When the accuracy value is higher, the algorithm will more likely correctly identify the samples' labels. As a result, we obtain the following:
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{7}$$
- *Precision:* This refers to the portion of relevant samples present within the retrieved samples. As a result, we obtain the following:

$$Precision = \frac{TP}{TP + FP}. \tag{8}$$

- *Recall:* This value corresponds to the fraction of retrieved samples to the overall number of relevant examples. Thus, recall and precision measures are rooted in the concept and assessment of relevance. We can express this as:

$$Recall = \frac{TP}{TP + FN}. \tag{9}$$

- *F1-Score:* This metric is known as the harmonic mean of Recall and Precision and is defined in the following manner:

$$F1 - Score = \frac{1}{\frac{1}{Recall} + \frac{1}{Precision}} = 2 * \frac{Precision * Recall}{Precision + Recall}. \tag{10}$$

- *Area Under Curve (AUC):* determines the best class prediction model by considering all potential thresholds. The primary objective of AUC is to overcome the issue of unbalanced samples within a dataset and to avoid overfitting the model to the class with a higher number of instances. We can express this as:

$$AUC = \frac{1}{2}\left(\frac{TP}{TP + FP} + \frac{TN}{TN + FP}\right). \tag{11}$$

2) Datasets: We conducted our experiments on a server provided by RL Automotive using the AutoAlign dataset, which contains data on the Temperature, Pressure, and Location of numerous fleets and vehicles. AutoAlign is an R&D development project to determine wheel misalignment in real time that, saves fuel costs and reduces air pollution.

3) Features: The AutoAlign dataset records the sensor data in the table. The vehicle sensor received data are shown in Table II. This paper considers various sample features like temperature, pressure, latitude, and longitude. We summarize them as follows:

- *Temperature:* This sensor has been specifically designed to measure a tire's highly transient surface temperature, providing invaluable information for chassis tuning and driver development.
- *Pressure:* The tire-pressure monitoring system, known as Pressure, monitors the air pressure within the pneumatic tires on vehicles. This system provides the driver with real-time information on tire pressure using a gauge, pictogram display, or a primary low-pressure warning light.
- *Latitude:* It is a GPS sensor to get a vehicle's position consisting of longitude and latitude.
- *Longitude:* It is a GPS sensor to get a vehicle's position consisting of longitude and latitude.

4) Parameter setting: Evaluating the performance of a model solely based on the training dataset can result in a biased score. To overcome this, the model is assessed using a separate held-out sample to provide an unbiased estimation of its performance. This approach, commonly called the train-test split, evaluates algorithms. In this study, the dataset is randomly divided for each test, with 60% designated as training samples, 20% as validation samples, and 20% as testing samples. We select four features from the dataset. Unique seeds are used for each of the 10 sets of training, validation, and testing. We repeated this process ten times for each algorithm and calculated the average results. The experiments were conducted on an Intel Xeon CPU E5-2667 3.3GHz virtual machine with 190 GB RAM and 32 CPU cores, running Ubuntu server 18.04.

TABLE II: Vehicle sensor received data; Temp:= Temperature; Press:= Pressure; Lat:= Latitude; Lng:= Longitude.

Timestamp	Sensor ID	Temp	Press	Lat	Lng
12/18/21 23:46	001e1c	6	124.154	51.036	-2.027
12/18/21 23:45	001e2	10	105.721	41.503	-3.631
12/18/21 23:45	001e58	9	132.086	51.543	-2.651
12/18/21 23:46	23097a	5	106.365	50.718	-3.083
12/18/21 23:46	005ec0	8	70.779	51.2	-1.111
12/18/21 23:46	002321cb	4	78.961	61.126	-1.35

B. Training Phase: Experimental results

We employed the Multilayer Perceptron (MLP) model to classify benign and malicious data. The benign data were obtained from the AutoAlign dataset, while the malicious data were generated using outlier data. This section tests our proposed attack algorithms (RLFA and SLFA) on the classifier we initially trained and validates our defence algorithms (KCD and LCD) on the AutoAlign dataset.

1) Comparing methods based on Accuracy and Loss: The goal of this test scenario is to compare the accuracy (as shown in Table III) and loss (as shown in Table IV) of the attack and defence methods when compared to data that has not been subjected to any attacks, i.e., no-attack on train, validation, and test data.

The classifiers' accuracy is noticeably above 99% when there is no attack. The loss value without an attack is 0.18%. However, the RLFA algorithm adds random noise to the samples, resulting in an accuracy value of about 52% but a loss value of about 18%. This method does not make specific modifications, so the accuracy is lower than the SLFA attack. Table III demonstrates the increase in accuracy through the proposed defence mechanisms.

TABLE III: Comparing the accuracy of algorithms on train, validation, and test samples (%).

Algorithms	Train	Valid	Test
No attack	99.795920	99.857146	99.833333
RLFA	80.020410	80.047619	52.700001
SLFA	87.653059	88.190478	68.766665
KCD	99.755102	99.190474	99.299997
LCD	87.877554	87.476188	99.366665

2) Comparing methods based on Recall, Precision, and F1-score: This scenario aims to provide a comparison between the defence algorithms (refer to figure 3a) and the attack procedure (refer to figure 3b) using data that does not include any attack, meaning no-attack. Specifically, in Fig 3a, we present the Recall, Precision, and F1-score for different defence methodologies. Precision and Recall measures show

TABLE IV: Comparing loss of algorithms on train, validation, and test data (%).

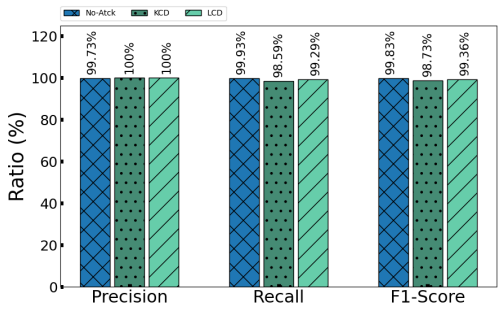
Algorithms	Train	Valid	Test
No attack	0.002305	0.002046	0.001836
RLFA	0.119700	0.121009	0.180553
SLFA	0.075652	0.073716	0.115980
KCD	0.002142	0.008091	0.006649
LCD	0.075199	0.078686	0.058709

the errors generated. The total amount of malicious detected can be measured by the recall value. That is, the proportion of correctly detected cases is the sum of all malicious cases (i.e., cases correctly detected by malicious plus cases falsely detected by benign). We aim to develop a machine-learning model to detect malicious behaviour with high recall values. Fig. 3a presents the dataset's Recall, Precision, and F1-score values. The figure illustrates the classification algorithms applied without an attack strategy, which resulted in a Recall and F1-score of 99%. Two important observations can be made from this figure: *i)* the KCD and LCD algorithm's Precision/Recall and F1-score values are close to those of No-Attack, indicating that our proposed defence algorithm can correctly identify benign samples; *ii)* the KCD algorithm outperforms the LCD algorithm in terms of Recall and F1-score values. Since our data contains malicious outlier data, the FP will be zero for all algorithms, affecting other metrics like Precision. Therefore, the Precision for all algorithms has the same value.

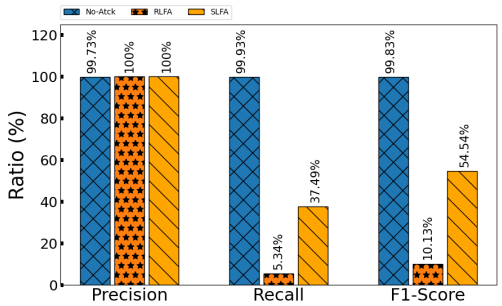
Fig. 3b presents Recall, Precision, and F1-score for attack algorithms, RLFA and SLFA, compared to the feature values of the dataset when there is no attack. This figure shows that the proposed attack method can deceive the ML model, cause misclassification, and significantly reduce the F1 score and recall values. Recall drops by approximately 46% for RLFA and about 62% for SLFA. The attacker aims to undermine the ML model's ability to classify data accurately. Therefore, our attack strategy is advantageous in such situations, and its harmful consequences are more visible in the dataset's features.

3) Comparing defence/attack algorithms based on FNR, TPR, and AUC: This section compares algorithms using FNR, TPR, and AUC, which are presented in Figs. 4a and 4b. A TPR value close to one (i.e., 100%) indicates higher precision in the model, whereas a value close to zero indicates poor performance in accurately identifying the samples. Comparing the TPR of different algorithms in Fig. 4a and 4b reveals that the KCD method outperforms the LCD method. Specifically, the KCD method has a rate of about 90% in identifying malicious samples. The FNR metric measures the rate of misclassification of data in a dataset. The rate of FNR is known to increase in the case of a flipping attack, while it can decrease when defence algorithms are employed.

Furthermore, based on the AUC and FNR values, we have a decrease in the AUC rate and an increase in the FNR rate after performing a label-flipping attack. FNR values are beneficial to the attacker as they increase during an attack. Defence



(a) Defence Algorithms



(b) Attack Algorithms

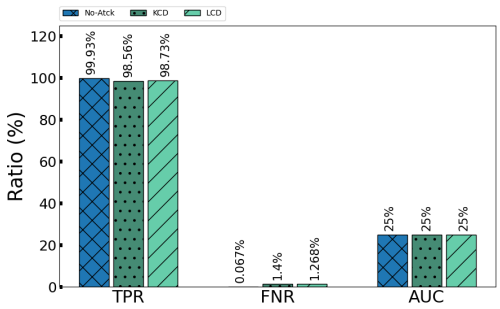
Fig. 3: The comparison between defence/attack algorithms based on Precision, Recall, and F1-Score. (No-Atck= No Attack)

strategies aim to minimize the FNR or the misclassification rate for malicious samples. Fig. 4a shows that our defence algorithms have a lower FNR, confirming our previous observations. Furthermore, since our data is composed of outliers, the FP rate is zero for all algorithms, which impacts the AUC values that are the same for all algorithms.

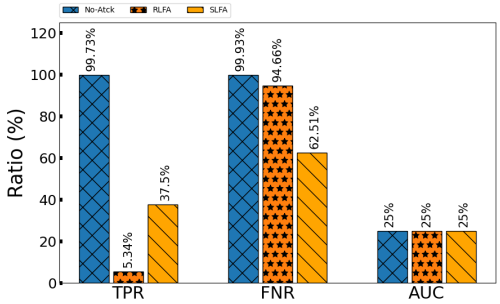
C. Testing Phase: Experimental results

This section presents experimental results for the RCA-IDS algorithm by introducing two scenarios, selected and full features, to verify real-time detection. First, we rank the features using the Random Forest Regressor (RF) algorithm to show which features impact the classification most. Then, we repeat our experiments for the higher-ranked features to determine which features more favourably affect the attack and defence algorithms. We select one feature (One-S-F), two features (Two-S-F), and three features (Three-S-F) with the highest ranking to evaluate our algorithms.

Table V presents the accuracy results of the attack and defence algorithms compared to the data without the attack based on the number of features we selected from all of them. The table indicates that the accuracy of the classifiers, in the absence of an attack, surpasses 99% when utilizing the full set of features. However, the accuracy decreases as the number of selected features reduces. The value of 99% shows how accurate our proposed AML models are, even in real-time scenarios. The RLFA algorithm randomly injects noise into the samples. Therefore, the accuracy of the RLFA algorithm is about 50%, which is slightly higher than the SLFA attack.



(a) Defence Algorithms



(b) Attack Algorithms

Fig. 4: The comparison between defence/attack algorithms based on AUC, TPR, and FNR. (No-Atck= No Attack)

From the results, we can see how much accuracy is increased through the proposed defence mechanisms.

TABLE V: The comparison between attack and defence algorithms in percent (%) based on Accuracy; Alg.:= Algorithm; -F:= Feature; -S:= Selected.

Alg.	Full-F	One-S-F	Two-S-F	Three-S-F
No-Atck	99.139	73.973	92.25	99.028
RLFA	50.195	50.028	50.028	50.084
SLFA	50.028	50.028	50.028	50.028
KCD	98.75	73.973	91.75	98.917
LCD	93.473	74.362	93.084	99.056

ACKNOWLEDGEMENT

This work has been funded by the European Space Agency (ESA) as part of the AutoTrust Project, Grant number 4000135646/21/UK/ND.

VII. CONCLUSION

This paper discusses a new real-time Cloud-based Adversarial IDS, RCA-IDS, to detect cyberattacks on CV systems. Two attacks, the Random-based Label Flipping Attack (RLFA) and the Silhouette-based Label Flipping Attack (SLFA), and defence algorithms, the K-mean-based Clustering Defence (KCD) and the Label-based Clustering Defence (LCD) algorithms, to mitigate these attacks introduced in this system. The performance of the defence algorithms using different features compares with the non-attack data in the AutoAlign dataset from the RL capital. An MLP classification algorithm is used to test our models. The results show that our proposed attacks can significantly reduce accuracy, from 99% (without attack) to 50% and 60% for RLFA and SLFA, respectively. However, after applying the defence algorithms, the accuracy improves to 99.99% and 99.8% for KCD and LCD, respectively. In future research, we suggest exploring semi-supervised methods that incorporate deep learning techniques such as autoencoders and different Generative Adversarial Networks (GANs), which can be combined with clustering methods in an ensemble learning framework to enhance the results against label-flipping attacks.

REFERENCES

[1] A. Lamssaggad, N. Benamar, A. S. Hafid, and M. Msahli, "A survey on the current security landscape of intelligent transportation systems," *IEEE Access*, vol. 9, pp. 9180–9208, 2021.

[2] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5341–5351, 2020.

[3] H. Liu, S. Zhang, P. Zhang, X. Zhou, X. Shao, G. Pu, and Y. Zhang, "Blockchain and federated learning for collaborative intrusion detection in vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 6073–6084, 2021.

[4] R. Jin, J. Hu, G. Min, and J. Mills, "Lightweight blockchain-empowered secure and efficient federated edge learning," *IEEE Transactions on Computers*, 2023.

[5] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.

[6] J. Mills, J. Hu, and G. Min, "Multi-task federated learning for personalised deep neural networks in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 630–641, 2021.

[7] N. Papernot *et al.*, "The limitations of deep learning in adversarial settings," in *Proc. of IEEE EuroS&P*, 2016, pp. 372–387.

[8] K. Grosse *et al.*, "Adversarial examples for malware detection," in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 62–79.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. of NIPS*, 2014, pp. 2672–2680.

[10] A. N. Bhagoji, D. Cullina, and P. Mittal, "Dimensionality reduction as a defense against evasion attacks on machine learning classifiers," *arXiv preprint arXiv:1704.02654*, 2017.

[11] A. Paudice, L. Muñoz-González, and E. C. Lupu, "Label sanitization against label flipping poisoning attacks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 5–15.

[12] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Advances in neural information processing systems*, 2013, pp. 1196–1204.

[13] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli, "Support vector machines under adversarial label contamination," *Neurocomputing*, vol. 160, pp. 53–62, 2015.

[14] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," *arXiv preprint arXiv:1803.09050*, 2018.

[15] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[16] A. I. Aviles-Rivero, N. Papadakis, R. Li, S. M. Alsaleh, R. T. Tan, and C.-B. Schonlieb, "Beyond supervised classification: Extreme minimal supervision with the graph 1-laplacian," *arXiv preprint arXiv:1906.08635*, 2019.

[17] "Label propagation," https://scikit-learn.org/stable/modules/label_propagation.html, [Online; accessed 15-July-2019].