A Cluster-Based Multidimensional Approach for Detecting Attacks on Connected Vehicles

Gianni D'Angelo, Arcangelo Castiglione, and Francesco Palmieri

***Abstract*—Nowadays, modern vehicles are becoming even more connected, intelligent, and smart. A modern vehicle encloses several cyber–physical systems, such as actuators and sensors, which are controlled by electronic control units (ECUs). Such ECUs are connected through in-vehicle networks, and, in turn, such networks are connected to the Internet of Vehicles (IoV) to provide advanced and smart features. However, the increase in vehicle connectivity and computerization, although it brings clear advantages, it introduces serious safety problems that can also endanger the life of the driver and passengers of the vehicle, as well as that of pedestrians. Such problems are mainly caused by the security weaknesses affecting the controller area network (CAN) bus, used to exchange data between ECUs. In this article, we provide two algorithms that implement a data-driven anomaly detection system. The first algorithm (cluster-based learning algorithm), is used to learn the behavior of messages passing on the CAN bus, for base-lining purposes, while the second one (data-driven anomaly detection algorithm) is used to perform real-time classification of such messages (licit or illicit) for early alerting in the presence of malicious usages. The experimental results, obtained by using data coming from a real vehicle, have shown that our approach is capable of performing better than other anomaly detection-based approaches.**

***Index Terms*—Clustering, controller area network (CAN) security, Internet of Vehicles (IoV), intrusion detection, *K*-means, machine learning (ML).**

I. Introduction

NOWADAYS, a modern vehicle is a set of interconnected cyber–physical systems. Vehicles are connected to private networks of the car manufacturer, as well as they are even more connected to the Internet using wireless radio technologies, such as 4G and Wi-Fi.

Many studies have shown that as early as 2015, 35% of new vehicles sold were connected to the Internet. Again, an Accenture forecast shows that by 2020 such a number of vehicles will increase to 98% and reach 100% in 2025 [1]. Due to the above reasons, a modern vehicle includes a complex cyber–physical network. It can be considered, a device of the Internet of Things (IoT), and not only a physical or mechanical device. Indeed, as shown in [2], the computer system of a vehicle is composed of over 100 million lines of code in its overall architecture higher than a modern operating system or a Boeing 757. As a consequence, vehicles are becoming even smarter, since manufacturers are enabling the connection of a vehicle with several entities, such as personal devices, vehicular *ad hoc* networks (VANETs), and the Internet, besides deploying innovative applications.

In detail, modern vehicles include several onboard computer devices, denoted as the electronic control unit (ECU), which controls and monitors the subsystems of a vehicle [3]. All the ECUs communicate with each other employing in-vehicle networks to enable advanced automotive features. In the last few years, many intelligent transportation systems (ITSs) applications have been introduced to improve road safety and drivers' experience. For example, cooperative adaptive cruise control (CACC) systems, fleet management systems, remote diagnostics, infotainment systems, remote engine start, eCall, and parking assistance [4]–[6]. As it is easy to observe, to operate appropriately, ITSs need to communicate frequently. This communication, which is carried out via the controller area network (CAN) bus [7], takes place through the exchange of messages between applications, sensors, and ECUs. More precisely, the CAN bus is the de facto standard of an in-vehicle network, and it provides a straightforward and reliable communication protocol that connects not only controllers and sensors but also the Internet. Again, the use of the CAN bus allowed the development and diffusion of vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communication interfaces [8].

As modern vehicles are increasingly equipped with more and more sophisticated electronic components and given their ever-increasing connectivity to external networks, it is easy to understand how security threats against such vehicles are growing rapidly and continuously. Also, unlike other smart devices, connected vehicles may have a further critical problem when compromised. The hacking of a vehicle could cause severe risks to people's lives, both inside and outside of the vehicle. In recent years, there have been many examples of both real-world attacks and proofs of concept against connected vehicles. In detail, ECUs can be compromised and intruded on. Practical experiments conducted in [9] have shown that an attacker can access an in-vehicle ECU remotely. Besides, various recent advisories [10] have also revealed that exploits to attack vehicles are publicly available. For this reason, an attacker could control some vehicle activities, including acceleration, steering braking, display, etc. Moreover, it is widely known that the CAN protocol is not secure. Indeed, it was designed to operate within a closed

Manuscript received April 15, 2020; revised June 20, 2020; accepted October 16, 2020. Date of publication October 22, 2020; date of current version August 6, 2021. *(Corresponding author: Gianni D'Angelo.)*

The authors are with the Dipartimento di Informatica, Università degli Studi di Salerno, 84084 Fisciano, Italy (e-mail: giadangelo@unisa.it; arcastiglione@unisa.it; fpalmieri@unisa.it).

Digital Object Identifier 10.1109/JIOT.2020.3032935

network environment. The main security weaknesses of the CAN are the following [11].

1) *Lack of Boundary Defense and Segmentation:* In a vehicle, the ECUs are usually grouped in sections. However, all such sections are connected by gateways and communicate through the CAN bus. Hence, a malicious user through the infotainment system could, for example, gain access to the anti-braking system.
2) *Lack of Device Authentication:* The CAN bus uses a broadcast communication, that is, a sent message reaches all nodes in the network. Again, any device can join the network communication by merely connecting to the bus, and the sender node is not authenticated. Thus, each node that has joined the network can send any CAN identifier it wishes (impersonation or masquerading attack).
3) *Use of Unencrypted Traffic:* The nodes of the CAN exchange messages in plaintext. Thus, the confidentiality requirement is not taken into account.
4) *Check of Availability:* Before each message is sent, the CAN protocol controls checks if the carrier is available. In case the carrier is busy, all the ECUs are blocked from transmitting, and the carrier is rechecked after a certain amount of time. Malicious users can take advantage of this mechanism to carry out a Denial-of-Service (DOS) attack.

We remark that a malicious user could access the CAN of a vehicle locally or remotely [5], [12], [13]. For example, remote attacks may be conducted via Wi-Fi, Bluetooth, cellular, radio data system, and telematics [5], [12]–[15]. On the other hand, local attacks may be carried out using remote keyless entry/start (RKE), onboard diagnostics (OBDs) connector, passive anti-theft system (PATS), tire pressure monitoring system (TPMS), and Bluetooth [12]. Again, recent studies demonstrate that the infotainment system is one of the main entry points of a vehicle [16].

Due to the reasons exposed above, information security and data protection are critical factors to be taken into consideration for the success of the newly emerging Internet of Vehicles (IoV). As a consequence, automotive cybersecurity is now considered a primary concern in the modern industry [17]. In the literature, several solutions have been proposed to prevent or at least mitigate the attacks arising from the security issues mentioned above. However, all these solutions require some modifications to the CAN protocol. An alternative approach that has gained increasing attention in the last decades to overcome such limitations is anomaly detection. It has been extensively used in many research fields and applications, such as trust-based recommendation systems [18], healthcare systems [19], aviation [20], and so forth [21]. Recently, anomaly detection has also proven to be effective in network traffic communications [22], [23], making it an optimal candidate for the automotive sector. In addition, anomaly detection is based on a data-driven model, usually empowered by learning systems. Therefore, it requires no modifications to either CAN protocol or bus. Accordingly, an anomaly detection system monitors the activities of the bus and raises an alarm if there is any unexpected event (early alerting). Although many studies have been undertaken in this area [24], most of the proposed methodologies and algorithms rely on monitoring the frequency of the messages or on analysis of the messages sequence [25], [26]. In fact, any attack may produce an alteration of both sequence and frequency and thus be detected. Nevertheless, this approach may fail if the attack causes slight changes in these features or relies on properly crafted obfuscation practices. Also, most of the existing solutions are not able to identify the specific illicit message and limit their action to raising a warning only.

To overcome these limitations, in this article, we propose a data-driven anomaly detection system able to learn the expected system behavior in a normal operating scenario, and immediately flag outliers with respect to the baseline by spotting the individual anomalous command and its operands. Our proposal uses data associated with different messages flowing on the CAN bus and tries mining useful features from them by working in an unsupervised way. Such features can be seen as representative of the CAN behavior. Any message assuming a behavior deviating from the learned one is to be considered an attack message. This leads to more granular anomaly detection, centered on evaluating any single message. To accomplish this, we proposed two algorithms: the former, cluster-based learning algorithm (CLA), is intended for learning such data behaviors, and the latter, data-centric anomaly detection algorithm (DADA), for detecting anomalies occurring on the CAN bus, from the observation of issued commands and associated operands/parameters.

Ultimately, we can summarize the main contributions as follows.

1) A data-driven approach is used to detect anomalies on CAN bus. The knowledge is derived by data flowing in the CAN bus during the normal operating scenario.
2) The anomaly detection is centered on a single message and its operands.
3) An unsupervised-based algorithm for learning the normal operating scenario is provided.
4) An algorithm for detecting anomalies on CAN bus and to perform early alerting is provided. The algorithm is able to detect any type of attack, such as injection, DoS, fuzzy, revolutions per minute (RPM), and GEAR attack.

This article is organized as follows. In Section II, we provide an overview of the recent related literature concerning the anomaly detection problem within the automotive sector. In Section III, we provide the background underlying the proposed contributions, in particular by focusing our attention on the K-means algorithm and the CAN bus protocol. In Section IV, we introduce our contributions, providing first the ideas that underlie it, and then we focus on the relevant details. In Section V, we show and describe the experimental results achieved through a testing activity conducted on a proof of concept of our proposal. Finally, in Section VI, we draw some conclusions and future research directions.

II. Related Work

Several anomaly based solutions to detect attacks on the in-vehicle network have been proposed.

Perraud [27] proposed an algorithm to deal with distributed DoS (DDoS) attacks on an in-vehicle connectivity unit (IVC-U). This algorithm aggregates the incoming traffic to create a single feature vector. This vector is used to detect a DDoS pattern, as an outlier for the clusters built during an observation window of the legitimate received traffic. Afterward, when a DDoS attack is detected, this algorithm splits legitimate flows, and DDoS flows through only the most discriminant subspace. Finally, this algorithm filters out the flows whose anomaly score in such a subspace is high.

Othmane *et al.* [28] characterized the "no attack" and "under-attack" state of a specific vehicle. This characterization relies on the analysis of injected speed and RPM messages through Pearson correlation, *K*-means cluster analysis, and hidden Markov models. Again, to create effective techniques for the detection of injected messages, this work suggests the use of more features than the data content of the CAN message, besides integrating the knowledge about how the ECUs interact.

To detect attacks on the CAN bus, Taylor *et al.* [29] propose the use of long short-term memory (LSTM) networks. In this work, the authors collect CAN bus data from the normal functioning of a 2012 Subaru Impreza, and inject some messages in the data set to simulate attacks. Again, they trained a detection model to forecast the next data byte sent by each ECUs. Thus, each deviation from the model is considered as an attack. This solution can detect anomalies with very low false-positive rates.

Song *et al.* [30] proposed a low impact intrusion detection scheme for in-vehicle networks. This algorithm analyzes the time occurring between CAN messages that result to be an essential feature for detecting attacks within the CAN traffic. Furthermore, this algorithm can detect all of the message injection attacks without false-positive errors.

Gmiden *et al.* [31] proposed an algorithm that does not require any modification to the CAN bus. It relies on the evaluation of the occurrence time intervals of the between CAN ID messages. A similar approach acting on the CAN ID sequences is used in [32].

Kang and Kang [33] introduced an intrusion detection technique that relies on deep neural networks, according to which the in-vehicle network packets exchanged among ECUs are used for extracting low-dimensional features and distinguish abnormal packets from normal ones. More precisely, this system continuously observes packet exchanges over the VANET and by performing feature extraction and providing real-time detection of attacks with sufficiently high accuracy. The used features make the technique very efficient and lightweight since they extracted on the fly from the packet stream flowing over the VANET.

Alshammari *et al.* [34] proposed the use of some machine learning (ML) techniques, that is, *k*-nearest neighbor and support vector machines, to cluster and classify the intrusions in VANETs. This proposal analyzes the offsets and time intervals between message requests and responses flowing over the CAN.

Hanselmann *et al.* [35] presented a solution called CANet. It makes use of a deep neural network, which is able to handle with a high dimensional CAN bus.

Ghaleb *et al.* [36] introduced a useful misbehavior detection model that operates in four phases, i.e., data acquisition, data sharing, analysis, and decision making. The training of such a model is based on historical data from both normal and abnormal data. At the same time, the classification makes use of an artificial neural network (ANN), and in particular of a feed-forward backpropagation neural network. To make their proposal more effective, the authors use the NGSIM real traffic data set.

Given the rise of the use of ML techniques in anomaly detection, Chockalingam *et al.* [37] considered the use of some ML algorithms to detect anomalies in CAN packets or packet sequences. More precisely, the authors assess the effectiveness of such algorithms to detect some frequent attacks on the CAN protocol. Finally, the authors also provide some implementation details of their proposal.

Finally, Berger *et al.* [38] performed a comparative assessment of different ML techniques concerning their usefulness for detecting intrusions and anomalies within automotive CAN-based networks.

However, a summary of the most recent approaches can be found in [24] and [39].

III. Background

In this section, we provide the background necessary to follow better and understand the contributions described in this article. More precisely, in this section, we provide an overview of the *K*-means algorithm and CAN bus protocol.

A. *K-Means*

Clustering is the process that enables the natural organization of similar objects into groups. It is an exploratory data mining process whose primary goal is ensuring that the objects belonging to a group are sufficiently different from the objects in other groups and vice versa, according to a specific similarity or distance metric. This is essentially a way of classifying unlabeled data in a certain number of aggregates, resulting in an unsupervised process. Several clustering flavors are available, namely, partitioning, density, and grid based, together with hierarchical clustering algorithms.

K-means is a widely used partitioning-based technique based on an iterative refinement mechanism. Objects are determined to belong to one of K clusters, where the number K is chosen in advance. Then, the object membership is determined by identifying the centroids of each cluster and associating each object with the cluster corresponding to the nearest centroid based on a chosen distance function that captures the similarity between different objects [40]. K points are randomly chosen as the initial centroids (or an initial random partitioning is performed). As a new object is added to a cluster, it modifies the centroid itself, which usually is not a point in the cluster but a kind of gravitational accumulation point within it.

K-means, that may be seen as a way of assigning each object to an equivalence class, represented by the cluster centroid, is mainly used due to its efficiency. Indeed, its computational complexity is $\mathcal{O}(nkl)$, where n is the order of the data set, i.e., the number of objects involved, k is the number of clusters into

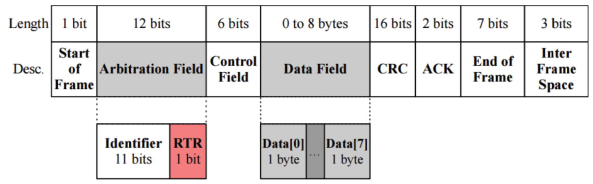

Fig. 1. Structure of the CAN data frame [26].

which the objects should be grouped, and l is are the iterations necessary for completing the whole clustering task. Again, in terms of space, the K-means complexity is $\mathcal{O}(k+n)$, besides needing further space for storing the data matrix. Besides, the "bootstrap" partition based on some chosen points behaving as initial centroids does not depend on the order n.

B. CAN-Bus

The CAN is a serial real-time communication protocol. It is a broadcast protocol, introduced by the automotive industry to simplify the wiring complexity of automobiles [41]. The onboard diagnostics uses the CAN (OBD) standard. Note that all modern cars and light trucks in the U.S. and EU must mandatorily adhere to such a standard.

An ECU uses the CAN protocol to transmit its frames over the CAN bus [7]. In detail, the CAN is a time-synchronized broadcast, multicast reception bus, in which two wires to connect ECUs are used. One wire is called the CAN Low (CANL), and the other is referred to as the CAN High (CANH).

Over the CAN bus, four types of message frames can be exchanged: 1) a *data frame* used to exchange data between the nodes; 2) a *remote frame* used to request the transmission of a specific identifier; 3) an *error frame* used by any node that detects an error; and 4) an *overload frame* used to inject a delay between *data* or *remote* frames [42]. As shown in Fig. 1, the *data frame* is composed of seven fields: 1) a *Start-of-Frame (SoF)* field, which denotes the start of a frame; 2) an *arbitration* field, which contains the identifier of the message (stored in the identifier field) and service bits; 3) a *control* field, which contains the length of the data included in the message, and two reserved bits; 4) a *data* field, which is 0–8 B long, and includes the data to be transferred; 5) a *CRC* field, which is used for error checking; 6) an *ACK* field, which is used to acknowledge the receipt of the message; and 7) End-of-Frame (EoF) field, which indicates the end of the frame [42].

Note that the arbitration field, also known as the *arbitration ID* or *CAN ID*, specifies the frame function and priority of the frame, with 0×000 representing the maximum achievable value and $0 \times 7FF$ the minimum one. Such priority is used to solve the conflict arising when multiple nodes compete for sending a frame concurrently. In detail, each CAN ID is strictly associated with a specific ECU, although an individual ECU can broadcast several different CAN IDs, as well as two distinct ECUs, are not allowed to use the same CAN ID for sending frames. When a frame is sent, every ECU listening on the CAN bus receives it and, depending on the CAN ID, determines its acceptance and further processing [7]. In this scenario, a malicious user could prevent ECUs from communicating, by injecting on the bus frames with CAN ID set to 0×000. This injection represents a straightforward but effective DoS attack. Indeed, since the CAN ID of a frame univocally determines its processing priority, repeatedly sending multiple frames with the CAN ID associated with the highest possible priority causes starvation over the CAN bus by preventing other ECUs to transmit their messages. This attack can lead the vehicle to an unstable state [43].

IV. Proposed Solution

Detecting anomalies in CAN data is an activity that differs from most of the ML applications because it is not possible to clearly know the data associated with an attack. Indeed, although the families of the attacks are well-known, their implementations and the employed data are often not known in literature because they can change. For this reason, it is not possible to collect a representative set of attack data to be used to train an ML-based classifier, mainly when the supervised approach is used. On the other hand, it is possible to capture CAN data of a vehicle during its normal behavior (free of attacks) under different conditions, such as driving style, environment conditions, vehicle brand, traffic condition, and so forth. Such data can be collected and used to gain insight into the normal behavior of the CAN bus. Any behavior deviating from the learned one is considered anomalous. As it will be shown in details in the next section, this approach can deal with masquerading, injection, DoS attacks, etc., and it can represent an alternative or a supplement to the message authentication code (MAC) for the protection of the CAN frames [44].

To accomplish this, we propose two algorithms: the former, named CLA, is used for learning the normal behavior of the CAN messages, while the latter, named DADA, for detecting anomalous usage patterns, that is the attacks accomplished by injecting malformed commands on the CAN bus, to perform early alerting.

A. CLA: Cluster-Based Learning Algorithm

The aim of this algorithm is to construct knowledge about normal CAN bus usage by finding out a set of behavioral signatures of the data field associated with any distinct CAN arbitration identifier ID_i occurring within an attack-free data set D, that is used for baselining. A multistep procedure is used for this purpose, where in the first step, the different arbitration identifiers and their associated command parameters are extracted from D and placed within different subsets. In the next step, a traditional K-means clustering of the subsets of D associated with each identifier ID_i leads to the construction of multiple sets of centroids, representing the admissible command parameters for each command identifier ID in the form of multiple gravitational attractors or accumulation points that individuate an admissibility zone within a multidimensional space. To accomplish this, the N^{ID_i} data observations associated with each specific identifier ID_i can be defined as the set of points $D^{\mathrm{ID}_i} = \{D_1^{\mathrm{ID}_i}, \ldots, D_{N^{\mathrm{ID}_i}}^{\mathrm{ID}_i}\}$. Each point is defined in a 8-D vector space, where each dimension is associated with a specific byte position in the data field of the CAN frames associated with a specific identifier, so that $D_k^{\mathrm{ID}_i} = [B_0, \ldots, B_7]$.

Such points are grouped into a given number of disjoint clusters $1 < K < N^{\mathrm{ID}_i}$, where at least a point must belong to each cluster and no cluster can contain all the points in D^{ID_i}, resulting into a partitioning $\pi_1^{\mathrm{ID}_i}, \ldots, \pi_K^{\mathrm{ID}_i}$ of the data set D^{ID_i} so that

$$\bigcup_{k=1}^{K} \pi_k^{\mathrm{ID}_i} = D^{\mathrm{ID}_i}$$

$$\pi_k^{\mathrm{ID}_i} \cap \pi_j^{\mathrm{ID}_i} = \emptyset \;\; \forall\, k, j \mid k \neq j \tag{1}$$

$$\emptyset \subset \pi_k^{\mathrm{ID}_i} \subset D^{\mathrm{ID}_i} \;\; \forall\, k. \tag{2}$$

Each partition $\pi_k^{\mathrm{ID}_i}$ is constructed according to an iterative schema, following the traditional K-means paradigm, starting from its centroid (8-D point) $\chi_k^{\mathrm{ID}_i}$ by using euclidean distance within an orthonormal basis of $\mathbb{N}^8$ for evaluating the similarity between points, with the aim of minimizing the following quantity:

$$\sum_{k=1}^{K} \sum_{D_j^{\mathrm{ID}_i} \in \pi_k^{\mathrm{ID}_i}} \|D_j^{\mathrm{ID}_i} - \chi_k^{\mathrm{ID}_i}\|^2. \tag{3}$$

The final partitioning $\Pi^{\mathrm{ID}_i} = \{\pi_1^{\mathrm{ID}_i}, \ldots, \pi_K^{\mathrm{ID}_i}\}$ of the data set D^{ID_i}, resulting in a stable position of the centroids $\chi_1^{\mathrm{ID}_i}, \ldots \chi_K^{\mathrm{ID}_i}$, once built on observations coming for normal operations, will be used to build the baseline describing the correct command behavior on the CAN-bus, in terms of behavioral signature of the data field associated with the specific arbitration identifier ID_i involved.

In the last step, starting from the final centroids of the aforementioned baseline partitioning, for each centroid $\chi_k^{\mathrm{ID}_i}$, an area $\Delta_k^{\mathrm{ID}_i}$ around it is estimated. The signature σ^{ID_i} of the considered arbitration identifier ID_i is thus represented by the set of these centroids along with their corresponding areas

$$\sigma^{\mathrm{ID}_i} = \left\{\left(\chi_1^{\mathrm{ID}_i}, \Delta_1^{\mathrm{ID}_i}\right), \ldots, \left(\chi_K^{\mathrm{ID}_i}, \Delta_K^{\mathrm{ID}_i}\right)\right\}. \tag{4}$$

The area $\Delta_k^{\mathrm{ID}_i}$ can be seen as a region in the space $\mathbb{N}^8$ characterized by having a lower bound ($LB_k^{\mathrm{ID}_i}$) and an upper bound ($UB_k^{\mathrm{ID}_i}$) equal to the minimum and maximum distance among the points of the kth cluster (associated with the partition $\pi_k^{\mathrm{ID}_i}$) and its centroid $\chi_k^{\mathrm{ID}_i}$, respectively, so that

$$LB_k^{\mathrm{ID}_i} = \min_{D_j^{\mathrm{ID}_i} \in \pi_k} \|D_j^{\mathrm{ID}_i} - \chi_k^{\mathrm{ID}_i}\| \tag{5}$$

$$UB_k^{\mathrm{ID}_i} = \max_{D_j^{\mathrm{ID}_i} \in \pi_k} \|D_j^{\mathrm{ID}_i} - \chi_k^{\mathrm{ID}_i}\| \tag{6}$$

$$\Delta_k^{\mathrm{ID}_i} = \left\{x \in D^{\mathrm{ID}_i} \;\middle|\; LB_k^{\mathrm{ID}_i} \leq x \leq UB_k^{\mathrm{ID}_i}\right\}. \tag{7}$$

In a bidimensional space, this would correspond to a circular crown around each centroid $\chi_k^{\mathrm{ID}_i}$. All the new points y, that have not considered in the baseline clustering step, when falling within this crown for at least a partition $\pi_k^{\mathrm{ID}_i}$ are to be considered licit parameters for the associated arbitration identifier ID_i (formally expressed by the function $\Lambda^{\mathrm{ID}_i}(y)$ assuming

Algorithm 1 CLA: Cluster-Based Learning Algorithm

Require: a) D, the attack free dataset; b) K, number of clusters.

Output: Σ, a Map associating any CAN identifier ID_i to the set of the centroids' coordinates, and the bounds of an area around the centroids.

```
1: Init:
     1) $\Sigma \leftarrow \emptyset$
     2) $U \leftarrow \emptyset$
2: Finding unique CAN identifiers ($ID_i$):
     1) $U = \{ID_i \in CAN\_ID \mid \exists!\; ID_i \in D\}$
     2) $u = |U|$, $u$, number of distinct $ID_i$
3: Extract set of data:
     1) For any $ID_i \in U$ do:
        a) $D^{ID_i} \leftarrow \{D_k^{ID_i}\}$,
           with $k = 1 \ldots N^{ID_i}$; $N^{ID_i}$, number of payloads
           associated with $ID_i$,
           and $D_k^{ID_i} = \left[B_0\, B_1\, B_2\, B_3\, B_4\, B_5\, B_6\, B_7\right]_k^{ID_i}$
4: Finding clusters, centroids, and bounds:
     1) For any $ID_i \in U$ do:
        a) Finding $K$ clusters:
           $\Pi^{ID_i} \leftarrow kmeans(D^{ID_i}, K)$
        b) Finding centroids:
           $\chi_k^{ID_i} \leftarrow centroid(\pi_k^{ID_i})$
           with $\pi_k^{ID_i} \in \Pi^{ID_i}$ and $k = 1 \ldots K$
        c) Finding bounds for any $k = 1 \ldots K$:
           $LB_k^{ID_i} \leftarrow \min_{D_j^{ID_i} \in \pi_k} \|D_j^{ID_i} - \chi_k^{ID_i}\|$,
           $UB_k^{ID_i} \leftarrow \max_{D_j^{ID_i} \in \pi_k} \|D_j^{ID_i} - \chi_k^{ID_i}\|$
5: Output:
     1) $\sigma^{ID_i} \leftarrow \left\{\left[\chi_k^{ID_i}\; LB_k^{ID_i}\; UB_k^{ID_i}\right]\right\}^K$,
        with $k = 1 \ldots K$
     2) $\Sigma \leftarrow \{< ID_i, \{\sigma^{ID_i}\} >\}^u$
```

a value 0), while new points falling out of it are illicit [$\Lambda^{\mathrm{ID}_i}(y)$ assumes a value 1]

$$\Lambda^{\mathrm{ID}_i}(y) = \begin{cases} 0, \text{ if } \exists\, k \mid y \in \Delta_k^{\mathrm{ID}_i} \\ 1, \quad \text{otherwise.} \end{cases} \tag{8}$$

Note that the centroid themselves may not be considered licit points because it may not fall into the crown area.

In Algorithm 1, a pseudocode of the proposed algorithm is shown. As depicted, once the initialization is concluded (step 1-Algorithm 1), the distinct CAN IDs (ID_i) are extracted (step 2). We remark that a single CAN ID is repeated many times on the CAN bus with a specific frequency and latency, thus there are many points associated with a single CAN ID, which are extracted in step 3, that is D^{ID_i}. Next (step 4.1a) a number of clusters associated with any ID are extracted using the K-means algorithm. After that, the centroids and their bounds are computed (step 4.1b and 1c). Finally, in step 5, the signature is provided as a set of vectors (Σ) including in sequence: the CAN ID, the centroids and their corresponding lower and upper bounds, respectively.

B. DADA: Data-Centric Anomaly Detection Algorithm

This algorithm is devoted to find out the malicious messages flowing into CAN bus. The message ($<\bar{\text{ID}}, D^{\bar{\text{ID}}}>$) is to be considered belonging to an attack when one of the following conditions occurs.

1) The CAN ID is unknown. That is, if the ID does not belong to the IDs found in the learning phase (step 2 of Algorithm 1), that is

$$\nexists\ \text{ID}_j \ \big|\ <\text{ID}_j, *> \in \Sigma \quad \wedge \quad \text{ID}_j \equiv \bar{\text{ID}}. \tag{9}$$

2) The ID is known, but the point associated with its data falls out of the areas associated with a certain number (*nPts*) of neighbor centroids learned in Algorithm 1. More specifically

$$\begin{aligned} &\exists\ \text{ID}_j \ \big|\ <\text{ID}_j, *> \in \Sigma \quad \wedge \quad \text{ID}_j \equiv \bar{\text{ID}} \quad \wedge \\ &D^{\bar{\text{ID}}} \notin \Delta_k^{\text{ID}_v} \quad \forall\ k, <\text{ID}_v, \left\{\sigma^{\text{ID}_v}\right\}> \in \Sigma \\ &\quad v = 1, \ldots, nPts. \end{aligned} \tag{10}$$

3) The ID is known, its point falls within the areas mentioned above, but the ID is different by the IDs associated with the above-mentioned neighbor centroids, that is

$$\begin{aligned} &\exists\ \text{ID}_j \ \big|\ <\text{ID}_j, *> \in \Sigma \quad \wedge \quad \text{ID}_j \equiv \bar{\text{ID}} \quad \wedge \\ &\exists\ k, \ <\text{ID}_v, \left\{\sigma^{\text{ID}_v}\right\}> \in \Sigma \ \big|\ D^{\bar{\text{ID}}} \ \in \ \Delta_k^{\text{ID}_v} \ \wedge \\ &\text{ID}_v \neq \bar{\text{ID}} \ \ \forall\ v. \end{aligned} \tag{11}$$

Algorithm 2 shows a pseudocode of the implementation. As depicted, given the data set of transitions (Attack_DS), the CAN ID (ID$_i$) and its relative message (D^{ID_i}) are first extracted (step 2.1a and 1b of Algorithm 2). Next, the typology of the incoming message is evaluated. First (step 2.1c), if the message ID is unknown, then it is considered to be an attack message.

Otherwise (step 2.1d), *nPts* points of Σ (learned in Algorithm 1) closer to D^{ID_i} are extracted through the usage of the K-nearest neighbor algorithm (*knnSearch*). Next, their IDs and associated bounds (LB^{pts}, UB^{pts}) are extracted.

If the point associated with the D^{ID_i} is out of these bounds, then the message is an attack (step 2.1(d)ivA). Contrarily, D^{ID_i} can be within one or more of these bounds; then, the corresponding IDs are compared. If there is at least one ID among the *nPts* IDs that is equal to the ID of the message under testing, then it is not an attack. Otherwise (step 2.1(d)ivB), the message is to be considered an attack. In other words, the message is considered licit if the ID is known, and its data falls within known areas having the same ID of the considered message.

Algorithm 2 DADA: Data-Centric Anomaly Detection Algorithm

Require: a) *Attack_DS*, the attack dataset; b) Σ, the Map container; c) *nPts*, number of neighbor points.

Output: *AD_List*, the entries list of the *Attack_DS* considered to be an attack.

```
1: Init:
     1) $AD\_List \leftarrow \emptyset$
2: Finding malicious CAN ID:
     1) For any $A_i \in Attack\_DS$ do:
        a) Extract $ID_i$
           $ID_i \leftarrow extractCAN\_ID(A_i)$
        b) Extract $D^{ID_i}$
           $D^{ID_i} \leftarrow extractData(A_i)$
        c) IF $< ID_i, * > \notin \Sigma$ THEN
           $AD\_List \leftarrow A_i$ (i.e., unknown ID)
        d) ELSE
           i) $pts \leftarrow knnSearch(\Sigma, D^{ID_i}, nPts)$
           ii) Extract IDs of $pts$:
               $ID^{pts} \leftarrow getID(pts)$
           iii) Extract bounds of $pts$:
               $LB^{pts} \leftarrow LB(pts)$
               $UB^{pts} \leftarrow UB(pts)$
           iv) Compute the distances among pts and $D^{ID_i}$:
               $distances \leftarrow \{d_k\}^{nPts}$
               with $d_k = Euclidean\_Dist(pts_k, D^{ID_i})$,
               and $pts_k \in pts$ , $k = 1 \ldots nPts$
               A) IF $(distances < LB^{pts})$ or $(distances > UB^{pts})$ THEN
                  $AD\_List \leftarrow A_i$ (i.e., out of ranges)
               B) ELSE_IF $ID_i \notin ID^{pts}$ THEN
                  $AD\_List \leftarrow A_i$, (i.e., ID mismatch)
3: Output:
   $AD\_List$
```

V. Experimental Results

In this section, we show the results obtained from the evaluation of our proposal when it is applied to detect well-known typologies of attacks. More precisely, we consider the following attacks.

1) *DoS Attack:* In this attack, messages with the highest priority (i.e., ID = 0×000) are injected on the CAN bus. Its intent is to increase the latency of the normal messages to delay or inhibit the CAN response.
2) *Fuzzy Attack:* Random spoofed CAN IDs and/or data values are injected on the CAN bus. This type of attack can produce bad functioning of the vehicle under attack. For example, the steering wheel trembles, turn signal lamps light irregularly, the gearbox changes automatically, and the instrument panel blinks arbitrary [26]. Therefore, unlike the DoS attack, the Fuzzy one intends to paralyze the vehicle rather than delay the normal messages.
3) *RPM Attack:* Malicious messages are injected on the CAN bus by using the same ID of the RPM messages. Because many behaviors of the vehicle are dependent on the RPM value, this attack may produce many typologies of dangerous functioning.

TABLE I
DATA SETS USED IN EXPERIMENTAL EVALUATION [45]

Dataset	# of normal msgs	# of illicit msgs	Total
Attack-free	988,872	–	988,872
DoS Attack	3,078,250	587,521	3,665,771
Fuzzy Attack	3,347,013	491,847	3,838,860
GEAR Attack	3,845,890	597,252	4,443,142
RPM Attack	3,966,805	654,897	4,621,702
Total	15,226,830	2,331,517	17,558,347

4) *GEAR Attack:* Malicious messages are injected on CAN bus by using the same ID of the GEAR messages.

In the following, we first describe the data sets that have been used for the experiments. Subsequently, we show the evaluation metrics used for estimating the performance. Next, we present the results obtained, and finally, we provide relative interpretations.

A. Data Set

The experiments have been carried out by using the car-hacking data sets provided by the Hacking and Countermeasure Research Lab (HCRL) [45]. More precisely, five data sets are provided, that is: 1) attack free; 2) DoS attack; 3) fuzzy attack; 4) spoofing of the drive gear; and 5) spoofing of the RPM gauge. Each data set came from real vehicles by connecting the CAN bus through the OBD-II port. Each data set includes a total of 30–40 min of CAN traffic, while the injected messages are performed for 3–5 s for a total of about 300 intrusions.

For a DoS attack, the most dominant message (0×000) is injected every 0.3 ms, while in a fuzzy attack, random CAN IDs and data values are injected every 0.5 ms. Finally, in a Spoofing attack (RPM and GEAR) the messages with CAN ID associated with RPM/GEAR are injected every 1 ms.

The data set is provided in table form with the following attributes.

1) *Timestamp:* Time in seconds.
2) *CAN ID:* The hexadecimal version of the CAN identifier.
3) *DATA[0-7]:* 8-B data field expressed in hexadecimal.
4) *Flag:* $\{T, R\}$. T represents the illicit message, while R is the normal message.

Table I shows a summary of the distribution of the messages in the data sets used in our experiments.

B. Performance Evaluation and Results Analysis

As shown in [27] any algorithm that needs to operate on the CAN bus should comply with the following requirements.

1) When there are no attacks, monitoring of the incoming traffic on the CAN bus should consume the least amount of CPU resources possible, as well as it should not degrade the quality of the connectivity, in particular for safety-critical services.
2) The detection time should be quite fast and any algorithm with a very long convergence time (in the range of seconds) should be excluded.
3) The computing power necessary to detect and mitigate an attack should be reasonable and compatible with the computing power of an IVC-U.

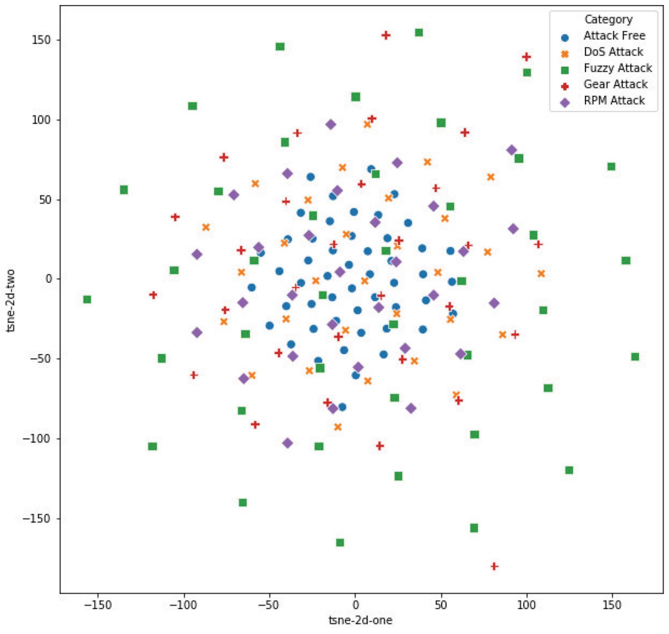

Fig. 2. Bidimensional t-SNE representation of some centroids. For the sake of clarity, only a few centroids associated with each ID are shown in a zoomed area around the center.

Accordingly, we tested the performance of the proposed solution by implementing all tests in the MATLAB $R2019a$ environment on a classical PC equipped with 64-b Intel Core i7-8565U CPU @ 1.80 GHz, and 16-GB RAM.

As mentioned earlier, our method makes use of an unsupervised approach. Accordingly, Algorithm 1 is only used in the training phase on normal traffic. More precisely, we used all normal traffic of Table I for training, while all illicit traffic for testing. Training was performed by using 300 clusters for any ID (i.e., $K = 300$ in Algorithm 1), and by considering 30 neighbor points (i.e., $nPts = 30$ in Algorithm 2) for anomaly detection.

To provide a visual interpretation of the functioning of our approach and, in particular, to show the capability of the Algorithm 2 in distinguishing different typologies of messages, in Fig. 2 is reported as a bidimensional representation of the centroids of any data set. In particular, for clarity's sake, only a few centroids associated with each ID are shown. We remark that during the training phase, only the normal messages are fed into Algorithm 1. Contrarily, in Fig. 2, the centroids of all the data sets are depicted, that is the Algorithm 1 is applied on all the data sets.

To accomplish this, we used the t-Distributed Stochastic Neighbor Embedding (t-SNE) [46]. t-SNE is an ML-based technique for dimensionality reduction, which is also well suited for the visualization of high-dimensional data sets.

More precisely, Fig. 2 depicts a zoomed area around the center (0, 0). As shown, centroids of the attack-free data set are scattered, which means that the data set includes data falling into many quite different ranges of values (areas around the centroids). This justifies our choice of $K = 300$, which guarantees a high granularity of data ranges. Indeed, the usage of a lower value of clusters would lead to have a lower number of areas, each with larger bounds. This means to consider a lot

TABLE II
EXPERIMENTAL RESULTS FOR EVERY DATA SET

Dataset	Acc.	TPR	TNR	PPV
Attack-free	1.0000	–	0.9998	–
DoS Attack	1.0000	1.0000	1.0000	0.9999
Fuzzy Attack	0.9996	0.9999	0.9996	0.9970
GEAR Attack	0.9998	1.0000	0.9998	0.9986
RPM Attack	0.9999	1.0000	0.9998	0.9990

of data that is not admissible as admissible. We remark that, according to our approach, the admissible data associated with an ID are those belonging to the area around the centroids of the ID in question. The extreme case is the usage of a single centroid for an ID. The area around this centroid would collect many not-admissible value ranges associated with the considered ID.

Furthermore, as shown in Fig. 2, many centroids of the attack-free data set are overlapped or very close to the centroids of other data sets. As a result, their areas may intertwine and thus provide errors in distinguishing IDs. To overcome this issue, we used $nPts = 30$ in Algorithm 2. Indeed, this ensures using many centroids of the attack-free data set to classify the involved message. The higher the number of points considered, the lower the probability of making mistakes.

To evaluate numerically the performance of our approach, we used some metrics derived by true positive (TP), true negative (TN), false positive (FP), and false negative (FN) observations. TP are the correctly classified illicit messages, TN are the correctly classified normal messages, FP are the incorrectly classified normal messages, and FN are the messages incorrectly classified as illicit. The metrics used are the following.

1) $Acc = [TP+TN/(TP+TN+FP+FN)]$: Accuracy is the total number of illicit (TP) and normal (TN) messages correctly classified with respect to all data sets.
2) $TPR = (TP/TP+TN)$: True positive rate, also known as Recall or Sensitivity. It is the amount of TP with respect to all illicit messages.
3) $TNR = (TN/TN + FP)$: True negative rate, also known as Specificity or Selectivity. It is the amount of TN with respect to all licit messages.
4) $PPV = (TP/TP + FP)$: Positive predictive value, also known as Precision. It is the amount of TP with respect to the total number of predicted illicit messages.

Note that these metrics range from 0 to 1. The minimum value indicates the worst performance, while the maximum value indicates the best one.

Table II shows the results obtained for any data set. As depicted, all the metrics achieved the maximum score (100%) for all data sets. More precisely, for DoS attacks, only 21 malicious messages were incorrectly classified as normal, and 40 normal messages as malicious ones. For fuzzy attacks, 26 illicit messages were incorrectly classified as normal ones, while 1473 normal messages were confused as illicit. For GEAR and RMP attacks, all the illicit messages were correctly classified as such. Instead, 816 and 686 normal messages were incorrectly classified as illicit ones, respectively.

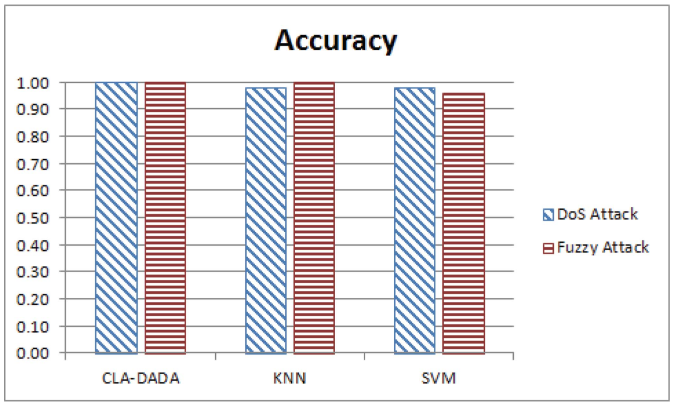

Fig. 3. Comparison of the accuracy for DoS and fuzzy attack data sets.

Ultimately, in Fig. 3 we provide a performance comparison with the state-of-the-art, which use approaches based on KNN and SVM [34].

As depicted, our approach outperformed the state-of-the-art ones for both DoS and fuzzy attacks, respectively. In addition, according to the requirements complying to [27], mentioned earlier, the implementation of our approach is rather simple, and the execution time of the Algorithm 2 allows the detection of attack messages in real time.

VI. CONCLUSION

In this article, the anomaly detection approach has been used to address the security problems arising in modern in-vehicle networks. More precisely, the attacks on the CAN bus have been considered. As it is known, the CAN bus lacks any built-in security mechanisms to deal with attacks coming from outside, since it was designed to operate in closed network environments, where all the communicating nodes trust each other [5]. More precisely, four different attacks on the CAN bus have been considered, that is: DoS, fuzzy, GEAR, and RPM attacks. In the literature, some solutions have been proposed to prevent or mitigate the effects of such attacks. However, many of the proposed solutions require the modification of the CAN protocol, which is not admissible. Also, most of them are based on monitoring the messages frequency or their latency over time, which may fail if the attack is characterized by an adaptive behavior. Besides, such solutions do not allow us to identify the specific malicious message.

The anomaly detection system presented in this article has proven to be effective in overcoming such limitations. It makes use of a data-driven approach, in which message classification is performed in an unsupervised way by using only the data associated with the CAN messages (IDs). We have provided an algorithm, named CLA, for mining clusters from data, which have proven to be effective in representing the behavior of normal messages flowing on the CAN bus. In particular, for any ID, CLA extracts a number of clusters and represents them through the usage of their centroids together with an area around them. These features have demonstrated to be capable of representing the message data also when the corresponding points cover a large area, that is the points are very scattered. Nevertheless, the number of clusters to be used is crucial for correct learning. In fact, although the usage of many clusters could increase the granularity of the features, it could lead to a higher computational load. On the other hand, using a few clusters could lead to a performance reduction. A self-adaptive approach, which is the subject of future work, would be desirable. We have also provided a second algorithm, named DADA, to use the features extracted by CLA and find out the illicit messages. Due to the implementation simplicity of this algorithm, it is suitable for searching malicious messages in real time. Indeed, the experimental results, obtained by using a notable data set coming from real vehicles, have confirmed the ability of our approach in spotting all the above-mentioned attacks with high precision. Indeed, the metrics (accuracy, sensitivity, specificity, precision) used to evaluate the performance of the proposal have achieved all the maximum rating for any attack. Besides, the efficiency of our proposal has been confirmed by comparison with other notable approaches available in the literature.

The excellent results obtained in this article encourage us to prove its effectiveness on other typology of attacks in the future. Nevertheless, we intend to use our approach also in many other applications in which data-driven anomaly detection is necessary.

References

[1] V. Zieglmeier, S. Kacianka, T. Hutzelmann, and A. Pretschner, "A real-time remote IDs testbed for connected vehicles," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput.*, 2019, pp. 1898–1905.

[2] M. Wolf, A. Weimerskirch, and C. Paar, "Security in automotive bus systems," in *Proc. Workshop Embedded Security Cars*, 2004, pp. 1–3.

[3] F. Yu, D.-F. Li, and D. Crolla, "Integrated vehicle dynamics control—State-of-the art review," in *Proc. IEEE Veh. Power Propulsion Conf.*, 2008, pp. 1–6.

[4] Y. Zhao, "Telematics: Safe and fun driving," *IEEE Intell. Syst.*, vol. 17, no. 1, pp. 10–14, May 2002.

[5] L. B. Othmane, H. Weffers, M. M. Mohamad, and M. Wolf, "A survey of security and privacy in connected vehicles," in *Wireless Sensor and Mobile Ad-Hoc Networks*. New York, NY, USA: Springer, 2015, pp. 217–247.

[6] J. den Hartog *et al.*, "Security and privacy for innovative automotive applications: A survey," *Comput. Commun.*, vol. 132, pp. 17–41, Feb. 2018.

[7] M. Farsi, K. Ratcliff, and M. Barbosa, "An overview of controller area network," *Comput. Control Eng. J.*, vol. 10, no. 3, pp. 113–120, 1999.

[8] S. Biswas, R. Tatchikou, and F. Dion, "Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety," *IEEE Commun. Mag.*, vol. 44, no. 1, pp. 74–82, Jan. 2006.

[9] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," in *Proc. Black Hat USA*, 2015, p. 91.

[10] UD of Homeland Security CISA Cyber + Infrastructure. (Jul. 2017). *ICS Advisory (ICSA-17-208-01)*. [Online]. Available: https://www.us-cert.gov/ics/advisories/ICSA-17-208-01

[11] L. Pan, X. Zheng, H. Chen, T. Luan, H. Bootwala, and L. Batten, "Cyber security attacks to modern vehicular systems," *J. Inf. Security Appl.*, vol. 36, pp. 90–100, Oct. 2017.

[12] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," in *Proc. Black Hat USA*, 2014, p. 94.

[13] S. Checkoway *et al.* "Comprehensive experimental analyses of automotive attack surfaces." in *Proc. USENIX Security Symp.*, vol. 4, 2011, pp. 447–462.

[14] S. Nie, L. Liu, and Y. Du, "Free-fall: Hacking tesla from wireless to can bus," in *Proc. Black Hat USA*, 2017, pp. 1–16.

[15] I. Rouf *et al.*, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study," in *Proc. USENIX Security Symp.*, vol. 10, 2010, pp. 116–127.

[16] S. Mazloom, M. Rezaeirad, A. Hunter, and D. McCoy, "A security analysis of an in-vehicle infotainment and app platform," in *Proc. 10th {USENIX} Workshop Offensive Technol. (WOOT)*, 2016, pp. 260–264.

[17] D. Ward, I. Ibarra, and A. Ruddle, "Threat analysis and risk assessment in automotive cyber security," *SAE Int. J. Passenger Cars Electron. Elect. Syst.*, vol. 6, pp. 507–513, Jan. 2013.

[18] G. D'Angelo, F. Palmieri, and S. Rampone, "Detecting unfair recommendations in trust-based pervasive environments," *Inf. Sci.*, vol. 486, pp. 31–51, Jun. 2019.

[19] L. F. M. Carvalho, C. H. C. Teixeira, W. Meira, M. Ester, O. Carvalho, and M. H. Brandao, "Provider-consumer anomaly detection for healthcare systems," in *Proc. IEEE Int. Conf. Healthcare Informat. (ICHI)*, 2017, pp. 229–238.

[20] G. D'Angelo, G. Cavaccini, and S. Rampone, "Shimming analysis of carbon-fiber composite materials with eddy current testing," in *Proc. 5th IEEE Int. Workshop Metrol. Aerosp. (MetroAeroSpace)*, 2018, pp. 68–73.

[21] G. D'Angelo, R. Pilla, C. Tascini, and S. Rampone, "A proposal for distinguishing between bacterial and viral meningitis using genetic programming and decision trees," *Soft Comput.*, vol. 23, no. 22, pp. 11775–11791, 2019.

[22] G. D'Angelo, F. Palmieri, M. Ficco, and S. Rampone, "An uncertainty-managing batch relevance-based approach to network anomaly detection," *Appl. Soft Comput.*, vol. 36, pp. 408–418, Nov. 2015.

[23] H. Xia, B. Fang, M. Roughan, K. Cho, and P. Tune, "A basisevolution framework for network traffic anomaly detection," *Comput. Netw.*, vol. 135, pp. 15–31, Apr. 2018.

[24] G. K. Rajbahadur, A. J. Malton, A. Walenstein, and A. E. Hassan, "A survey of anomaly detection for connected vehicle cybersecurity and safety," in *Proc. IEEE Intell. Veh. Symp.*, 2018, pp. 421–426.

[25] E. Seo, H. M. Song, and H. K. Kim, "GIDs: GAN based intrusion detection system for in-vehicle network," in *Proc. 16th Annu. Conf. Privacy Security Trust (PST)*, 2018, pp. 1–6.

[26] H. Lee, S. H. Jeong, and H. K. Kim, "OTIDs: A novel intrusion detection system for in-vehicle network by using remote frame," in *Proc. 15th Annu. Conf. Privacy Security Trust*, 2017, pp. 57–66.

[27] E. Perraud, "Machine learning algorithm of detection of DoS attacks on an automotive telematic unit," *Int. J. Comput. Netw. Commun.*, vol. 11, no. 1, p. 23, Apr. 2019.

[28] L. B. Othmane, L. Dhulipala, M. Abdelkhalek, M. Govindarasu, and N. Multari, "Detection of injection attacks in in-vehicle networks," Dept. Comput. Sci., Iowa State Univ., Ames, IA, USA, Rep., 2019.

[29] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA)*, 2016, pp. 130–139.

[30] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network," in *Proc. IEEE Int. Conf. Inf. Netw. (ICOIN)*, 2016, pp. 63–68.

[31] M. Gmiden, M. H. Gmiden, and H. Trabelsi, "An intrusion detection method for securing in-vehicle can bus," in *Proc. 17th Int. Conf. Sci. Techn. Autom. Control Comput. Eng. (STA)*, 2016, pp. 176–180.

[32] M. Marchetti and D. Stabili, "Anomaly detection of can bus messages through analysis of ID sequences," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2017, pp. 1577–1583.

[33] M.-J. Kang and J.-W. Kang, "A novel intrusion detection method using deep neural network for in-vehicle network security," in *Proc. IEEE 83rd Veh. Technol. Conf. (VTC Spring)*, 2016, pp. 1–5.

[34] A. Alshammari, M. A. Zohdy, D. Debnath, and G. Corser, "Classification approach for intrusion detection in vehicle systems," *Wireless Eng. Technol.*, vol. 9, no. 4, pp. 79–94, 2018.

[35] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "CANET: An unsupervised intrusion detection system for high dimensional can bus data," *IEEE Access*, vol. 8, pp. 58194–58205, 2020.

[36] F. A. Ghaleb, A. Zainal, M. A. Rassam, and F. Mohammed, "An effective misbehavior detection model using artificial neural network for vehicular ad hoc network applications," in *Proc. IEEE Conf. Appl. Inf. Netw. Security (AINS)*, 2017, pp. 13–18.

[37] V. Chockalingam, I. Larson, D. Lin, and S. Nofzinger, "Detecting attacks on the can protocol with machine learning," in *Proc. 8th Annu. EECS 588 Security Symp.*, 2016, pp. 1–7.

[38] I. Berger, R. Rieke, M. Kolomeets, A. Chechulin, and I. Kotenko, "Comparative study of machine learning methods for in-vehicle intrusion detection," in *Computer Security*. Cham, Switzerland: Springer, 2018, pp. 85–101.

[39] S.-F. Lokman, A. T. Othman, and M.-H. Abu-Bakar, "Intrusion detection system for automotive controller area network (CAN) bus system: A review," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, p. 184, Jul. 2019. [Online]. Available: https://doi.org/10.1186/s13638-019-1484-3

[40] J. Blömer, C. Lammersen, M. Schmidt, and C. Sohler, "Theoretical analysis of the *k*-means algorithm—A survey," in *Algorithm Engineering*. Cham, Switzerland: Springer, 2016, pp. 81–116.

[41] S. C. Hpl, "Introduction to the controller area network (CAN)," Texas Instrum., Dallas, TX, USA, Rep. SLOA10, 2002.

[42] BOSCH, *CAN Specification. Version 2.0. 1991*, Robert Bosch GmbH, Stuttgart, Germany, 1991.

[43] C. Miller and C. Valasek, "Adventures in automotive networks and control units," in *Proc. DEFCON Conf.*, vol. 21, 2013, pp. 260–264.

[44] O. Avatefipour and H. Malik, "State-of-the-art survey on in-vehicle network communication (CAN-BUS) security and vulnerabilities," 2018. [Online]. Available: arXiv:1802.01725.

[45] H. K. Kim. *Car-Hacking Dataset for the Intrusion Detection*. Accessed: Apr. 10, 2020. [Online]. Available: http://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset

[46] L. van der Maaten and G. Hinton, "Visualizing data using *t*-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Oct. 2008.

Gianni D'Angelo received the Italian "Laurea" degree (cum laude) in computer engineering and the Ph.D. degree in computer science from the University of Sannio, Salerno, Italy, in 1998.

He is a Research Fellow and a Contract Professor with the University of Salerno, Fisciano, Italy. He gained experience in the world of the pattern recognition, decision making, deep learning, evolutionary algorithms, neural networks, fuzzy logic, ANFIS systems, genetic algorithms, and parallel programming applied in various scientific and industrial fields. He is also engaged in research and development of Web and mobile applications involving Internet technologies and embedded devices. He has authored numerous articles published in international journals, books, and conferences. His research interests concern with the development of soft computing algorithms for HPC and parallel computing for knowledge discovery in big data context.

Dr. D'Angelo currently serves as a Reviewer, an Editorial Board, and a Guest Editor for several international journals.

Arcangelo Castiglione received the M.S. and Ph.D. degrees in computer science from the University of Salerno, Fisciano, Italy, in 2011.

He is currently an Assistant Professor with the Department of Computer Science, University of Salerno. His research mainly focuses on cryptography, information security, computer security, and digital watermarking.

Dr. Castiglione is an Associate Editor for the *IET Cyber–Physical Systems: Theory and Applications*, the *Journal of High Speed Networks*, the *International Journal of Embedded Systems*, *Future Internet*, *Bulletin of Electrical Engineering and Informatics*, and *Connection Science*, as well as he has been a Guest Editor for several Special Issues and Volume Editor for Lecture Notes in Computer Science. He has been a member of several program committees for international conferences, and a Reviewer for several scientific journals and conferences. He serves as a Secretary the IEEE Technical Committee on Scalable Computing and the IEEE Systems, Man, and Cybernetics Technical Committee of Cybermatics.

Francesco Palmieri received the Italian "Laurea" degree and the Ph.D. degree in computer science from the University of Salerno, Fisciano, Italy, in 1989.

He is a Full Professor with the University of Salerno. Previously, he has been an Assistant Professor with the Second University of Naples, Caserta, Italy, and the Director of the Telecommunication and Networking Division of the Federico II University, Naples, Italy. At the start of his career, he also worked for several international companies on networking-related projects. He has been closely involved with the development of the Internet in Italy as a Senior Member of the Technical-Scientific Advisory Committee and of the CSIRT of the Italian NREN GARR. He has published a large number of papers in leading technical journals, books, and conferences. His major research interests concern high performance networking protocols and architectures, routing algorithms, and network security.

Prof. Palmieri currently serves as the Editor-in-Chief of an international journal the *Journal of High Speed networks* and is part of the Editorial Board or an Associate Editor of several other well reputed ones (that is, the *IEEE Transactions on Dependable and Secure Computing*, *Future Generation Computer Systems*, *Applied Soft Computing*, and *Soft Computing*).