# Detection of Message Injection Attacks Onto the CAN Bus Using Similarities of Successive Messages-Sequence Graphs

Mubark Jedh, Lotfi Ben Othmane, *Senior Member, IEEE*, Noor Ahmed, and Bharat Bhargava, *Life Fellow, IEEE*

***Abstract*—The smart features of modern cars are enabled by a number of Electronic Control Units (ECUs) components that communicate through an in-vehicle network, known as Controller Area Network (CAN) bus. The fundamental challenge is the security of the communication link where an attacker can inject messages (e.g., increase the speed) that may impact the safety of the driver. Most of existing practical IDS solutions rely on the knowledge of the identity of the ECUs, which is proprietary information. This paper proposes a message injection attack detection solution that is independent of the IDs of the ECUs. First, we represent the sequencing of the messages in a given time-interval as a direct graph and compute the similarities of the successive graphs using the cosine similarity and Pearson correlation. Then, we apply threshold, change point detection, and Long Short-Term Memory (LSTM)-Recurrent Neural Network (RNN) to detect and predict malicious message injections into the CAN bus. The evaluation of the methods using a dataset collected from a moving vehicle under malicious RPM and speed reading message injections show a detection accuracy of 97.32% and detection speed of 2.5 milliseconds when using a threshold method. The performance metrics makes the IDS suitable for real-time control mechanisms for vehicle resiliency to cyber-attacks.**

***Index Terms*—Industry applications, security, information security, intrusion detection, intelligent transportation systems, transportation, vehicle, mathematics, algorithms, detection algorithms.**

## I. Introduction

THE growing market explosion on modern cars with high premium prices is driven by the increased consumer awareness for their safety features and superior functionalities. This is credited to a number of Electronic Control Units (ECUs) components that communicate through an in-vehicle network, known as Controller Area Network (CAN) bus [1]. Within a single vehicle, there is a complex network of around one hundred collaborating ECUs, as depicted in Figure 1. These ECUs use a large software base of about 100MB to control the functionalities of the vehicle through message exchanges using the CAN bus [2]. There are two types of CAN: (1) CAN-H (high), a fast bus for communicating critical data, such as the engine, transmission, and speed messages, and (2) CAN-L (low), a slow bus for communicating non-critical data such as infotainment data [2]. Most importantly, it's used for safety mechanisms such as; collision avoidance, anti-lock brakes, traction control, and electron stability control.

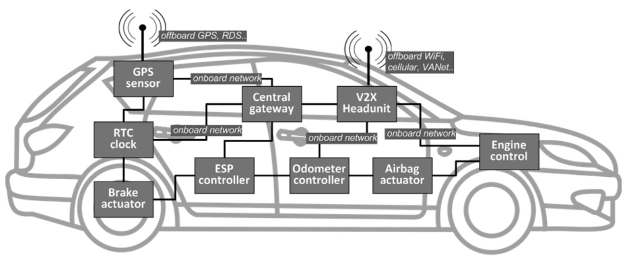


Fig. 1. Example of connected vehicle [3].

Furthermore, the CAN bus enables intercommunication link within the vehicle and external vehicles and devices through WiFi, Bluetooth, or cellular networks. This capability is exploited by the Intelligent Transportation System (ITS) applications, such as infotainment systems, fleet management systems, parking assistance, remote diagnostics, eCall, remote engine start, and Cooperative Adaptive Cruise Control (CACC) systems. These applications communicate with the ECUs of the vehicle through the CAN bus to improve the experience of the customers [3]–[5] by, for example, reducing the speed and activating the brakes of the vehicle.

The CAN bus was designed as a stable, safe, and flexible closed network without considering security, specifically, authentication and authorization mechanisms. In addition, the extension of the network through the On-Board Diagnostics (OBD) to provide ways to report self-diagnosed errors and malfunctions through On-Board Units (OBUs) contributed to expanding the attack surface (the set of ways an attacker can compromise the vehicle) through the CAN bus [7]–[9]. Upstream's research team identified 367 publicly reported incidents for a decade long [6]. The analysis of these incidents shows an exponential growth of attacks, as depicted by Figure 2a. Among these attacks, 27% involved taking control of the car, as depicted by Figure 2b.

To remedy these security problems, a wide array of defensive security solution schemes has been proposed. Most

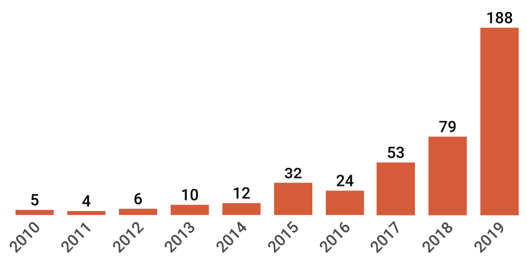
Manuscript received December 24, 2020; revised May 17, 2021 and June 20, 2021; accepted July 3, 2021. Date of publication July 19, 2021; date of current version August 30, 2021. This research is partly funded by Iowa State University's Regents Innovation Fund (RIF). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Kai Zeng. *(Corresponding author: Lotfi Ben Othmane.)*

Mubark Jedh and Lotfi Ben Othmane are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA (e-mail: lbenothmane@icloud.com).

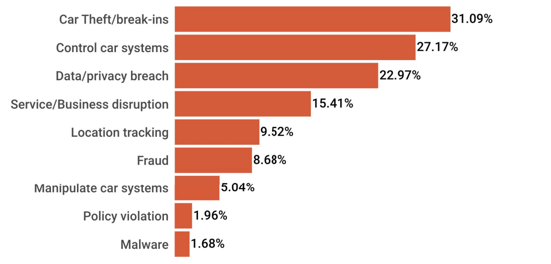
Noor Ahmed is with the Air Force Research Laboratory, Rome, NY 13441 USA.

Bharat Bhargava is with the Department of Computer Science, Purdue University, West Lafayette, IN 47907 USA (e-mail: bbshail@purdue.edu).

Digital Object Identifier 10.1109/TIFS.2021.3098162



(a) Increase of cyber-attacks on connected vehicles.



(b) Impacts of cyber-attacks on connected vehicles.

Fig. 2. Growth and distribution of cyber-attacks on connected vehicles between 2010 and 2019 [6].

of these solutions typically require the modification of the CAN protocol, which is not practical for aftermarket vehicles [3], [4]. Others have attempted to devise machine learning-based solutions driven by syntactic data through simulations or data related to researchers' devices to detect malicious behaviors [10], which limits the practicality of their efficacy.

Many of the practical IDS solutions require the knowledge of CAN ID of the messages (the CAN messages include an ID that indicates the information embedded in the message, e.g., speed and brake) which is proprietary to the car manufacturers [11]. The information is used to identify abrupt values changes that indicate attacks. An alternative approach is to use OBDII to extract CAN messages exchanged by the vehicle' ECUs and translate the CAN IDs of the messages to their corresponding Priority ID (PID), as in [5]. This approach prevents attackers who may reverse engineer the IDS device from recovering the semantics of CAN IDs and use them for attacks on vehicles. Unfortunately, this approach has several imitations. First, a reasonable proportion of the CAN message types are not captured (or ignored) through OBDII because of lack of equivalent PIDs for every CAN IDs. For instance, song and Kim [12] were able to match the PIDs of only nine out of 40 ECUs to their corresponding CAN IDs. In addition, we observed in our previous studies [5], [13] with a Ford vehicle for the importance of the differences in the size of CAN message trace and PID messages trace.[1] Second, PID messages are acquired through query/response mechanisms [5], which returns a limited set of *fresh* messages. These limitations prevent the development of an effective machine-learning-based IDS for the CAN bus.

In this paper, we answer the question: *Can we detect cyber-attacks on a moving vehicle without the need to know the IDs of the vehicle's ECUs?* We hypothesize that there is a pattern of messages sequences between the collaborating ECUs of the vehicle, and injecting CAN messages disrupts the sequences' pattern. To answer this question, we first develop a direct graph that represents the sequence relations between ECUs messages based on their CAN ID, which we call *Messages-Sequence Graph (MSG)*. Then, we use a sliding-window approach to compare the similarity between the graphs computed from the sequences of messages sent through the CAN bus in successive time windows using the *cosine similarity* and *Pearson correlation* metrics. Next, we apply the *threshold, change point detection, and LSTM-RNN* techniques to detect and predict injection attacks on the CAN bus. The main contributions of the paper are:

1) An effective sequential CAN Bus message injection attacks detection mechanism that uses similarity metrics of successive messages-sequence graphs that has an accuracy of 97.32% and detection speed of 2.5 milliseconds.
2) An effective LSTM-RNN-based Machine Learning (ML) technique for detecting messages injections attacks on CAN Bus from MSGs.
3) A change-point detection technique for detecting messages injection attacks on CAN Bus from MSGs.

The paper is organized as follows. Section II discusses the security issues of the CAN protocol and Section III reports about related work. Then, Section IV presents the research method, Section V describes the evaluation methods and Section VI reports the results. Section VII concludes the paper.

## II. Security Issues on Can Protocol

The Controller Area Network (CAN) is a network protocol developed by Robert Bosch in 1986 [1] to communicate the ECUs that control the behaviors of the vehicles' mechanical and electrical components [14]. All cars manufactured in USA after 2008 support the CAN protocol and offer a means to interface with it through the On-Board Diagnostics (OBD)-II port, which is usually located under the steering wheel. The OBD is implemented in the vehicles to provide ways to report self-diagnosed errors and malfunctions [15] and is mandatory for all cars and light trucks sold in the United States and the European Union. This port gives direct access to the CAN network, which inherently creates the attack surface, the set of ways an attacker can penetrate the vehicle.

By design, CAN bus is resilient, robust, and easy to wire but has a set of security weaknesses. The fundamental security issues are: (1) The CAN bus data frame has no source identifier field to identify the legitimacy of the sending ECU. As a result, ECUs cannot trust messages based on their sources. (2) The protocol does not protect the confidentiality of CAN messages. Attackers can read the messages exchanged in the CAN bus and infer information that could be used to stage attacks, including associating CAN ID with ECUs. (3) The use of priority in gaining the right to send messages could be easily used to flood the bus of a vehicle with messages that have

[1] We observed that synchronization-like messages, for example, do not have corresponding PIDs.

small CAN ID and prevent other ECUs from communicating and makes their service unavailable, i.e., Denial of Service (DOS) attack. frame [16].

Typically, attackers inject messages into the CAN bus, suppress legitimate messages (i.e., legitimate messages that have higher priority IDs than injected messages are ignored) or compromise an ECU to behave maliciously through either direct or indirect access points. Direct access point attacks include the OBD-II port, the CD player or the USB port [17]. For instance, the attacker can inject malicious CAN messages through the USB port, encode a malicious software onto a CD to exploit the entertainment system of a car or use it to access the CAN bus or update the firmware of an ECU. The indirect access points are short and long-distance wireless access points connected to the CAN bus. These include Bluetooth, on broad Wi-Fi such as Vehicle-to-Vehicle (V2V) devices, remote keyless entry, Tire Pressure Monitoring System (TPM), Global Positioning System (GPS) and cell phone network.

The attack surface increases as more features, such as vehicle apps used for telematics services, road-side assistance, V2V applications, and remote diagnostic, are added in the car. Connecting vehicles to external entities through wireless devices expose the security weaknesses of the CAN bus [3]. Wolf *et al.* were among the pioneers in describing the weaknesses of the CAN bus, including unauthorized access into the CAN bus and lack of confidentiality and integrity checks of CAN messages [18]. Further, Hartzell *et al.* discussed the impacts of the common entry points, limited bandwidth, multi-cast messaging, lack of encryption of messages, and multi-system integration on the security of the CAN bus [17].

Valeseka and Miller demonstrated that hijacking connected vehicles is possible. They identified remotely the IP address of their Jeep car from the Sprint Cellular network and took over the vehicle's critical features, including disabling breaks, controlling the steering wheels, and turning on/off windshield wipers while the car is moving [19]. Recently, Golston and Green reported successful exploits of vulnerabilities to take control of recent Tesla cars [20], [21]. An experimental security analysis of the attack surface of the connected vehicle including the short and long-range wireless channels, the entertainment systems (i.e., CD player, iPod port, USB port) and the electric charge port was performed by Koscher *et al.* [22], who showed the wide range of potential attacks on connected vehicles. Othmane *et al.* [23] surveyed security experts and validated Koscher *et al.*'s insights that attacks on connected vehicles are not lab experiments anymore. Thus, an effective and practical detection and prediction of injections of CAN messages is critical for connected and autonomous vehicles.

## III. Related Work

Attackers take control of connected vehicles by injecting messages into their in-vehicle networks, mainly their CAN bus. Several hardware and software-based encryption methods were proposed to prevent eavesdropping CAN messages. For instance, Farag *et al.* proposed CANTrack, which encrypts the data payload field of CAN messages to prevent access to them by non-legitimate entities [24]. On the one hand, message authentication mechanisms are considered to detect and/or prevent injection attacks. For instance, Hiroshi *et al.* proposed the use of a secret key that would be distributed to legitimate ECUs of the given vehicle [25]. Each of the legitimate ECUs of the vehicle must reject CAN messages that are not authenticated with the key. Authentication-based mitigation solutions have been heavily explored [26]–[29], but they cannot be used for aftermarket vehicles.

Other notable mitigation techniques have been reported in e.g., [3], [4]. In general, some of these proposed mitigation solutions trade performance and security requirements given the computation and communication constraints of the ECU used in vehicles and time-criticality of some of the messages (i.e., braking). Others, require modification of the currently well-vetted CAN bus protocol (with respect to safety) which makes them not practical, especially for aftermarket vehicles.

Early detection of message injection attacks on CAN bus methods were based on frequency, entropy, and correlation [30]. For instance, Taylor *et al.* proposed an Intrusion Detection System (IDS) based on messages frequency using the Hamming distance between successive message data fields to detect attacks [31]. The methods were validated using syntactic dataset, a data set that is constructed by adding and removing messages to the set of the CAN messages collected in normal driving conditions. IDSs that are based on frequency, timing, and entropy are effective for only attacks that disturb the frequencies of CAN bus messages [10] because they assess the collision management in the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) algorithm used by the CAN bus.

CAN messages compete on the CAN Bus; the message with the lowest CAN ID is injected first in the bus. Thus, frequency-based IDS would work in injecting messages with low CAN ID (the frequency of messages in a time-step becomes high) but may fail to detect attacks that inject messages with high CAN ID—as the frequency of injected messages is low. Our method is more sensitive to the injection of messages. It accounts for each injection as one legitimate two-messages sequence is missing and one illegitimate two-messages sequence exists. Unfortunately, we cannot implement the frequency-based method and compare it to our method because we did not slice our data based on the time-window when we collected it.

Recently, there has been increased attention for the use of machine learning techniques based on feature extraction from the CAN bus data frame messages to identify attacks. At the core of these solution methods is discriminating messages associated to attacks and those that are not, with acceptable accuracy and false positive [32]. Neural Network (NN) has been the commonly used ML-based approach for designing IDSs for the CAN bus, e.g., [11], [33]–[35]. The main problem with this supervised learning method is that it requires extensive time to develop high-performance IDS models from labeled data [11] and the trained models are specific for vehicle' make and model, and driver. The unsupervised ML-based IDSs use mostly message-timing, message-frequencies, and message-latency [36], [37]. These methods identify the CAN bus fuzzing through the assessment of the contention algorithm used in the CSMA/CD protocol.

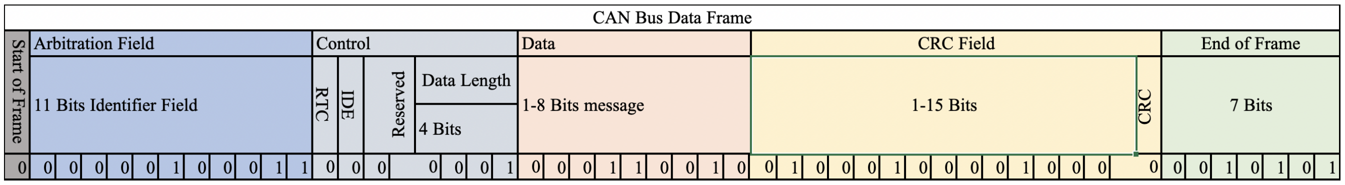


Fig. 3. CAN message data frame format.

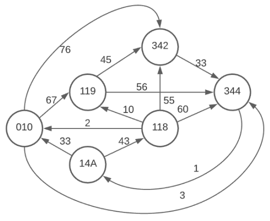
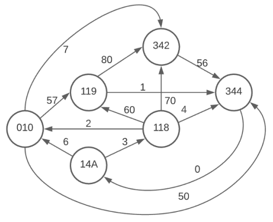



Fig. 4. Similarity of two messages-sequence graphs at successive time-window $t$ (left) and $t+1$ (right). The labels of the nodes are the CAN ID and the labels of the edges are number of times a message with the CAN ID source of the edge is followed by a message with the CAN ID destination of the same edge during the time-window. For example, 33 messages and 56 messages with ID 344 followed messages with ID 342 at resp. time-windows $t$ and $t+1$, which indicate a possible change of the behavior of the vehicle.

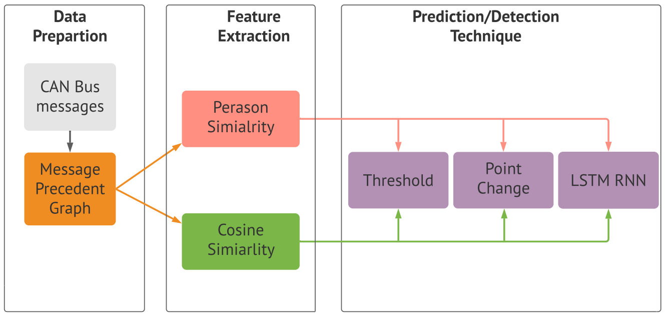


Fig. 5. High-level view of our solution methodology.

```
(1522004165.988068) can0 264#0003948C0C00FD83
(1522004165.988292) can0 04A#00D8D8D8FED8D8D8
(1522004165.988807) can0 010#0DFB00D4009380A1
(1522004165.992318) can0 050#8000880000000000
```

Fig. 6. An Illustration of CAN data stream format– Each row corresponds a Timestamp followed by the CAN interface name, the CAN ID of the ECU (i.e., 264), followed by "#" and the actual data in hexadecimal format.

The common data sources that are used to assess the accuracy of the ML-based IDS solutions include; data from the owner devices [38], synthetic/artificial data [39], simulated data [40], and data from a stationary/parked vehicle [36], [41], [42]. This limits the confidence in the results and threatens its validity [43]. To the best of our knowledge, our previous work [10] and the works of Stachowski [11] are the only two studies that used datasets collected from moving vehicles under messages injection attacks. Both studies require knowing the association of CAN ID to the vehicle's ECUs, which is proprietary information and dependent on the make and model of the vehicle.

Wu *et al.* [37] and Young *et al.* [44] provide comprehensive surveys on IDS for connected and autonomous cars. Unlike most machine learning-based studies in the literature, our proposed technique neither depends on the make or model of the car nor its proprietary information (i.e., CAN ID); it uses a combination of graph-based and machine learning techniques.

## IV. Research Method

We approached the problem of detecting and predicting message injection attacks on modern cars by capturing the patterns of the sequencing of the CAN messages exchanged among the ECUs. As depicted by Figure 5, data collected from the CAN bus are represented as a direct *Message-Precedence* graph (left box). Then, the *Pearson and Cosine* similarities of successive graphs are used as the ML features (middle box). Finally, *LSTM-RNN, Threshold, and Point Change* prediction is used to predict messages injection (right box). We validate our methodology using datasets collected from a moving car while fabricated speed and RPM messages are being injected into its CAN bus. We describe these methods in the following subsections.

### A. Message-Sequence Graph

Technically, CAN is a time-synchronized broadcast, multi-cast reception message network bus. Figure 3 above illustrates the CAN message format. Each message consists of a data frame, remote frame, error frame, or overload frame. The data frame is used to exchange data between the nodes, the remote frame is used to request the transmission of a specific identifier, the error frame is transmitted by any node detecting an error, and the overload frame is used to inject a delay between data or remote frames [1].

We observed by monitoring the CAN Bus of the vehicle that the sequence of CAN IDs of messages are almost the same when the car is parked and that there are patterns of CAN ID sequences associated with actions such as increase of speed. Figure 6 shows a snapshot of a message captured from the CAN bus. Each message has a freshness tag (like a timestamp) followed by the CAN bus ID, the CAN ID of the ECU, and the actual data. The ECU of a given vehicle collaborate by exchanging successive messages through the CAN bus to perform specific actions. For example, a CAN message representing an increase of the fuel will usually be followed by a message representing an increase of RPM and most likely a message representing an increase of the speed [10]. It is intuitive to see that each of these messages have a sequence pattern.

*We hypothesize that there are patterns of sequences of messages exchanged between the collaborating ECUs of the vehicle and injecting CAN messages disrupts the* *messages-sequence patterns*. The idea is to capture the pattern of the sequences of the messages and represent it as a graph where the nodes represent the CAN ID and the edges represent the sequences of the messages of that given CAN ID. Note that the messages captured are independent of vehicle's ECUs.

Graphs are an effective tool for modeling relationships [42] and are commonly used in applications like fraud detection and social networking anomaly detection [45]. Marchetti and Stabili first used graphs to construct an IDS for CAN Bus [46] and achieved 100% accuracy by evaluating the method using synthetic dataset. Islam *et al.* evaluated Marchetti and Stabili method using dataset of a parked vehicle [47] and found the accuracy to be 86.23% [42]. Furthermore, they proposed the use of the variation of the number of edges of the MSGs (size of the cliques of the graphs) as an indicator of attacks and Chi-square to compare the data computed from a test dataset to data calculated from a baseline datasets [42] with 100% accuracy for spoofing attacks. We implemented the method and tested it using the datasets that we collected from a moving vehicle [13]. We found that the method reports attacks when using a dataset representing a moving vehicle in the presence of speed reading injection ($X^2(1, N = 58) = 9,56$, $p < 0.01$) and RPM readings injection ($X^2(1, N = 58) = 29,18$, $p < 0.01$). Then, We split the dataset representing normal behavior into two sets—the two sets are baselines. The method reports that the two sets are independent, which implies (according to the paper) that one dataset set is related to an attack—which is not true. We note that the first part of the normal dataset is related to an increase in the vehicle's speed, and the second part of the normal dataset is related to a decrease in the vehicle's speed ($X^2(1, N = 27) = 5,27$, $p < 0.01$). This implies that the method is effective in detecting attacks on parked vehicles but is ineffective on moving vehicles. We note that the variation in the sizes of the graphs' cliques is high in moving vehicles, which makes the method inefficient.

We opted to use the network flow of the graphs rather than the size of the MSG-cliques. Formally, let $E(N_1, N_2)$ be an edge that represents a CAN message with CAN ID $N_1$ followed by a CAN message with CAN ID $N_2$. Let the label of edge $E(N_1, N_2)$ represent the frequency of CAN messages with CAN ID $N_1$ followed by CAN messages with CAN ID $N_2$. We call this graph *Messages-Sequence Graph (MSG)* as shown in Figure 4, and the construction of the graph sequence is illustrated in Algorithm 1.

In Algorithm 1, we first create a dictionary of the CAN ID exchanged in the CAN bus–The CAN ID become the labels of the rows and columns on a matrix representing the edges of the MSG in line 2. Then, it loops over all the CAN messages that were exchanged during the time window $w$ (we use windows of e.g., 100 successive messages) and increases the label of the edge linking the node representing the CAN ID of a given message to the node repressing the CAN ID of the previously processed message in lines 3-7. Equation 1 represents the distribution of the messages-sequences at time $t$.

$$D(t) = E(N_i, N_j)(t) \tag{1}$$

**Algorithm 1** Message Graph Sequencer

**Input:** $CANData$: A batch of CAN messages with size window
**Input:** $window$: Size of the batch messages
**Output:** $MSG$: message precedence graph

```
1: function COMPUTEMSG($data$, $window$)
2:     Call CreateDictionary()
3:     for $k \leftarrow 1$ to $N$ do
4:         key ← concat(CANData[$j$]['CAN ID']
5:         + CANData[$j$+1]['CAN ID'])
6:         MSG[key] ← MSG[key] +1
7:     end for
8:     return $MSG$
9: end function
```

### *B. Feature Extraction*

We hypothesised that the MSG representing the messages exchanged in a CAN bus during a time slot $t$ with size $w$ is $similar$ to the MSG representing the CAN messages exchanged during the following time slot $t+1$ with same size $w$ in the case of normal driving behavior and that injection of messages into the CAN bus disrupts this pattern. We formulate the *Similarity* concept for our IDS using Equation 2. That is, the similarity *Sim* at time $t + 1$ is the similarity of the distributions of the messages-sequences $D$ at time $t$ and at time $t + 1$.

$$Sim(t + 1) = Similarity(D(t), D(t + 1)) \tag{2}$$

There are several similarity metrics that measure the similarity between two graphs [48] including Cosine similarity, Pearson and Cramer correlation, chi-squared, T-test and Levene's tests. We briefly describe the two methods that we selected for our study and the rationale behind our choice.

*1) Cosine Similarity:* measures the cosine angle between two non-zero vectors and determines whether the angle point is in the same direction or not. It helps to tease apart the types and relationships of vertices in social networks [48]. Simply, it's metric measures the angle between two vectors as formulated by Equation 3, where $x$ and $y$ are two vectors. Cosine Similarity is often used to measure document similarity, mainly for plagiarism detection. Simply, the metric compares two documents by measuring the similarity of the vectors of the frequencies of the words in both documents, then outputs a number between 0 and 1. The closer the number is to 1, the more similar the two vectors.

$$Cosim(x, y) = \frac{\sum_{i=1}^{n}(x_i \times y_i)}{\sqrt{\sum_{i=1}^{n} x_i^2} \times \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{3}$$

The cosine similarity for the two MSGs of Figure 4 is 0.66.

*2) Pearson Correlation:* The correlation measures the strength of the linear association between two non-zero vectors and is given by Equation 4. It is widely used to measure similarity [48]. It compares the similarity and returns values between 1 and $-1$. The closer the value to 1 or $-1$, the strong the modeled relationship is.

$$PC(x, y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})}} \tag{4}$$

The Pearson correlation coefficients of the two MSGs of Figure 4 is 0.63.

The Pearson correlation measures the strength of the relationships between two variables and is a de facto metric for measuring similarity between two datasets. The cosine similarity method is commonly used in forensics investigation, malware analysis, and IDS [49], [50]. This paper is the first to report the use of these metrics to construct IDS from CAN Bus data. Further, experimented with other metrics, such as t-test and Levene's test, which exhibited much lower accuracy [51] and were excluded from the paper.

### C. Prediction of Messages Injections

In conjunction with *Cosine and Pearson* similarity metrics, we employed three techniques to predict the messages injection attacks, which are: threshold, LSTM-RNN and Change Point Detection (CPD). In general, the threshold is commonly considered as the base case, and the LSTM and CPD are commonly used IDS solutions [52].

*1) Threshold:* A threshold is a selected value of the given similarity metric that is believed to provide better accuracy in detecting injection of messages. To identify "good" thresholds for the three selected similarity metrics, we variate the threshold values for the given metric and compute its accuracy until we observe an "optimal" value.

*2) Recurrent Neural Network-Long Short-Term Memory:* The technique was first introduced in 1997 and become the core methodology deep learning [53]. It is a gradient-based architecture developed for modeling time-series data with long-term dependencies [54]. The design solves the problem of vanishing gradient by allowing errors to be back-propagated through time. The LSTM was constructed using the sequential model of a linear stack of layers, which are recurrently repeating memory blocks. It is very powerful in sequence prediction problems because it can store past information for a long time.

*3) Change Point Detection:* The problem could be formulated as follows [55]. We observe a sequence of observations $y_i/i = 1, \dots, n$, indexed by some meaningful ordering, such as time or location in the sequence. Change Point Detection (CPD) is concerned about testing the null hypothesis:

$$H_0 : y_i \sim F_0, \quad i = 1, \dots n, \tag{5}$$

against the single change-point alternative:

$$H_1 : \exists 1 \leqslant \tau \leq n, y_i \sim \begin{cases} F_1, & i < \tau \\ F_0, & \text{Otherwise} \end{cases} \tag{6}$$

CPD estimates the change-points-location where the distribution of a sequence of observations abruptly changes [55], [56] and is commonly used for detecting anomalous behavior in sequences of observations [57]. The sequence of observations used in this work are the similarities between consecutive MSGs.

TABLE I
DATASET SIZE

| No | Description | # of CAN messages |
|---|---|---|
| 1 | CAN Data for no injection of fabricated messages | 23,963 |
| 2 | CAN Data with injection of "FFF" as the speed reading | 88,492 |
| 3 | CAN Data with injection of "FFFF" as the RPM reading | 30,308 |

## V. EVALUATION METHODS

In this section, we discuss our experimental evaluation setup and the parameters used for our study.

### A. Evaluation Datasets

In our previous study [10], we collected a log of CAN bus messages for (1) normal driving behavior, (2) injection of fabricated speed reading messages onto the CAN bus, and (3) injection of fabricated RPM reading messages onto the CAN bus of an in-motion Ford Transit 500 2017. The data set is available in [13]. Table I shows the number of CAN messages that were used in the research. Figure 7 shows the speed and RPM readings and their relationships in the three datasets [10]. The plot at the left shows that the speed of the vehicle varies between 0 and almost 30 mph and the RPM readings varies from almost 0 to almost 2300 units during a normal driving scenario. The plot at the center shows that the speed of the vehicle in this drive test reached almost 11 mph and the RPM readings oscillates between the actual values and the value that corresponds to the injected hex value "FFFF", i.e., 120000. The plot at the right shows the RPM readings of the vehicle in this drive test reached almost 1800 units and the speed readings oscillates between the actual values and the value that corresponds to the injected hex value "FFF", i.e. almost 25 mph.

The Pearson correlation between the speed readings and RPM readings represented by the orange lines in Figure 7 shows a strong relationship (the coefficient is 0.85) between the 2 quantities in the plot at the left, weak relationship (the coefficient is 0.33) in the plot at the center, and no relationship (the coefficient is -0.013) in the plot at the right [10]. This suggests that the ECUs of the vehicle may act on the injected RPM readings but may not act on the injected speed readings that they receive. The difference between the correlation coefficients hints to the impact of messages injection on the collaboration between the ECUs of the car, which we explore in this paper.

### B. Evaluation Method for the Similarity Techniques

We first construct the MSGs from each time-window (default 100) successive messages from the three CAN log datasets described in Table I. Then, the similarity metrics are used to compute the similarity values of the MSGs series for each of the datasets.

To assess the efficacy of each of the similarity metric, we plot the similarity values as time-series data to observe

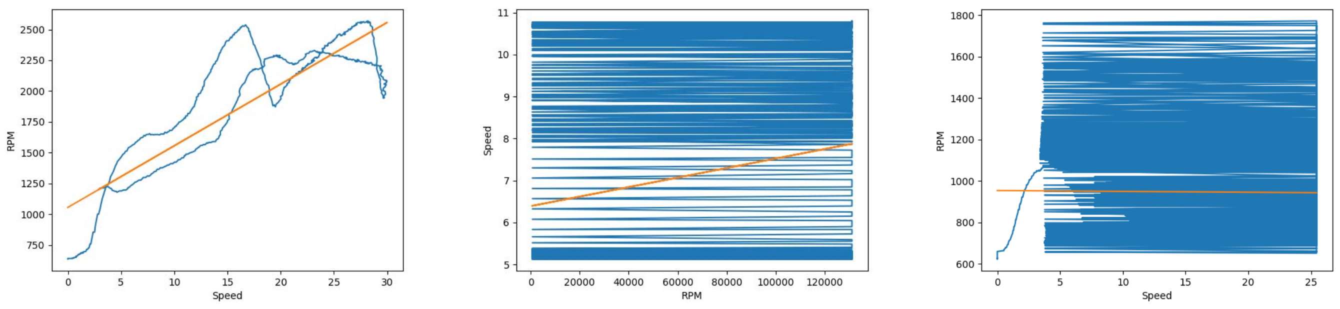


Fig. 7. An illustration of *speed* and *RPM* readings on the three datasets for a normal vehicle operation (left), and the impact on message injection attacks for RPM reading on speed reading (center), and on speed reading on RPM reading (right).

TABLE II
NORMALITY TEST FOR METRIC VARIABLES (SHAPIRO-WILK)

| Data | Cosine Similarity | Pearson Similarity |
|---|---|---|
| Normal | $1.17^{-5}$ | $1.12^{-5}$ |
| RPM | $7.48^{-4}$ | $1.79^{-3}$ |
| Speed | $9.995^{-12}$ | $9.899^{-12}$ |

the tendencies of the values computed from the three datasets. Then, we plot the distributions of the frequencies of the similarity values to observe whether similarity values computed from the no injection of fabricated messages dataset differ or not from the similarity values computed from the RPM and speed injection of fabricated messages datasets. Next, we use t-test [58] to statistically validate the difference between the two distributions of similarity values computed from the no injection of fabricated messages dataset and the distributions of similarity values computed from respectively RPM and speed injection of fabricated messages datasets.

In addition, we set thresholds for each of the similarity metrics and compared the similarity values computed from the MSGs to these thresholds. A similarity value below the threshold of the given similarity indicates injection of messages at the related time-window. Subsequently, the accuracy of the method in detecting injection of messages onto the CAN bus is computed. The best thresholds that we identified are 0.87 for both the cosine similarity and Person correlation. Section VI-A discusses the results of the evaluation.

## C. Evaluation Method for the Change Point Detection Technique

The CPD method estimates the point of change of a population from a sample data. The method uses Markov-Chain Monte Carlo (MCMC) to sample the data and Bayesian inference to detect the point at which the mean of the population changes. The parameters of the model are the mean and the standard deviation of the population and the distributions of the data before the change point and after the change point. We choose the Normal distribution for both distributions. We tested the normality of cosine similarity values and Pearson correlation coefficients computed from the three datasets using the Shapiro-Wilk [59]. The p-values of the t-tests are provided by Table II. The values indicate that all the datasets follow the Normal distribution. We used PYMC [60], [61] to evaluate the capability of the change point detection method to identify injection of CAN messages.

Using CPD method to identify injection of messages from the similarity data, we compared the identified point of changes to the actual point of changes (i.e., the time we started injecting messages) for each of the datasets. In addition, we compare the *strength of change* for each of the datasets, which is the proportion of the difference between the mean before the change point and after the change point to the average of the two means. We use threshold 1% to interpret this strength of the change; a strength of change above 1% implies there is a change and a strength of change below 1% implies there is no change. Section VI-B discusses the results of the evaluation.

## D. Evaluation Method for the RNN-LSTM

Initially, we developed prediction models using the Recurrent Neural Network (RNN)-Long Short-Term Memory (LSTM) method from the three raw-datasets described in Table I. The models had low performance. To have datasets that have a balanced number of records wrt. injection/no injection of CAN messages, we constructed two datasets out the three row datasets as follows:

- We appended the injection of speed reading messages onto the CAN bus dataset to the no injection of fabricated messages dataset to form the *constructed speed readings injection dataset*.
- We appended the injection of RPM reading messages onto the CAN bus dataset to the no injection of fabricated messages dataset to form the *constructed RPM readings injection dataset*.

We used LSTM-RNN, as implemented in package Keras [62], to predict injection of messages in the CAN bus. Table III lists the parameters that were used in the study. The sequences of MSG similarities are fed onto the first/input layer of the LSTM model. The output of the input layer is passed on to the second layer. The output of the second hidden layer is then passed on to the output/dense layer, which maps the output values to binary values, representing the states injection of CAN messages and no injection of CAN messages. The experiment is performed for both dataset and the accuracy of the method is computed and reported in Section VI-C.

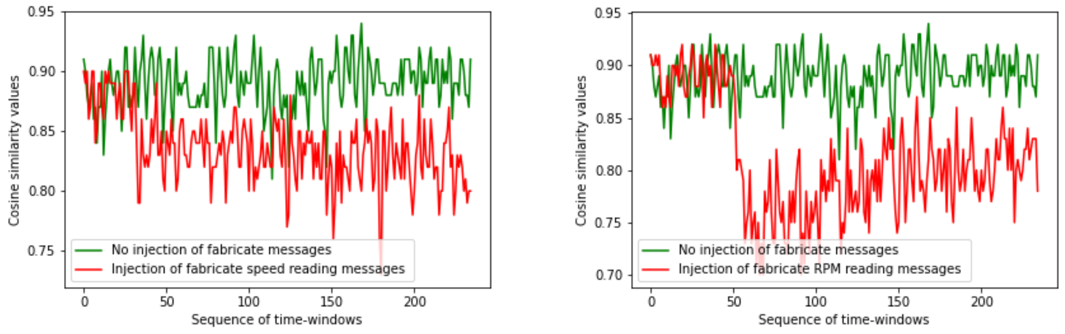


Fig. 8. Impact of the injection of speed reading messages (left) and RPM reading messages (right) on the of cosine similarity values of the MSGs of successive time-windows; the cosine-similarity is higher without the injection of speed and RPM reading messages.

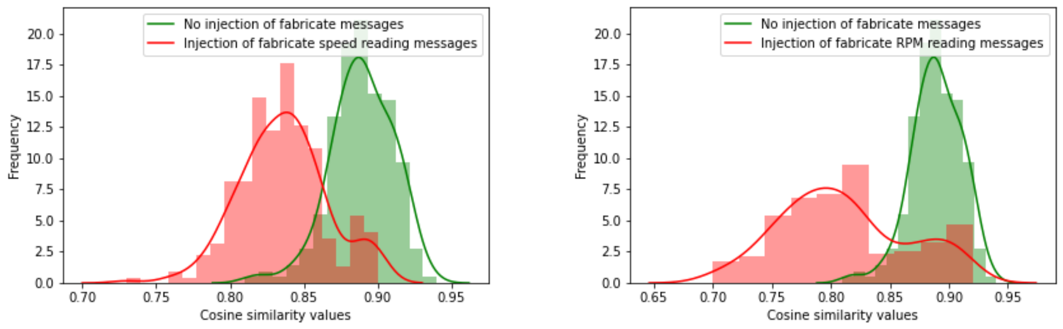


Fig. 9. Impact of the injection of speed reading messages (left) and RPM reading messages (right) on the distribution of the frequencies of the cosine similarity values of MSGs of successive time-windows.

TABLE III
RNN-LSTM Model Parameters

| Parameters | Value |
|---|---|
| Input layer | 42 units |
| Second hidden layers | 12 units |
| Dense/Output layer | 1 units |
| Dropout rate | 20% |
| Adam optimizer learning rate | 0.01 |
| Batch Size | 128 |
| Number of epochs | 128 |
| Ratio of the training dataset | 2/3 |

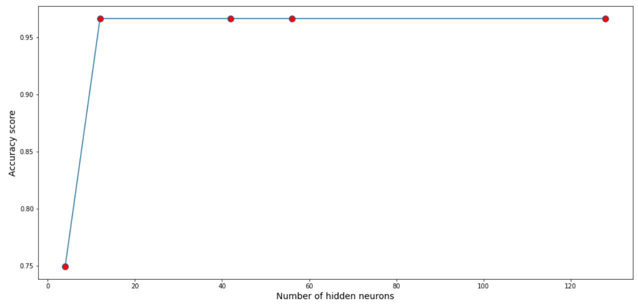


Fig. 10. The impact of the number of hidden neurons on the accuracy of the models generated from the Pearson similarities data.

Note that we adopted the commonly used RNN configuration parameters as specified in Table III, including the ratio of training dataset and identified the other parameters by visualizing the impacts of the variations of the parameters values on the accuracy of the generated prediction models. For instance, Figure 10 and Figure 11 show the impact of the number of hidden neurons and the number of epochs on the accuracy of the models respectively. In addition, inspecting the Tensorborad Graph Visualization to detect over-fitting and under-fitting of the generated models.

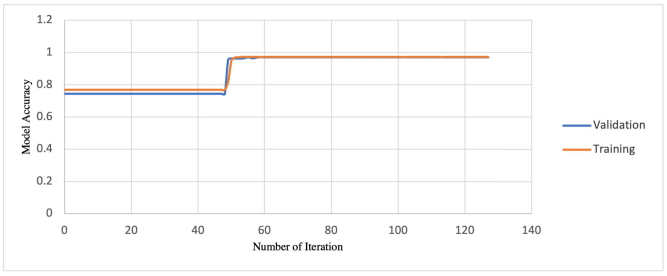


Fig. 11. Impact of the number of Epochs on the accuracy of the models generated from the Pearson similarities data.

## VI. Results of the Evaluation of the Detection on Messages Injection Using Similarity of Messages-Sequences Graphs

This section describes the capability of MSG similarity metric in conjunction with threshold, RNN-LSTM, and CPD to detect injection on CAN messages.

### A. *Evaluation of the Threshold Technique*

This subsection describes the results of using cosine similarity and Pearson correlation of MSGs and thresholds to detect injection of CAN messages.

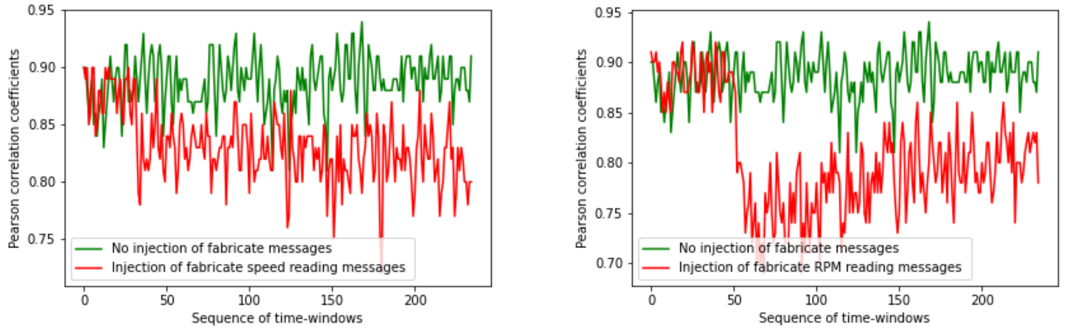


Fig. 12. Impact of the injection of speed reading messages (left) and RPM reading messages (right) on Pearson correlation coefficients of MSGs of successive time-windows; the Pearson correlation coefficients are higher without the injection of speed and RPM reading messages.

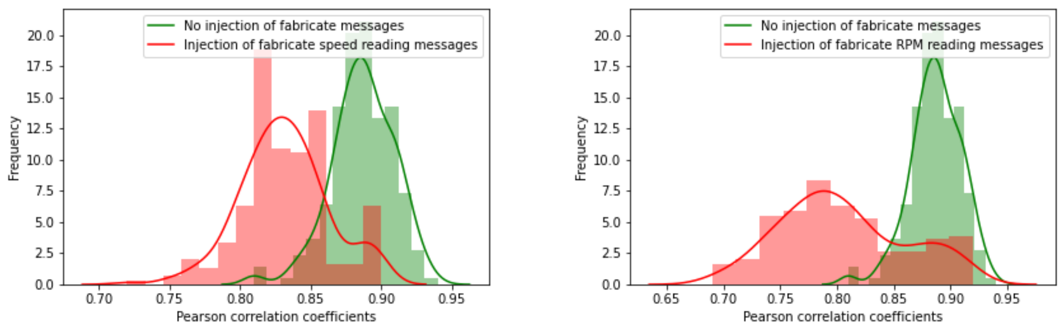


Fig. 13. Impact of injection of speed reading messages (left) and RPM reading messages (right) on the distribution of the frequencies of the Pearson correlation coefficients of MSGs of successive time-windows.

*1) Cosine Similarity Method:* Figure 8 and Figure 9 show respectively the sequence of cosine similarity values and the distribution of the cosine similarity values computed from the three datasets. We observe from Figure 8 that the plot of the sequence cosine similarity values extracted from the no injection of fabricated messages dataset is higher than the sequence of the cosine similarity values extracted from the injection of fabricated reading messages dataset.

We observe from Figure 9 that the distribution of the cosine similarity values extracted from the no injection of fabricated messages dataset is different from the distribution of the cosine similarity values extracted from the injection of fabricated reading messages dataset in red color. The t-test confirms that injection of fabricated messages onto the CAN bus impacts the sequence of messages exchanged in the CAN bus. The t-test's p-value for the difference of the mean of the cosine similarity values extracted from the injection of speed reading messages dataset and the mean of the cosine similarity values extracted from the no injection of fabricated messages dataset is 1.12 $e-74$ and the t-test's p-value for the difference of the mean of the cosine similarity values extracted from the injection of RPM reading messages dataset and the mean of the cosine similarity values extracted from the no injection of fabricated messages dataset is 2.61 $e-69$.

*2) Pearson Correlation Method:* Figure 12 shows the sequence of correlation coefficients extracted from the three datasets over time, and Figure 13 shows the distribution of correlation coefficients extracted from the three datasets.

We observe from Figure 12 that the plot of the sequence Pearson correlation coefficients extracted from the no injection of fabricated messages dataset is higher than the sequence of the Pearson correlation coefficients extracted from the injection of fabricated reading messages dataset. We observe from Figure 13 that the distribution of the Pearson correlation coefficients extracted from the no injection of fabricated messages dataset is different from the distribution of the Pearson correlation coefficients extracted from the injection of fabricated reading messages dataset. The t-test confirms that injection of fabricated messages onto the CAN bus impacts the sequence of messages exchanged in the CAN bus.

The t-test's p-value for the difference of the mean of the Pearson correlation coefficients extracted from the injection of speed reading messages dataset and the mean of the cosine similarity values extracted from the no injection of fabricated messages dataset is $9.466e-75$ and the t-test's p-value for the difference of the mean of the cosine similarity values extracted from the injection of RPM reading messages dataset and the mean of the cosine similarity values extracted from the no injection of fabricated messages dataset is $1.6e-70$.

*3) Accuracy of the Threshold Method:* Table IV and Table V summarize the accuracy of the threshold-based method in detecting injection of messages using the two similarity metrics applied to successive MSGs when using the three datasets. Table VI shows also that the cosine similarity and Pearson correlation have low false-positive rates for the case of no injection of fabricated messages dataset. We conclude that cosine similarity and Pearson correlation exhibit excellent

TABLE IV

PERFORMANCE OF THE THREE SIMILARITY METRICS WHEN USING THE INJECTION OF RPM READINGS DATASET

| Metrics | Accuracy (%) | False Positive rate (%) | Threshold |
|---|---|---|---|
| Cosine similarity | 96.65 | 3.34 | 0.87 |
| Pearson correlation | 97.32 | 2.67 | 0.87 |

TABLE V

PERFORMANCE OF THE SIMILARITY METRICS WHEN USING THE INJECTION OF SPEED READING MESSAGES DATASET

| Metrics | Accuracy (%) | False Positive rate (%) | Threshold |
|---|---|---|---|
| Cosine similarity | 89.20 | 10.80 | 0.87 |
| Pearson correlation | 90.57 | 9.43 | 0.87 |

TABLE VI

PERFORMANCE OF THE SIMILARITY METRICS WHEN USING THE NO INJECTION OF FABRICATED RPM READING MESSAGES DATASET

| Metric | Accuracy (%) | False Positive rate (%) | Threshold |
|---|---|---|---|
| Cosine similarity | 89.4 | 10.6 | 0.87 |
| Pearson correlation | 86.8 | 13.20 | 0.87 |

TABLE VII

COMPARING THE IDENTIFIED POINT OF CHANGES TO THE APPROXIMATE ACTUAL POINT OF CHANGE. (WE SET THE POINT OF CHANGE AS THE MEAN OF THE ESTIMATED POINT OF CHANGE RANGES WITH 94%.)

| Data | Cosine similarity | Pearson similarity | Actual POC |
|---|---|---|---|
| Injection of RPM reading messages | 66 | 54 | 34 |
| Injection of speed reading messages | 391 | 468 | 53 |
| No injection of fabricated messages | 169 | 174 | 0 |

accuracy in detecting injection of speed and RPM reading messages.

## B. Change Point Detection Method

This subsection describes the capability of the CPD method in conjunction with the cosine similarity and Pearson correlation of MSGs to detect injection of messages onto the CAN bus.

Table VII and shows the identified point of changes and the actual point of changes and Table VIII provides the strength of changes computed from the three datasets for both similarity techniques. Discussions on the capability of the two similarity metrics follow.

*1) Cosine Similarity:* Figure 14 shows the identified change points from the cosine similarity sequences computed from the injection of RPM reading messages, injection of speed reading messages and no injection of fabricated messages datasets. The change points are at windows 66, 391, and 169 for resp. The injection of RPM readings, injection of speed readings and no injection of fabricated messages datasets. We observe that the method detects quickly the change for the case of injection of RPM reading messages dataset (after 32 time-windows) but with significant delay for the case of injection of speed reading dataset dataset.

TABLE VIII

RATIO OF THE DIFFERENCE BETWEEN THE MEAN BEFORE THE CHANGE POINT AND AFTER THE CHANGE TO THE AVERAGE OF THE TWO MEANS (IN PERCENTAGE)

| Data | Cosine Similarity | Pearson Similarity |
|---|---|---|
| Injection of RPM reading messages | 8.65 | 10.96 |
| Injection of speed reading messages | 2.07 | 2.31 |
| No injection of fabricated messages | 0.81 | 18.12 |

We also observe that the method wrongly detects a change for the case of no injection of fabricated messages dataset. We observe, however, that the strength of change is 8.65% for the injection of RPM readings, 2.07% for the case of injection of speed readings dataset, and 0.81% for the case of no injection of fabricated messages dataset. Therefore, the strength of the change metric indicates that the method detects change for the case of injection of RPM readings and injection of speed readings and does not detect change for the case of no injection of fabricated messages dataset.

*2) Pearson Correlation:* Figure 15 shows the identified change points from the Pearson correlation coefficients sequences computed from the injection of RPM reading messages, injection of speed reading messages, and no injection of fabricated messages datasets. The change points are at windows 54, 468, and 174 for resp. The injection of RPM reading messages, injection of speed reading messages and no injection of fabricated messages datasets. We observe that the change was quickly detected for the case of injection of RPM readings dataset (after 20 time-windows) but with a significant delay for the case of injection of speed readings dataset.

We also observe that the method wrongly detects a change for the no injection of fabricated messages dataset. We observe, however, that the strength of change is 10.96% for the injection of RPM reading messages dataset, 2.31% for the case of injection of speed reading messages dataset, and 0.75% for the case of the no injection of fabricated message dataset. Therefore, the strength of the change metric indicates that the method detects change for the case of injection of RPM reading messages and injection of speed reading messages and does not detect change for the case of no injection of fabricated message dataset.

*3) Summary:* We observe that cosine similarity and Pearson correlation detect injection of RPM messages quickly but slow to detect injection of speed messages. In addition, we observe that change strength allows detecting message injection.

## C. LSTM-RNN Prediction Method

We discuss in the following the performance of LSTM-RNN in conjunction with the similarities of MSGs in predicting injection of messages onto the CAN bus.

Table IX provides the accuracy of the LSTM-RNN method in predicting injection of CAN messages from a constructed

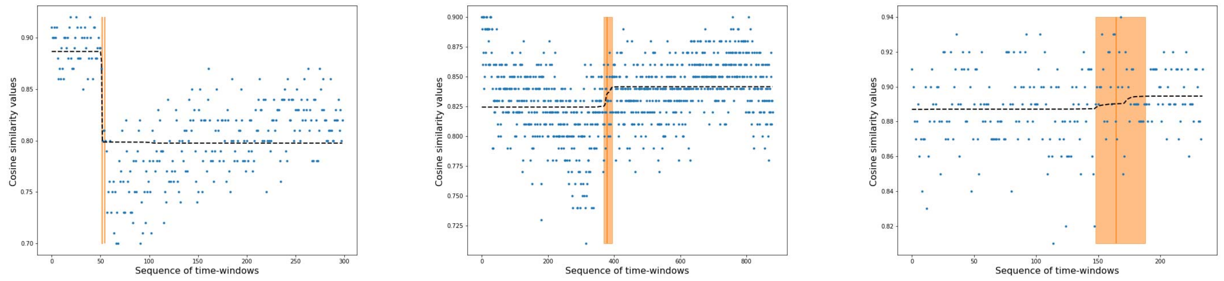


Fig. 14. Detection of point of change of successive MSGs from the Cosine Similarity when RPM reading messages injected (left), speed reading messages injected (center), and normal operation (right).

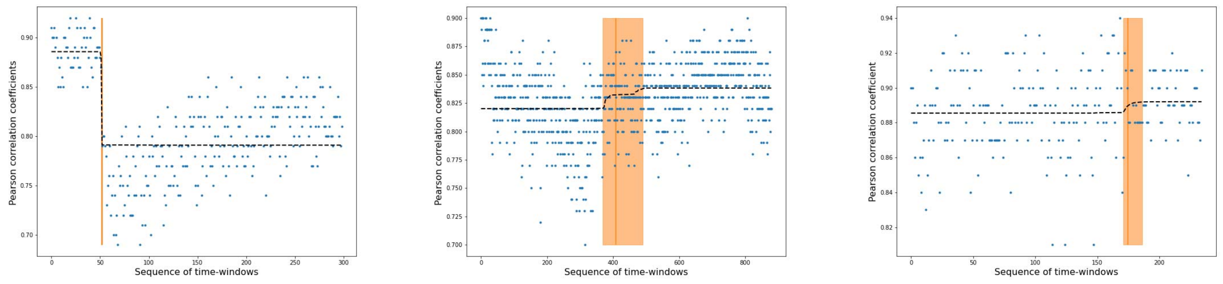


Fig. 15. Detection of point of change of successive MSGs from the Pearson correlation coefficients when RPM injected (left), speed message injection (center), and normal operation (right).

TABLE IX
ACCURACY OF LSTM-RNN IN PREDICTING INJECTION OF CAN MESSAGES

| | Constructed injection of speed reading messages dataset | | Constructed injection of RPM reading messages dataset | |
|---|---|---|---|---|
| Window size | Cosine similarity | Pearson correlation | Cosine similarity | Pearson correlation |
| 100 | 96.93 | 96.93 | 97.32 | 96.8 |
| 200 | 98.45 | 98.45 | 79.35 | 55 |
| 300 | 73.43 | 73.43 | 59.01 | 88.52 |
| 400 | 93.75 | 96.9 | 88.89 | 88.89 |
| 500 | 80.26 | 100 | 92.23 | 97.1 |
| 600 | 96.82 | 96.8 | 37.93 | 37.9 |
| 700 | 98.145 | 98.145 | 92 | 92 |
| 800 | 93.5 | 93.5 | 90.48 | 85.7 |
| 900 | 97.5 | 97.56 | 83.33 | 94.44 |
| 1000 | 97.73 | 97.3 | 100 | 100 |

injection of speed and RPM readings datasets considering different window sizes. We observe that the window size impacts the performance of LSTM-RNN in detecting injection of CAN messages. The results show that

1) the accuracy for the cosine similarity varies between 73.43% and 98.45% for the case of constructed injection of speed reading messages dataset and between 37.93% and 100% for the case of constructed injection of RPM reading messages dataset;
2) the accuracy for the Pearson correlation varies between 73.43% and 100% for the case of constructed injection of speed reading messages dataset and between 37.9% and 100% for the case of constructed injection of speed reading messages dataset.

We conclude that LSTM-RNN exhibits excellent accuracy in predicting injection of CAN messages onto the CAN bus from the constructed injection of speed and RPM readings messages dataset when using cosine similarity and Pearson correlation to compute the similarities of the consecutive MSGs.

## *D. Evaluation of the Speed of Attack Detection*

We evaluated the techniques using an Intel Dual-Core i5, with 2.3 GHz CPU and 8 GB RAM, MacBook Pro. We used Python 3.7 and Keras for processing the data.

Table X shows the CAN message injections detection speed of the algorithms used in the study. We observe that the LSTM-RNN and CPD can detect CAN bus message injections on average, in respectively, 6.03 seconds and 10.06 respectively. We note that both average values are well above the normal breaking reaction time, between 0.5 and 2 seconds. We observe, however, that the threshold-based method can detect CAN bus message injection in about 1.57 milliseconds when using cosine similarity metrics and 2.44 milliseconds when using Pearson correlation. Thus, the latter method is suitable for IDS-based real-time resiliency-control mechanisms for cyber-attacks.

## *E. Impact of the Results*

Although several machine learning-based message-injection attack-detection studies in the literature, only a few techniques were evaluated using CAN Bus datasets of parked or moving vehicles under cyber-attacks. Table XI compares the IDS solutions that we developed with the state of the art solutions that were evaluated using datasets collected from parked and moving vehicles in presence of CAN message injection

TABLE X

EVALUATION OF THE ATTACK DETECTION SPEED OF THE SOLUTIONS FOR WINDOW 200 MESSAGES

| Dataset | Size | # Cluster | AvgTime Cosine sim. (msec.) | Time Pearson corr (msec) | LSTM-RNN (sec.) | Change point Cosine sim. (sec) | Change point Pearson corr(sec) |
|---|---|---|---|---|---|---|---|
| No injection of fabricated messages | 23963 | 235 | 1.67 | 2.56 | N/A | 9.95 | 9.86 |
| Injection of speed reading messages | 88492 | 880 | 1.64 | 2.48 | N/A | 10.74 | 10.26 |
| Injection of RPM reading messages | 30308 | 299 | 1.50 | 2.41 | N/A | 11.77 | 11.52 |
| Constructed injection of speed reading messages | 112455 | 1115 | 3.17 | 4.97 | 5.64 | N/A | N/A |
| Constructed injection of RPM reading messages | 54271 | 534 | 3.31 | 4.97 | 6.42 | N/A | N/A |

TABLE XI

STATE-OF-THE-ART IDS VS. OUR IDS SOLUTION

| IDS solution | Description | ML type | Knowledge of CAN ID | Parked/Mo | Accuracy | Detection speed | Suitable for RT control |
|---|---|---|---|---|---|---|---|
| IDS Supplier 1 [11] | Detection of cyclic messages injection provided by IDS Supplier 1 | Sup. | yes | moving | 1.0 | - | no |
| IDS Supplier 2 [11] | Detection of cyclic messages injection provided by IDS Supplier 2 | Sup. | yes | moving | 0.83 | - | no |
| Previous work [10] | Detection of injection of fabricated messages using HMM | Unsup. | yes | moving | 0.80 | - | no |
| [36] | Detected unknown attacks by training Generative Adversarial Nets (GAN) on normal datset | Unsup | no | parked | 0.98 | 0.18 | yes |
| [63] | Detected intrusion attack using Convolution Neural Network (CNN) and Gated Recurrent Unit (GRU) | Unsup | no | parked | 0.94 | - | no |
| [42] | Use of graph-based solution | Unsup. | no | parked | 1.0 | 258.9 $\mu$ sec. | yes |
| [42] | Use of graph-based solution | Unsup. | no | moving | - | 0.91 sec | yes |
| LSTM-MSGs | Detection of injection of fabricated messages using MSG, Pearson correlation similarity with LSTM-RNN | Sup. | no | moving | 0.98 | 63.85 sec. | no |
| Change point-MSGs | Detection of injection of fabricated messages using MSG, Pearson correlation similarity with change point detection | Sup. | no | moving | 0.98 | 13.62 sec. | no |
| Threshold-MSGs | Detection of injection of fabricated messages using MSG, Pearson correlation similarity with threshold | Unsup. | no | moving | 0.97 | 2.45 msec. | yes |

Note:
(1) LSTM stands for Long Short-Term Memory, Recurrent Neural Network (RNN), MSG stands for Messages-Sequence Graph, Sup. is for supervised ML technique, Unsup. is for unsupervised ML., and RT for real-time.
(2) IDS solutions that detect message injection in less than two seconds (i.e., breaking reaction time) are considered suitable for Real-Time (RT) control mechanisms.

attacks. Among these solutions, two pre-commercials IDSs (Supplier 1 and Supplier 2) that were evaluated by Stachowski *et al.* [11], and the IDS that we proposed earlier [10]. One of the NN-based IDS that was evaluated by Stachowski *et al.* on three vehicles makes (each was tested for several hours) showed complete accuracy (1.0) but (1) requires knowledge of the association of CAN IDs to the corresponding ECUs for each of the vehicle-makes, (2) sensitive to change to the vehicle configuration (the authors developed a separate machine learning model for each of the three vehicles), (3) requires extensive machine-learning model training time, and (4) believed to have a high attack detection speed, which makes it not practical for real-time application, such as activate braking to counter speed increase attack.[2] Note that the solution discussed in [42] has higher false alarm rate when using our datasets, as discussed in subsection IV-A.

[2]The authors did not report attack detection speed for the method, but we expect that the method would require high attack detection speed (i.e., above two seconds) like our LSTM-RNN method.

Most of the practical CAN Bus injection detection methods use models developed from a baseline dataset, e.g., a neural network model. The baseline models are closely related to ECUs set used by the vehicle. Changing the ECUs set by, e.g., adding or removing devices requires developing a new model, which makes the solution not appropriate for the after-market vehicles. In addition, the baseline model is developed with the assumption that the car is not under attack when the baseline dataset is collected, which cannot be guaranteed for after-market vehicles as well. Therefore, our proposed method does not use a baseline model.

Our proposed IDS solution does not require the knowledge of the association of CAN IDs to the corresponding ECUs of the vehicle and shows good accuracy, up to 0.98, when a supervised ML (RNN-LSTM) is used, and with high accuracy when an unsupervised method (threshold technique) is used, up to 0.97. Additionally, the threshold-Consine Similarity method showed a quick detection of injection of fabricated messages, 1.5 to 2.6 milliseconds, as discussed in the previous section. This shows that our scheme is a practical *generic* IDS solution as it does not use proprietary information from the car manufacturers. Note that we used in this study three datasets related to one driver and one vehicle. We consider experimental evaluation for a set of vehicles and drivers in our future work.

The methods that we proposed in this paper captures the legitimate CAN ID sequence patterns. We expect that the method will also be effective in detecting Denial of Service attacks in networks and insider attacks which we consider in our future work.

## VII. Conclusion

The number of cyber-attacks on connected vehicles is increasing. The research community has proposed anomaly-based Intrusion Detection System (IDS) solutions to address this problem. The main advantage of this solutions is that it does not require modification to the CAN protocol. Existing IDSs either use the entropy of messages or require the knowledge of the IDs of the different sensors/actuators of the test vehicle. This paper investigates the use of Messages-Sequence Graph (MSG), which models the sequences of CAN messages in a time window to detect message injection. The study found that the cosine similarity and Pearson correlation of the sequence of MSGs are effective in detecting injection of RPM and speed messages with 98.45% accuracy and speed detection time of 1.5 to 2.64 milliseconds.

## Acknowledgment

The authors thank Bhagath-Kuma Veerannagari for developing some of the functions used in the research.

## References

[1] R. Bosch GmbH. (1991). *CAN Specification v2.0*. [Online]. Available: http://esd.cs.ucr.edu/webres/can20.pdf

[2] D. K. Nilsson, U. E. Larson, and P. H. Phung, "Vehicle ECU classification based on safety-security characteristics," in *Proc. IET Road Transp. Inf. Control Conf. ITS United Kingdom Members' Conf. (RTIC)*, 2008, pp. 1–7.

[3] L. Ben 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: Vehicular and Space Applications*. New York, NY, USA: Springer, 2015, pp. 217–247.

[4] V. H. Le, J. den Hartog, and N. Zannone, "Security and privacy for innovative automotive applications: A survey," *Comput. Commun.*, vol. 132, pp. 17–41, Nov. 2018.

[5] L. Ben Othmane *et al.*, "Demo: A low-cost fleet monitoring system," in *Proc. IEEE Int. Smart Cities Conf. (ISC2)*, Sep. 2018, pp. 1–2.

[6] Upstream Auto. (2020). *Upstream Security's Global Automotive Cybersecurity Report*. Accessed: Feb. 2020. [Online]. Available: https://www.upstream.auto/upstream-security-global-automotive-cybersecu%rity-report-2020/

[7] C. Miller and C. Valasek. (2013). *Adventures in Automotive Networks and Control Units*. [Online]. Available: http://illmatics.com/carhacking.pdf

[8] R. Brandom, "The scariest thing about the Chrysler hack is how hard it was to patch," Dept. Comput. Sci., Verge, Tech. Rep., Jul. 2015. [Online]. Available: https://www.theverge.com/2015/7/24/9036153/chrysler-hack-vulnerability-automobile-car-software-security

[9] J. Golson. (Sep. 2016). *Car Hackers Demonstrate Wireless Attack on Tesla Models*. [Online]. Available: https://www.theverge.com/2016/9/19/12985120/tesla-model-s-hack-vulnerab%ilitykeen-labs

[10] L. Ben Othmane, L. Dhulipala, N. Multari, and M. Govindarasu, "On the performance of detecting injection of fabricated messages into the CAN bus," *E Trans. Depend. Sec. Comput.*, early access, Apr. 23, 2020, doi: 10.1109/TDSC.2020.2990192.

[11] S. Stachowski, R. Gaynier, and D. J. LeBlanc. (Apr. 2019). *An Assessment Method for Automotive Intrusion Detection System Performance*. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/41006

[12] H. M. Song and H. K. Kim, "Discovering CAN specification using on-board diagnostics," *IEEE Des. Test. Comput.*, vol. 38, no. 3, pp. 93–103, Jun. 2021.

[13] L. Ben othmane and L. Dhulipala, "Injection of RPM and speed reading messages onto the CAN bus of a moving vehicle," *IEEE Dataport.*, to be published, doi: 10.21227/s1jy-h433.

[14] T. G. F. Vahid. (1999). *Embedded System Design: A Unified Hardware/Software Approach*. Accessed: Feb. 5, 2020. [Online]. Available: http://dsp-book.narod.ru/ESDUA.pdf

[15] B. Electronics. *OBD-II On-Board Diagnostic System*. Accessed: Jan. 2019. [Online]. Available: http://www.obdii.com/connector.html

[16] O. Avatefipour and H. Malik, "State-of-the-art survey on in-vehicle network communication (CAN-bus) security and vulnerabilities," 2018, *arXiv:1802.01725*. [Online]. Available: https://arxiv.org/abs/1802.01725

[17] S. Hartzell, C. Stubel, and T. Bonaci, "Security analysis of an automobile controller area network bus," *IEEE Potentials*, vol. 39, no. 3, pp. 19–24, May 2020.

[18] M. Wolf, A. Weimerskirch, and T. Wollinger, "State of the art: Embedding security in vehicles," *EURASIP J. Embedded Syst.*, vol. 2007, pp. 1–16, Jan. 2007.

[19] C. Valasek and C. Miller. (2015). *Adventures in Automotive Networks and Control Units*. Accessed: Feb. 1, 2020. [Online]. Available: https://ioactive.com/pdfs/IOActive_Adventures_in_Automotive_Networks_an%d_Control_Units.pdf

[20] G. Golson. (2016). *Car Hackers Demonstrate Wireless Attack on Tesla Models*. Accessed: Feb. 3, 2020. [Online]. Available: https://www.theverge.com/2016/9/19/12985120/tesla-model-s-hack-vulnerabilit%y-keen-labs

[21] A. Greenburg. *This Bluetooth Attack Can Steal a Tesla Model X in Minutes*. Accessed: Feb. 3, 2020. [Online]. Available: https://www.wired.com/story/tesla-model-x-hack-bluetooth/

[22] K. Koscher *et al.*, "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Secur. Privacy*, May 2010, pp. 447–462.

[23] L. Ben Othmane, R. Fernando, R. Ranchal, B. Bhargava, and E. Bodden, "Likelihood of threats to connected vehicles," *Int. J. Next-Gener. Comput.*, vol. 5, pp. 290–303, Nov. 2014.

[24] W. A. Farag, "CANTrack: Enhancing automotive CAN bus security using intuitive encryption algorithms," in *Proc. 7th Int. Conf. Modeling, Simulation, Appl. Optim. (ICMSAO)*, Apr. 2017, pp. 1–5.

[25] H. Ueda, R. Kurachi, H. Takada, T. Mizutani, M. Inoue, and S. Horihata, "Security authentication system for in-vehicle network," SEI, Oaks, PA, USA, Tech. Rev., Oct. 2015, pp. 5–9, vol. 81. [Online]. Available: https://global-sei.com/technology/tr/bn81/pdf/81-01.pdf

[26] A. Herrewege, D. Singelée, and I. Verbauwhede, "CANAuth—A simple, backward compatible broadcast authentication protocol for CAN bus," in *Proc. ECRYPT Workshop Lightweight Cryptogr.*, Dresden, Germany, Jan. 2011, p. 7.

[27] B. Groza, S. Murvay, A. van Herrewege, and I. Verbauwhede, "LiBrA-CAN: A lightweight broadcast authentication protocol for controller area networks," in *Cryptology and Network Security*. Berlin, Germany: Springer, 2012, pp. 185–200.

[28] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata, "CaCAN—Centralized authentication system in CAN," in *Proc. Embedded Secur. Cars (Escar) Eur.*, Nov. 2014, pp. 1–9.

[29] R. Kurachi, T. Pyun, S. Honda, H. Takada, H. Ueda, and S. Horihata, "CAN disabler: Hardware-based prevention method of unauthorized transmission in CAN and CAN-FD networks," in *Proc. Embedded Secur. Cars (Escar)*, Jun. 2016, pp. 1–7.

[30] M. Bozdal, M. Samie, S. Aslam, and I. Jennions, "Evaluation of CAN bus security challenges," *Sensors*, vol. 20, pp. 16–17, Apr. 2020.

[31] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," in *Proc. World Congr. Ind. Control Syst. Secur. (WCICSS)*, Dec. 2015, pp. 45–49.

[32] M.-J. Kang and J.-W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," *PLoS ONE*, vol. 11, no. 6, pp. 1–17, Jun. 2016.

[33] 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. 184, no. 1, pp. 1–7, 2019.

[34] S.-F. Lokman, A. T. B. Othman, and M.-H. Abu-Bakar, "Optimised structure of convolutional neural networks for controller area network classification," in *Proc. 14th Int. Conf. Natural Comput., Fuzzy Syst. Knowl. Discovery (ICNC-FSKD)*, Jul. 2018, pp. 475–481.

[35] S. Longari, D. H. N. Valcarcel, M. Zago, M. Carminati, and S. Zanero, "CANnolo: An anomaly detection system based on LSTM autoencoders for controller area network," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1913–1924, Jun. 2021.

[36] 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, Secur. Trust (PST)*, Aug. 2018, pp. 1–6.

[37] W. Wu *et al.*, "A survey of intrusion detection for in-vehicle networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 919–933, Mar. 2020.

[38] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *Proc. 25th USENIX Conf. Secur. Symp.*, Austin, TX, USA, Aug. 2016, pp. 911–927.

[39] 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)*, Oct. 2016, pp. 130–139.

[40] M. Levi, Y. Allouche, and A. Kontorovich, "Advanced analytics for connected car cybersecurity," in *Proc. IEEE 87th Veh. Technol. Conf. (VTC Spring)*, Jun. 2018, pp. 1–7.

[41] G. D'TAngelo, A. Castiglione, and F. Palmieri, "A cluster-based multidimensional approach for detecting attacks on connected vehicles," *IEEE Internet Things J.*, early access, Oct. 22, 2020, doi: 10.1109/JIOT.2020.3032935.

[42] R. Islam, R. U. D. Refat, S. M. Yerram, and H. Malik, "Graph-based intrusion detection system for controller area networks," *IEEE Trans. Intell. Transp. Syst.*, early access, Oct. 1, 2020, doi: 10.1109/TITS.2020.3025685.

[43] D. S. Cruzes and L. Ben Othmane, *Empirical Research for Software Security: Foundations and Experience*. New York, NY, USA: Taylor & Francis Group, LLC, 2017, ch. Threats to Validity in Software Security Empirical Research, pp. 275–300.

[44] C. Young, J. Zambreno, H. Olufowobi, and G. Bloom, "Survey of automotive controller area network intrusion detection systems," *IEEE Des. Test. Comput.*, vol. 36, no. 6, pp. 48–55, Dec. 2019.

[45] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: A survey," *Data Mining Knowl. Discovery*, vol. 29, no. 3, pp. 626–688, 2015.

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

[47] 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, Secur. Trust (PST)*, Aug. 2017, pp. 57–5709.

[48] M. Newman, *Networks—An Introduction*. London, U.K.: Oxford Univ. Press, 2010.

[49] H. Kwon, T. Kim, S. J. Yu, and H. K. Kim, "Self-similarity based lightweight intrusion detection method for cloud computing," in *Intelligent Information and Database Systems*, N. T. Nguyen, C.-G. Kim, and A. Janiak, Eds. Berlin, Germany: Springer, 2011, pp. 353–362.

[50] J.-H. Park, M. Kim, B.-N. Noh, and J. B. D. Joshi, "A similarity based technique for detecting malicious executable files for computer forensics," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, Sep. 2006, pp. 188–193.

[51] (2020). *Using Messages Precedence Similarity to Detect Message Injection in in-Vehicle Network*. [Online]. Available: https://lib.dr.iastate.edu/creativecomponents/651/

[52] T. Andrew, J. Bryans, and S. Shaikh, "Towards viable intrusion detection methods for the automotive controller area network," in *Proc. 2nd Comput. Sci. Cars Symp. Future Challenges Artif. Intell. Secur. Auton. Vehicles (CSCS)*, Munich, Germany, Sep. 2018, pp. 1–9.

[53] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

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

[55] H. Chen and N. Zhang, "Graph-based change-point detection," *Ann. Statist.*, vol. 43, no. 1, pp. 139–176, Feb. 2015.

[56] D. Barry and J. A. Hartigan, "A Bayesian analysis for change point problems," *J. Amer. Stat. Assoc.*, vol. 88, no. 421, pp. 309–319, Mar. 1993.

[57] H. Olufowobi *et al.*, "Anomaly detection approach using adaptive cumulative sum algorithm for controller area network," in *Proc. ACM Workshop Automot. Cybersecur. (AutoSec)*, Richardson, TX, USA, 2019, pp. 25–30.

[58] W. S. Gosset, "The probable error of a mean," *Biometrika*, vol. 6, no. 1, pp. 1–25, Mar. 1908.

[59] S. Shapiro and M. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, nos. 3–4, pp. 591–611, 1965.

[60] A. Patil, D. Huard, and C. Fonnesbeck, "PyMC: Bayesian stochastic modelling in Python," *J. Stat. Softw.*, vol. 35, no. 4, pp. 1–81, 2010.

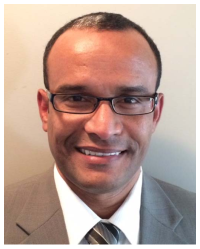
[61] M. D. Homan and A. Gelman, "The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo," *J. Mach. Learn. Res.*, vol. 15, no.1, pp. 1593–1623, Jan. 2014.

[62] F. Chollet *et al.* (2015). *Keras*. Accessed: Feb. 2020. [Online]. Available: https://github.com/fchollet/keras

[63] A. R. Javed, S. U. Rehman, M. U. Khan, M. Alazab, and T. Reddy, "CANintelliIDS: Detecting in-vehicle intrusion attacks on a controller area network using CNN and attention-based GRU," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1456–1466, Apr. 2021.

**Mubark Jedh** received the B.S. degree from Northern Illinois University, USA, in 2017, and the M.S. degree from Iowa State University, USA, in 2020, where he is currently pursuing the Ph.D. degree.

**Lotfi Ben Othmane** (Senior Member, IEEE) received the B.S. degree in information systems from the University of Sfax, Tunisia, in 1995, the M.S. degree in computer science from the University of Sherbrooke, Canada, in 2000, and the Ph.D. degree from Western Michigan University (WMU), USA, in 2010. He is currently an Assistant Teaching Professor with Iowa State University, USA. Previously, he was the Head of the Department of Secure Software Engineering, Fraunhofer SIT, Germany. He works currently on engineering secure cyber-physical systems.



**Noor Ahmed** received the B.Sc. degree from Utica College in 2002, the M.Sc. degree from Syracuse University in 2006, and the Ph.D. degree from Purdue University in 2016, all in computer science. He has been a Computer Scientist with the Air Force Research Laboratory, Rome, NY, USA, since 2003. His research interests focus on security in cloud computing and SOA, and semantic computing, reliability, and resiliency in distributed systems with special emphasis on moving target defense, blockchain, and cyber-physical systems.

**Bharat Bhargava** (Life Fellow, IEEE) received the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA. He is with the Department of Computer Science, Purdue University. He is a fellow of IETE.