Towards Resiliency of Heavy Vehicles through Compromised Sensor Data Reconstruction

Hossein Shirazi
Colorado State University
Fort Collins, Colorado, USA
Shirazi@colostate.edu

William Pickard
Colorado State University
Fort Collins, Colorado, USA
William.Pickard@colostate.edu

Indrakshi Ray
Colorado State University
Fort Collins, Colorado, USA
Indrakshi.Ray@colostate.edu

Haonan Wang
Colorado State University
Fort Collins, Colorado, USA
wanghn@stat.colostate.edu

ABSTRACT

Almost all aspects of modern automobiles are controlled by embedded computers, known as Electronic Control Units (ECUs). ECUs are connected with each other over a Controller Area Network (CAN) network. ECUs communicate with each other and control the automobile's behavior using messages. Heavy vehicles, unlike passenger cars, are constructed using ECUs manufactured by different Original Equipment Manufacturers (OEMs). For reasons of interoperability, the Society of Automotive Engineers (SAE) mandates that all ECUs should communicate using the standardized SAE-J1939 protocol that gives semantics to the signals transmitted on the CAN network. Security concerns have been historically ignored in protocols and standards. Consequently, an ECU having malicious code can spoof other ECUs, *e.g.*, a message can be injected through the OBD-II port or the telematics unit into the internal network to interfere with the behavior of the vehicle. Intrusion Detection Systems (IDS) have been proposed and utilized to detect various types of security attacks. However, such systems are only capable of detecting attacks and cannot mitigate them. A compromised ECU may generate invalid data values; even if such invalid values are detected, there is still a need to counter their effects. Almost all prior works focus on detecting attacks. We demonstrate how to make the vehicle resilient to attacks. We analyze the log files of real driving scenarios and show ECUs are significantly dependent on other ECUs to operate. We demonstrate that parameters of a compromised ECU can be reconstructed from those of other non-compromised ECUs to allow the vehicle to continue operation and make it resilient to attacks. We achieve this by modeling the behavior of an ECU using the multivariate Long Short-Term Memory (LSTM) neural network. We then reconstruct compromised ECU values using information obtained from trustworthy ECUs. Despite some levels of errors, our model can reconstruct trustworthy data values that can be substituted for values generated by compromised ECUs. The error between the reconstructed values and the correct ones is less than 6% of the operating range for the compromised ECU, which is significantly low and can be substituted. Our proposed approach makes the vehicle resilient without requiring changes to the internal architecture.

CCS CONCEPTS

• **Security and privacy** → **Intrusion detection systems**; • **Computing methodologies** → **Learning latent representations**.

KEYWORDS

Heavy Vehicle, Security, Resiliency, LSTM Network

ACM Reference Format:
Hossein Shirazi, William Pickard, Indrakshi Ray, and Haonan Wang. 2022. Towards Resiliency of Heavy Vehicles through Compromised Sensor Data Reconstruction. In *Proceedings of the Twelveth ACM Conference on Data and Application Security and Privacy (CODASPY '22), April 24–27, 2022, Baltimore, MD, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3508398.3511523

1 INTRODUCTION

In the United States, trucking is a critical part of the nation's economic infrastructure carrying 62.7% of the tonnage and 61.9% of the value of all goods shipped, coming to a total of $11.2 billion as of 2016. This amount is projected to increase to $18.7 billion by 2045. The increase in e-commerce since the year 2000 has only served to increase truck traffic in urban and residential areas [3]. As a result, cybersecurity attacks that target heavy vehicles, including trucks, represent a severe threat to both the nation's infrastructure and economy. Moreover, the increased use of embedded computers, networked controllers, and associated software has caused the attack surface of heavy vehicles to expand. To ensure interoperability between components and provide a uniform diagnostic standard, heavy vehicle embedded networks use a standardized protocol known as the Society of Automotive Engineers (SAE) standard SAE-J1939 [2]. These standards were designed without considering any security issues.

A lack of attention to security during the design of vehicle network standards has left them vulnerable to attacks [16, 29]. Specific to heavy vehicles, Mukherjee et al. [20] demonstrated multiple forms of Denial-of-Service (DoS) attacks that targeted the SAE-J1939 protocol directly. Injection attacks against the SAE-J1939 protocol were demonstrated by Burakova et al. [5] that are capable of taking over select functions of a heavy vehicle, such as disabling a driver's ability to accelerate. Previously, security concerns in-vehicle networks were downplayed because of the need for physical access. However, a more comprehensive analysis of vehicle attack surfaces has found several attack vectors that can be exploited remotely [6]. In heavy and commercial vehicles, the rise of telematics opens up a significant attack vector that exposes the SAE-J1939 embedded network. Such attacks can damage the heavy vehicle, jeopardize its behavior, and even cause loss of life. In June 2020, the UNECE World Forum for Harmonization of Vehicle Regulations (WP.29) adapted a new international automotive security regulation. While such attempts can mitigate the cybersecurity risks posed to vehicles, there is a need to mitigate attacks for vehicles on the roads.

1.1 Limitation of Existing Approaches

With an ever-increasing attack surface and a lack of built-in security features, reactive security mechanisms that can detect and mitigate the effects of attacks will be critical to securing heavy vehicle networks. Most of the works in heavy vehicle security either focus on simulating attacks or develop Intrusion Detection Systems (IDS) that can detect attacks and notify the system [7, 8, 21, 23, 25]. Existing IDSs can only detect compromised devices through anomalous values generated by them. Along with IDSs, we need mechanisms that will make vehicles resilient to attacks. In other words, when an anomalous value has been detected by a compromised or spoofed device, we need mechanisms that will replace the compromised values with correct ones. Such mechanisms will make heavy vehicles more resilient to attacks.

1.2 Our Approach

Our research is a step towards making heavy vehicles more resilient to cyber-attacks. We develop a machine learning model of the system with reasonable accuracy and robustness in the event of an attack. Specifically, we demonstrate that if an Electronic Control Unit (ECU) has been compromised, how its data values can be reconstructed from non-compromised values. Our proposed approach has the capability to mitigate the effects of that attack once it has been detected by some existing IDS.

In a heavy vehicle, the ECUs communicate with each other using the SAE-J1939 protocol. For this, various types of messages are sent, some periodic and some aperiodic. Some messages are broadcast, whereas others are destination-specific. We begin by developing a model of the data transmitted using the SAE-J1939 protocol in a heavy vehicle that can serve as a fingerprint for the system. This fingerprint can learn the behavior of the ECUs embedded in the vehicle. We use Long Short-Term Memory (LSTM) neural network, which is a form of Recurrent Neural Network (RNN) that has proven effective in modeling complex multivariate time series data [12, 14], for our work. Our approach is to train multivariate LSTM models when multiple values generated from ECUs have been used as input so that the models may learn the relationships between the different vehicle parameters.

Subsequently, the multivariate approach will leverage the additional information gained from the dynamical system to fine-tune the fingerprint. Our core hypothesis is that LSTM models that can learn relationships between large number of inputs will perform better than LSTM models that rely on fewer inputs. While the stochastic nature of neural network training means there is no way to guarantee what influence each input will have on the final model, we will use the Pearson correlation coefficient technique to examine the models after they have been trained to determine the number of inputs used by the models.

Once an attack is detected, our proposed approach can reconstruct the vehicle's parameter data that has been compromised by the attacker, taking the use of multivariate LSTM. If the multivariate model can learn enough relationships between inputs, a target output can be reconstructed using only these alternate inputs while ignoring the original, compromised data. Such data can be used to mitigate the effects of an attack if one is detected and has the added benefit that reconstructed data are immune from the effects of the compromised data.

We empirically show how our proposed approach can reconstruct compromised sensor values. During experiments, we use Controller Area Network (CAN) log files of a real-driving vehicle and train LSTM network. We assume that each sensor value obtained from the ECU can be compromised regardless of adversary access and knowledge. We reconstruct each sensor value using other trustworthy values that exist on the CAN network. Our results significantly improve existing statistical approaches; the error, at most, is only 6% of operating ranges for sensor values in all cases we have conducted our experiments.

The contributions of this paper are:

- We state the limitations of existing work which focus on IDSs that can detect attacks but are unable to mitigate them.
- We develop and evaluate multivariate LSTM reconstructive models for SAE-J1939 heavy vehicle network vehicle data. Such a model can be used as a trusted source of data in the presence of an attack.
- We demonstrate that for specific target inputs, the multivariate LSTM models use data from multiple inputs to inform and improve the accuracy of the trained models. We compared our results with two predictive models of Naïve PCHIP. These two models use values of targeted Suspect Parameter Number (SPN) for prediction. While our proposed reconstructive model does not use past values of targeted SPN, it's performance is comparable to that of predictive models.
- We illustrate how compromised data can be reconstructed using specially crafted LSTM models. These models only draw on inputs other than the target output to reconstruct the target input, thereby preventing the compromised data from affecting the prediction.

The rest of the paper is organized as follows. In Section 2, we provide the background of CAN and SAE-J1939 protocols followed by related literature on defense and attacks against heavy vehicles. In Section 3, we explain our data collection and preprocessing. In Section 4, we describe the proposed approach with three methods of Naïve, linear, and LSTM. In Section 5, we create the experimental configuration and discuss our results. In Section 6, we conclude the paper and mention some future work.

2 PREREQUISITE AND RELATED WORK

2.1 Standards

2.1.1 CAN Bus. Development of the CAN started at Robert Bosch GmbH and was released in 1986 [1]. CAN employs a two-wired differential bus, which supports speeds of up to 1 Mbit/s. The initial version, CAN 1.0, used 11 bits for the identifier fields, which has since been upgraded to 29 bits in CAN 2.0 called an extended version. The protocol establishes a priority mechanism to prevent message collision on a shared bus; a lower value of the identifier specifies a higher priority for the sender [22]. CAN 1.0 was proposed in a time when neither the Internet nor concepts of *virus* and *worm* were prevalent [8] and security was not a concern. This is evident in the fact that the CAN protocol alone does not address any security concerns.

2.1.2 SAE-J1939 . The CAN bus forms the lower layer of a vehicle's network. SAE-J1939 constitutes the upper layers of a heavy vehicle's network. The SAE-J1939 protocol allows ECUs from multiple vendors to communicate. SAE-J1939 defines five layers in the seven-layer OSI network model, including the CAN ISO 11898 specification, and uses only extended frames with a 29-bit identifier for the physical and data-link layers. *Protocol Data Unit (PDU)* is a block of information transferred over the internal network. Each *PDU* in the SAE-J1939 protocol consists of seven fields: *priority (P)*, *extended data page (EDP)*, *data page (DP)*, *PDU format (PF)*, *PDU specific (PS)* (which can be a destination address, group extension, or proprietary), *source address (SA)*, and *data field.* There is also a reserved field *Reserved (R)* with a one-bit length for further usage. These fields are all packed in one data frame and sent over the physical media to other network devices. The combination of the *PF* and the *PS* fields derives two important parameters: Parameter Group Number (PGN) and Destination Address (DA). PGN is used to group similar vehicular parameters in one single message frame. Each parameter is referred to as SPN. Each SPN defines how the application layer can interpret some portion of the data field. DA specifies the destination of the message.

2.2 Introduced Defense Mechanisms

Daily et al. [11] were instrumental in connecting security researchers with the heavy vehicle industries. The authors prototyped a remotely accessible testbed to evaluate and improve existing and new domain-specific security technologies. The system relies on embedded Linux-based node controllers that can simulate the sensor inputs to various heavy vehicles ECUs. The node controller also monitors and affects the flow of network information between the ECUs and the vehicle communications backbone on the SAE-J1939 and J1708 networks. Many concepts from the IT networks, such as securing the network, cryptography, proposed attacks, and defense mechanisms, have been adapted for the internal network of automobiles. The proposed countermeasures can be classified into *proactive* and *reactive* mechanisms [19].

2.2.1 Proactive Mechanisms. Proactive mechanisms focus on improving protocols, applications, systems, *etc.* to prevent the occurrence of any attacks. These mechanisms are not foolproof [19], but can be remarkably effective. The CAN and SAE-J1939 protocols do not support authentication and encryption, so a wide range of attacks can be launched. Towards this end, Murvay and Groza [22] proposed a mechanism to include message authentication on the protocol and evaluated the overall overhead on network communication. However, even with an authentication mechanism on the CAN bus, the maximum payload length is only 8 bytes, so the available space for an encrypted Message Authentication Code (MAC) is minimal [7]. Multiple solutions have been proposed to address this limitation. These include sending MAC in multiple data frames, using multiple CRC fields, or exploiting an out-of-bound channel for message authentication [7, 24, 26, 28]. Although proactive mechanisms can prevent attacks, they require changing the protocols, applications, and hardware. These types of solutions are unrealistic as they do not consider vehicles that are currently operational.

2.2.2 Reactive Mechanisms. Reactive mechanisms detect an attack or an impending attack and reduce its impact on the victim's vehicle at the earliest and provide a response mechanism to either block the attack or alert other systems [19] while trying to minimize the number of *false-alarms* [1].

The physical signal characteristics have been recently used to fingerprint ECUs connected to the CAN bus using voltage or time. Murvay and Groza [21] have authenticated messages on the CAN bus using physical characteristics of ECUs. The approach requires measuring the voltage, filtering the signals, and calculating mean square errors to uniquely identify ECUs. The signals from different ECUs showed minor variations in, for example, how the fast-rising edge is set up or how stable a signal is. Although these characteristics remain unchanged over a period of several months, there are certain limitations. Examples include results varying with changes in temperature [10]. Furthermore, their method was evaluated on the low-speed bus with trivial ECUs, not on the high-speed bus with critical ECUs. In addition, the authors did not consider collision situations that would impact the identification mechanism [9].

Cho and Shin [7] proposed a time-based IDS. The clock offsets and skews of asynchronous nodes depend solely on their local clocks, thus distinct from others. Their method detects anomalies based on clock skew of different ECUs connected to the bus. The approach measured intervals of periodic messages for different ECUs to measure unique clock skews for each ECU. However, this method can be defeated in specific ways. For example, if an adversary can match the clock skews of a tampered device with that of the actual one, this approach does not work [27]. Moreover, this approach does not work for non-periodic messages.

Machine learning algorithms are widely used for levels higher than the physical layer. Learning algorithms have the ability to learn patterns and detect any deviation outside of an acceptable threshold. Hence, these algorithms have been widely employed to create detection mechanisms. Kang and Kang [15] proposed an intrusion detection model that discriminates between regular and abnormal packets in an embedded vehicles network by using a Deep Neural Network (DNN). As this algorithm tested only one type of attack, it is unclear how this mechanism will detect more complex attacks. Furthermore, it authenticates targeted fields with other current field values but does not take into account previous

[1]A false-alarm occurs when an alert is not due to an actual attack.

values of the fields. An attack message that is consistent with other feature values can bypass this algorithm.

Chockalingam et al. [8] investigated the use of different machine learning algorithms in detecting anomalies in CAN packets or packet sequences and compared the algorithms. They used Long Short-Term Memory (LSTM) to consider the sequence of inputs for labeling the dataset. While they did not report the accuracy, the Area Under Curve (AUC) for Receiver Operating Characteristic (ROC) curve varies from 82% to 100% in different situations.

Narayanan et al. [23] introduced OBD Secure-Alert, which detects abnormal behavior in vehicles as a plug-n-play device on vehicles. They used Hidden Markov Model (HMM) to decide whether a vector is normal or not. HMM considers just a single previous vector, so Narayanan et al. added more previous messages to each learning vector.

Shirazi et al. [25] presented a modular machine learning-based approach to obtain a fingerprint of a vehicle's embedded network. Specifically, this approach uses data taken from the SAE-J1939 SPNs, which represents the actual physical state of the vehicle consisting of values captured through sensor readings and command/control messages, not just relying on metrics gleaned from the data link layer. By using the actual vehicle data off of the SAE-J1939 bus, the intrusion detection system is able to detect anomalies originating from almost any source. The authors collected datasets from a heavy vehicle under various real-world driving scenarios injected with labeled anomalies. The authors were able to detect 98–99% of the malicious messages injected to the bus. Using this approach, attacks that alter the data contained in the SAE-J1939 protocol can be easily detected; such situations occur when a trusted ECU in the embedded vehicle network becomes compromised. However, this approach has the following limitations. First, the system relies on labeled anomalous data for constructing the supervised training dataset. Second, the approach only detects malicious behavior and does not mitigate attacks.

2.3 Attacks Against Heavy Vehicles

The SAE-J1939 communication protocol used in heavy vehicle embedded networks and the CAN network bus it is built on are both unencrypted and lack any way to authenticate packet source. Because of this, attacks have been demonstrated by Burakova et al. [5] that require nothing more than an ability to inject packets onto the target's network bus. The attacks described in the paper used two different strategies for injecting packets into the network. The first is a spoofing attack that utilized specially crafted packets to alter the output of the heavy vehicle's controllers, such as the instrument cluster. The authors precisely controlled the output of the vehicle's speedometer, tachometer, etc. just by manipulating values within specific SPNs. This type of attack occurs because of the lack of security inherent in the underlying protocols. However, it can be observed that the SPN values engineered by the attackers will deviate from the valid SPN values purposefully and, therefore, the actual physical state of the vehicle. A basic example of this is altering the speedometer reading while the vehicle is parked. While the speedometer value may indicate the vehicle is moving, the wheel speed sensors, engine rpm, and other readings would contradict this.

The authors' second form of attack was a replay attack where they recorded protocol traffic during specific periods of operation, such as driving down the highway, and then replayed that sequence of messages at other times. Using this method, the vehicle could be tricked into accelerating while at a standstill without input from the driver. While a replay attack may not allow an attacker to alter the vehicle state as precisely as a spoofing attack, the replay attack has several advantages that make it harder to detect. If implemented carefully, a replay attack may almost perfectly replicate not just the protocol message contents but even the lower-level timing of the bus. Also, because the data being replayed is from the actual running of the vehicle, the replayed data will naturally be very accurate to the original physical system. Detecting this attack should still be possible because the vehicle state encoded in the replayed packets will conflict with the actual vehicle state that is still being transmitted by the real hardware.

2.4 Remote Attacks Against Passenger Vehicles

While heavy vehicles and passenger vehicles do not share the same upper-layer protocol, they both use the same underlying CAN bus for communication between the embedded controllers. Many of the same forms of attack that are effective against passenger vehicles can be used against heavy vehicles. Miller and Valasek [18] targeted passenger vehicles and successfully attacked a vehicle remotely over a cellular modem. The severity of the vulnerabilities demonstrated and the impact the findings had, including a vehicle recall by Chrysler, is significant.

This work is significant because the attack into the internal embedded network came from a telematics system. These systems are generally not considered an essential controller in a vehicle's design and operation. Further, the vulnerable device was a combined audio, display, telematics unit, sometimes referred to as an "infotainment" system. While the remote attack over a cellular network was severe, the vulnerable infotainment system also had vulnerable WiFi and Bluetooth transceivers. These examples serve to illustrate how, as vehicle network complexity and connectivity increase, the attack surface will also continue to increase. Thus, we cannot assume that attacks will only come from foreign devices connected to the vehicle network in the future. Due to the increased attack surface and benefits of remote exploitations, attacks will likely occur due to the use of legitimate OEM hardware devices that have been compromised remotely.

3 DATA PREPARATION AND FEATURE ENGINEERING

This section details the collection and feature engineering steps used for preparing the data used in the experiments.

3.1 Data Collection and Representation

We use the previously captured CAN bus log messages at the University of Tulsa to create the dataset. For security reasons, we cannot release information about the model or manufacturer of the vehicle. The vehicle was driven on a closed course for more than 12 minutes, and a 1.4 million CAN messages were captured. For capturing messages, a SAE-J1939 protocol logger is connected to the vehicle's diagnostic port during driving. In total, there were 96

Figure 1: Engine Speed (SPN 190) and Wheel speed (SPN 84) of the vehicle during the logged experiment.

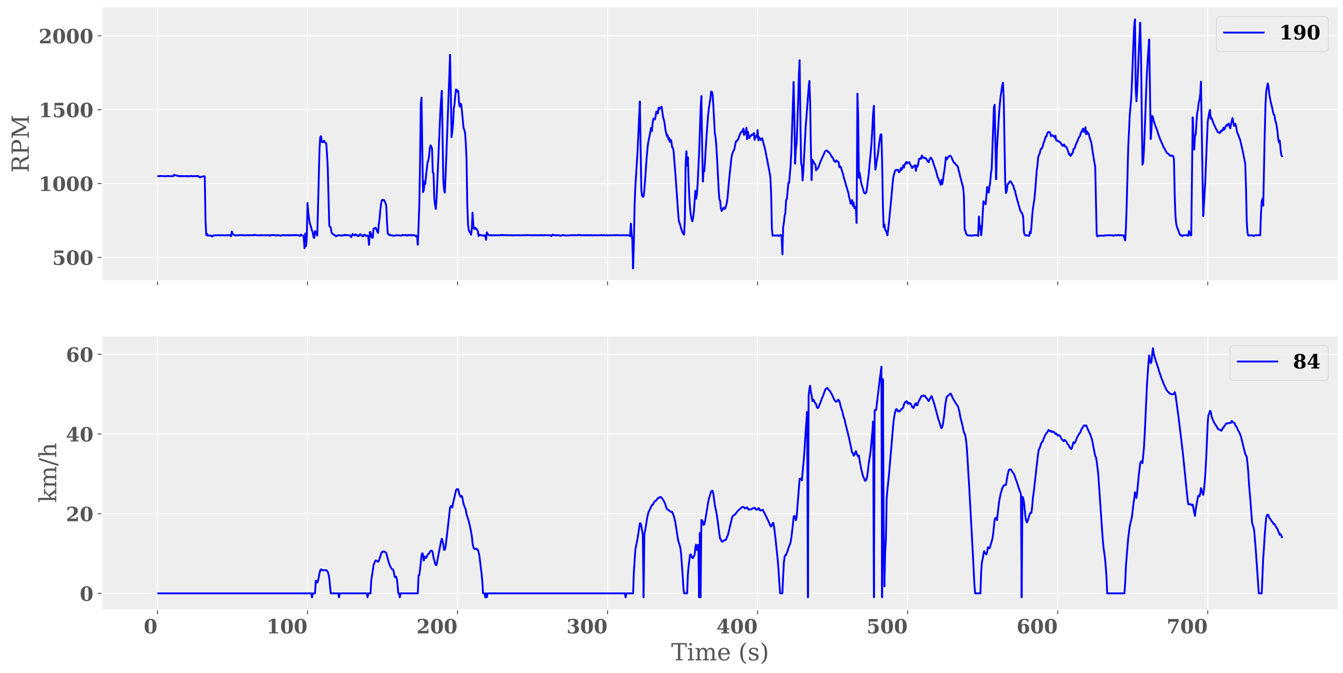


Table 1: Listing of all SPNs we conducted experiments in this study, their operating range (or data range), units and transmission rates

Parameter Group Number (PGN)	PGN Label	SPN	SPN Label	Data Range	Units	Rate
FD94	Electronic Engine Controller 7	27	EGR Valve Position	0 to 160.6	%	100 ms
FEF1	Cruise Control Speed	84	Wheel-Based Vehicle Speed	0 to 251	km/h	100 ms
FEDB	Engine Fluid Level Pres.	157	Engine Fuel Injector Metering Rail Pres.	0 to 251	MPa	500 ms
FEF2	Fuel Economy (Liquid)	183	Engine Fuel Rate	0 to 3212	L/h	100 ms
FEF2	Fuel Economy (Liquid)	184	Engine Instantaneous Fuel Economy	0 to 125	km/L	100 ms
F004	Electronic Engine Controller 1	190	Engine Speed	0 to 8031	rpm	20 ms
FEBF	Wheel Speed Information	904	Front Axle Speed	0 to 250	km/h	100 ms
FEA6	Intake Manifold Information 1	1127	Engine Turbocharger 1 Boost Pres.	0 to 8031	kPa	500 ms
FDB3	AT 1 Outlet Gas 2	3246	AT 1 Diesel Particulate Filter Outlet Temp.	-273 to 1734	℃	500 ms
FD8C	AT 1 Gas Parameters	3609	AT 1 Diesel Particulate Filter Intake Pres.	0 to 6425	kPa	500 ms
FD3E	AT 1 SCR Exhaust Gas Temp. 1	4360	AT 1 SCR Intake Temp.	-273 to 1735	℃	500 ms
FD3E	AT 1 SCR Exhaust Gas Temp. 1	4363	AT 1 SCR Outlet Temp.	-273 to 1735	℃	500 ms
FD20	AT 1 Diesel Oxidation Catalyst	4765	AT 1 Diesel Oxidation Catalyst Intake Temp.	-273 to 1735	℃	500 ms
FD20	AT 1 Diesel Oxidation Catalyst	4766	AT 1 Diesel Oxidation Catalyst Outlet Temp.	-273 to 1735	℃	500 ms
FCFD	Electronic Engine Controller 9	5313	Commanded Engine Fuel Rail Pres.	0 to 251	MPa	100 ms
FCC5	AT 1 SCR Exhaust Gas Temp. 2	5862	AT 1 SCR Intermediate Temp.	-273 to 1735	℃	500 ms

different PGNs in the log file, and we used SAE-J1939 standard to convert raw data from CAN network into SPNs values.

3.2 SPN Selection.

Since the learning algorithms need adequate data to learn the pattern, we had to select SPNs which had at least 500 messages in the log file. Applying this condition, 16 SPNs were selected from multiple subsystems representing different physical measurements such as temperature, pressure, RPM, *etc.* The SPNs selected were of the "Measured" type, *i.e.* continuously sampled. A summary of the SPNs used as input for this experiment can be found in Table 1.

Figure 1 shows physical movement of the vehicle, during this experiment, with plotting two SPN values of Engine Speed (SPN 190) and Wheel Speed (SPN 84). For example, the vehicle's speed has been decreased and increased multiple times, and there are periods during which the vehicle was stationary.

3.3 Data Representation.

In this article, we denote our multivariate input matrix as $X \in \mathbb{R}^{n \times \tau}$ where n is the number of inputs to the model, and τ is the number of input samples. Further, our time series input is represented as a vector sequence:

$$x^{(t)} = \left[x_1^{(t)}, \ldots, x_{n-1}^{(t)}, x_n^{(t)}\right]^\top, \; t = 1, 2, \ldots, \tau \tag{1}$$

3.4 Data Downsampling

Multivariate LSTM models are not designed to handle inputs of different sampling rates. All input sequences to an LSTM model must be of equal length. The SAE-J1939 protocol carries SPN data from a wide variety of sensors and ECUs that have varying sampling rates from as little as 20 ms to as much as a minute. Some SPNs may even be sampled asynchronously. Neither the CAN protocol nor SAE-J1939 have a shared master clock and neither make any timing guarantees resulting in samples that may not be aligned.

There are multiple approaches to resampling multi-rate signals, including interpolation and decimation, which use frequency domain analysis. However, these approaches are computationally intensive and introduce assumptions that may bias the training of the LSTM model. Downsampling by throwing out intermediate values introduces aliasing noise into the signal, but we believe that the LSTM model can overcome this noise to produce good predictive results.

We chose to model the effect of a one-sample deep buffer where the model input at each sequential time step is equal to only the most recently received value for that SPN. For each input sequence x_i for $i = 1, 2, \ldots, n$, timestamps were recorded for each packet reception. These timestamps comprise an index series S_i, with each element of the series $s_i^{(t)}$ representing the time of reception for each data point $x_i^{(t)}$. A new index series $s_i'^{(t)}$ is generated for all inputs with a new index $t = 1, 2, \ldots, \tau$ and a new sampling period $T =$ 500ms, the longest sampling period of the SPNs in the experiment. To simulate the buffer, for each new time step tT, the most recent past value is selected:

$$s_i'^{(t)} = \max\big(\{s \in S_i \mid s \leq tT\}\big) \tag{2}$$

The resampled input sequence x_i' is then given by:

$$x_i'^{(t)} = x_i^{(s_i'^{(t)})} \tag{3}$$

3.5 Data Scaling

To aid in the training and performance of neural networks, it is standard practice to rescale inputs to the range $[0, 1]$. Simple min-max scaling is utilized where each scaled input is given as:

$$x_i^* = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \tag{4}$$

Output predictions $\hat{x}^*$ from the model for each target SPN i must be re-scaled back to the original range of the SPN:

$$\hat{x}_i = \hat{x}_i^*\big(\max(x_i) - \min(x_i)\big) + \min(x_i) \tag{5}$$

4 OUR APPROACH

In this section, we review the theory behind each of the two approaches of *predictive* and *reconstructive* followed by threat adversary and threats against this system. Predictive models use previous values of targeted SPNs to predict values in the future. For predictive models, we propose (i) naïve, and (ii) Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) method. Reconstructive models does not use targeted SPNs and reconstruct targeted SPNs by using other SPNs. We propose LSTMs method for reconstructive approach. Predictive and reconstructive models will be utilized to perform single-time step prediction iteratively over the input data sequence.

4.1 Predictive Approach

4.1.1 Naïve Method. The input data used in this experiment comes from a range of sources with very different dynamics, from slow-moving temperature values to fast-moving, state-dependent boost pressure values, which makes finding a statistical model to serve as a point of comparison to our proposed LSTM model difficult. To serve as point of comparison for the prediction performance, we will use the naïve method:

$$\hat{x}_i^{(t+1)} = x_i^{(t)} \tag{6}$$

Equation 6 shows that the output of the naïve method for each given input is equal to the input, and the error would be the changes of input values between two consequent timestamps.

4.1.2 PCHIP. PCHIP is an statistical approach that is being used to approximate some function f, with $y = f(x)$. This algorithm uses monotonic cubic splines to find the values for new points. That preserves monotonicity in the interpolation data and does not overshoot if the data is not smooth. It also could be used to extrapolate values for unseen out-of-bounds points.

4.2 LSTM Reconstructive Approach

In this section, we will give a brief overview of the functioning of an individual LSTM cell within a larger model [12, 14]. SPN values on the CAN bus network are a stream of packets consisting of multiple sub-sequences where the data exhibits temporal dependencies within each sub-sequence. LSTM networks have been highly effective in modeling such temporal sequences as demonstrated in problems like unconstrained handwriting recognition [4], speech recognition [13], image captioning [31] and anomaly detection [30], among others. LSTMs exhibit two levels of recurrence: one within each cell where a self-loop exists between the components of the cell, and another is the recurrence among the cells that form the outer RNN structure. For each input time step t, it uses both the input $x^{(t)}$ and the output from the previous time step $h^{(t-1)}$ to calculate the output $h^{(t)}$. h for "hidden" is used in machine learning contexts to output a single layer. LSTMs improve upon standard RNNs with two key elements. First is the addition of a cell state variable c that is passed on during each time step in addition to the previous output value $h^{(t-1)}$. This internal state acts as the cell's long-term memory. The second element is three gates $f^{(t)}$, $i^{(t)}$ and $o^{(t)}$, the forget, input, and output gates respectively that control the flow of information through the cell. Each of these gates is controlled by a set of weights W, U, b, the input weights, recurrent weights and, biases respectively. The equations for the gates are defined as:

$$f^{(t)} = \sigma\Big(W_f x^{(t)} + U_f h^{(t-1)} + b_f\Big) \tag{7}$$

$$i^{(t)} = \sigma\Big(W_i x^{(t)} + U_i h^{(t-1)} + b_i\Big) \tag{8}$$

$$o^{(t)} = \sigma\Big(W_o x^{(t)} + U_o h^{(t-1)} + b_o\Big) \tag{9}$$

Where σ is the sigmoid recurrent activation function with a range $[0, 1]$, which allows gates to turn "off" and "on" respectively. There is also a set of weights and biases for the update of the cell state:

$$c^{(t)} = f^{(t)} c^{(t-1)} + i^{(t)} \tanh\Big(W_c x^{(t)} + U_c h^{(t-1)} + b_c\Big) \tag{10}$$

The output of the cell h is computed using the tanh activation function:

$$h^{(t)} = o^{(t)} \tanh\Big(c^{(t)}\Big) \tag{11}$$

4.2.1 LSTM Model Architecture.

Our model input will have a depth of 5, 10, or 20 time steps input per iteration for this experiment. We evaluate which model gives the best performance. Besides, for each compromised SPN, we will use the other 15 SPNs to predict only one value ahead of the time. Thus, our output will be predicting one-time steps in the future for one feature at a time. Each of the 16 SPNs will have a separate model trained for them.

The model architecture selected for this experiment has a stack of two LSTM layers with 64 units each. A final densely connected layer takes the output of the second hidden LSTM layer and outputs the final prediction.

4.2.2 Model Training.

A 75/25 train/test split was used when partitioning the data. Due to the sequential nature of time series data, care must be taken to ensure that out-of-order samples are not presented to the model. The last 25% of the training data was selected as the test set, and all performance evaluations were done against that subset.

For training, Mean Squared Error (MSE) was used for the loss function. The Adam optimizer was used with a learning rate of 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ [17]. Learning rate decay was not employed. The training was performed for 1000 epochs with a batch size of 100. For computational efficiency, checkpointing is used to save the best-performing model.

4.3 Threat Modeling

We have the following assumptions to model the adversary and attacks. We assume that the adversary can receive and decipher all packets on the CAN bus. Besides, the adversary can send messages that are fully compatible with SAE-J1939 on the bus, such as sending messages to a specific ECU or broadcasting messages. In addition, the adversary can send messages in the desired frequency and priority. As there is no security mechanism in the SAE-J1939 the adversary can impersonate other ECUs and send messages on their behalf.

We also assume an IDS mechanism in the network can detect and label adversarial behavior and compromised devices and SPN. This IDS is required for our proposed model to know which signal is non-trustworthy and needs to be reconstructed.

5 EXPERIMENTS AND EVALUATION

In evaluation, one step ahead prediction is performed over the test data sequence for each of the three predictive models: (i) naïve, (ii) PCHIP, and (iii) LSTM reconstruction. We use naïve and PCHIP for comparison. Two models are using a targeted SPN value for prediction, while the LSTM reconstruction model does not have access to targeted SPN and uses other SPN values for prediction. In other words, two models of naïve and PCHIP predict SPN values based on previous values of the same SPN, but LSTM models do not do so. Our hypothesis here is that our predictive LSTM model can be used to generate trustworthy values for a compromised SPN if an IDS detects it without using any of its past compromised values.

5.1 Correlated SPNs

We assume that each SPN in the system can be compromised. Thus, we create a specific LSTM model to re-construct targeted SPN values. This enables the resiliency of the system when a threat persists. For each LSTM model of targeted SPN, we need to specify what other SPNs are required to be added in the input vector of LSTM models for the reconstruction. In other words, which other SPNs can be used to reconstruct values of compromised SPN. Using all available SPNs may be one approach; however, all SPNs may not be useful or related and may not converge in the training phase. Therefore, for each SPN, we used Pearson correlation coefficient to detect related SPNs of each targeted SPN. This model measures the linear correlation between two sets of data. One limitation is that this approach only considers the linear relationship between variables. We cover this limitation with testing LSTM models with all available SPNs.

Pearson correlation coefficients range between -1 and $+1$, with $+1$ representing a positive correlation and -1 representing a negative correlation. A positive correlation means when the variable increases, the other one increases as well, and vice versa. A negative correlation means variables are changing in different directions, *e.g.* when a variable increases, the other one decreases. 0 also represents no linear correlation. Our study only needs to know which two pairs of values are moving together, regardless of directions. Thus, we use absolute values of coefficients instead. Figure 2, shows correlation between each pairs of SPN in our experiment. For example, $(84, 904)$ and $(4363, 5862)$ are highly correlated, while $(157, 4363)$ are not linearly correlated.

5.2 Selected Input SPNs for Compromised SPN

For visualizing correlation between different SPNs, we draw two directed graphs of correlated SPNs relation. Each graph, in Figure 3, shows top 5 (or 10) most related SPNs. Each node in the graph representing one SPN in our study has 5 (or 10) incoming nodes from the most correlated other SPNs. For example, Node 184 (SPN 184) has five incoming nodes from 904, 84, 1127, 27, 183.

These two graphs also show which SPNs are correlated to others and which ones are not. For example, 9 SPNs are dependent on SPN 4765 while no other SPNs are dependent on SPN 184 when we only consider the top 5 correlated SPNs in Figure 3a.

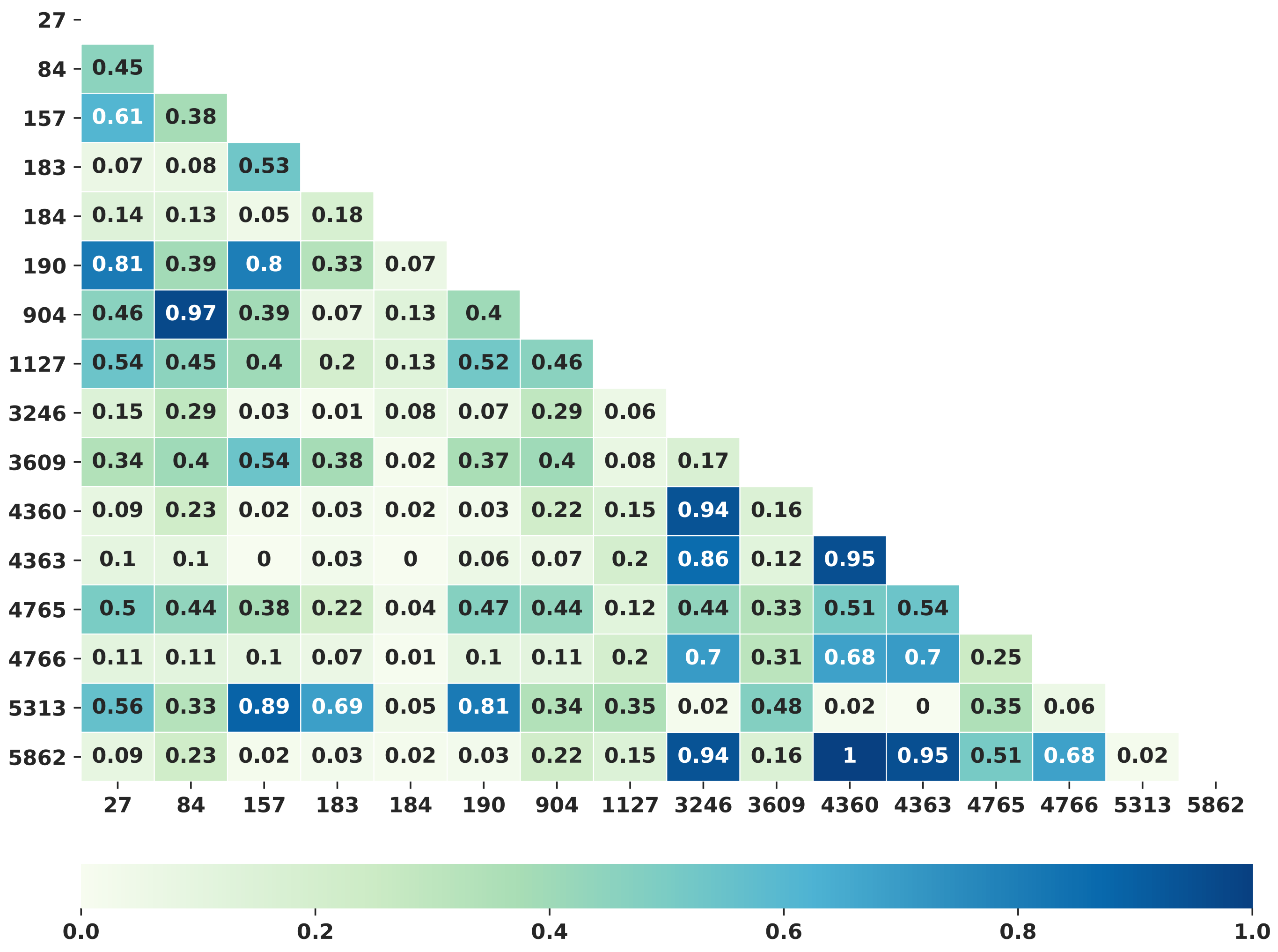


Figure 2: Coefficient heat map matrix. This heat map matrix shows the coefficient values when Pearson algorithm was used. The larger values (represented in darker color) show two SPNs are more related. The values close to zero (are shown in lighter color) mean two SPNs are not linearly related.

For each SPN, we created three LSTM models. The difference between the three networks is the number of SPNs that have been used as input for predicting the targeted SPN. For the first network, we selected 5 and 10 most related SPNs from Figure 2. We also created another model with all 15 available SPNs. We call these multivariate LSTM models as LSTM$_5$, LSTM$_{10}$, and LSTM$_{15}$ respectively.

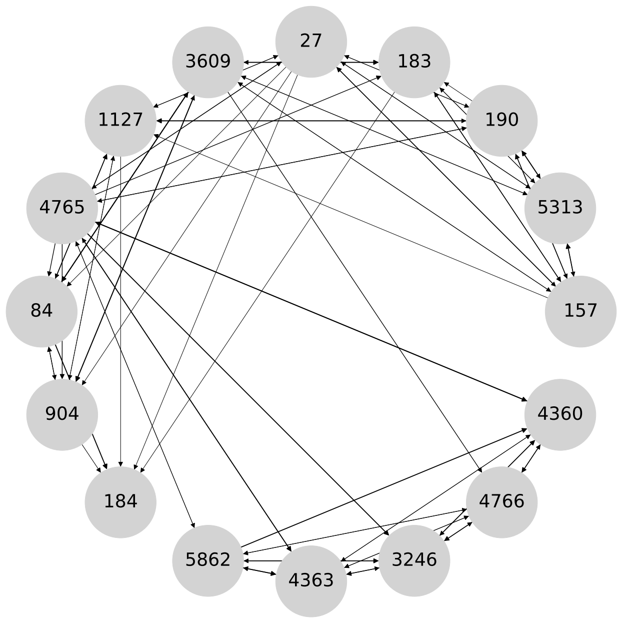
5.3 Predictive and Reconstructive Performance

We define two tasks to create predictive and reconstructive models. Two models of naïve and PCHIP predict new values of a targeted SPN based on previous values of the *same* SPN. Thus, we call this task prediction. LSTM models do not have access to previous values of the same targeted SPN. These models reconstruct targeted SPN values based on values from *other* SPNs. This task is called reconstruction. The comparison between these two approaches is unfair, as these algorithms are not conducting the same task. LSTM model is conducting a relatively more complex task.

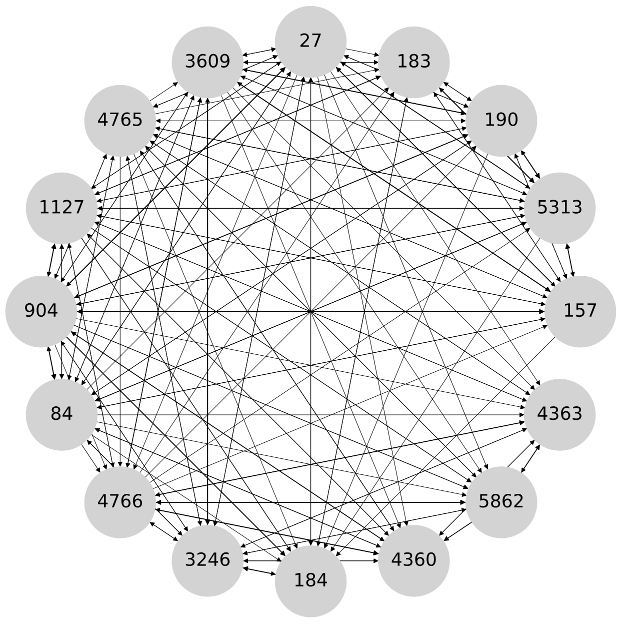
To evaluate predictive and reconstructive performance, we used mean error between the actual SPN values x_i and the predicted or reconstructive values $\hat{x}_i$. Evaluation results can be seen in Table 2.

It can be seen in Table 2 that for some tested SPNs, the multivariate LSTMs model provides similar performance to naïve and PCHIP methods. Nonetheless, LSTM models have shown reasonably good performance or even better. For example, Turbocharger Boost Pressure (SPN 1127), is one which shows improvement.

In addition LSTM models that use more SPNs to reconstruct targeted SPN are performing better except a few SPNs like (4363, 4765, and 4766). We added SPNs sorted based on correlation. It shows that when more correlated SPNs were available, the model reconstructed targeted SPN better. For the exception, the difference is not significant. It means all correlated SPNs have been added, and adding new SPN does not improve the performance.



(a) SPNs correlated to 5 most related SPNs



(b) SPNs correlated to 10 most related SPNs

Figure 3: A graph shows each SPN related to which other SPNs. Left (3a) shows each SPN is related to top 5 most related SPNs and right (3b) shows top 10 SPNs.

Table 2: Reconstructive LSTM models performance as compared to predictive models of Naïve and PCHIP. LSTM networks are reported based on the number of SPNs used for reconstruction, 5,10 and, 15 SPNs. Best performance for each approach are highlighted in bold.

	SPN		***Predictive***		***Reconstructive***		
#	**Label**	**Unit**	**Naïve**	**PCHIP**	**LSTM$_5$**	**LSTM$_{10}$**	**LSTM$_{15}$**
157	Injector Metering Rail Pres.	MPa	**9.76**	16.62	21.17	18.29	**15.81**
1127	Turbocharger Boost Pres.	kPa	12.38	**12.04**	18.53	15.46	**11.53**
3609	Particulate Filter Intake Pres.	kPa	**0.49**	0.66	1.03	0.99	**0.98**
5313	Commanded Fuel Rail Pres.	MPa	**10.39**	20.73	21.24	20.08	**19.8**
3246	Particulate Filter Outlet Temp.	℃	**0.17**	0.18	**14.27**	14.57	14.97
4360	SCR Intake Temp.	℃	**0.12**	0.18	10.14	10.36	**9.94**
4363	SCR Outlet Temp.	℃	**0.11**	0.16	43.18	**42.68**	43.19
4765	Catalyst Intake Temp.	℃	**1.18**	2.39	16.15	**12.92**	14.36
4766	Catalyst Outlet Temp.	℃	**0.21**	0.43	30.16	**23.97**	24.04
5862	SCR Intermediate Temp.	℃	**0.12**	0.18	10.89	10.53	**10.51**
190	Engine Speed	rpm	**48.74**	56.21	173.9	160.90	**149.82**
904	Front Axle Speed	km/h	0.98	**0.79**	5.33	5.22	**5.16**
183	Fuel Rate	L/h	**3.57**	6.09	8.51	8.41	**8.22**
184	Instantaneous Fuel Economy	km/L	**2.66**	6.82	7.83	6.72	**6.23**
84	Wheel-Based Vehicle Speed	km/h	0.95	**0.58**	8.38	7.16	**6.57**
27	EGR Valve Position	%	**4.44**	9.36	14.90	**11.01**	11.9

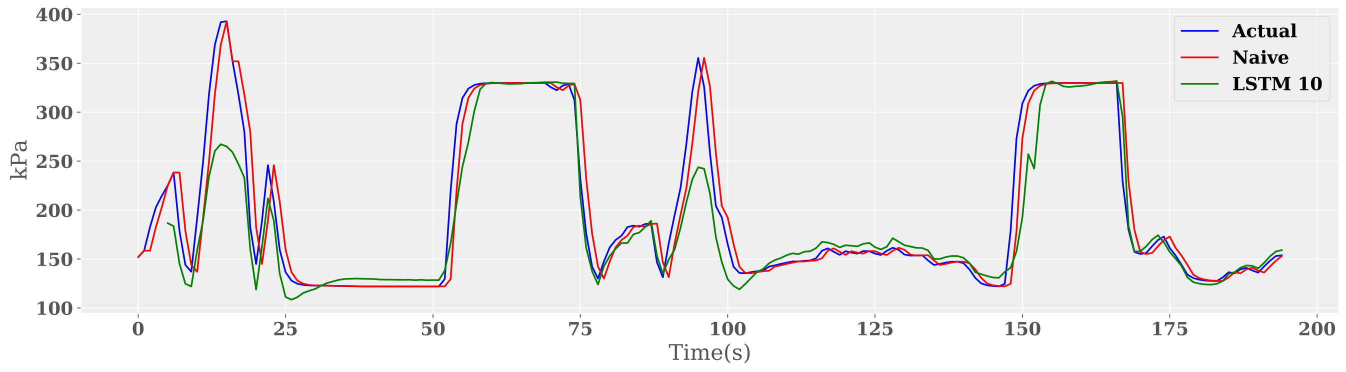


Figure 4: Comparison of reconstructed LSTM model predictions to the naïve model for Turbocharger Boost Pressure (SPN 1127). The period displayed is from the test data split during the opening of a blow-off valve to illustrate the transient response of the model. An important feature of the LSTM model is that it does not lag the actual data, and in some SPNs, the model is leading the actual data.

Not all SPNs tested proved to be a good fit for the multivariate LSTM model, with some performing worse than the baseline predictive models. In particular, SPNs relating to temperature performed the worst, as can be seen in Table 2 where SPNs relating to temperature have been grouped in the middle section. One reason for that is that these temperature values were relatively slow-moving, and some were monotonically increasing over the test interval, two features that are not ideal for LSTM models. Additionally, as we will see below, the temperature models did not make heavy use of other input variables to help improve their predictions. In the future, it will likely be necessary to perform additional feature engineering on temperature data to improve the performance of the LSTM models.

Further insight can be gained by examining Fig. 4 where we can see that unlike the naïve method, which lags the actual values by one-time step, both the LSTM model and the LSTM reconstruction closely follow changes in the actual value and, in some cases lead it. This ability to anticipate sudden state changes in the dynamical system indicates that the model uses information from other inputs to help its future predictions. If the model were only relying on current and past values of boost pressure, an aperiodic signal subject to sudden transients like the opening of the blow-off valve, it would not have the necessary information to anticipate these changes see a lag in the predicted values.

5.4 Normalized Error

One limitation in the previous comparison is that we cannot compare the performance of models among different SPNs because the operating range for each SPN is different thus, the error unit also will be different. For example, Engine Speed (SPN 190) ranges from 0 to 8032 but SCR Intermediate Temperature (SPN 5862) ranges from −273.0 to 1735. SAE-J1939 defines this operating ranges for each SPN. To address that limitation, we defined a unit-less error metric that does not consider the range or unit of SPNs as follows:

$$\Theta = \left| \frac{\sigma - (dr_{max} - dr_{min})}{dr_{max} - dr_{min}} \right| \tag{12}$$

dr_{min} is the minimum value in the SPN data range, dr_{max} is the maximum value in the SPN data range, σ is the mean of the error, and Θ is a normalized error with respect to the range of data.

If this new error metric is low, the predicted values range similar to the actual values, and the model can precisely predict the targeted SPN. However, if the error values are high, it means the model failed to predict the compromised SPN precisely and cannot be used when a threat persists.

The intuition behind this metric is two-fold. First, it makes it possible to consider the error of different SPNs regardless of the data range. Second, it better understands the volume of error among different SPNs as it is normalized.

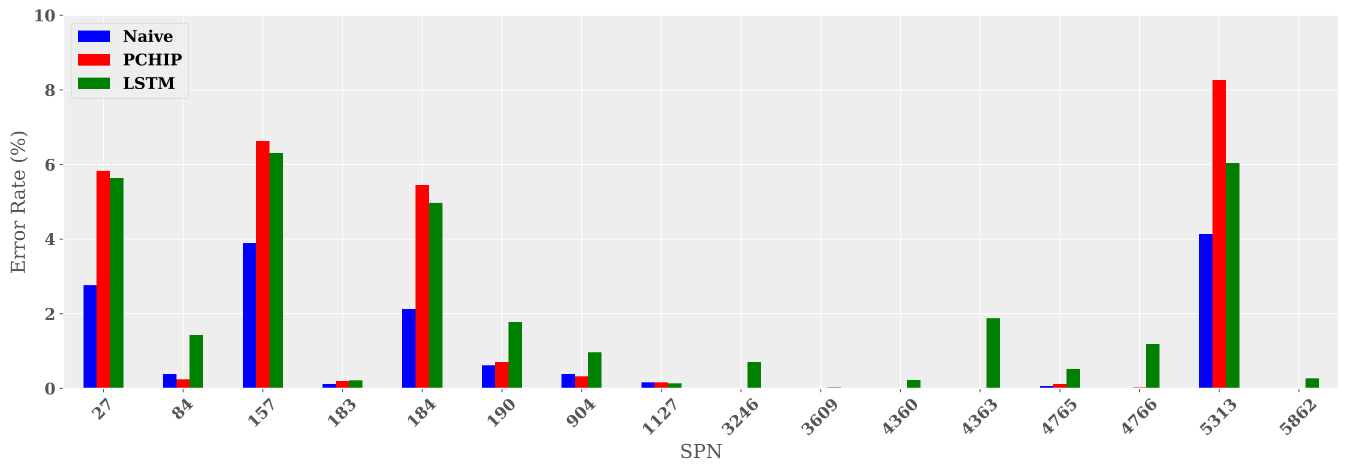
Figure 5 shows the performance of three models on all 16 SPNs. For some of SPNs like 3609, 1127, and 4360 the error level is significantly low (less than 1%), but, for some other SPNs like 27 and 157, the error is more (larger than 5%). However, in all cases, the error rate is not significant. Another observation here is the comparison between different prediction models. LSTM models perform reasonably well in all cases but have lower performance than naïve or PCHIP models. This performance can still be accepted in case of compromised SPN to make the system resilient and be operational.

6 CONCLUSION

This work presented a multivariate LSTM approach to creating a fingerprint of the data carried on an embedded heavy vehicle network utilizing the SAE-J1939 protocol. We addressed the lack of resiliency in existing IDS for heavy vehicles. While current IDS can only detect a security incident, our proposed approach utilized trusted SPNs to reconstruct compromised values. We also have shown that our approach can learn the complex dynamical relationships between inputs and that the resulting models can then leverage this information to outperform the naïve statistical approach. The performance of the models shown here proves the viability of such a mitigation strategy.

The experiment presented here only uses a small subset of the SPNs carried within the SAE-J1939 protocol, but this method can be extended and scaled to include many more. With more SPNs

Figure 5: Error rate of predictive models (naïve or PCHIP) and constructive model (LSTM).



available for training, the greater the possibility for additional learning leading to improved prediction performance. Future work will include increasing the number of SPNs available for training new models. One limitation of our proposed approach is that it only works when only one SPN is compromised. If more than one SPN is compromised, this approach may not work very well. In the future, we will work to address this limitation. We believe that once these issues have been addressed, we can truly integrate our approach with an anomaly detection mechanism and provide robust intrusion detection and mitigation system.

ACKNOWLEDGEMENT

This work was supported in part by funding from NSF under Award Numbers CNS 1715458, ATD 2123761, CNS 1822118, NIST, ARL, Statnett, AMI, and Cyber Risk Research. We would like to thank Mr. Stephen Raio and Mr. Sidney Smith from ARL for encouraging us in pursuing this line of work. The content is solely the responsibility of the authors and does not necessarily represent the official views of the agencies that funded this work.

REFERENCES

[1] 2015. Road vehicles — Controller area network (CAN). https://www.iso.org/standard/63648.html
[2] 2018. Serial Control and Communications Heavy Duty Vehicle Network. https://doi.org/10.4271/J1939_201808
[3] 2018. Transportation Statistics Annual Report 2018. *USDOT Bureau of Transportation Statistics* (2018). https://doi.org/10.21949/1502596
[4] Thomas M Breuel, Adnan Ul-Hasan, Mayce Ali Al-Azawi, and Faisal Shafait. 2013. High-Performance OCR for Printed English and Fraktur Using LSTM Networks. In *12th International Conference on Document Analysis and Recognition.* 683–687.
[5] Yelizaveta Burakova, Bill Hass, Leif Millar, and André Weimerskirch. 2016. Truck Hacking: An Experimental Analysis of the SAE J1939 Standard. In *10th USENIX Workshop on Offensive Technologies (WOOT 16).*
[6] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *20th USENIX Security Symposium (USENIX Security 11).*
[7] Kyong-Tak Cho and Kang G. Shin. 2016. Fingerprinting Electronic Control Units for Vehicle Intrusion Detection. In *25th USENIX Security Symposium, USENIX Security 16.* 911–927.
[8] Valliappa Chockalingam, Ian Larson, Daniel Lin, and Spencer Nofzinger. 2016. Detecting Attacks on the CAN Protocol With Machine Learning. *Annu EECS* 558, 7.
[9] Wonsuk Choi, Hyo Jin Jo, Samuel Woo, Ji Young Chun, Jooyoung Park, and Dong Hoon Lee. 2018. Identifying ECUs using inimitable characteristics of signals in controller area networks. In *Transactions on Vehicular Technology.*
[10] Wonsuk Choi, Kyungho Joo, Hyo Jin Jo, Moon Chan Park, and Dong Hoon Lee. 2018. Voltageids: Low-level communication characteristics for automotive intrusion detection system. In *Transactions on Information Forensics and Security.*
[11] Jeremy Daily, Rose Gamble, Stephen Moffitt, Connor Raines, Paul Harris, Jannah Miran, Indrakshi Ray, Subhojeet Mukherjee, Hossein Shirazi, and James Johnson. 2016. Towards a Cyber Assurance Testbed for Heavy Vehicle Electronic Controls. *SAE International Journal of Commercial Vehicles* 9, 2 (2016), 339–349.
[12] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to Forget: Continual Prediction with LSTM. *Journal of Neural Computation* 12, 10 (2000), 2451–2471.
[13] A. Graves, N. Jaitly, and A. Mohamed. 2013. Hybrid speech recognition with Deep Bidirectional LSTM. In *IEEE Workshop on Automatic Speech Recognition and Understanding.* 273–278.
[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Journal of Neural Computation* 9, 8 (1997), 1735–1780.
[15] Min-Ju Kang and Je-Won Kang. 2016. A Novel Intrusion Detection Method Using Deep Neural Network for In-Vehicle Network Security. In *83rd IEEE Vehicular Technology Conference (VTC Spring).* 1–5.
[16] Kaveh Bakhsh Kelarestaghi, Kevin Heaslip, and Ryan M. Gerdes. 2018. Vehicle Security: Risk Assessment in Transportation. *arXiv preprint arXiv* abs/1804.07381 (2018).
[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[18] Charlie Miller and Chris Valasek. 2015. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA* 2015 (2015), 91. http://illmatics.com/Remote%20Car%20Hacking.pdf
[19] Jelena Mirkovic and Peter Reiher. 2004. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *ACM SIGCOMM Computer Communication Review* 34, 2 (2004), 39–53.
[20] Subhojeet Mukherjee, Hossein Shirazi, Indrakshi Ray, Jeremy Daily, and Rose Gamble. 2016. Practical DoS Attacks on Embedded Networks in Commercial Vehicles. In *Information Systems Security.* Vol. 10063. Cham, 23–42.
[21] Pal-Stefan Murvay and Bogdan Groza. 2014. Source identification using signal characteristics in controller area networks. In *Signal Processing Letters.*
[22] Pal-Stefan Murvay and Bogdan Groza. 2018. Security Shortcomings and Countermeasures for the SAE J1939 Commercial Vehicle Bus Protocol. *IEEE Transactions on Vehicular Technology* 67 (2018), 4325–4339.
[23] Sandeep Nair Narayanan, Sudip Mittal, and Anupam Joshi. 2016. OBD_SecureAlert: An Anomaly Detection System for Vehicles. In *International Conference on Smart Computing.*
[24] Dennis K Nilsson, Ulf E Larson, and Erland Jonsson. 2008. Efficient in-vehicle delayed data authentication based on compound message authentication codes. In *Vehicular Technology Conference.*
[25] Hossein Shirazi, Indrakshi Ray, and Charles Anderson. 2020. Using Machine Learning to Detect Anomalies in Embedded Networks in Heavy Vehicles. In *Foundations and Practice of Security*, Vol. 12056. Springer International Publishing, 39–55.

[26] Christopher Johnathan Szilagyi. 2012. *Low cost multicast network authentication for embedded control systems.* Ph. D. Dissertation. Carnegie Mellon University.

[27] Adrian Taylor, Sylvain Leblanc, and Nathalie Japkowicz. 2016. Anomaly detection in automobile control network data with long short-term memory networks. In *International Conference on Data Science and Advanced Analytics*.

[28] Anthony Van Herrewege, Dave Singelee, and Ingrid Verbauwhede. 2011. CANAuth-a simple, backward compatible broadcast authentication protocol for CAN bus. In *Workshop on Lightweight Cryptography*.

[29] Marko Wolf and Robert Lambert. 2017. Hacking trucks-cybersecurity risks and effective cybersecurity protection for heavy duty vehicles. *Automotive-Safety & Security 2017-Sicherheit und Zuverlässigkeit für automobile Informationstechnik* (2017).

[30] Di Wu, Zhongkai Jiang, Xiaofeng Xie, Xuetao Wei, Weiren Yu, and Renfa Li. 2020. LSTM Learning With Bayesian and Gaussian Processing for Anomaly Detection in Industrial IoT. *IEEE Transactions on Industrial Informatics* 16, 8 (2020), 5244–5253.

[31] Xinxin Zhu, Lixiang Li, Jing Liu, Ziyi Li, Haipeng Peng, and Xinxin Niu. 2018. Image captioning with triple-attention and stack parallel LSTM. *Neurocomputing* 319 (2018), 55–65.