A Unified Time Series Analytics based Intrusion Detection Framework for CAN BUS Attacks

Maisha Maliha
Western Michigan University
Kalamazoo, MI, USA
maisha.maliha@wmich.edu

Shameek Bhattacharjee
Western Michigan University
Kalamazoo, MI, USA
shameek.bhattacharjee@wmich.edu

ABSTRACT

Modern smart vehicles have a Controller Area Network (CAN) that supports intra-vehicle communication between intelligent Electronic Control Units (ECUs). The CAN is known to be vulnerable to various cyber attacks. In this paper, we propose a unified framework that can detect multiple types of cyber attacks (viz., Denial of Service, Fuzzy, Impersonation) affecting the CAN. Specifically, we construct a feature by observing the timing information of CAN packets exchanged over the CAN bus network over partitioned time windows to construct a low dimensional representation of the entire CAN network as a time series latent space. Then, we apply a two tier anomaly based intrusion detection model that keeps track of short term and long term memory of deviations in the initial time series latent space, to create a 'stateful latent space'. Then, we learn the boundaries of the benign stateful latent space that specify the attack detection criterion. To find hyper-parameters of our proposed model, we formulate a preference based multi-objective optimization problem that optimizes security objectives tailored for a network-wide time series anomaly based intrusion detector by balancing trade-offs between false alarm count, time to detection, and missed detection rate. We use real benign and attack datasets collected from a Kia Soul vehicle to validate our framework and show how our performance outperforms existing works.

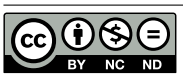
CCS CONCEPTS

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

KEYWORDS

anomaly detection; intrusion detection, vehicle CAN bus security

ACM Reference Format:
Maisha Maliha and Shameek Bhattacharjee. 2024. A Unified Time Series Analytics based Intrusion Detection Framework for CAN BUS Attacks. In *Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy (CODASPY '24), June 19–21, 2024, Porto, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3626232.3653249



CODASPY '24, June 19–21, 2024, Porto, Portugal

ACM ISBN 979-8-4007-0421-5/24/06.
https://doi.org/10.1145/3626232.3653249

1 INTRODUCTION

A modern smart vehicle contains an intra-vehicular network, called a *Controller Area Network (CAN)* that supports intra-vehicle communication between various electronic components of a car using a CAN protocol. Such electronic components are known as *Electronic Control Units (ECUs).* The intra-vehicle communication between various ECUs is supported by a communication standard called the *CAN protocol.* The CAN protocol has become a widely used communication standard in the realm of smart vehicles [20].

In CAN communication, each ECU node is uniquely *identified by a CAN ID.* All ECUs have access to a common bus channel, which means that any ECU node can send messages. Therefore, even if the attacker has access to at least one of the ECUs, it can negatively influence the CAN messages from other ECU nodes, since all ECU nodes must adhere to common BUS contention rules [9]. Finally, the broadcast nature of the protocol makes it very flexible and easy to use but also allows attackers to observe messages and misuse this property by flooding the CAN network illegitimately.

Till date, several proof of concept attacks exploiting CAN bus vulnerabilities have been shown. For example, researchers at Software Engg. Institute (SEICMU), demonstrated a Jeep Cherokee hack via exploiting CAN [16]. Similarly, researchers in Korea Univ. [18] demonstrated various kinds of attacks by exploiting CAN BUS vulnerabilities on a Kia Soul car: such as DoS(Denial of Service), Fuzzing, and Impersonation, cause vehicle malfunctions that affect the physical safety of passengers.

1.1 Related Work and Limitations

The body of existing works can be classified under the following categories: i) ECU ID specific approaches; ii) Standard Machine Learning Classifiers iii) Time windowed statistics.

ECU ID specific approaches involve designing the intrusion detection system based on training observations from each ECU ID [28], [13], [26], [21]. Hence, these approaches require separate detection per ECU. The total number of ECUs in a modern car varies between 80 to 150. Hence, a large number of models per ECU is required which negatively affects scalability. Furthermore, generalizability is another disadvantage of these methods since every car manufacturer has a different system of CAN ID assignment to its ECUs, making such methods, car specific.

Standard Machine Learning Classifiers apply known methods such as One-Class Support Vector Machine (OCSVM) [2], [1] and Generative Adversarial Network (GAN) training mechanism [8] to detect anomalous behavior in the CAN bus. The major disadvantage of this class of method is the lack of explain-ability of detection and benign inferences. Furthermore, the detection is not in real time and happens at the end of the test set. Additionally, the framework in [2] calculates false alarm rate and miss-detection rate but not time to

detection. The works [1, 8] do not evaluate the rate of false alarm and miss-detection which are key aspects of evaluating intrusion detection methods.

Time windowed statistics is an approach that uses temporal patterns of CAN messages for anomaly based detection of CAN bus attacks. In [12], an inter-arrival time was computed between ECU node IDs. Then accuracy, recall, and the actual runtime were calculated for detection. In [17], the researchers created a function that counts the number of unique CAN IDs within a specific time window. They evaluated its performance by presenting only an AUC (Area Under the Curve) score. However, false alarm or miss-detection and time to detection of attacks were not reported. Additionally, [24] explains that the above approaches were not suitable for detecting advanced attacks such as *impersonation*.

The work [18] used an "offset" measure - defined as the number of CAN messages observed between a remote and a data frame for a given ECU node. Offset based detection is very cumbersome as the range of possible offsets can vary arbitrarily. Hence, the method needs to profile patterns of each offset separately for each ECU node. Furthermore, [18] did not report false alarm or missed detection. In contrast, the work in [10], used a "Clock Offset" based Intrusion Detection System (COIDS) that observed the time intervals between periodic messages transmitted in the network. COIDS estimated these time differences, known as clock offsets, between the transmitter ECUs' clocks. These clock offset measurements were used as unique fingerprints for each transmitter ECU to construct a baseline of normal clock behavior. Just like [18], the work in [10] does not report false alarm or missed detection, and also has the disadvantage of ECU node specific approaches.

1.2 Our Contributions

We propose a unified framework that can detect different CAN BUS attack types such as DoS, Fuzzy, Impersonation. We summarize our key contributions in the following list:

(1) We construct a time series invariant feature by observing the inter-arrival time of CAN packets(Data and Remote frames) over the CAN bus network over partitioned time windows (in an ECU Node ID agnostic manner), that constructs a stable low dimensional representation of the entire CAN network benign behavior as a *time series latent space.*

(2) Then, we construct a two tier anomaly based intrusion detection model that keeps track of short term and long term memory of deviations in the initial time series latent space to create a *stateful latent space.*, that establishes the profile of benign behavior in a lower dimensional manifold.

(3) Then, we learn the boundaries of the stateful latent space under benign conditions to establish the attack detection criterion, by leveraging the benefits of quantile L1 regression learning.

(4) Most importantly, we formulate a preference based multi-objective optimization formulation to find hyper-parameters of the proposed model such that it optimizes the security objectives tailored for a network-wide time series anomaly based intrusion detector by balancing trade-offs between false alarm count, time to detection and missed detection rate.

(5) We use real benign and attack datasets collected from a Kia Soul vehicle to validate our framework and show how our performance outperforms existing works using the same dataset.

(6) We perform security evaluation assuming strategies attacker can employ given the knowledge of this proposed method. Then we quantify the extent of possible degradation in attack detection performance and associated trade-offs.

Benefits of our work: First, our proposed technique supports real-time detection, i.e., it does not need the entire test set of attacks to infer attacks. Second, our approach is unified - i.e., it is not Node ID specific or attack type specific approach. Specifically, the same architecture is used to detect different attack types such as denial-of-service (DoS), fuzzy and impersonation attacks. Only the values of parameters and hyperparameters need to be different. Third, our model parameters and hyperparameter learning are optimized by balancing multiple conflicting security objectives necessary for network wide time series anomaly based attack detection.

The rest of the paper is organized as follows: Sec. 2: introduces the details of the CAN BUS and protocol frame format. Sec. 3 describes the dataset and attack implementation details; Sec. 4 is the proposed framework; Sec. 5 gives test set set up, attack detection criterion and experimental evaluation; Sec. 6 offers security evaluation of our proposed method if the attacker has knowledge of our method; finally Sec. 7 offers a conclusion.

2 CAN SYSTEM DESCRIPTION

Here, we introduce the details of the CAN network and protocol.

2.1 CAN BUS Network

Here, we present the details of ECUs and CAN Network:

ECUs: ECUs are small embedded computers that perform one or more functions such as engine control, transmission control, ABS (Anti-lock Braking System), airbag control, and more. An ECU broadcasts CAN messages containing appropriate commands, to all other ECUs, via the CAN network [25]. The intended ECU receiver accepts while other ECUs discard the CAN messages. A modern vehicle may have anywhere between 70 ECUs [15] up to 150 ECUs in luxury cars. Each ECU is uniquely represented by at least one or more CAN IDs. The CAN IDs are assigned to ECUs such that they implicitly denote priority. A lower CAN ID hex code indicates a higher priority ECU and hence wins the CAN bus contention.

CAN Topology: The physical media that supports communication between ECUs, is the CAN bus, where the ECUs are connected via a bus topology; hence the term CAN bus network. A CAN bus network allows ECUs to communicate over a wide range of distances, at a high speed, without much wiring cost.

CAN protocol: The CAN protocol supports communication from multiple ECUs simultaneously. As multiple ECUs transmit messages, only one message can occupy the CAN bus at any given time point. The message which has the lowest CAN ID value in its identifier field has the higher priority to gain the occupancy of the CAN bus i.e. a high priority CAN packet. The lower priority ECU node will terminate its transmission while the high priority node continues to transmit. The low priority node attempts to transmit again and transmits once the bus is free. This is how the CAN protocol resolves collisions during message transmission.

2.2 CAN Frame Format

Below we describe the various fields of the CAN version 2.0A frame:
Start of Frame(SOF): represents a 'dominant 0' to signal to the rest of the network that a CAN node wants to communicate.

ID: This frame identifier defines the priority of each message in the CAN network. A lower ID value has higher priority to occupy the CAN bus.
Remote Transmission Request(RTR): RTR indicates whether a node forwards data or requests dedicated data from another node
Control: The control field contains a 'dominant 0' for 11 bit Identifier, known as Identifier Extension Bit (IDE). It also has the Data Length Code(DLC) of 4 bits which specifies the length of the transmitted Data.
Data: The data field contains the actual data values also known as payload(0-8 bytes)
Cyclic Redundancy Check(CRC): CRC checks for data integrity
Acknowledgement(ACK): ACK slots indicate that the data was received and processed correctly.
End of Frame(EOF): EOF represents the end of the CAN frame with a recessive '1'.

2.3 Response Mechanism of CAN Frame:

A typical CAN message is called a CAN frame. Two fundamental types of frames is relevant to our study: (1) *remote frame* and (2) *data frame*. These frame types play a role in the request response mechanism of the CAN protocol.
Remote Frame: When the RTR bit of a CAN frame is 'recessive 1', the frame is known as the remote frame. Remote frame is a control frame and does not contain any data. Instead, it works as a request frame that requests the desired data from a particular ECU node. When that node receives a remote frame, it responds with a corresponding data frame in response to the remote frame request.
Data Frame: When the RTR bit is a 'dominant 0', the frame is called a data frame and is in response to a remote frame. The Data frame contains the same ECU node ID as the remote frame along with the actual payload and is responsible for transmitting the message from a source to the destination ECU.

3 ATTACKS AND DATASET DESCRIPTION

Here, we first give a general dataset description used in our paper, followed the description of attack and some implementation details.

3.1 Dataset Description

We use publicly available real world OTIDS dataset [19], to evaluate our framework. There are a total of 2,369,397 messages in this dataset, which were gathered from the CAN bus network of a Kia Soul vehicle during a state without attack. Additionally, this dataset has three distinct attack types: DoS, Fuzzy and Impersonation, which is described later in the paper. There are 656,578 messages for DoS attack attack, 591,989 messages for fuzzy attacks, and 995,471 messages for impersonation attacks.

These datasets were gathered by recording CAN traffic through the OBD-II port of a Kia-Soul vehicle. Attack nodes were constructed using an Arduino with a CAN shield and a Raspberry Pi3 with PiCAN2 to perform DoS, fuzzy, and impersonation attacks. For the impersonation attack, the impersonating node was programmed to choose a specific identifier and transmit data frames periodically while responding immediately with a data frame upon receiving a remote frame. The impact of the attacks on the actual vehicle was also verified by [19]. The following describe of each of the attack types:

3.2 DoS Attack

A denial of service (DoS) attack is a flooding strategy where an adversary overwhelms the CAN network with excessive number of CAN packets marked with higher priority CAN messages. In OTIDS dataset [19], the adversary floods the network with high priority CAN packets where the identifier field has IDs - 0x000. This allows the adversary to occupy the CAN bus more often, by winning the BUS contention process. Hence, the legitimate ECU nodes back off and get delayed access to the CAN BUS and sometimes get blocked from accessing the BUS entirely. In [19], ECUs with the following CAN IDs 0x153, 0x164,0x1F1,0x220,0x2C0,0x4B0,0x4B1,and 0x5A0 get impacted due to DoS attack. The DoS attack type was first shown by Miller & Valasek [22], which affect functioning of steering control unit, causing the car to be unresponsive to sharper turns. Similarly, depending on the ECU being denied, the physical impacts of DoS attacks could vary.

3.3 Fuzzy Attack

In Fuzzy attack, the adversary injects semi-random or random data into the network's communication process to create unexpected consequences. Fuzzy attack involves actual CAN IDs that are typically broadcast on the bus or generate random CAN IDs to carry out the attack [14]. In the case of OTIDS dataset which we are using, the attacker follows the first approach to create fabricated CAN messages. Specifically, the attacker injects spurious CAN messages with IDs that appear in the benign traffic and inserts arbitrary data in the data field of the spurious CAN message. In this way, fuzzy attack injects various types of CAN frames with random data in order to compromise these CAN IDs 0x164, 0x1F1, 0x220, 0x2C0, 0x4B0, 0x4B1, and 0x5A0 [18]. The work [11], showed the practical impact of the same fuzzy attack on a KIA Soul vehicle - such as high beeping sounds, errors in the navigation system, and compromised functionality of the accelerator pedal.

3.4 Impersonation Attack

Impersonation (or masquerade) attacks refer to the malicious act of an attacker pretending to be a legitimate node on the network. The attacker uses a weakly compromised node to first suspend message transmission of a targeted CAN ID. Subsequently, the attacker uses the OBD port-II, to inject a fabricated message. In OTIDS dataset, the CAN ID 0x164 which appears in normal vehicle operations was first suspended. Then the attacker injects a message with the same CAN ID as an impersonating node to respond to remote frames intended for the legitimate ECU with CAN ID 0x164.

There are a number of impacts of impersonation attack. For example, the work [7] showed impersonation attacks, in which an attacker compromises two ECUs, to mount this attack. This is done by the attacker first monitoring and learning the CAN ID of the ECU, then injecting attack messages of the same ID and frequency. This type of attack is commonly referred to as a "masquerade attack" in other papers. Miller et al. [23] demonstrated that they controlled the Jeep Cherokee's ABS (an ECU) by a masquerade attack.

4 TECHNICAL FRAMEWORK

The technical portion of this paper is divided into six parts: *(i)* Theoretical Intuition (Sec 4.1): which describes the domain specific

Table 1: Notation

Symbol	Description
ID	Set of CAN IDs
$\sigma_{R(T)}$	Standard deviation of ratio values $R(T)$
e	Scalar factor of $\sigma_{R(T)}$
l	Window length
fl	Frame length
w_1, w_2	Quantile regression weights
t	Timestamps
T	Time window
$t_s(ID)$	Time interval
B_{RTR}	Set of RTR bits(0,1)
$\eta_{t_S}(T)$	Total number of the time intervals in a window
∇(T)	Stateless Residual
HM(T)	Harmonic means of the time intervals
AM(T)	Arithmetic means of the time intervals
R(T)	Ratio of HM(T) to AM(T)
$\mu_{R((T)}$	Mean of the ratio values $R(T)$
SM_{high}	The upper safe margin
SM_{low}	The lower safe margin
$RUC(T)$	Stateful Residuals
$\tau^*_{max}, \tau^*_{min}$	upper & lower threshold of stateful latent space
k	Individual ON-cycle
n	Total number of ON-cycle
$f(n)$	Number of ON-cyle where attack is detected
d_k	Time to detection in each ON-cycle
dT	Average time to detection

challenges and theoretical origins of our work. *(ii)* Time Series Latent Space(Sec 4.2): is the first phase of the detection method which creates a low dimensional time series representation from the raw CAN traffic; *(iii)* Stateful Latent Space(Sec 4.3): that introduces the idea of stateless residuals and stateful residuals to create a stateful latent space that is stable and sparse; *(iv)* Learning Thresholds of Stateful Latent Space(Sec 4.4): this part explains learning the thresholds of stateful latent space; *(v)* Finding Hyper Parameter with Multi- Objective Optimization (Sec: 4.5): explains the security trade-offs of choosing the optimal parameters for the proposed approach. *(vi)* Sec. 4.6 discusses how the theoretical model is implemented with the dataset to derive the optimal model that is then applied to the test set. A summary of the notations can be found in Table 1.

4.1 Theoretical Intuition

First, we get the theoretical intuition on the scientific idea and its relevance to the CAN BUS security problem.

4.1.1 Requirements: First, let us understand the requirements:

Real Time Detection: For anomaly based intrusion, learning the profile of benign behavior is required. Any observations that do not adhere to the profile of benign behavior are viewed as an intrusion. While anomaly detection techniques can be modeled in various ways; since CAN bus attacks create an immediate civilian impact, we must take a time series approach that enables real time detection in the test/deployment set.

Invariant Properties: For high fidelity time series attack detection in cyber physical systems, the NIST guidelines [27] opine that the metric of anomaly detection must have the following properties:

(1) Stationarity in Benign Situations: In the absence of attacks, anomaly detection metrics should show minimal changes across time and history. Metrics with such properties are called invariants.

(2) Fluctuations in Invariant Under Cyber Attacks: The invariant should have inherent mathematical properties that will produce fluctuations, such that values taken by the invariant are completely different when under attack versus when under benign; i.e. invariant will tend to form separate clusters under benign versus under attack. This requirement maximizes detection accuracy.

Unified Approach: Smart cars have limited computer power unlike enterprise networks. Additionally, there are several types of cyber attacks affecting the CAN, and there are several ECU Node IDs. Current detection methods are either attack specific (needs different methods to detect each attack type), or ECU Node Specific (needs to fingerprint each ECU node)- i.e. they are *'not'* unified. Having multiple instances of attack detection per attack type and per ECU node increases on-board computing and memory requirements which is not ideal. Hence, we seek to develop a unified approach.

4.1.2 Domain Specific Challenges. The invariant design is particularly challenging because of two reasons: Previous theoretic works on time series invariant design [5, 27], work on signal translated data which are exchanging data on the same physical quantity between nodes in the network. In our problem, this is not the case, because each ECU node carries a payload that is specific to its function. Hence, we need to identify a universal feature that has properties that will lead to in-variance under benign conditions, while showing deviations regardless of the attack type.

Furthermore, the number of ECUs in a car varies between tens upto hundreds. It is well known that the clustering requirement is difficult to achieve or explain in higher dimensions, popularly known as the "curse of dimensionality". Hence, we need a dimensionality reduction technique that will map the entire CAN bus network's CAN data at any time point into a lower dimensional latent space that will enable lightweight real time anomaly detection.

While methods like PCA (Principal Component Analysis) and t-SNE(t-Distributed Stochastic Neighbor Embedding) are commonly used for dimensionality reduction, these methods do not automatically evolve with time and require costly re-calculations on every time window of detection. Furthermore, both of these popular approaches make parametric assumptions on the raw data features, which did not work for the problem we have. Hence, we need to apply a different technique to guarantee this.

4.1.3 Theoretical Origins. Our framework is inspired from theoretic advances in a series of works [4, 5]. For a set of random variables that are weakly positive correlated, it was proven in [4] that the ratio between harmonic to arithmetic mean of the individual values of the random variables remain stationary in their time series. The magnitude of positive covariance determines strength of the stationarity and invariance. The work [4], also proved that any event that violates the space/time positive covariance structure will cause fluctuations in the values taken by the time series of the ratio between Harmonic Means to Arithmetic Mean of such random variables.

4.1.4 Hypothesis. Given the above known scientific facts from [4], if analogical random variables containing similar properties can be constructed, then it can be utilized to find an invariant in a lower dimensional space that is stable enough to create a time series profile of benign behavior. The first novelty of the paper is the identification of such random variables that contain these desired properties. The second novelty is to adapt the NIST's guidelines (meant for physics driven frameworks) for a data driven method like ours. Another important novelty is that the previous works in network-wide time series anomaly-based intrusion detection in CPS including [4, 27], do not take a security principled approach for

finding the hyper-parameters of a two-tier NIST model. We show a way to find the hyper-parameters as a preference-based multi-objective optimization problem, that makes sense for network wide time series anomaly-based intrusion detection.

4.2 Time Series Latent Space

Generating the time series latent space is described in two stages: (1) the extraction of a unified feature set, that serves as input to the invariant calculation; (2) the invariant calculation process over the unified feature that produces a time series latent space.

4.2.1 **Unified Feature Set Extraction**. For unified attack detection, we establish a common link that exists between different attacks and different ECU Node IDs affected by the cyber attacks.

For DoS attack, since the legitimate ECU nodes on average back-off more due to higher amounts of high priority messages, the time intervals between the requested remote frames and response data frames of the attack-affected ECUs tend to *stochastically* increase, while the benign ECUs request response intervals do not.

For Fuzzy Attack, the attack increases the processing load on the receiver ECUs as random CAN IDs are injected in the CAN network. Also, unexpected random data increases the ECU node processing time, thus the response of the data frame from receiving ECU is delayed. Again, the time interval between the requested remote frame and the responded data frame of affected ECUs increases stochastically.

For Impersonation attack, the attacker first suspends communication from the targeted legitimate ECU node. Then a fully compromised ECU node sends a response data frame to the requested remote frame originally meant for the targeted ECU node ID. Again, the anatomy of this attack creates a subtle but inevitable stochastic increase in the time interval between the response and the data frame for the targeted ECU node ID.

From the above three paragraphs, we conclude the common link between all the attacks: i.e., subtle random and stochastic increases in the time interval between remote and data frames for a subset of CAN IDs. Hence, using such a feature will enable unified attack detection. However, we cannot profile the probability distribution or covariance for detection because these are steady state approaches (i.e., they are accurate over bigger samples of data).

We divide the time domain into discrete time windows of length l, each denoted by T. Our idea is that at each time window T, we record the time intervals $t_s^{ID_i}(T)$ between a remote frame and its corresponding data frame, for each of the i-th CAN ID ID_i observed within the T-th time window. The recording of the time interval is possible by mirroring the CAN bus traffic to the OBD-Port II, and then compute the time intervals.

Now, we introduce the calculation of each $t_s^{ID_i}(T)$. Let the set of CAN IDs with a pair of remote frame and data frame observed in the T-th time window be $\boldsymbol{ID} = \{ID_1, \cdots, ID_i, \cdots, ID_{\eta_{t_s}(T)}\}$, where each ID_i corresponds to a specific CAN identifier and $\eta_{t_s}(T)$ is the total number of time interval pairs observed in the T-th time window. Additionally, we denote $B_{RTR} = \{0, 1\}$ as the set of RTR bits, representing the two values for the RTR field. Formally, each time interval pair is given by:

$$t_s^{\boldsymbol{ID_i}}(T) = t(B_{RTR} = 1, \boldsymbol{ID_i}) - t(B_{RTR} = 0, \boldsymbol{ID_i}) \quad (1)$$

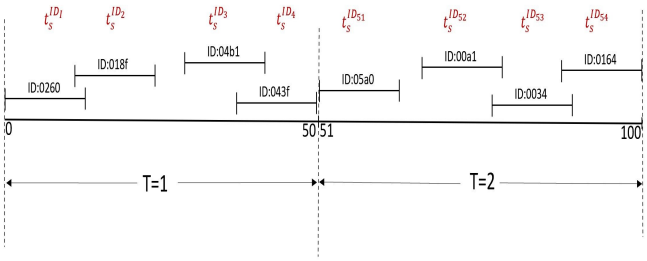


Figure 1: Time Intervals (t_s^{ID}) in Time Windows(1 & 2)

where the time interval associated with a particular ID is defined as $t_s^{\boldsymbol{ID_i}}$ which is determined by the difference between the timestamps of the RTR bit 1 ($t(B_{RTR} = 1, \boldsymbol{ID})$) and the RTR bit 0 ($t(B_{RTR} = 0, \boldsymbol{ID})$) for a particular ID.

Why this Feature Choice For Invariant: From steady state analysis of the benign part of the dataset, we found the covariance structure of time intervals observed between remote and corresponding data frame between different CAN IDs in the car follows a generally weakly positive covariance structure. As evident from Fig. 2, the figure depicts the Spearman's rank correlation between various $t_s^{ID_i}$ is generally positive. Since we proved that various $t_s^{ID_i}$ follows a generally positive covariance structure, then intelligently adapting the theory in [4] will construct a time series of the ratio between harmonic to arithmetic mean of $t_s^{ID_i}(T)$ enabling a profile of benign behavior in real time via a lower dimensional mapping.

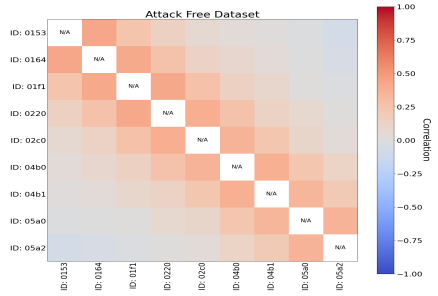


Figure 2: Spearman's Rank Correlation between ECU CAN IDs for Benign Data shows a predominantly positive covariance structure as evident from mostly red shades)

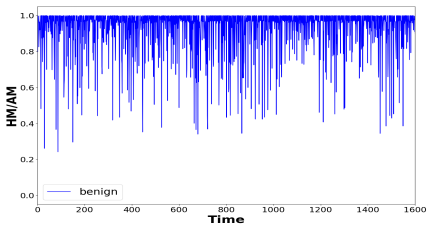


Figure 3: Ratio for Benign data:Tier 1 Latent Space

4.2.2 ***Behavioral Invariant:***. In this section, we propose the mechanism of constructing the Harmonic Mean to Arithmetic Mean ratio of the different $t_s^{ID_i}$ at each time window T as invariant which characterizes a 2-dimensional latent space-time series embedding.

Formally, let the harmonic mean, $HM(T)$ and arithmetic mean, $AM(T)$ of the time interval at time window T be defined as:

$$HM(T) = \frac{\eta_{t_s}(T)}{\sum_{i=1}^{\eta_{t_s}(T)} \frac{1}{t_s^{ID_i}(T)}} \quad AM(T) = \frac{\sum_{i=1}^{\eta_{t_s}(T)} t_s^{ID_i}(T)}{\eta_{t_s}(T)} \tag{2}$$

The $AM(T)$ and $HM(T)$ of the time intervals feature in the $T-th$ time window, where $t_s(ID_i)$ represents the i-th time interval and $\eta_{t_s}(T)$ denotes the total count of time interval pairs occurring within that T-th time window. At the end of each window T, we calculate the ratio of $HM(T)$ and $AM(T)$, denoted as $R(T)$. Formally:

$$R(T) = \frac{HM(T)}{AM(T)} \tag{3}$$

where $0 \leq R(T) \leq 1$, as $HM(T) \leq AM(T)$.

Fig. 3, visualizes the $R(T)$ for the benign portion of the dataset and it is clear that most of the ratio samples are near 1, but due to high noise and the fact that the strength of correlation is not very high, many $R(T)$ show sharp downward deviations which makes it unsuitable for high fidelity detection. This indicates that we need to take additional steps to get a more stable latent space that is consistent to as per NIST requirements [27].

4.3 Constructing a Stateful Latent Space

Traditionally, a threshold-based method is used to track changes in the difference between the time series' real value and its smoothed expected value over time. However, this approach either results in more false alarms or missed detections, but cannot balance both. As a remedy [27], mandates the notion of stateless residuals and stateful residuals, which we merge into our problem. The term residual is defined by the difference between a time series invariant and a certain smoothed (or expected) value of the invariant. Stateless residuals only keep track of the difference at a given time instant and have no memory of the past. Stateful residuals on the other hand keep a memory of stateless residuals from this time window to a certain recent past (i.e. a higher order markov model). Now we show how we adopt this concept:

4.3.1 ***Stateless Residuals***. The stateless residual for us will be the instantaneous difference between the ratio metric, denoted as $R(T)$, and *some measure of expected boundaries called safe expected margin*. In our approach, we determine the mean, $\mu_{R(T)}$ and the standard deviation of the $\sigma_{R(T)}$ from the probability distribution of the ratio values $R(T)$ in the training dataset.

We define the safe expected margin as a region of $R(T)$ samples, fall within some range of the $\mu_{R(T)} \pm e \times \sigma_{R(T)}$, where e is a scalar factor of the standard deviation within the range of $(0, 5]$. Formally, the upper expected safe margin (denoted by SM_{high}) and the lower expected safe margin (denoted by SM_{low}) given by the following:

$$SM_{\text{high}} = \mu_{R(T)} + e \cdot \sigma_{R(T)} \tag{4}$$
$$SM_{\text{low}} = \mu_{R(T)} - e \cdot \sigma_{R(T)} \tag{5}$$

where e is a hyper-parameter controlling the width of the safe expected margin. A larger e widens the safe margins, leading to a decrease in false alarms but also increase the chances of missed detections for smaller attacks.

Now we calculate the stateless residuals. Stateless residual mathematically is the "signed distance" between the observed ratio values $R(T)$ and the expected safe margins. We define three categories of stateless residuals and they are calculated by the following:

$$\begin{aligned} \nabla^+(T) &= R(T) - SM_{high}, \quad \text{if} \quad R(T) > SM_{high} \\ \nabla^-(T) &= R(T) - SM_{low}, \quad \text{if} \quad R(T) < SM_{low} \\ \nabla^{(0)}(T) &= 0, \quad \text{if} \quad SM_{low} \leq R(T) \leq SM_{high} \end{aligned} \tag{6}$$

In Eqn. 6, the $\nabla^{(0)}(T)$, corresponds to those ratio samples that stay within the expected margin are perfect inliers. The $\nabla^+(T)$ is a positive stateless residual obtained, if the sample ratio value $R(T)$ is greater than the upper expected safe margin, while a negative stateless residual $\nabla^-(T)$ is obtained when $R(T)$ falls below the lower expected safe margin. *In general, the notation $\nabla(T)$ refers to any stateless residual (i.e., $\nabla^+(T), \nabla^-(T), \nabla^{(0)}(T)$).*

Cyber attacks need to continue for some time to have a tangible impact on the system; while deviations due to noise are more random and sporadic over time. This fact can be leveraged to reduce the false alarms without sacrificing missed detection, by calculating a stateful residual from stateless residuals [27]. In our context, this means tracking the pattern of a longer term history of *consecutive* $\nabla(T)$ values apart from just the stateless residual every time window. Next, we formally define the stateful residual, which will form the final latent space that characterizes the underlying structure of benign data.

4.3.2 ***Stateful Residual***. Now we show the calculation of stateful residuals. At each time window T, the framework keeps a sum of the stateful residuals $\nabla(T)$ over a sliding frame of length fl number of time windows. Mathematically, it is defined by:

$$RUC(T) = \sum_{T=T-fl}^{T} \nabla(T) \tag{7}$$

Note that $RUC(T)$ can be *positive, negative, or zero*, depending on the realizations of $\nabla(T)$ over time. The Fig. 4 depicts all the $RUC(T)$ samples obtained (by plugging in the optimal hyperparameter values into the model, which is shown later in Sec 4.5), which specifies the stateful latent space.

Let the set of positive, negative, and zero $RUC(T)$ samples observed during the benign (training) set be denoted as RUC^+, RUC^-, $RUC^{(0)}$. We can see that a major portion of the stateful latent space is zero (i.e., the $RUC^{(0)}$ are in the majority). The RUC^- are the values less than zero and in fact RUC^+ has zero samples. This indicates a sparse and stable stateful latent space.

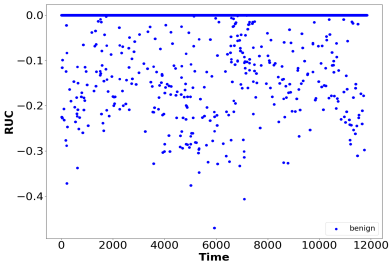


Figure 4: Stateful Latent Space Embedding that specifies benign Data from the entire CAN bus network

4.4 Learning Thresholds of Stateful Latent Space with Quantile L1 Regression

In this section, we focus on learning the boundaries/threshold that specify the profile of benign operation of the CAN BUS network from the stateful latent space. The $RUC^{(0)}$ set does not play a role in the threshold identification since these are zeros and are considered perfect in-liers. The set of RUC^{-} will be used to learn the lower threshold boundary of the stateful latent space (denoted as τ^*_{min}) while the RUC^{+} will be used to learn the upper boundary of the stateful latent space (denoted as τ_{max}).

Note, that an alternative for establishing a threshold is to use the most extreme value observed in the RUC^{-} and RUC^{+} set as a threshold - but this creates an overly optimistic model which has higher time to detection and higher missed detections. Hence, a notion of best fit threshold on the RUC^{-} and RUC^{+} sets is more appropriate for attack detection threshold design.

The attack detection threshold can be viewed as time invariant straight lines that identify the upper and lower best fit. Hence, employing a regression that learns the bias term only by taking the set RUC^{-} as inputs will give lower threshold. Similarly, a regression that learns the bias term with RUC^{+} as its training input points will help to find upper threshold. In given dataset, we did not observe any positive RUC. Hence, the problem reduces to doing a regression to just learn the lower threshold τ^*_{min} over the set RUC^{-}.

Now in standard machine learning, regression problems use an L2 norm (MSE loss function) that assumes that regression errors follow a Gaussian distribution. However, for our problem, the RUC^{-} feature space does not adhere to a Gaussian distribution due to complex relationships between ECU request-response patterns. Moreover, L2 norm assumes constant variance in the errors across observations, known as homoscedasticity. However, the variability in the regression errors for the observed latent space is not constant over time. Given the violation of such assumptions, we conclude that we cannot use simple linear regression to find the threshold.

To address the aforementioned challenges, now we introduce the algorithm that learns the threshold from input points in RUC^{-} and RUC^{+} sets, using a less well-known quantile weighted L1 regression technique. The learning approach is identical for both thresholds, hence the Algorithm below only shows how to learn the optimal lower detection threshold i.e. τ^*_{min}. In this dataset, we did not find any positive RUC values, hence τ^*_{max} is a degenerate case, which means τ^*_{max} is effectively 0. Practically, we only need to learn τ^*_{min}.

Explanation of Algorithm 1: Algorithm 1 calculates the optimal threshold, τ^*_{min} by iterating over the candidate threshold hypothesis space τ^{-} where τ^{-} is an instance from the hypothesis/candidate space. The elements in RUC^{-} represented by r, form training inputs to Algorithm 1.

We also need to give more weight to points in the outer region compared to the inner region, to reduce the false alarms, since these are part of benign behavior. The quantile L1 regression gives different weights (w_1 & w_2) to regression errors depending on their importance and calculates the weighted L1 errors. The $RUC^{-} < \tau$ correspond to inner points, while $RUC^{-} > \tau$ indicated outer points. With $w_2 > w_1$ and $w_1 + w_2 = 1$, in Algorithm 1, the outer points get more penalty that helps in false alarm reduction. The L1 norm does not incur large penalties for occasional outliers in the outer region of the latent space during benign operation, preventing the learnt threshold from getting too influenced by outliers in the outer region. This ensures that our learning approach accounts for both the need of detecting the anomaly accurately and the necessity of balancing the false alarms and missed detection error trade-offs.

Algorithm 1 Calculate τ^*_{min}

```
r ∈ RUC⁻
for τ⁻ ∈ [τ_min] do
  cost_ = 0
  penalty_ = 0
  for r ∈ RUC⁻ do
    if r < τ then
      cost_ = w1 · |r − τ⁻|
    else
      penalty_ = w2 · |r − τ⁻|
    end if
  end for
end for
τ*_min = argmin_τ (1/η_RUC⁻) |sum(cost_) + sum(penalty_)|
```

Eventually, running Algorithm 1 produces τ^*_{min}; which corresponds to that τ which minimizes the mean of quantile weighted L1 loss values over all the training data points appearing in RUC^{-}.

4.5 *Finding Optimal Hyperparameters with Multiple Objective Optimization*

Here we propose a method to learn the optimal values of the set of hyperparameters $\{e, fl, l, w_1, w_2\}$ - which are collectively the hyper-parameters of our anomaly detection model. Ideally, we need to pick the set of hyper-parameters that give the best security performance, but we do this in a novel way that balances various security objectives concurrently, unlike standard hyper-parameter tuning used in machine learning.

We posit that multiple objectives need to be satisfied for optimal security performance. Time to detection, false alarms, and missed detection rates are typical factors in assessing security performance. However, most anomaly detection frameworks treat each of these factors in silos or do not take an approach that balances all these factors in the correct sequence of importance.

As far as time series anomaly detection is concerned for a network wide detector (such as ours), the most important is the false alarm count and not false alarm rate (due to base rate fallacy [3]), followed by time to detection, and at last the missed detection rate. This preference order can be reasoned by the following explanation:

The missed detection rate is least important in our context, since at the network-wide level, we need to just raise an alarm only once to indicate the presence of an attack. Once the RUC violates the threshold, even if some of the RUC samples following that, do not violate the detection threshold, the system as a whole has already raised an alarm indicating an attack. Missed detection rate is very important for host level intrusion detection methods but is not very important in network wide detection. The missed detection rate still makes sense to evaluate since it gives an average sense of what fraction of the attack samples fall outside the detection thresholds during the attack lifetime; giving us an idea of what to expect for

other attack realizations that is not represented in a given dataset. Then, the time between the actual launch of an attack and the actual detection made by our framework - (i.e., time to detection) is more critical than the missed detection rate due to the immediate nature of the impact of CAN bus attacks on the vehicle's operation.

Now let us compare the time to detection versus false alarm count. Note, we are considering false alarm count instead of rate, because rates are misleading in time series anomaly detection. The rate entirely depends on how often the detector makes a decision and the duration of the validation and test sets. Hence, false alarm rates will not paint an accurate picture [3, 27]. Since the actual prior probability of attack is very low, the benign situation is the most commonly encountered one; hence minimizing false alarm counts should get the highest preference, followed by time to detection, and then missed detection rate.

From the above explanation, it makes sense to model the problem as a preference based multi-objective optimization problem [6] rather than an ideal multi-objective optimization problem, since the objectives in our case are not of equal priority. Now, let us formally define our preference based multi-objective optimization in the context of our problem. Let $\mathbf{X}$ be the set of decision variables (which constitutes all the unknown hyper-parameters of the proposed model), such that $\mathbf{X} = \{w_1, w_2, e, fl, l\}$, i.e., the unknown hyper-parameters of our model. The decision variables affect the set of multiple objective functions denoted by $\mathbf{F}$, where $\mathbf{F}(\mathbf{X}) = \{f_{FA}(\mathbf{X}), f_{TD}(\mathbf{X}), f_{MD}(\mathbf{X})\}$ constituting false alarm count, time to detection and missed detection rate respectively as individual objectives; all of which need to be minimized. The optimization can be defined as follows:

$$\begin{aligned}
&\min && f_{FA} && (8a)\\
&\min && f_{TD} && (8b)\\
&\min && f_{MD} && (8c)\\
&\text{subject to} && 0 < w_1 < 1, \quad \forall \quad w_1 \in \mathbb{R} && (8d)\\
& && 0 < w_2 < 1 \quad \forall \quad w_1 \in \mathbb{R} && (8e)\\
& && e > 0, \quad \forall \quad e \in \mathbb{R} && (8f)\\
& && fl > 1, \quad \forall \quad fl \in \mathbb{I} && (8g)\\
& && l > 1 \quad \forall \quad l \in \mathbb{I} && (8h)
\end{aligned}$$

Since our problem is based on a multiobjective optimization problem, we found *Pareto optimality* is the best concept to choose the optimal solution, as it finds solution for multiple decision criteria. Pareto optimization produces set of optimal solutions called *Pareto Front* based on priorities. This optimization process finds out *Pareto Front* depending on the decision variables such as regression error's weights w_1 and w_2, scalar factor e determining the width of safe margin, frame length fl and time window length l.

To solve Multi-Objective Optimization Problems (MOOP), the following approaches are popular: (1) genetic (e.g. NSGA-II) and (2) classical (e.g., ϵ-constraint method, weighted sum/metric). While classical approaches are able to produce one deterministic set of optimal solutions, genetic algorithms produce multiple sets of optimal solutions through repetitive calculations of various genetic operators. This means the genetic algorithms have higher complexity and most often require an additional step to find a usable solution from the set of solutions. Since smart vehicles have limited local computing capabilities, classical approaches have a clear advantage.

With the classical approach, two main methods are weighted-sum and the ϵ-constraint method. The weighted sum is difficult to apply in our context as the weight assignments are difficult, due to each objective being in a different unit of scale. Furthermore, convexity is a required condition for the weighted sum method, which is often difficult to guarantee in real world problems.

Since our problem has three different objectives with a clear preference order, the ϵ-constraint method is appropriate since it reformulates the problem for each objective individually (thus creating sub-problems) and then executes the decision variable solution iteratively in the order of preference. The ϵ-constraint method is also suited for non-convex problems. The order of priority is as follows: minimizing the false alarm, time to detection and lastly missed detection. Another requirement of ϵ-constraint method is incorporating user specified constraints on the other objectives, while the priority objective is being optimized in a given iteration.

The user specified constraints while minimizing false alarm for time to detection(ϵ_{TD}) is less than 0.5 s and missed detection rate(ϵ_{MD}) is less or equal to 0.55.The value for ϵ_{TD} indicates the maximum acceptable duration for detecting an intrusion within the system. Keeping the importance of quick response to any anomalies in mind, the model's requirement is defined as less than 0.5 sec to detect any intrusion in the network. Setting ϵ_{MD} at 0.49 is to ensure that the system's primary focus is capturing the majority of the threats while some anomalies go undetected. *The implementation details of our multi-objective optimization for cross validation phase can be found in Sec. 4.6.*

4.6 Details of Model Implementation

This section concerns how we applied the theoretical model discussed between Sec. 4.2-Sec. 4.5 with our dataset. Note that the original dataset contains one benign dataset and three attack datasets. While our training dataset includes only benign samples, the cross validation sets and the test sets have both benign and attack samples.

First, we divide the original benign dataset into two parts equally. The first 50% is used as the training dataset. The remaining 50% of the benign dataset is again split into two equal parts: The first portion of the latter 50% is used as the benign part of the cross validation set, while the remaining portion is set aside to form the benign portion of the testing set. Then, we extract the last 50% of the entries from each of the three attack datasets and append it to the *common* benign part of cross validation set - to create three cross validation sets one per attack type.

Then, for each of the three cross validation sets, we find the corresponding optimal hyper-parameters set e^*, fl^*, l^*, w_1^*, w_2^* (by solving the multi-objective optimization) that produces three combinations of optimal hyperparameters for each corresponding attack type. The optimal model hyperparameters obtained after applying the MOOP on the three validation sets are shown in Table 2.

We use these values from Table 2 and plug them into our model over the training dataset to obtain three distinct sets of optimal stateful latent spaces, i.e., $RUC(T)$ values. Then we apply Algorithm 1 on the three optimal latent spaces producing three different thresholds τ_{min}^* and τ_{max}^* one for each attack type.

Table 2: Pareto Front Set

Attack Type	e	Frame Length	Time Window	w_1	w_2
DoS Attack	2.7	9	50	0.9	0.1
Fuzzy Attack	3	7	90	0.9	0.1
Impersonation Attack	5	5	150	0.9	0.1

5 TESTING AND EXPERIMENTAL EVALUATION

The experimental results section is divided into three subsections: (a) Calculation of Latent Space in Test Set and Detection Criterion (b) Illustrative Visualization Results; (c) Performance Evaluation.

5.1 Test Set Calculation of Latent Space

The test set has both benign and attack datasets similar to cross validation. We extracted the last 25% of the benign dataset and the first 50% of the attack dataset for each of the attack types (Dos, Fuzzing, Impersonation) and appended them to form one continuous test dataset per attack type. Therefore, the corresponding attack starts somewhere in the middle of the test set and this time point is noted which gives the ground truth start time of the actual attack. The time to detection is defined as the difference between the ground truth attack start time stamp and the timestamp when the RUC in the test set violates any of the learned thresholds. Note that the original benign dataset is larger than the attacks even with smaller percentages, the benign samples are over-represented in the test set (Table 3) which allows for a fair false alarm assessment.

Table 3: % of Benign and Attack in test set

Attack Type	Benign(%)	Attack(%)
DoS	64.34	35.66
Fuzzy	66.68	33.32
Impersonation	54.34	45.66

For each of the three test sets, we plug in the optimal hyperparameter values into our model to calculate the $RUC^{test}(T)$ (i.e. the stateful latent space in the test set) and compare it with the corresponding thresholds learnt during training. For each window T, we calculate the stateless residual ($\nabla^{test}(T)$) followed by the stateful residual $RUC^{test}(T)$ values in the test set, using the $\mu_{R(T)}$ and $\sigma_{R(T)}$, which are the mean and the standard deviation of the ratio samples calculated over the training set.

The $RUC^{test}(T)$ calculated over the benign part should ideally lie within the thresholds (τ^*_{min} and τ^*_{max}) while the $RUC^{test}(T)$ values calculated over the attack part of the test set should fall outside the threshold range (τ^*_{min} and τ^*_{max}). Formally, this is written as:

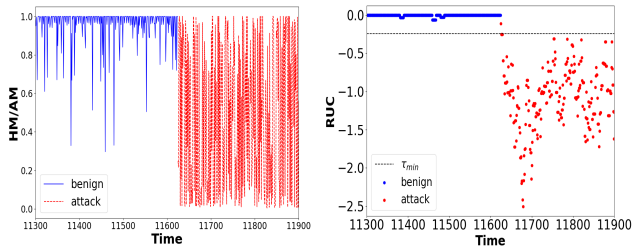
$$RUC^{test}(T) = \begin{cases} \in [\tau^*_{\min}, \tau^*_{\max}], & \text{Benign;} \\ \notin [\tau^*_{\min}, \tau^*_{\max}], & \text{Attack} \end{cases} \quad (9)$$

5.2 Illustrative Visualization Results

Here, we provide a visualization of the ratio and RUC values under benign and attack parts of the test set. We show some time domain pictures of how the RUC in the test set violates the learnt attack detection thresholds (learnt from the training phase) during the attack part of the test set, but $RUC(T)$ remains within the threshold under the benign part of the test set. This illustration is further divided into four subsections, each for (i)DoS, (ii)Fuzzy, (iii) Impersonation and (iv) Explainability-which gives a reason behind the observed visualization under benign and attack parts of the test set.

5.2.1 **DoS Attack.** For Dos attack, we show two visualizations. The first visualization (See Fig. 5(a)) shows the ratio metric during the test set, indicating a drop when DoS attack is initiated. The second visualization is the RUC metric, shown in Fig. 5(b).

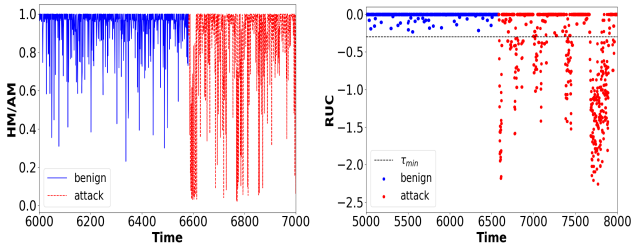
The RUC feature in Fig. 5(b) shows that the attack successfully violates the learnt threshold (the black dashed line) during the test set. The RUC feature is then used to detect the attack. This highlights the ability of the RUC to detect malicious traffic and identify DoS attacks.



(a) (b)

Figure 5: Test Set with DoS Attack (a) Ratio (b) RUC

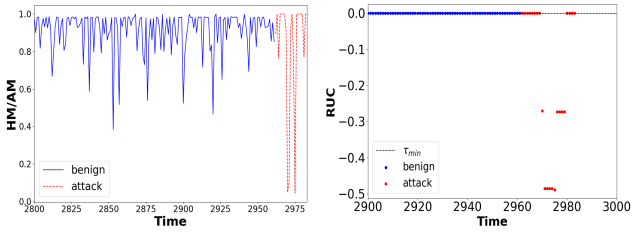
5.2.2 **Fuzzy Attack.** For the fuzzy attack, Fig. 6(a), we observe that the ratio metric drops when the attack starts, but the extent of the drop is not very obvious. This shows the need for the stateful residual which is depicted in Fig. 6(b). The Fig. 6(b) clearly shows that the attack samples violate the learnt threshold τ_{min}.



(a) (b)

Figure 6: Test Set with Fuzzy Attack (a) Ratio (b) RUC

5.2.3 **Impersonation Attack.** For impersonation attacks, the attackers attempt to imitate the typical behavior of an ECU node. Hence, detecting this attack is more difficult. The ratio metric during the impersonation attack is shown in Fig. 7(a) and the RUC characteristic is shown in Fig. 7(b). The presence of the attack is nonetheless clearly demonstrated by the RUC feature, which violates the threshold.



(a) (b)

Figure 7: Test set with Impersonation (a) Ratio (b) RUC

5.2.4 **Explain-ability of Deviations and Detection.** We explain the reason why the ratio and latent space show sharp decreases in the visualization. Fig. 8 shows that the Spearman's rank correlation calculated under attacks has a largely uncorrelated covariance structure when compared to the earlier Fig. 2 under benign data which showed a positive covariance structure. The drop in covariance explains the decrease in ratio and the RUC.

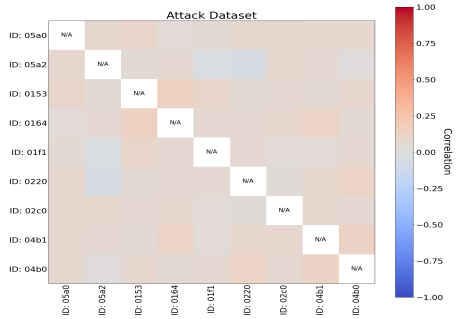


Figure 8: Spearman's Rank Correlation between ECU CAN IDs for DoS Attack Dataset shows predominantly negative or uncorrelated covariance structure compared Fig. 2 which had a generally positive covariance structure

5.3 Performance Evaluation and Results

The performance evaluation (Section 5.3), gives the numbers on the final performance achieved by our optimally learnt model during the test set. We report the final performance evaluation using three metrics- time to detection of false alarm count, and missed detection rate in the test set, to assess the performance of our approach. The subsection is further divided into four parts, one each for performance metrics under DoS, Fuzzy, Impersonation attacks and the fourth is the comparison of our performance with existing works using the same dataset.

Table 4 is a summary of the numbers obtained for performance evaluation under each of three attacks. Below we discuss observations for each attack type:

5.3.1 **DoS Attack Performance.** Table 4, first row, corresponds to the performance, obtained from the test set of DoS attacks. The RUCs of the test set were calculated using the hyperparameter values, shown in Table 2. After that, the values were compared with the thresholds to measure the performance.

One notable aspect of the evaluation results is the absence of any false alarms, indicating that the intrusion detection system successfully differentiated between normal and attack traffic.

Our framework exhibits a time to detection of only 0.036215 seconds for DoS attack Attack, demonstrating our ability to quickly and accurately detect a DoS attack. Finally, the missed detection rate of 0.01291513 suggests that most of the attack samples violate the attack detection threshold.

Table 4: Experimental Results of the Optimal Model.

Attack Type	FA Count	Time to Detection	MD Rate
DoS	0	0.036215 sec	0.01291513
Fuzzy	0	0.300889 sec	0.54775549
Impersonation	0	0.11819791 sec	0

5.3.2 **Fuzzy Attack Performance.** Table 4, second row, corresponds to the performance obtained from the test set with Fuzzy attacks. We achieved a false alarm count of 0 for fuzzy attacks Thus the most common disadvantage of anomaly based attack detection has been avoided for Fuzzy attacks showing a similar absence of false alarms. This is not surprising given how the priorities in the multiobjective optimization have been set. If one is willing to sacrifice a few false alarms, the time to detection and missed detection will see further improvement.

The intrusion detection system was able to identify the fuzzy attack within 0.3008 sec, which should enable any CAN bus attack mitigation technique to respond quickly.

Finally, the missed detection rate is 0.54775549, i.e. 54% of the attack samples did not violate the threshold and 46% of the samples violated the attack detection threshold. While the missed detection may seem a touch on the higher side, this is not a problem for network wide time series anomaly detection frameworks, because the attack needs to be inferred just once.

5.3.3 **Impersonation Attack Performance.** Table 4, third row, corresponds to the performance obtained from the test set with Impersonation attacks. The hyperparameters, referred to in Table 2), are achieved via cross-validation.

We observe that false alarm just like DoS and Fuzzy attacks is zero. To conclude, our framework does not suffer from false alarms for any attack due to the way to tune the hyperparameters of the model. Thus, we escape the problems with the base rate fallacy that plagues all anomaly based intrusion detection methods.

When compared to DoS attacks, impersonation attacks took 0.118 sec and the missed detection rate under impersonation attack was also found to be zero with the optimal model.

Table 5: Comparison with Previous Works

Attack	Metric	**Proposed**	COIDS [10]	OTIDS [18]	OCSVM [2]
DoS	FA	0	NA	NA	6.45%
	MD	1.29%	NA	NA	2.99%
	TD	0.036	0.204	0.846 sec	NA
Fuzzy	FA	0	NA	NA	NA
	MD	54.77%	NA	NA	NA
	TD	0.3001	0.219	0.892 sec	NA
Imperso -nation	FA	0	NA	NA	NA
	MD	0	NA	NA	NA
	TD	0.118	0.235	0.972 sec	NA

5.3.4 **Comparison with Previous Works.** In this section, we provide a comparison with previous works, shown in Table 5. We compare our framework with COIDS [10], OTIDS [18] and OCSVM [2]; all of which used the same dataset [19].

False Alarms: Let us compare the false alarm (FA) performance of our approach with rest of the methods. Except for OCSVM none of the other methods using this dataset to calculate FA. Our framework with a 0% false alarm rate for every attack, outperforms OCSVM with a 6.45% FA.

Time to Detection: Now let us compare the time to detection achieved by our proposed approach with the COIDS, OTIDS and OCSVM methods for each of the DoS, Fuzzy and Impersonation.

For DoS attacks, our framework detects the attack in 0.036 sec outperforming the COIDS method (with 0.204 sec) and the OTIDS method (with 0.846 sec) for COIDS, while OCSVM did not report it.

For Fuzzy attacks, our framework detects the attack in 0.30 sec. The COIDS method (0.21 sec) and OTIDS method (with 0.892 sec)

for COIDS, while the OCSVM method did not report time to detection. Our framework takes 0.10 seconds more compared to COIDS method. However, the disadvantage of COIDS method is it does not make an effort to measure the other important metrics of false alarm and missed detection.

For impersonation attacks, our approach detects the attack in 0.11 seconds, thus outperforming COIDS (with 0.235 sec) and OTIDS (with 0.972 sec.), while OCSVM did not report time to detection.

Missed Detection: Amongst existing works only OCSVM paper measures the missed detection rate for only DoS attack attack. However, our framework has an MD of 1.29% outperforming OCSVM with an MD of 2.99%.

6 SECURITY EVALUATION FOR EVASION

Security by obscurity is not a good practice. Hence, now we discuss strategies that attackers might employ to circumvent or degrade detection success when attackers have complete knowledge of our detection model and show how much our performance degrades. The attacker could exploit our method to bypass detection by implementing all attack types, using an ON-OFF strategy, i.e., alternating between equal periods of attacks being injected (called ON cycle) and no attack being injected (called OFF cycle). The intuition is that if the attacker restricts the time duration of each ON cycle to be equal to or less than the known frame length (fl) of our detection model, smaller deviations will trigger in the stateful latent space; which may not violate optimal detection threshold τ^*_{min}.

However, we prove that progressive degradation of time to detection and missed detection rate happens *only for very short ON periods*. Such short on-off periods are unlikely to have tangibly physical impacts on the vehicle.

Now we discuss how we simulated the ON-OFF strategy using the dataset. For an adversary, the attack parameter is the length of each ON/OFF period, which could be any number that is beyond the defender's control. Hence, we create multiple variations of the attack test sets with various ON-OFF periods in the following way:

Preparing Test Set with ON-OFF strategy: To simulate the on-off strategy using the OTIDS dataset, we did the following: For each attack type, we constructed a test set by arranging attack and benign samples from the original dataset sequentially, given a particular ON-OFF ratio. We tested different on-off cycle's lengths that examine the impact of this attack parameter on our detection.

For example, if the ON-OFF period is 5, we prepare an attack dataset where 5 ratio samples from the attack dataset are followed by 5 ratio samples from the benign data and another 5 from the attack test set then repeat. We are using the original dataset to create on on-off strategy in the test set. Therefore, there is no chance of incorrect evaluation.

Metrics of evaluation under Complete Knowledge: We measured the average time to detection and the missed detection rate of our method for various ON-OFF period lengths.

Average Time to Detection and Missed Detection: Let n be the total number of ON-cycles in the new test set, then $f(n)$ signifies the number of the ON-cycles where the attack got detected.

Let k be a variable representing each individual ON-cycle, such that $k \in \{1, n\}$. For each ON cycle where the attack gets detected by our method, we record the corresponding time to detection, denoted as d_k. Now, the time to detection for a candidate on-off attack can be represented as $d_1, d_2, ..., d_n$. If $f(n)$ is the number of ON-cycles(where attack is detected), the average time to detection dT can be determined as follows:

$$dT = \frac{\sum_{k=1}^{n} d_k}{f(n)} \tag{10}$$

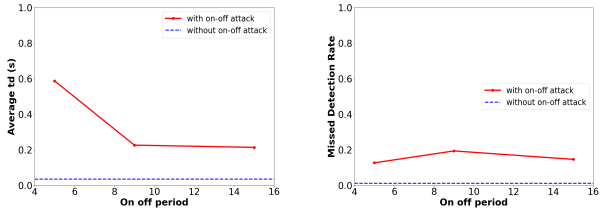
The number of attack samples that did not violate the threshold are counted as missed detection.

Below we discuss the performance of our proposed model for every attack under the on-off cycle attack strategy.

6.0.1 ***DoS Attack.*** In our detection model for DoS attack, the optimal frame length is 9. Therefore, for testing, we calculate the average time to detection and missed detection rate for three candidate ON attack periods = 5, 9, 15. The 5 and 9 correspond to attacks less than equal to the optimal frame length, while 15 correspond to higher than optimal fl ON periods.

From Fig. 9(a), we see when ON and OFF period = 5, the system takes 0.5523 sec on average to detect. The blue dashed line in Fig. 9(a), is the time to detection (0.036 sec) without an on-off attack. Hence, in the worst case, the difference is 0.514 seconds between our defense model and the attacker's stealthiest response. We can also see from Fig. 9(a), that as the ON-OFF periods become equal to or greater than the frame length the red line converges towards the baseline time to detection.

In contrast, Fig. 9(b), shows that the highest missed detection rate is 0.1949 if the on-off period length is equal to the optimal frame length. The attacker may optimally attack by aligning their on-off lengths that match the optimal frame length to create a difference of about 0.16 from the baseline. For all other candidate On-off periods, missed detection rate drops and converges to the baseline (i.e., blue dashed line).



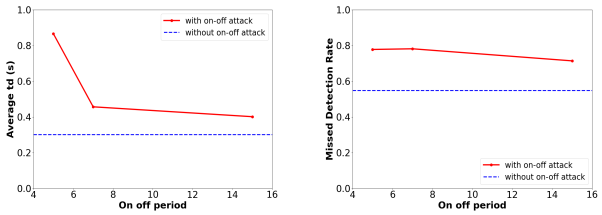
(a) (b)

Figure 9: DoS Attack (a) Degradation in Avg. Time to Detection(s) (b) Missed Detection Rate

6.0.2 ***Fuzzy Attack.*** For fuzzy attack, the optimal frame length is 7. Similar to the reasons given under DoS attack attack, we try the following on-off periods, viz., 5, 7 and 15.

Figure 10(a) compares the time to detection with and without on-off attacks. As on-off period increases, the average time to detection decreases. The worst case happens for an ON-OFF period of 5 (shorter than optimal fl of 7), where average time to detection is 0.8661 seconds, while the time to detection without on-off is 0.3 sec, given a worst-case difference of only 0.56 seconds.

Figure 10(b) shows when length of the on-off cycle increases, the missed detection rate decreases. The missed detection rate is higher at 0.7819 when the on-off window is exactly the same as the frame length. *However, the difference is not practically significant since the attack still gets quickly detected for all On-off periods.*

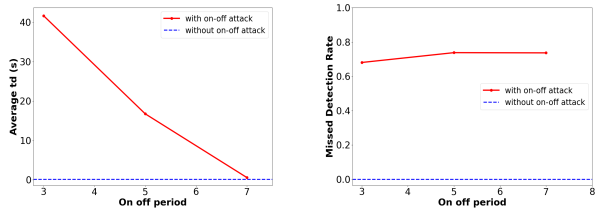


(a) (b)

Figure 10: Fuzzy Attack (a) Degradation in Avg. Time to Detection(s) (b) Missed Detection Rate

6.0.3 ***Impersonation Attack.*** In impersonation attack, the optimal frame length is 5. Fig. 11(a) shows the average time to detection with varying on-off period such as 3,5 and 7. The baseline time to detection without an on-off attack is 0.11 seconds (dashed blue line) and we observe that under on-off, time to detection varies between 41.5 seconds in the worst case (for ON period of 3) to 0.49 sec (for ON period=7). However, on-off cycle length of 3 or 5, may be too short to have an impact on the vehicle. Although this is hypothetical, we are considering the worst-case scenario, which can be verified by offensive cybersecurity researchers in future.

From Fig. 11(b), we can see that the missed detection rate increased (0.738) when the on-off cycle is 5, equal to optimal frame length but slightly less for on-off cycles 3 and 7. When the on-off cycle duration is 3, less than the optimal frame length, the average time to detection is 41.515 sec, but the time to detection rapidly drops to zero as the ON period increases slightly.



(a) (b)

Figure 11: Impersonation Attack (a) Degradation in Avg. Time to Detection (b) Missed Detection Rate

Discussion: One disadvantage of our work is that although highly accurate, it needs to run three copies of the same model architecture with three pairs of cross validated hyperparameters concurrently to detect the presence of any one or more of the attacks. However, since our method is very lightweight, it is much better compared to ECU node-specific, offset-specific and neural network based techniques. We keep the measure of complexity for this as future work.

7 CONCLUSION

In this paper, we have introduced a common model architecture that maps the entire CAN bus network's data into a stateful latent space embedding which can be used to detect three different types of attacks: Denial-of-service (DoS) attack, Fuzzy attack and Impersonation attack. Furthermore, we showed that preference based multi-objective optimization to learn the hyper-parameters of our model can balance various security performance objectives. Furthermore, we showed the benefits of quantile L1 regression for learning the threshold such that attacks can be detected in real time by comparing it with a stateful latent space embedding.

Acknowledgements: The work is supported by NSF SATC Grant # CNS-2030611.

Credit Author Statement M. Maliha - *Formal Analysis, Investigation, Software, Implementation*;
S. Bhattacharjee - *Conceptualization, Methodology.*

REFERENCES

[1] M. Al-Saud, A. M. Eltamaly, M. A. Mohamed and A. Kavousi-Fard, "An Intelligent Data-Driven Model to Secure Intravehicle Communications Based on Machine Learning",*IEEE Transactions on Industrial Electronics*, Vol. 67(6) pp. 5112-5119, 2019.
[2] O. Avatefipour, A. S. Al-Sumaiti, A. M. El-Sherbeeny, E. M. Awwad, M. A. Elmeligy, M. A. Mohamed & H. Malik"An Intelligent Secured Framework for Cyberattack Detection in Electric Vehicles' CAN Bus Using Machine Learning", *IEEE Access*, Vol. 7, pp. 127580-127592, 2019.
[3] S. Axelsson, "The base-rate fallacy and the difficulty of intrusion detection", *ACM Trans. Inf. Syst. Secur.* Vol. 3(3), pp. 186-205, Aug. 2000.
[4] S. Bhattacharjee and S. K. Das, "Detection and Forensics against Stealthy Data Falsification in Smart Metering Infrastructure," *IEEE Transactions on Dependable and Secure Computing*, Vol. 18(1), pp. 356-371,2021.
[5] S.Bhattacharjee, P. Madhavarapu, S. Silvestri, S.K. Das, " Attack Context Embedded Data Driven Trust Diagnostics in Smart Metering Infrastructure", *ACM Trans. on Privacy and Security*, Jan 2021.
[6] E. Burke, G. Kendall, "SEARCH METHODOLOGIES: Introductory Tutorials in Optimization and Decision Support Techniques", Springer, 2005.
[7] K. T. Cho, & K. G. Shin,"Fingerprinting electronic control units for vehicle intrusion detection",*USENIX Security Symposium*,Vol. 40, pp. 911-27,2016.
[8] T. Fiorese, & P. Montino, "Learning-based Intrusion Detection System for On-Board Vehicle Communication", *ITASEC*,pp. 180-192,2021.
[9] S. Fröschle and A. Stühring, "Analyzing the capabilities of the CAN attacker" *In Computer Security–ESORICS 2017: 22nd European Symposium on Research in Computer Security*, pp. 464-482, 2017.
[10] S. Halder, M. Conti, and S. K. Das., "COIDS: A Clock Offset Based Intrusion Detection System for Controller Area Networks. *Proceedings of the 21st International Conference on Distributed Computing and Networking*. pp. 1–10,2020.
[11] M. Han, B. Kwak, H. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis", *Vehicular Communications*, Vol. 14, pp. 52-63, 2018.
[12] M. L. Han, B. I. Kwak and H. K. Kim, "Event-Triggered Interval-Based Anomaly Detection and Attack Identification Methods for an In-Vehicle Network", *IEEE Transactions on Information Forensics and Security* , Vol. 16, pp. 2941-2956, 2021.
[13] Y. He, Z. Jia, M. Hu, C. Cui, Y. Cheng and Y. Yang, "The Hybrid Similar Neighborhood Robust Factorization Machine Model for Can Bus Intrusion Detection in the In-Vehicle Network", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23(9), pp. 16833-16841, 2021.
[14] R. Hu, Z. Wu, Y. Xu & T. Lai, "Multi-attack and multi-classification intrusion detection for vehicle-mou.nted networks based on mosaic-coded convolutional neural network, *Scientific Reports*, Vol. 12(1), pp. 1-16.,2022.
[15] H. Kimm & H. S. Ham, "Integrated fault tolerant system for automotive bus networks" *2010 Second International Conference on Computer Engineering and Applications*.Vol. 1, pp. 486-490,2010.
[16] C. King, D. Klinedinst. "Vehicle Cybersecurity: The Jeep Hack and Beyond." Carnegie Mellon University, Software Engineering Institute's Insights, May 23, 2016.
[17] T. Kuwahara, Y. Baba, H. Kashima, T. Kishikawa, J. Tsurumi, T. Haga, Y. Ujiie,T. Sasaki, & H. Matsushima, "Supervised and Unsupervised Intrusion Detection Based on CAN Message Frequencies for In-vehicle Network",*Journal of Information Processing*, Vol. 26, 2018.
[18] H. Lee, S. H. Jeong and H. K. Kim, "OTIDS: A Novel Intrusion Detection System for In-vehicle Network by Using Remote Frame", *2017 15th Annual Conference on Privacy, Security and Trust (PST)* pp. 57-5709, 2017.
[19] H. Lee, S. H. Jeong and H. K. Kim, "CAN Dataset for intrusion detection (OTIDS). [Online]: http://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset", 2018.
[20] C. W. Lin, and A. Sangiovanni-Vincentelli, "Cyber-security for the controller area network (CAN) communication protocol." *2012 International Conference on Cyber Security*,pp. 1-7,IEEE,2012.
[21] S. Longari, D. H. Nova Valcarcel, M. Zago, M. Carminati and S. Zanero, "CANnolo: An Anomaly Detection System Based on LSTM Autoencoders for Controller Area Network," *IEEE Transactions on Network and Service Management*, Vol. 18(2),pp. 1913-1924, 2021.
[22] C. Miller & C. Valasek, "Adventures in automotive networks and control units". *Def Con 21*, pp. 15-31,2013.
[23] C. Miller & C. Valasek, "Remote exploitation of an unaltered passenger vehicle", *Black Hat USA 2015*,(S 91), pp. 1-91,2015.
[24] S. Rajapaksha, H. Kalutarage, M. Omar Al-Kadri, A. Petrovski, G. Madzudzo, and M. Cheah. 2023. "AI-Based Intrusion Detection Systems for In-Vehicle Networks: A Survey. *ACM Comput. Surv.* Vol. 55(11), 2023.
[25] L. Ran, W. Junfeng, W. Haiying and L. Gechen, "Design method of CAN BUS network communication structure for electric vehicle" *International Forum on Strategic Technology 2010*, pp. 326-329,2010.
[26] D. Tanaka, M. Yamada, H. Kashima, T. Kishikawa, T. Haga and T. Sasaki, "In-Vehicle Network Intrusion Detection and Explanation Using Density Ratio Estimation", *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 2238-2243, 2019.
[27] D. Urbina, J. Giraldo, A. Cardenas, J. Valente, M. Faisal, N. Tippenhauer, J. Ruths, R. Candell, H. Sandberg, "Survey and New Directions for Physics-Based Attack Detection in Control Systems", *NIST Grant/Contract Reports NIST GCR 16-010*, 2016.
[28] P. Wei, B. Wang, X. Dai, L. Li, & F. He, "A novel intrusion detection model for the CAN bus packet of in-vehicle network based on attention mechanism and autoencoder", *Digital Communications and Networks*, Vol.9(1), 14-21,2023.