Towards a Comprehensive Evaluation of Voltage-Based Fingerprinting for the CAN Bus

Ashton McEntarffer
Clemson University
Clemson, SC, USA
amcenta@clemson.edu

Linxi Zhang
Central Michigan University
Mt Pleasant, MI, USA
zhang15l@cmich.edu

Yu Wei Liu
Clemson University
Clemson, SC, USA
yuwei@clemson.edu

Mert D. Pesé
Clemson University
Clemson, SC, USA
mpese@clemson.edu

Abstract

The Controller Area Network (CAN), a standard communication protocol in modern vehicles, lacks inherent security features, making it susceptible to attacks. While various defense mechanisms have been proposed, their practical implementation in resource-constrained vehicles remains limited. This paper presents a comprehensive evaluation framework for voltage-based fingerprinting, a promising technique for identifying and mitigating CAN bus attacks. This framework compares the performance of four different machine learning (ML) models, analyzes the impact of distinct sections within the CAN voltage frame, explores various waveform and feature types, and considers practical deployment factors such as detection latency and sampling rate. Notably, the paper investigates the CAN ringing phenomenon and its potential for efficient Electronic Control Unit (ECU) identification. Results demonstrate that the proposed framework offers robust classification performance while ensuring real-world feasibility.

CCS Concepts

• **Security and privacy → Intrusion/anomaly detection and malware mitigation**; **Network security**.

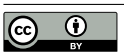
Keywords

CAN Bus, Fingerprinting, Ringing, Automotive Security

ACM Reference Format:
Ashton McEntarffer, Linxi Zhang, Yu Wei Liu, and Mert D. Pesé. 2025. Towards a Comprehensive Evaluation of Voltage-Based Fingerprinting for the CAN Bus. In *The 40th ACM/SIGAPP Symposium on Applied Computing (SAC '25), March 31-April 4, 2025, Catania, Italy*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3672608.3707981

1 Introduction

Since its introduction in 1986, the Controller Area Network (CAN) has been the *de-facto* in-vehicle network protocol in passenger vehicles. A seminal work from 2010 [8] showed the lack of fundamental security protection on the CAN bus which allowed attackers to inject arbitrary CAN messages to vehicles, making them misbehave. The exploration of CAN vulnerabilities manifested in the Jeep hack in 2015 when researchers were able to remotely control a Jeep Cherokee on the highway [5].



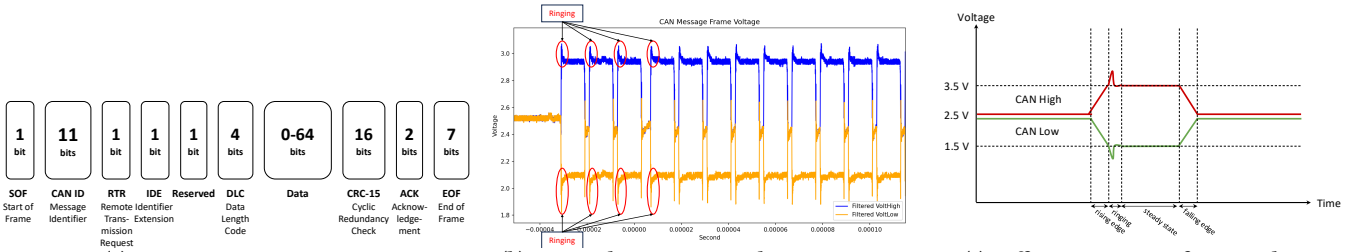
SAC '25, March 31-April 4, 2025, Catania, Italy

ACM ISBN 979-8-4007-0629-5/25/03
https://doi.org/10.1145/3672608.3707981

Although a myriad of literature on CAN defenses have since been published [1], particularly to protect the message integrity of CAN frames against spoofing attacks, practical deployment of these countermeasures in real production vehicles is still lacking [11]. Broadly speaking, there are two general ways to detect integrity attacks, namely by (i) using cryptography or (ii) intrusion detection systems (IDSes) that detect attack patterns or attacking nodes and optionally mitigate them. One key reason behind the adoption of cryptography are resource-constrained in-vehicle computers called Electronic Control Units (ECUs) in vehicles, not having the memory and CPU power for cryptographic overhead. Poor hardware also impacts the detection (and mitigation) latency which can lead to deadline misses in hard real-time systems such as vehicles.

IDSes can be implemented as a centralized, standalone monitoring ECU or distributed on several ECUs, with the majority of work focusing on the former [13]. Furthermore, they are grouped into four broad categories: (1) Fingerprint-based techniques usually operate on the physical layer and leverage signals such as time and voltage to distinguish between different ECUs, identifying attacking ECUs. Next, (2) parameter monitoring-based techniques operate on the message level and can leverage features such as the inter-arrival time between CAN messages. (3) Information theory-based approaches operate on the data-flow level and can analyze the entropy of CAN frames (or parts of them) to detect deviations which can be linked to an intrusion. Finally, (4) machine learning-based techniques are data-based, with the majority of the surveyed work in this category.

Each category has its unique drawbacks in terms of classification performance and other functional metrics such as latency or memory consumption (which are not even always evaluated). Voltage-based fingerprinting is increasingly being used for ECU identifications in recent years [2–4, 14]. It relies on unique hardware voltage characteristics that differ from ECU to ECU. Typically, a machine learning (ML) classifier is used to learn the different ECUs' fingerprints and distinguish them from each other. Compared to the other methods, voltage-based methods for intrusion detection are more difficult to evade [9]. While an in-vehicle ECU's software may be vulnerable to remote exploitation, the inherent features of its voltage signal, and thus its unique fingerprint, are challenging to manipulate in a controlled manner. However, we found that existing studies mostly (i) focus on one specific ML algorithm, (ii) do not analyze differences between unique sections of the collected voltage signal, as well as extracted features and (iii) are not designed with practical deployability in mind by making unrealistic assumptions.



(a) CAN Data Frame Syntax (b) CAN Voltage Diagram showing Ringing (c) Different Sections of CAN Voltage.

Figure 1: Overview of CAN frames and voltage characteristics.

In this paper, we address aforementioned shortcomings by comparing four different ML models on voltage data against other. Second, we define five distinct sections inside a CAN voltage frame and analyze the fingerprinting performance for each of these sections to find the most *suitable* voltage data inside a CAN frame. We also compare the impact of extracted time-domain versus frequency-domain features on classification performance, as well as the choice of raw versus differential voltage signals. All in all, these distinctions yield a total of 120 different combinations that can be considered during evaluation. Third, our work puts a special focus on certain metrics such as detection latency and sampling rate to consider real-world deployability. For instance, this paper is the only one besides Xun *et al.* [14] that considers latency in their evaluation.

The main contributions of this paper are as follows:

- We provide a comprehensive voltage fingerprinting framework to compare different combinations of ML models, frame sections, waveform types, as well as feature types on voltage data collected from the CAN bus. The datasets, code and results are publicly available under https://anonymous.4open.science/r/CAN_voltage_fingerprinting-8D5F.
- To the best of our knowledge, we are the first to analyze the CAN ringing phenomenon for voltage-based CAN fingerprinting. Analyzing the entropy of the ringing sections of a CAN frame, we observe that they have lower entropy than the other sections, meaning they have higher information density. It can furthermore be shown that ringing sections alone can be a fast approximate when a system cannot afford more complex setups.
- We demonstrate that various combinations in this paper offer similar, if not better, performance and latency compared to existing voltage fingerprinting techniques.

2 Background and Threat Model

2.1 CAN Primer

Vehicle-mounted ECUs are the source of vehicle sensor data. Typically, an in-vehicle network (IVN) is used to connect these ECUs, with the CAN bus being the most extensively used technology in vehicles today. The structure of a CAN 2.0A data frame, which is the most popular type of data frame used in CAN, is shown in Figure 1a.

CAN is a broadcast-based bus that uses differential voltage signaling to represent zeros (dominant bits) and ones (recessive bits). If a zero and a one are being transmitted by two ECUs at the same time, the zero wins. As a result, the lower message identifier (CAN ID), the higher the priority of the message. The same CAN ID is usually never transmitted by two or more ECUs.

ECUs connected to CAN will now be referred to as *CAN nodes*. A CAN node is typically made up of three primary parts that are arranged in various tiers of the OSI stack: the CAN controller, CAN transceiver, and microcontroller unit (MCU). The latter two are essential elements of CAN communication, whereas the MCU is responsible for executing the program.

CAN controllers function at the *data link layer*, utilizing specific CAN message information (CAN ID, DLC, and Data) from the MCU application to construct a whole CAN frame, which is essentially a digital bitstream. Two interfaces, CAN_TX for outgoing data and CAN_RX for inbound data, are provided by each CAN controller to the bottom *physical layer*. Moreover, the CAN controller carries out the fundamental operations of the CAN protocol.

CAN transceivers function on the *physical layer*. They create a bitstream for CAN_RX from the voltage and convert digital bitstreams from CAN_TX to an analog voltage (between 0 and 5 V). The two CAN_H (CAN High) and CAN_L (CAN Low) levels of differential voltage signaling are used by CAN. Both CAN_H and CAN_L have voltage levels of 2.5 V when transmitting a recessive (1) bit. CAN_H and CAN_L would use 3.5 V and 1.5 V, respectively, while transmitting a dominating (0) bit.

2.2 CAN Ringing

The ringing section of a bit in a digital signal refers to the transient oscillation or resonance that occurs immediately following a transition from one state (e.g., low voltage) to another (e.g., high voltage). This phenomenon is characterized by a series of decaying oscillations around the new steady-state value. It is caused by the physical properties of the electronic circuitry, such as capacitance and inductance, which resist sudden changes in voltage or current. The ringing section typically settles down to the steady-state value of the bit over a period of time, which can vary depending on the specific characteristics of the circuit. Figure 1b depicts a CAN frame with highlighted ringing envelopes.

2.3 Threat Model

Adversaries are primarily focused on (i) compromising the ECU(s) or (ii) adding external ECU(s). The former is similar to a remote compromise of an existing ECU (e.g., as seen in the Jeep hack [5]). The attacker controls an existing ECU and transmits CAN

messages with a spoofed CAN ID (and payload), impersonating another legitimate node in the network. On the other hand, the latter requires an attacker to physically connect a new ECU to the CAN bus (e.g., through the OBD-II port). The new adversarial node can then transmit spoofed CAN messages as well, impersonating other in-vehicle ECUs.

This threat model is consistent with existing literature. Some existing proposals [2, 4, 10] focus on detecting damaged internal ECUs. That is, they cannot handle the situation where an attacker adds an external ECU to launch attacks. Some other approaches [3, 6, 7, 9] consider both attack vectors to ensure robust defense mechanisms against a broad spectrum of potential threats to CAN bus. In this work, we cover both attack vectors.

3 Related Work

Several methods have been proposed for ECU fingerprinting and intrusion detection, leveraging signal processing, statistical analysis, and machine learning techniques. Despite their contributions, existing approaches face limitations.

One major limitation is the reliance on a narrow set of machine learning algorithms. Xun *et al.* [14] propose a hybrid approach combining FeatureBagging and Convolutional Neural Networks (CNN), achieving up to 98.85% accuracy. However, despite the promise of such hybrid models, the approach remains centered on deep learning, introducing computational challenges that limit its applicability in resource-constrained, real-time environments such as vehicles. Similarly, Choi *et al.* [3] use Support Vector Machines (SVM), while Kneib and Huth [6] employ logistic regression for intrusion detection. While effective within controlled environments, these methods rely on singular algorithms that may overlook the broader complexity of ECU behaviors in diverse settings.

Another shortcoming is the lack of detailed analysis of the differences between unique sections of the collected voltage signals. Existing studies fail to sufficiently explore how different sections of the signal, such as rising edges or falling edges, can impact the accuracy of ECU fingerprinting. Murvay and Groza [10] introduce signal processing techniques based on mean square errors and convolutions, but their analysis remains restricted to specific features, overlooking other aspects of the signal that could offer deeper insights. This focus on a limited set of features may reduce the adaptability of these methods in real-world scenarios.

Finally, the practical implementation of these methods in real automotive environments is often hindered by unrealistic assumptions. Many approaches assume high sampling rates or computational resources that are beyond the capabilities of typical in-vehicle systems. For example, both Levy *et al.* [9] and Popa *et al.* [12] rely on a 500 MS/s sampling rate, which far exceeds the capability of most onboard analog-to-digital converters (ADCs), typically capped at 12.5 MS/s. These high data acquisition rates introduce significant computational overhead, making the systems impractical for real-time deployment in real vehicles. Viden, introduced by Cho *et al.* [2], presents a more deployable solution with its lower 50 KS/s sampling rate, but still relies on proprietary voltage profiles, which may not be easily adaptable across different vehicle models or under varying environmental conditions. Additionally, the VehicleEIDS system by Xun *et al.* [14] offers robustness through the use of differential voltage signals, but its deployment is still limited by the need for vehicle-specific tuning and calibration.

In contrast, our approach addresses the shortcomings by (1) comparing four different ML models on voltage data to enhance adaptability, (2) analyzing five distinct sections of the CAN voltage frame to identify the most informative data segments, and (3) focusing on practical detection latency and sampling rate to ensure real-world deployability. Our framework not only improves classification performance but also facilitates efficient deployment in cars. A comparison of our work with prior art is shown in Table 5.

4 System Design

The proposed framework reads the sampled CAN_H and CAN_L raw voltages from the CAN bus. The system operates on a CAN frame-level because this ensures that extracted features are properly associated with a single ECU. At a sampling rate of 6.25 MS/s, roughly 3000 voltage samples are collected per CAN frame.

Figure 2 provides an overview of our framework. First, the CAN frame is split into four distinct sections that are a contribution of this paper. Then, features are extracted from each section (see Section 4.1). For the specific ringing section, a preprocessing step is necessary (see Section 4.2) before feature extraction can be performed on the voltage data. Finally, the multi-class classification process is explained in Section 4.3 to determine if the message traffic from an ECU is adversarial or not.

4.1 Feature Extraction

Whether to train the models or make classifications, statistical features must be extracted from the CAN frame. There are two types of *data representations*: (i) Raw data where both CAN_H and CAN_L voltages are analyzed separately, and (ii) differential data where only the differential voltage between CAN_H and CAN_L are analyzed for features.

Next, *sections* within the CAN frame are identified. Sections consist of rising edge, ringing envelope, steady state, and falling edge. Figure 1c depicts the four different sections. Please note that these sections do not occur at every bit of a CAN frame, but only when there is a bit transition, i.e., from dominant (0) to recessive (1) or vice versa. We further refer to a sequence of contiguous bits with the same bit state (either sequence of 0s or 1s) as *current bit state*. Rising edge refers to the section of a bit transition starting at the point where the voltage differential is greater than 0.08 V, and ending at the local extremum voltage value located past the threshold of recessive state bit sampling.

The ringing envelope is defined as the section of a bit starting at the local extremum voltage value located past the threshold of recessive state bit sampling, and continuing until the voltage has decayed to a stable value. The end of this decay is identified as the point where the slope of the decay curve has fallen below a significant threshold value. This is typically 0.01 V for CAN_H, and 0.005 V for CAN_L by our experimental observations. The methods used to identify these points are expanded upon in Section 4.2.

Steady state refers to the section after a ringing envelope. It is characterized by maintaining a steady voltage value with little to no significant oscillations, beginning from the end point of the

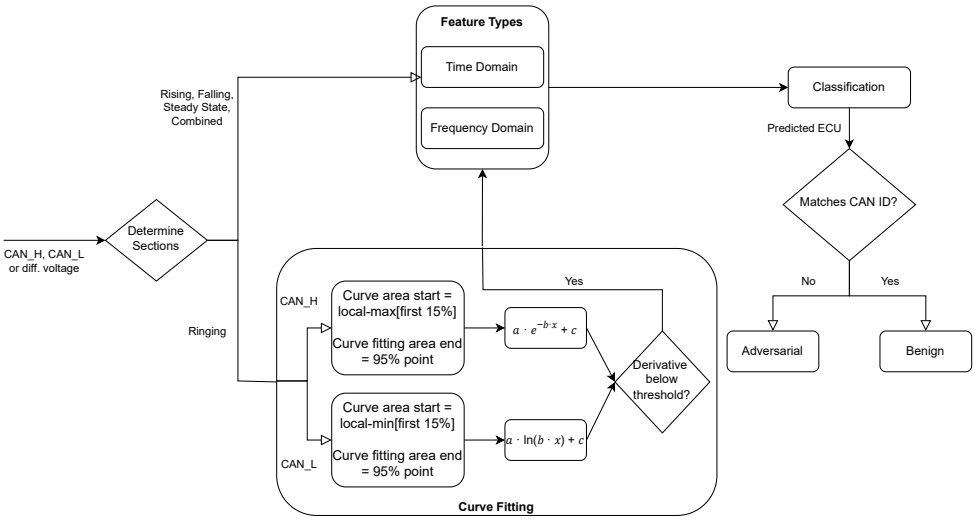


Figure 2: System Design Overview of our Framework

ringing section and ending at the point where the voltage value falls beneath the threshold of recessive state bit sampling.

Finally, *falling edge* is the section of a bit transition starting at the point where the voltage value falls below the threshold of recessive state bit sampling and ends at the point where the voltage differential falls below 0.08 V. It follows the steady state section.

For the various sections, the following features can be extracted from the voltage time domain: Mean, standard deviation, mean deviation, skewness, kurtosis, root mean square, maximum, minimum. Frequency domain features are also calculated from the data using Fast Fourier Transform (FFT), including centroid, entropy, spread, skewness frequency, average spectrum, variance, kurtosis frequency, and irregularity. These features are consistent with existing literature [14]. Features are extracted and compiled as a single sample, which is passed on to model training and classification, which are expanded upon in Section 4.3.

4.2 Ringing Extraction

Once a ringing section is identified, a preprocessing step called *curve fitting* to calculate the envelope of oscillation is defined. Using this envelope, we can determine the start and end points of this section.

First, the raw voltage data is passed to the function defined in Algorithm 1. Lines 2-23 outline the process to determine the current bit state. This is necessary, as the curve fitting works with the raw voltage information of the CAN frame, and does not initially know the current bit state. The sections of raw voltage data starting where the differential voltage between CAN_H and CAN_L rises above 0.08 V and ending when the differential falls below 0.08 V are identified and recorded. Once a bit has been identified, it is appended to a list of bits. The final bit is removed, as this is the ACK bit sent by the replying ECU, and might skew the classification of the sample if its information is included in the feature calculation.

Lines 24-31 in Algorithm 1 outline the identification of *curve areas*. This process narrows down the range of the current bit state that might contain the ringing sections to attain more accurate results from the curve fitting process. Specifically, the rising edge and falling edge of the current bit state would skew the curve downwards as the majority of their signals are below the threshold for recessive bit state voltages while the majority of the ringing section is above this threshold. To perform this range narrowing, the start point of the area of the current bit state to undergo curve fitting is defined as the maximum (for CAN_H) or minimum (for CAN_L) voltage value of the first 15% of the current bit state, which is also the end of the rising edge. The end point of the area of the current bit state to undergo curve fitting is defined as 95% of its width, as this is roughly before the end of the steady state section of the current bit state. After each curve area is identified, it is appended to a list of curve areas.

Lines 32-42 in Algorithm 1 outline the curve fitting process itself. For each curve fitting area in the curve fitting areas list, both CAN_H and CAN_L are analyzed separately. CAN_H curve fitting areas are fit with an exponential decay curve function of the form $a \cdot e^{-b \cdot x} + c$. CAN_L curve fitting areas are fit with a logarithmic growth function of the form $a \cdot \ln(b \cdot x) + c$. CAN_H is fit with an exponential decay curve because the ringing section on CAN_H represents a decay from the peak value of the rising edge section of CAN_H to the beginning of the steady state section. CAN_L is fit with a logarithmic growth curve because the ringing section on CAN_L represents a rise in voltage from the minimum value of the rising edge section of CAN_L to the beginning of the steady state section. To allow for dynamic curve fitting, the initial parameters of each curve can be set based on the values of CAN_H and CAN_L. For the exponential decay curve, a represents the initial magnitude of the curve, and is set equal to the range of the curve area of CAN_H. b represents the rate of decay, and is set equal to $1/t_H$ where t_H represents the timespan of the curve area of CAN_H. c represents the constant vertical offset of the curve and is set equal to the minimum value of the curve area of CAN_H. For the logarithmic growth curve, a represents the initial magnitude of the curve, and is set equal to the range of the curve area of CAN_L. b represents the scaling factor of the growth rate, and is set here to a value such as 0.01 as to modulate the growth factor for ease of processing. c

Algorithm 1 find_ringing(CAN_H, CAN_L)

Initialize an empty list bits ← [ ]
Set on_bit ← **False**
Set start_time ← **None**
for each index i in data **do**
  voltage_diff[i] ← |CAN_H$_i$ − CAN_L$_i$|
  voltage_diff[$i+1$] ← |CAN_H$_{i+1}$ − CAN_L$_{i+1}$|
  if voltage_diff[i] > 0.08 **then**
    if on_bit == **False and** voltage_diff[i+1] > voltage_diff[i] **then**
      start_time ← data.time$_i$
      on_bit ← **True**
    end if
  else if voltage_diff[i] < 0.08 **then**
    if on_bit == **True and** voltage_diff[i+1] < voltage_diff[i] **then**
      end_time ← data.time$_i$
      if start_time exists **then**
        Append (start_time, end_time) to bits
        start_time ← **None**
      end if
      on_bit ← **False**
    end if
  end if
end for
Remove the last element from bits ▷ Remove ACK bit
Initialize an empty list curve_areas ← [ ]
for each bit in bits **do**
  H_curve_start ← max value from first 15% of bit
  L_curve_start ← min value from first 15% of bit
  curve_end ← 95% length of bit
  Extract {H,L}_curve_area from bit using indices
  Append ({H,L}_curve_area to curve_areas
end for
Initialize an empty list ringing ← [ ]
for each ({H,L}_curve_area in curve_areas **do**
  Normalize H_curve_area and L_curve_area
  Fit exponential decay: H_curve = $a \cdot e^{-b \cdot x} + c$
  Fit logarithmic function: L_curve = $a \cdot \ln(b \cdot x) + c$
  Calculate diff of the fitted curves
  Locate |dy/dx$_{\text{high}}$| < 0.01 and |dy/dx$_{\text{low}}$| < 0.005
  {H,L}_ringing ← {H,L}_curve up to diff
  Append (H_ringing, L_ringing) to ringing
end for
return ringing

represents the constant vertical offset of the curve and is set equal to the minimum value of the curve area of CAN_L. For both curves, x represents the voltage values.

Finally, the end of the ringing section, and beginning of the steady state, is defined as the point where the derivative of the ringing curve falls below a threshold value, typically between 0.01 and 0.005, and may differ between CAN_H and CAN_L due to physical differences in the behavior of voltage levels in the two channels. After each ringing section is identified, it is appended to a list of ringing sections. Once all ringing sections in a CAN frame are identified, the list of ringing sections is passed on to the feature extraction module.

4.3 Classification

Four ML models are used in this paper: Support Vector Machine (SVM), Random Forest (RF), Neural Network (NN), and Convolutional Neural Network (CNN). For training, we utilize an 80/20 training-testing split, data normalization, and class weight balancing to account for model underrepresentation that may arise from random sampling. During training, we use the CAN ID of a message as label for each data sample.

During classification, probabilities are estimated for each ECU, and those probabilites are utilized to create a softmax classification layer for the SVM and RF models. The SVM model is compiled with a linear kernel, as the output data classification should be able to be differentiated using linear separation. Also, the ECU labels of the extracted data are one-hot encoded to improve performance in the NN and CNN models. By converting the labels from categorical to numerical data, classification latency is decreased and results more clearly convey the models' accuracy. The NN consists of a dense layer of 128 neurons, a dense layer of 64 neurons, and a dense layer of 32 neurons. ReLu activation and a dropout of 0.3 is used between dense layers. Finally a softmax layer of 8 neurons is used to classify the sample to an ECU. Categorical cross entropy is used for the loss function, and Adam optimization is used to speed up gradient descent. The CNN automatically reshapes the input data to be rectangular before processing the data. It consists sequentially of a 1D convolutional layer of 32 neurons, a kernel size of 3, with ReLu activation function, a second 1D convolutional layer of 64 neurons, kernel size of 3, with ReLu activation. It then flattens the data and passes it to a 128 neuron dense layer with ReLu activation. Finally, a dropout of 0.5 is applied, and the data is passed to a softmax function to classify the ECU. The model utilizes Adam optimizer with categorical cross entropy loss.

The classification is performed for each current bit state, i.e., contiguous bit segments of either a dominant or recessive state. On average, there are 20 of them in a CAN frame. Ideally, for each CAN frame, each single classification output shall be identical. For a CAN frame transmitted by an adversarial ECU, the classification output is more likely to be different for each current bit state as it cannot match each state to a specific ECU. Knowing this, a threshold-based system can be developed to detect adversarial ECUs. For example, if 4 or more classifications out of the last 20 current bit states are different from the other 16, it is classified as coming from an unknown adversarial ECU. Note that this threshold-based detection is not evaluated in Section 5, but is reserved for future work.

5 Evaluation

5.1 Experimental Setup

Our setup consists of a CAN bus of eight ECUs, with one ECU being adversarial (ECU 8). Each ECU is represented by an Arduino Uno R3 with a Seeed Studio CAN-BUS Shield V2. Each benign ECU is transmitting messages with a unique CAN ID, ranging from 0x01 to 0x07. The voltage readings from the CAN bus are collected at 6.25 MS/s sampling rate using the PicoScope 2204A. A varying number of 200 - 2000 message frames are collected for each ECU which is representative of real CAN traffic. In total, over 10000 frames were collected during our experiments. To process the signal features,

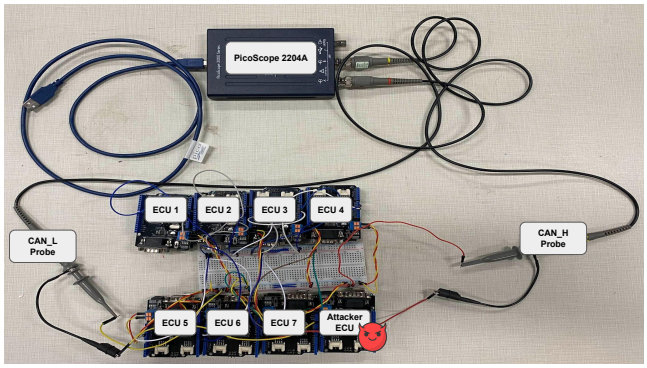


Figure 3: Experimental testbed with 7 benign and one adversarial Arduino

Table 1: Entropy data for all ECUs

ECU	H Ringing	L Ringing	H Other	L Other	H Factor	L Factor
1	3.90	3.90	4.79	4.79	8.90	8.90
2	3.82	3.83	4.73	4.73	9.10	9.00
3	3.85	3.86	4.73	4.73	8.80	8.70
4	3.81	3.81	4.71	4.7	9.00	8.90
5	3.96	3.96	4.84	4.84	8.80	8.80
6	3.85	3.85	4.73	4.73	8.80	8.80
7	3.89	3.89	4.78	4.78	8.90	8.90
Adv	3.87	3.87	4.76	4.76	8.90	8.90
Average	3.86	3.87	4.75	4.75	8.90	8.86

the ACK bit was excluded, as this includes voltage features from ECUs separate from the message sender.

After sampling the CAN bus using the Picoscope, a Raspberry Pi 4 extracted the features and performed the multi-class classification. Latency was measured on the Raspberry Pi to provide a realistic representation of in-vehicle ECUs. During model training, a high performance cluster with 10 CPU cores, 256 GB Memory, 5 Nvidia Tesla A100 GPU was used. In regards to ringing section identification, the average ending and starting point of all ECUs is collected for CAN_H and CAN_L channels using the procedure described in Algorithm 1. These points are then used to extract features from all ECUs during signal feature extraction. All models utilize Python 3.6.8 with Keras 2.6 and Tensorflow 2.6 framework. Note that the curve fitting for ringing envelope identification utilizes Scipy 1.5.4.

5.2 Ringing Entropy

After analyzing the ringing sections for their information content, it appears that they are far more information-dense than other sections. This information density is quantified through its Shannon entropy, which is a calculation of the uniqueness of the ringing section relative to the other sections:

$$H(X) = -\sum p(X) \log p(X) \tag{1}$$

While other sections (rising edge, falling edge, steady state) had an average Shannon entropy of around 4.75 for CAN_H and CAN_L (depicted as H and L in Table 1), the average entropy of CAN_H and CAN_L for the ringing sections was 3.86 and 3.87, meaning the ringing sections were over 8 times more information dense.

Table 2: Data Representation Type Performance Metrics

Metric	Raw Average	Differential Average
Precision	0.8557	0.6688
Recall	0.8764	0.7077
F1 Score	0.8621	0.6731
Accuracy	0.8732	0.7036
False Positive Rate	0.0183	0.0424
False Negative Rate	0.1236	0.2923
Classification Latency (μs)	150.85	144.58

One takeaway from this finding includes the fact that it might be possible to create voltage-based identification systems that only require features extracted from ringing sections, if their performance would be on par with the other sections.

5.3 Fingerprinting Performance

5.3.1 Experiment 1: Data Representation Type Comparison. Table 2 shows that raw data provides considerably better performance metrics than differential data. Latency is also comparable between the two data representation types, which indicates that CAN_H and CAN_L should be used individually for voltage fingerprinting instead of the differential voltage.

5.3.2 Experiment 2: Signal Section Comparison. Table 3 shows that combined rising edge, steady state, and falling edge signal sections provide the highest classification performance, displayed by the highest F1 Score of 0.9247. Findings also show that steady state sections provide the highest individual section performance, signified by an F1 Score of 0.8097, while rising edge and falling edge sections provide the lowest performance, signified by an F1 Score of 0.7147 and 0.7049 respectively. Ringing sections had comparable performance to steady state sections, with an F1 Score of 0.7606. This implies that higher performance sections consisting of ringing section and steady state sections might be ideal. Note that latency for classification was highest for ringing and steady state sections, at 153.40μs and 156.04μs, respectively.

5.3.3 Experiment 3: Feature Type Comparison. Table 3 shows that combined voltage and frequency features provide the highest classification performance. While F1 Score for voltage and frequency data separately is 0.7727 and 0.7093, respectively, F1 Score for combined voltage and frequency data is 0.8208. The classification latency for combined features is the highest, at 149.55 μs, though this isn't considerably higher than the latency for voltage and frequency, which is 149.07 μs and 144.53 μs, respectively.

5.3.4 Experiment 4: Model Type Comparison. Table 3 shows that CNN, NN and RF models provide the highest classification performance. F1 Score for CNN, NN, and RF models are 0.7676, 0.7014, and 0.7119 respectively, while the F1 Score for SVM models is considerably lower at 0.5024. Note that while the performance of NN and CNN models are comparable to RF, they have higher classification latency at 124.62 μs and 147.71 μs, compared to the RF classification latency of 31.18 μs.

5.3.5 Optimal Combinations. Utilizing the data outlined in the previous subsections and in Table 3, it is clear that certain combinations of data representation types, signal sections, feature types, and

Table 3: Average Signal Section, Feature Type, Model Performance Performance Metrics

Category	Precision	Recall	F1 Score	Accuracy	False PositiveRate	False NegativeRate	Avg. ClassificationTime per Sample (μs)
Rising	0.6715	0.7174	0.6779	0.7008	0.0427	0.2826	143.39
Ringing	0.7611	0.7730	0.7606	0.7803	0.0318	0.2270	153.40
Steady	0.8045	0.8292	0.8097	0.8281	0.0248	0.1708	**142.36**
Falling	0.6570	0.7049	0.6652	0.6975	0.0431	0.2951	143.38
Combined	**0.9173**	**0.9359**	**0.9247**	**0.9353**	**0.0094**	**0.0641**	156.04
Voltage	0.7675	0.7960	0.7727	0.7943	0.0295	0.2040	149.07
Frequency	0.7047	0.7423	0.7093	0.7300	0.0386	0.2577	**144.53**
Combined	**0.8147**	**0.8380**	**0.8208**	**0.8408**	**0.0229**	**0.1620**	149.55
SVM	0.5404	0.5092	0.5024	0.6195	0.0579	0.4908	51.66
RF	0.7311	0.7017	0.7119	0.7621	0.0353	0.2983	**31.18**
NN	0.6999	0.7390	0.7014	0.7278	0.0388	0.2610	124.62
CNN	**0.7623**	**0.7921**	**0.7676**	**0.7884**	**0.0304**	**0.2079**	147.71

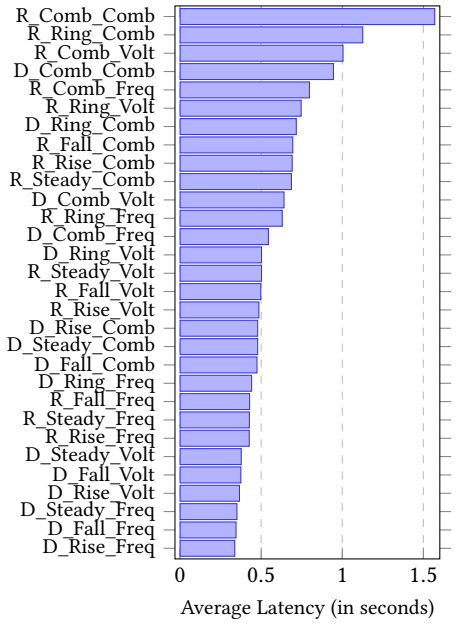


Figure 4: Average Feature Extraction Latency (RPi 4)

model types provide higher performance than others. Summarizing these findings, it would appear that the highest performance could be attained by implementing a CNN that analyzes both CAN_H and CAN_L data, and combined frequency and voltage features extracted from combined rising edge, steady state, and falling edge sections. Furthermore, poor performance can be expected from SVM models that analyze differential frequency data from falling or rising edge sections. Latency is comparable between combinations, though SVM and RF are the fastest. Ringing sections provide a level of performance between combined and steady state sections, and rising and falling edge sections. This implies that solely looking at ringing or steady state sections could provide a quick solution for classification needs while not significantly sacrificing performance.

5.4 Fingerprinting Latency

Total latency is important to consider in terms of practical performance. Fingerprinting pipelines must be able to extract voltage data from the CAN bus, extract features from the voltage data, and classify the samples from the features in fractions of a second. Hardware limitations must also be brought into account, as higher sampling rates require more advanced devices that might not be feasible for most in-vehicle settings. To represent the total latency from our analysis, we define the following equation:

$$t_T = t_F + t_C + t_S \approx t_F + t_C \tag{2}$$

In this equation t_T represents the total fingerprinting latency. It is equal to the feature extraction latency t_F plus the sample classification latency t_C plus the CAN frame sampling latency t_S (which is negligible). Table 3 shows that sample classification latency is insignificant compared to the feature extraction latency which is reported in Figure 4. It shows that the *Differential-Rising Edge-Frequency* combination has the lowest latency at 0.3373 seconds, while *Raw-Combined Section-Combined Feature* has the highest latency at 1.57 seconds. It appears that combinations with higher latency tend to be those with better classification performance.

Table 4: Performance Metrics for Adversarial ECU

Metric	Mean	Stddev
Precision	0.1217	0.1079
Recall	0.0155	0.0344
F1 Score	0.0224	0.0448
Accuracy	0.0797	0.1641
False Positive Rate	0.2248	0.1022
False Negative Rate	0.2248	0.1022
Classification Latency (μs)	38.0	19.5

5.5 Adversarial Classification

All experiments so far were conducted on fingerprinting benign ECUs. We analyzed the performance of our classification models on an additional adversarial ECU to gather information about the ability of our models to avoid accepting adversarial data as normal data. The adversarial ECU was set up to broadcast messages with an existing CAN ID, e.g., 0x05 representing ECU 5. After testing each model on features collected from an adversarial ECU, we report the mean and standard deviation of performance metrics in Table 4.

Comparing the classification performance shown in Table 2 with the adversarial classification performance from Table 4, it is clear that the classification performance of the models is significantly lower when tested against unrecognized adversarial signal features.

Table 5: Comparison of Voltage-Based IDS

	Murvay *et al.* [10]	Choi *et al.* [3]	Choi *et al.* [4]	Scission [6]	Xun *et al.* [14]	**Our Work**
Sampling Rate	2 GS/s	2.5 GS/s	2.5 GS/s	20 MS/s	12.5 MS/s	**6.25 MS/s**
Signal	High, Low	Diff	Diff	Diff	Diff	**Diff; High, Low**
Match IDs for each ECU	-	-	Protocol ID	Proprie-tary	Clock Skew	**Protocol ID**
Platform	Testbed	Testbed	Car	Car	Car	**Testbed**
Model Perfor-mance	N/A Mean Squared Error	96.48% SVM	94.14% SVM	99.85% Logistic Regres-sion	98.85% Feature-Bagging - CNN	**96.13% CNN, Various Models**
Latency	-	-	-	-	3 ms	1570 ms

This verifies our claims from Section 4.3 as each current bit state is classified inconsistently. This implies that threshold-based techniques for CAN frame classification would be optimal in identifying adversarial CAN frames.

5.6 Comparison with Related Work

Table 5 shows how the results from our analysis compare to the current state-of-the-art. Looking at the methods outlined by [10], [3], [4], [6], and [14], they used significantly higher sampling rates than the 6.25 MS/s used in our approach. This exemplifies the fact that we utilize far more realistic sampling rates and hardware compared to existing studies, meaning our results more accurately represent how these approaches might be applied in a real automotive environment. Also note that we are the first to analyze both High, Low and Differential data in finding optimal features for model performance. In addition to those advancements, we show similar intrusion detection F1 Score performance without needing to utilize more complicated model architectures such as Feature-Bagging CNN models. Overall, we believe our work is among the first to truly display how voltage-based IDSes could be practically implemented in a general in-vehicle environment. While our latency might be higher than that shown in [14] (evaluated on an HPC-grade Intel Xeon Gold 5218R CPU, NVIDIA GeForce RTX 3080 and 128 GB RAM), we are among the first to demonstrate voltage-based IDS on low-resource systems (such as the Raspberry Pi 4) comparable to those expected in many real world environments.

6 Conclusion

This paper provides a thorough examination of voltage-based fingerprinting for securing CAN bus communication in vehicles. The evaluation framework incorporates a range of ML models, signal sections, feature types, and data representations, offering a multifaceted perspective on the technique's effectiveness. Key findings include that (1) raw voltage data, as opposed to differential data, yields superior performance metrics; (2) Combining rising edge, steady state, and falling edge sections provides the best classification performance; (3) Steady state and ringing sections demonstrate comparable performance and could be utilized for efficient ECU identification; (4) CNNs, NNs, and RFs exhibit the highest classification performance; (5) Analyzing adversarial ECU behavior reveals the need for threshold-based techniques to effectively detect malicious CAN frames. Significantly, the paper highlights the potential of utilizing the CAN ringing phenomenon for fast and efficient ECU identification. With its lower entropy and distinct characteristics, the ringing section presents a promising avenue for future research. The framework's focus on practical considerations like sampling rate and latency ensures its applicability in real-world automotive environments. This research contributes to the advancement of CAN bus security by offering a comprehensive and deployable solution for identifying and mitigating potential attacks.

Acknowledgments

This project is partially supported by the Clemson University Creative Inquiry program, as well as Clemson University's Virtual Prototyp- ing of Autonomy Enabled Ground Systems (VIPR-GS), under Coop- erative Agreement W56HZV-21-2-0001 with the US Army DEVCOM Ground Vehicle Systems Center (GVSC). DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited. OPSEC #9270

References

[1] Mehmet Bozdal, Mohammad Samie, Sohaib Aslam, and Ian Jennions. 2020. Evaluation of can bus security challenges. *Sensors* 20, 8 (2020), 2364.
[2] Kyong-Tak Cho and Kang G Shin. 2017. Viden: Attacker identification on in-vehicle networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1109–1123.
[3] 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. *IEEE Transactions on Vehicular Technology* 67, 6 (2018), 4757–4770.
[4] 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. *IEEE Transactions on Information Forensics and Security* 13, 8 (2018), 2114–2129.
[5] Andy Greenberg. 2015. Hackers Remotely Kill a Jeep on the Highway—With Me in It. https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway.
[6] Marcel Kneib and Christopher Huth. 2018. Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 787–800.
[7] Marcel Kneib, Oleg Schell, and Christopher Huth. 2019. On the robustness of signal characteristic-based sender identification. *arXiv preprint arXiv:1911.09881* (2019).
[8] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, et al. 2010. Experimental security analysis of a modern automobile. In *2010 IEEE symposium on security and privacy*. IEEE, 447–462.
[9] Efrat Levy, Asaf Shabtai, Bogdan Groza, Pal-Stefan Murvay, and Yuval Elovici. 2023. CAN-LOC: Spoofing detection and physical intrusion localization on an in-vehicle CAN bus based on deep features of voltage signals. *IEEE Transactions on Information Forensics and Security* (2023).
[10] Pal-Stefan Murvay and Bogdan Groza. 2014. Source identification using signal characteristics in controller area networks. *IEEE Signal Processing Letters* 21, 4 (2014), 395–399.
[11] Mert Dieter Pese. 2022. *Bringing Practical Security to Vehicles*. Ph. D. Dissertation.
[12] Lucian Popa, Bogdan Groza, Camil Jichici, and Pal-Stefan Murvay. 2022. Ecuprint—physical fingerprinting electronic control units on can buses inside cars and sae j1939 compliant vehicles. *IEEE Transactions on Information Forensics and Security* 17 (2022), 1185–1200.
[13] Wufei Wu, Renfa Li, Guoqi Xie, Jiyao An, Yang Bai, Jia Zhou, and Keqin Li. 2019. A survey of intrusion detection for in-vehicle networks. *IEEE Transactions on Intelligent Transportation Systems* 21, 3 (2019), 919–933.
[14] Yijie Xun, Zhouyan Deng, Jiajia Liu, and Yilin Zhao. 2023. Side Channel Analysis: A Novel Intrusion Detection System Based on Vehicle Voltage Signals. *IEEE Transactions on Vehicular Technology* (2023).