vProfile: Voltage-Based Sender Identification on Controller Area Networks

by

Nathan Dingming Liu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2021

Author's Declaration

This thesis consists of material all of which I authored or co-authored (see Statement of Contributions included in the thesis). This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Every chapter except Chapter 5 is based on a paper that was co-authored by myself, Carlos Moreno, Murray Dunne, and my supervisor Professor Sebastian Fischmeister entitled "vProfile: Voltage-Based Anomaly Detection in Controller Area Networks" that was presented at the virtual 2021 Design, Automation and Test in Europe Conference (DATE) [22]. Nathan Liu was the sole author of the remainder of this thesis.

We conducted this research at the University of Waterloo under the supervision of Professor Sebastian Fischmeister. Carlos Moreno suggested the initial idea and Nathan Liu implemented it and created the experiments. Carlos Moreno and Murray Dunne assisted with the mathematical aspects of the algorithm and statistical analysis of experimental results.

Abstract

Modern vehicles are becoming more accessible targets for cyberattacks due to the proliferation of wireless communication channels. The intra-vehicle Controller Area Network (CAN) bus lacks sender authentication, exposing critical components to interference from less secure, wirelessly compromised modules.

To address CAN's vulnerability, this thesis proposes vProfile, a sender identification system based on voltage fingerprints of electronic control units (ECU). vProfile exploits the physical properties of ECU output voltages on the CAN bus to determine the authenticity of bus messages, which enables the detection of both hijacked ECUs and external devices connected to the bus. We show the potential of vProfile using experiments on two production vehicles with precision and recall scores of over 99.99%. We also show the impact of temperature and battery voltage variations on vProfile and provide a method to adapt to those changes. The improved identification rates and more straightforward design of vProfile make it an attractive improvement over existing methods.

Acknowledgements

I want to thank my supervisor Professor Sebastian Fischmeister for providing me with the opportunity to be a member of his research group. I participated in many engaging and industry-relevant projects and learned plenty from his grammar lessons. I would also like to thank Carlos Moreno and Murray Dunne for sharing their expertise that helped me tremendously and the other members of the Real-time Embedded Software Group for making my Master's degree an enjoyable experience.

Finally, I would like to thank Feridun Hamdullahpur, the former President and Vice-Chancellor of the University of Waterloo, for considering me vital to the institution's academic mission.

Dedication

I dedicate this thesis to my parents Lixin Liu and Yinglin Ma and my grandparents Shichen Liu, Shude He, Chunsheng Ma, and Chuanen Jiang. I cannot hope to repay their endless love and support and, although some of them are no longer with us, I know that they will continue to watch over me.

Table of Contents

List of Tables

List of Figures

Abbreviations

A/C Air Conditioning 35

ACK Acknowledgment 3, 9, 41

BCH Bose–Chaudhuri–Hocquenghem 9

CAN Controller Area Network 1–7, 10, 13, 14, 16, 23, 24, 30, 33, 35, 42, 45, 46

CIDS Clock-Based Instrusion Detection System 4

CRC Cyclic Redundancy Check 9

DLC Data Length Code 9

DoS Denial-of-Service 2

ECM Engine Control Module 10, 11, 33

ECU Electronic Control Unit 1–5, 8, 11, 13, 14, 16–18, 24, 26, 28–31, 33, 34, 41–43, 45

EOF End-of-Frame 9

ID Identifier 7, 8, 10, 14

IDS Intrusion Detection System 1–4, 46

LUT Lookup Table 17, 43

MSE Mean Square Error 3

PGN Parameter Group Number 10, 11

RTR Remote Transmission Request 9

SA Source Address 8, 10, 11, 13, 16, 17, 24, 31, 43, 45

SOF Start-of-Frame 9, 17

SRR Substitute Remote Request 9

Chapter 1

Introduction

Modern vehicles are becoming more connected, with newer models including cellular network interfaces to augment their comfort and convenience. However, having more connectivity implies new attack vectors [5]. These new entry points allow attackers to remotely infiltrate systems without physical access, which is particularly concerning when considering the fundamentally insecure yet widely-used Controller Area Network (CAN) bus for intra-vehicle communication.

CAN is an unauthenticated channel, designed in an era where cybersecurity was of little concern for automobiles due to their limited connectvitiy [17]. The lack of authentication means that a node on the bus, called an electronic control unit (ECU), can imitate any other ECU without raising suspicion. This deficiency is a significant flaw since ECUs can control critical system aspects, such as the brakes and airbags in a car.

Authentication methods based on cryptographic schemes, like message authentication codes, are quick to present themselves but are challenging to implement in CAN communications. The issue with using cryptography is that CAN has a low bandwidth, allowing only 8 bytes of data per message. Furthermore, additional computational power would be needed to achieve the recommended security levels while maintaining real-time constraints [19]. Instead of an active approach, one can passively monitor the bus traffic or its analog-domain information for unusual behavior. This method is typically known as an intrusion detection system (IDS). Research on IDSs exists in more than just the automotive industry [4, 35].

This thesis proposes vProfile, a system for verifying the origin of CAN messages by using the distinct electrical properties of a transmitting ECU. Specifically, we use the CAN bus voltage as a reputable source of analog-domain information. vProfile can integrate

into an IDS to enable message sender identification and the ability to detect unauthorized transmissions. Because of the versatility of CAN, with use in medical equipment, marine electronics, and factory automation [31], vProfile's applicability exceeds the vehicular domain.

1.1 Problem Statement

As mentioned previously, there are significant vulnerabilities in CAN's design and current literature shows several different successful attacks on CAN networks, including the work by Miller and Valasek on a Jeep Cherokee [24, 25]. The authors send arbitrary CAN messages by infiltrating the infotainment system through the car's cellular connection. Miller and Valasek were able to remotely control the engine and brakes of the vehicle, among other components. Other works also describe this message injection attack [3, 17]. Additional attack types include:

- Inducing faults to disable an ECU [6, 15, 32]
- Denial-of-Service (DoS) [3, 20]
- Malicious code injection [3, 17, 20]

Electronics contributed to about 10% of a car's cost in 1980 and [11] anticipates that this will increase to 50% by 2030, meaning that CAN's deficiencies are becoming harder to ignore. This has garnered the attention of both industry, where there exist companies that specialize in vehicular IDSs [1, 12], and government agencies [10, 30, 33]. With fully autonomous vehicles on the horizon, the human override disappears, leaving the passengers of compromised vehicles and other road users entirely at the attacker's mercy.

Thus, the problem statement for this thesis is: given a system that communicates through CAN, can we effectively and efficiently monitor the transmissions and detect messages from sources that should not be authorized to send them (i.e., impersonator devices).

1.2 Related Work

We present some existing literature that also uses analog-domain information to detect intrusions on CAN networks.

1.2.1 Voltage-Based

These works, similar to vProfile, also sample analog CAN voltages and exploit some inherent, immutable properties: a method by Murvay and Groza [28]; Cho and Shin's Viden [7]; Kneib and Huth's Scission [19]; a system by Choi, Jo, Woo, Chun, and Park [8]; Choi, Joo, Jo, Park, and Lee's VoltageIDS [9]; and Foruhandeh, Man, Gerdes, Li, and Chantem's SIMPLE [13]. These are often referred to as voltage-based IDSs. All of the methods that ran experiments on actual vehicles measured the CAN voltage through an existing connection in each system, such as the OBD-II port.

Murvay and Groza use a sampling rate of 2 GS/s at 12 bits for a 10 kb/s bus. They first remove noise from the input voltage data with a low pass filter and then store the output as a fingerprint for each ECU. They investigate three signal processing techniques: mean square error (MSE), convolution, and mean-value. This method's disadvantages are the high sampling rate and the high false classification rate. Murvay and Groza used an oscilloscope in their work. However, an with the specifications required for this method is expensive (e.g., the Texas Instruments ADC12D1000). They performed the MSE and convolution experiments on two sets of ten CAN transceivers and yielded an average false positive rate of 3.1% (with a standard deviation of 3.6%) and an average false negative rate of 6.0% (with a standard deviation of 17.9%).

Viden is not so much an IDS but rather a method to enhance an existing IDS by providing the ability to identify the attacking device. It uses a sampling rate of 50 kS/s for a 500 kb/s CAN bus. Viden creates multiple sets of tracking points from *non-ACK* voltage samples (voltages not taken from the acknowledgment (ACK) slot of a message) and uses them to create a *voltage profile* where each profile is unique to an ECU. When the IDS detects an intrusion, Viden must first create an attack profile using multiple malicious messages before identification can occur. This method's drawbacks include the complicated voltage profile creation process, the need to create an attack profile before classification, and the requirement of an underlying IDS.

Scission; Choi, Jo, Woo, Chun, and Park's method; and VoltageIDS extract features from the signal to train a machine learning algorithm. Scission samples a 500 kb/s CAN bus at a rate of 20 MS/s. The message is split into bits and binned into one of three groups based on certain criteria. It then uses the Relief-F algorithm supplied by the Weka 3 toolkit for time domain feature selection. Scission uses the logistic regression machine learning algorithm for training and classification. Choi, Jo, Woo, Chun, and Park's system samples at 2.5 GS/s and uses a resolution of 12 bits. They use LibXtract to calculate 40 features in both the time and frequency domain and keep the best eight time domain and best nine frequency domain features as ranked by the FEAST toolbox. Model training

uses a supervised learning algorithm. This method must first extract its 17 features from a message before it can be classified. This approach's two main disadvantages are the high sampling rate and how it can miss messages due to its long processing time. During the 1.02 ms required to extract all 17 features, the authors found that two messages can be transmitted on average, assuming a 500 kb/s bus at 50% load. The authors of VoltageIDS tried sampling rates from 20 MS/s to 2.5 GS/s and found that 250 MS/s at 8 bits for a 500 kb/s bus still produced acceptable results (though 2.5 GS/s did exhibit the highest scores). They extract and compute the mean for the dominant bit steady states and the rising and falling edges. Next, up to 20 features are computed for each of the three sections, resulting in up to 60 total features. They tried Linear Support Vector Machines and Bagged Decision Trees but found that the former performed more favorably for this application. For detection, they must first extract the same features used for training from the incoming CAN message. All three methods use elaborate pre-processing and feature selection processes that introduce non-negligible delay into the detection pipeline.

SIMPLE uses a 1 MS/s sampling rate at 12 bits for its real-vehicle tests. For a differential signal, it samples every dominant and recessive state in a CAN message eight times and then computes their sample-wise averages, resulting in 16 *features*. SIMPLE does this for messages from each ECU. Next, it performs Fisher-Discriminant Analysis to reduce the dimension of the features. It then uses a binary search algorithm to find Mahalanobis distance thresholds for each ECU based on equal error rates. For detection, incoming CAN frames undergo the same feature extraction procedure as the training data. SIMPLE then computes the Mahalanobis distance between the features of the incoming message and the stored template for the claimed transmitting device and compares the result to the previously calculated threshold.

1.2.2 Timing-Based

The other main type of system uses timing characteristics present on the network to detect anomalous behavior. We briefly summarize some existing techniques.

Gutierrez, Juliato, Ahmed, and Sastry propose a message-based time-series IDS which detects deviations in expected traffic patterns for advanced driver-assistance system messages [16]. Cho and Shin present Clock-based Instrusion Detection System (CIDS) in [6] where they estimate each ECU's accumulated clock offset (the difference between the ideal and actual arrival time of periodic messages) using the recursive least squares algorithm. In [34] and [36], the authors, similar to CIDS, fingerprint ECUs using their clock offsets. However, they additionally consider how the offsets change with fluctuating temperature

and leverage that in their algorithm. Moreno and Fischmeister attach two differential amplifiers to the bus, which allows them to locate the position of the transmitting device using the propagation delay difference [26].

1.3 Contribution

This thesis proposes vProfile, a voltage-based sender identification system for CAN networks that exploits the unique analog voltage waveform characteristics of messages sent from different ECUs. The main contribution is a technique that:

- Uses a lightweight detection algorithm that uses only a single feature
- Exhibits low misclassification rates on two real-life vehicles
- Minimizes latency since it requires analyzing only a section at the beginning of messages
- Is simple to retrofit because it needs only a connection to the CAN bus
- Has a higher potential to be implemented on less expensive embedded hardware due to its low sampling rate, resolution, and computational requirements

We also present three enhancements that can improve vProfile's detection rate and allow it to react to environmental changes.

1.4 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 gives background on CAN and concepts relating to vProfile's operation. Chapter 3 describes vProfile's implementation; Chapter 4 shows our experimental procedures and results for classification rate, environmental effects, and latency; and Chapter 5 discusses three optional enhancements to vProfile that can improve detection rates, including an online model update algorithm. Finally, Chapter 6 closes with a summary of the thesis, some limitations of vProfile, and future work.

Chapter 2

Background

This chapter presents background information on CAN as well as mathematical and theoretical concepts related to our proposed technique. The main topics we discuss are CAN's physical and transfer layers and the CAN bus voltage property that inspired vProfile.

2.1 CAN Overview

Robert Bosch GmbH released CAN in 1986, creating a protocol that is now ubiquitous not only in the automotive industry but also in aerospace and factory control, among others [29]. We can attribute CAN's popularity to its low cost, light protocol management, deterministic arbitration, and its inherent error detection and retransmission features [18, 29].

2.1.1 Physical Layer

There are two main physical layer implementations, single-wire and two-wire. We will focus on two-wire as described in ISO 11898-2 because it is the most popular variant, including in the vehicles we used for experiments. Henceforth, all mentions of CAN will mean the two-wire version.

CAN uses a differential signaling scheme where CAN High (CAN_H) and CAN Low (CAN_L) identifies the two wires. The bus can take one of two complimentary values: *dominant* or *recessive*. These represent logical '0' and logical '1', respectively, assuming

a wired-AND bus implementation (assumed for the rest of the thesis) [2]. The bus holds the recessive state when idle. Typically, CAN_H is driven to 3.5 V and CAN_L is driven to 1.5 V to send dominant and both CAN_H and CAN_L maintain 2.5 V to send recessive, but the actual state is determined by the differential voltage. This is visualized in Figure 2.1.

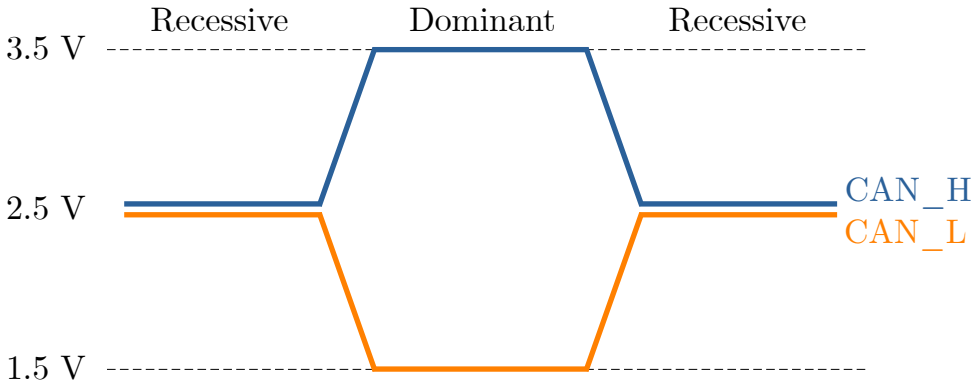


Figure 2.1: CAN bus differential signaling.

Synchronization

CAN is asynchronous and uses bit transitions to maintain synchronization. To ensure that there are a sufficient number of transitions, a bit of opposing polarity is inserted after five consecutive bits of the same value in a process known as *bit stuffing*. This is necessary because CAN uses non-return-to-zero coding [2].

2.1.2 Transfer Layer

The most recent revision of CAN is version 2.0, which specifies two different message formats: the standard frame format with an 11-bit identifier (ID) (CAN 2.0A) and the extended frame format with a 29-bit ID (CAN 2.0B) [2]. Because vProfile testing occurred on vehicles that employ the SAE J1939 protocol, which is based on CAN 2.0B, this thesis explains only the extended frame format and all future mentions of data fields are with respect to J1939 [21].

Message Frame Format

CAN specifies four frame types:

- Data frames
- Remote frames
- Error frames
- Overload frames

We focus on the data frame as it is arguably the most important type for intrusion detection. Figure 2.2 shows the format of an extended frame and Table 2.1 describes each field in detail.

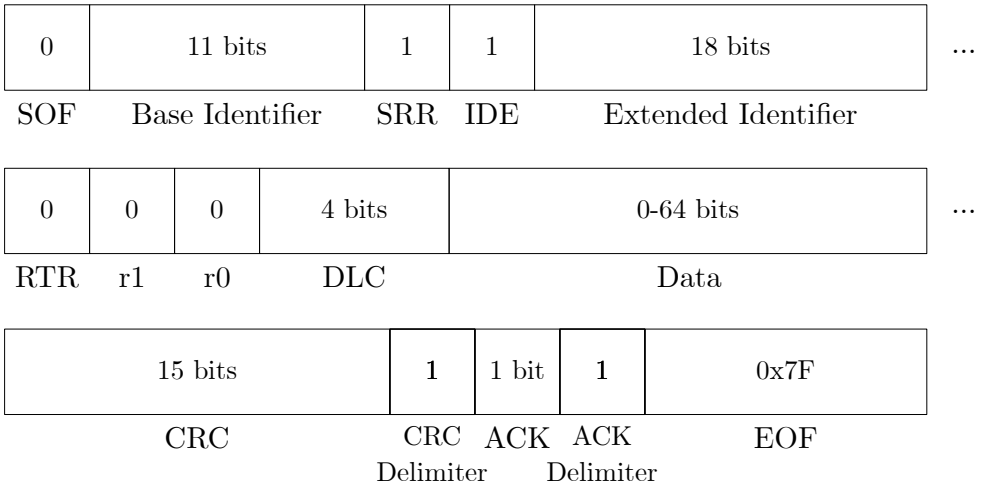


Figure 2.2: CAN extended frame format.

Each ID can map to only a single ECU, but each ECU can send multiple IDs. Thus, the ID can uniquely identify the sender of a legitimate message. The source address (SA) in the last 8 bits of the 29-bit ID exhibits this property, so vProfile needs only the SA to detect intrusions.

Arbitration

Whenever the bus is free, any ECU may begin transmitting. If more than one ECU starts transmitting messages simultaneously, the conflict is resolved by bitwise arbitration using the *arbitration field*, consisting of the Base ID, SRR, IDE, Extended ID, and RTR fields. This arbitration technique guarantees that neither information nor time is lost, making

Field	Length (bits)	Description
Start-of-Frame (SOF)	1	Marks the start of a frame and must be ‘0’.
Base Identifier	11	First part of the unique sender identifier.
Substitute Remote Request (SRR)	1	Substitutes the RTR bit in standard frames. Must be ‘1’.
Identifier Extension Bit (IDE)	1	Specifies the frame format. Must be ‘1’ for extended frames.
Extended Identifier	18	Second part of the unique sender identifier.
Remote Transmission Request (RTR)	1	Specifies the frame type. Must be ‘0’ for data frames.
Reserved Bits (r0, r1)	2	Reserved for future use. Must both be sent ‘0’ but receivers accept either value.
Data Length Code (DLC)	4	Length of the data (0-8 octets).
Data	0-64	Transmitted data.
Cyclic Redundancy Check (CRC)	15	Redundancy check using Bose–Chaudhuri–Hocquenghem (BCH) code.
CRC Delimiter	1	Must be ‘1’.
ACK Slot	1	Sent ‘1’ and any receiver can assert ‘0’ for correctly received, valid messages.
ACK Delimiter	1	Must be ‘1’.
End-of-Frame (EOF)	7	Marks the end of the frame and must all be ‘1’.

Table 2.1: CAN extended frame fields.

it suitable for safety-critical systems. During arbitration, every transmitter compares the value of the bit they are transmitting with the bus' value. If the values are equal, the sender may continue. If the values do not match, however, then that unit has lost arbitration and must stop transmission immediately [2]. Because the bus is wired-AND, it will take the value '0' if at least one '0' is currently being transmitted. Figure 2.3 illustrates this, where ECU 1 loses to ECU 0 at bit 7.

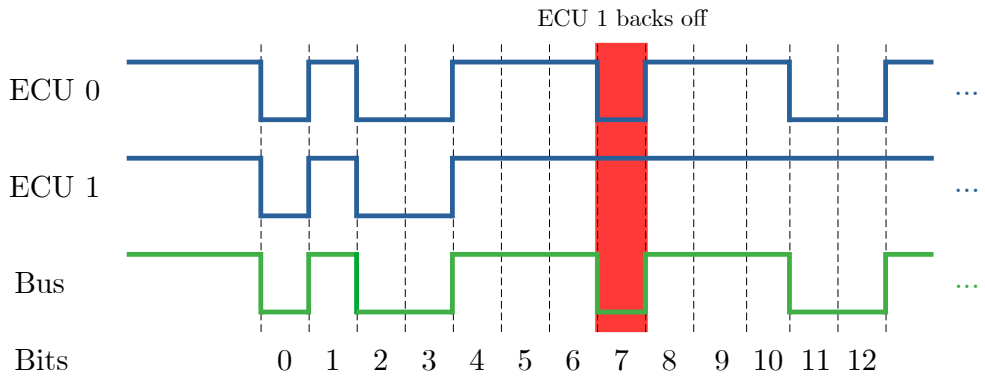


Figure 2.3: CAN bus arbitration example where ECU 1 loses to ECU 0 during bit 7.

Because '0' takes precedence over '1', smaller values preempt larger ones, meaning that CAN has an inherent message priority scheme. Figure 2.4 shows the format of the J1939 ID and Table 2.2 describes each field in detail. We see that the Priority field first determines a message's priority, then its Parameter Group Number (PGN), and finally its SA. For example, the SA of the Engine Control Module (ECM) is usually '0' and the PGN for messages about engine speed is also commonly '0'.

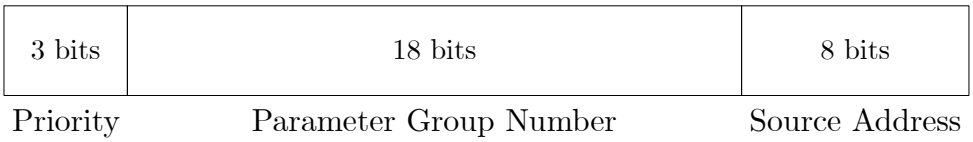


Figure 2.4: J1939 ID format.

2.2 vProfile Preliminaries

This section will describe some prerequisite concepts behind vProfile's theory.

Field	Length (bits)	Description
Priority	3	Controls the message's priority for arbitration.
PGN	18	Indicates the type of message (e.g., cruise control).
SA	8	Indicates the origin ECU (e.g., ECM).

Table 2.2: J1939 ID fields.

2.2.1 Immutable ECU Property

Minute inconsistencies in manufacturing introduce random physical differences in each ECU that are unpredictable and uncontrollable. These variations suggest that each ECU exhibits unique electrical characteristics. Thus, we can not only uniquely identify ECUs based on these electrical properties, but they are practically impossible for an adversary to imitate [14]. Figure 2.5 shows the unique properties of ECUs where we plot the rising and falling edges (together with the steady state between them, these form an *edge set*) of 200 voltage traces from two ECUs on a 2006 Sterling Acterra truck.

Though the traces originate from multiple SAs, it is clear that there are two distinct waveforms, one for each ECU. The authors of [9] also use edge sets in their detection algorithm because of their unique properties inherent to the transmitting device.

2.2.2 Distance Metrics

vProfile uses a distance metric to determine how similar the waveform of an incoming message is to what it should be according to its SA. We describe the metrics that we explored below.

Euclidean Distance

The first iteration of vProfile used Euclidean distance, a well-known and widely-used distance metric, and measures the distance between two points in Euclidean space. For an

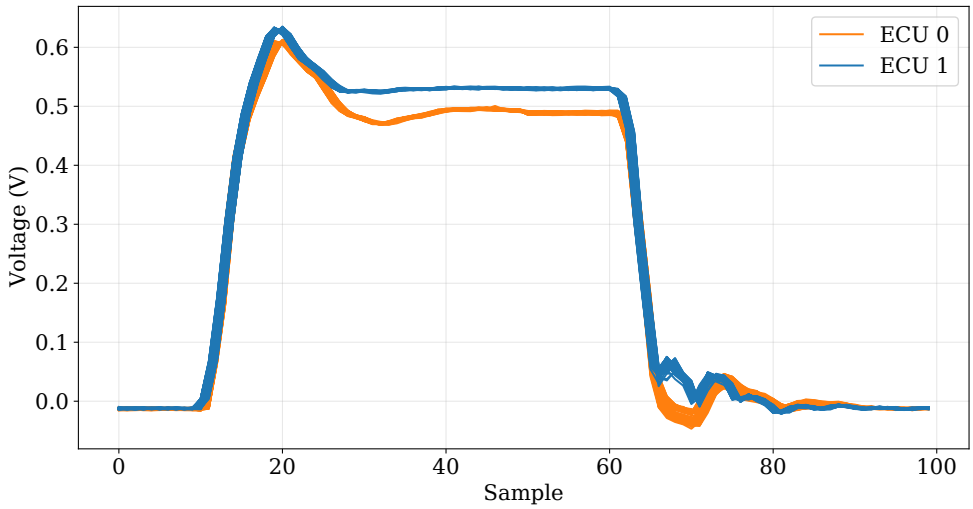


Figure 2.5: Voltage differences of messages from two different ECUs and the similarities of messages from the same ECU.

N-dimensional dataset, the Euclidean distance between points $\mathbf{x}$ and $\mathbf{y}$ is defined by

$$D_{\mathrm{E}}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^{\mathrm{T}} (\mathbf{x} - \mathbf{y})} \tag{2.1}$$

where $\mathbf{x} = (x_1, x_2, ..., x_N)^{\mathrm{T}}$ is and $\mathbf{y} = (\mu_1, \mu_2, ..., \mu_N)^{\mathrm{T}}$. For vProfile, the number of samples in an edge set defines the dimensionality. Taking Figure 2.5 as an example where each edge set is 100 samples, the dataset would have 100 dimensions.

Mahalanobis Distance

Mahalanobis distance measures the distance between a point and a distribution d. For an N-dimensional dataset, Mahalanobis distance is defined by

$$D_{\mathrm{M}}(\mathbf{x}, d) = \sqrt{(\mathbf{x} - \boldsymbol{\mu}_d)^{\mathrm{T}} \Sigma_d^{-1} (\mathbf{x} - \boldsymbol{\mu}_d)} \tag{2.2}$$

where $\mathbf{x} = (x_1, x_2, ..., x_N)^{\mathrm{T}}$ is the observation, $\boldsymbol{\mu}_d = (\mu_{d,1}, \mu_{d,2}, ..., \mu_{d,N})^{\mathrm{T}}$ is the mean of the distribution, and Σ_d is the covariance matrix of the distribution [23]. Note that (2.2) reduces to the Euclidean distance between the observation and distribution mean if Σ_d is the identity matrix.

Chapter 3

vProfile Overview

In this chapter, we provide a detailed description of vProfile. We explain our threat model, including the two types of attackers that we detect and their capabilities. We then describe vProfile's three operational stages from how it acquires the CAN voltages to model training and intruder detection.

3.1 Threat Model

Based on how an adversary could gain access to the CAN bus, we consider two types of intruders:

- *Hijack intruders*
- *Foreign intruders*

Hijack intruders gain bus access by hijacking an existing ECU and can then send messages crafted by the attacker using that ECU. Foreign intruders gain access by connecting a new device to the bus, which they can then use to send crafted messages. We assume that the foreign device did not exist during model training.

Both hijack and foreign intruders can craft messages that appear legitimate at the binary level, including the SA and other metadata bits. Due to the broadcast nature of CAN, there is no need for an intruder to identify themselves with a unique SA to receive responses. Intruders that transmit using a unique SA can be trivially detected. However,

those that transmit under the ID of an existing, legitimate ECU cannot be detected within the confines of the CAN protocol; this necessitates using information from a different domain.

3.2 Functionality

This section describes the three main stages of vProfile:

- *Preprocessing*
- *Training*
- *Detection*

Both Training and Detection require the Preprocessing stage.

3.2.1 Preprocessing

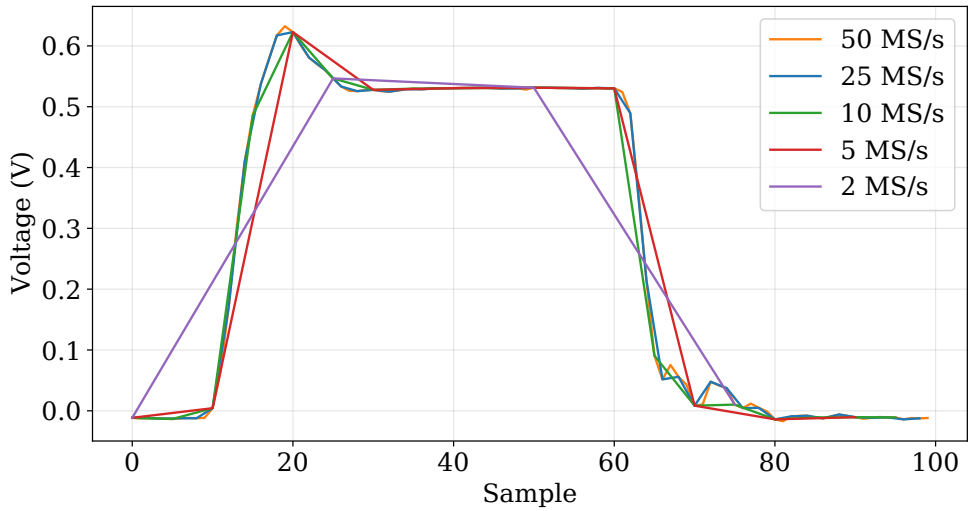
During Preprocessing, we sample the CAN bus voltage and then extract an edge set from every message we encounter. Thus, we can separate Preprocessing into two steps: *Sampling* and *Edge Set Extraction*.

Sampling

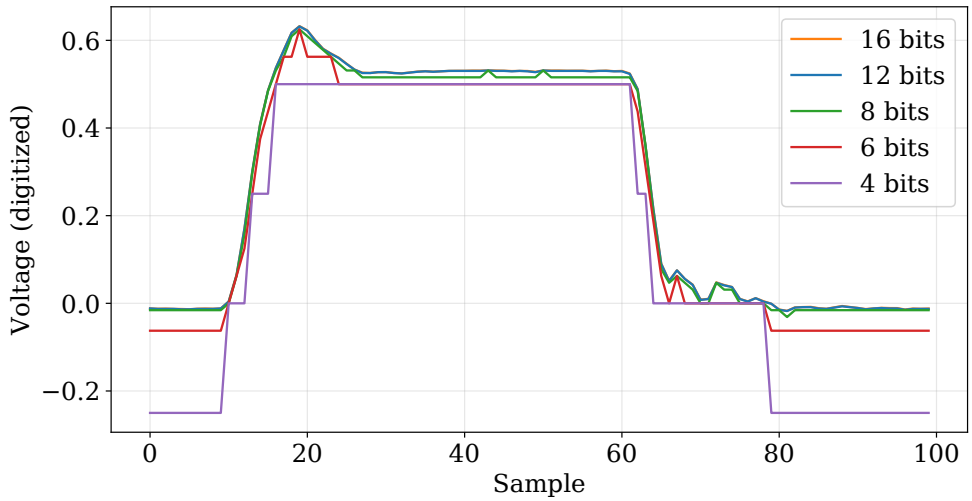
First, we must sample the bus to read its voltage level. As observed in Section 1.2, there are many choices for sampling rate and resolution. Ideally, we want as much information as possible, but there is a point of diminishing returns. The drawback of increased detail is an additional computational cost, which may be infeasible on embedded processors. Sampling rates and resolutions that are too low result in a loss of identifying voltage waveform characteristics, simplifying attacks on vProfile for foreign intruders as it facilitates imitating the waveform of target ECUs.

Figure 3.1 shows the effects of changing the sampling rate or resolution while the other is held constant using an edge set from a 2006 Sterling Acterra with a 250 kbps bus. We laterally scale the traces for easier comparison in Figure 3.1a. For Figure 3.1b, we drop the least significant bits for the lower resolutions and the negative voltages are an artifact

of the conversion from offset binary to volts. We observe that a sampling rate of around 10 MS/s and a resolution of 8 bits is the limit before we start deviating significantly from the original shape.



(a) Reducing the sampling rate with a constant resolution of 16 bits.



(b) Reducing the resolution with a constant sampling rate of 50 MS/s.

Figure 3.1: Effects of reducing the sampling rate and resolution on an edge set.

A good choice for sampling rate and resolution depends on the ECU voltage characteristics of the system on which vProfile runs. For example, a system with many ECUs with similar edge set waveforms would benefit from a higher sampling rate and resolution. In comparison, a system with fewer ECUs and distinct waveforms could tolerate a lower sampling rate and resolution. The experiments in Section 4.3 helped us select sampling rates and resolutions for our test vehicles.

Edge Set Extraction

Next, we extract edge sets from the sampled bus voltage. Per Section 2.2.1, the interesting signal characteristics for vProfile reside in edge sets. In the absence of ECU malfunctions, these characteristics do not change throughout a transmission, so only one edge set is required for every message [8]. Minimizing the delay from sampling to detection means that the edge set should be extracted as early in the transmission as possible. Because of the possibility of collisions during arbitration, any bits within the arbitration field are considered unstable. Thus, we use the first edge set after the arbitration field. Also, the message's SA is decoded and paired with its edge set because we would lose that information otherwise. We present an algorithm to extract the SA and edge set from the voltage trace of a CAN message.

First, we define some constants:

- *Bit width*: The number of samples per bit of the message. For a sampling rate of 10 MS/s on a 250 kb/s bus, we found the bit width to be roughly 40 samples/bit.
- *Bit threshold*: The value that indicates the presence of an edge. It should approximately horizontally bisect the rising edge. Sections of a trace that start below this threshold and end above it are rising edges and the opposite is true for falling edges. For Figure 2.5, a threshold of 38,000 is a good starting point.
- *Prefix length*: The number of samples before the threshold point to extract. We found two samples to be sufficient for a sampling rate of 10 MS/s on a 250 kb/s bus.
- *Suffix length*: The number of samples after the threshold point to extract. We found 14 samples to be sufficient for a sampling rate of 10 MS/s on a 250 kb/s bus

We use the prefix and suffix lengths to minimize redundant steady-state data while capturing all of the rising and falling edges since the steady-state values are very stable,

assuming regular system operation. We also use the lengths to normalize the edge set length as there could be up to five bits of the same value before a falling edge.

The algorithm traverses the message while counting the number of bits and ignores any stuffing bits. Referencing Table 2.1 and assuming that SOF is bit 0, we know that the SA corresponds to bits 24 to 31 and bit 33 is the first bit after the arbitration field. To maintain synchronization while ingesting the message, we align ourselves to the center of every edge we encounter. Once we reach bit 33, we iterate until the first rising edge and extract a prefix-length number of samples before and suffix-length number of samples after. We then find the falling edge and extract the same number of samples. Algorithm 1 summarizes the extraction procedure.

3.2.2 Training

The Training stage of vProfile uses the collection of edge sets and their corresponding SAs from Preprocessing to train a model. Since an ECU can send messages using multiple SAs, we must cluster the edge sets by their SA, where a cluster represents all of the messages sent by an ECU. If one is fortunate enough to be provided with a database containing the target system's ECUs and their valid SAs, it is straightforward to cluster using the database as a lookup table (LUT). If one does not have access to such information, we can group the data by SA and then calculate the distance between the edge sets of every pair of SAs and cluster those with the smallest distance.

The model contains each cluster's mean and a LUT that maps valid SAs to their cluster. The model also needs a detection threshold so that vProfile can handle traces that do not belong to any cluster. We determine this threshold by finding the largest distance from an edge set to its cluster's mean for each cluster. Algorithm 2 summarizes the training procedure.

3.2.3 Detection

Given an edge set to classify, vProfile uses the cluster-SA LUT first to check if the observed SA exists in the system. If it exists, vProfile then uses the LUT to get the cluster according to the reported SA (the *expected* cluster), predicts the cluster based on the edge set's distance to the cluster means (the *predicted* cluster), and compares the two results. If the expected and predicted clusters differ, vProfile raises an alarm. If they match, vProfile compares the minimum calculated distance to the detection threshold with some configurable margin added to account for additional deviation. If the distance is greater than

the sum of the threshold and margin, vProfile raises an alarm. Otherwise, the message is considered legitimate. Selecting an appropriate margin is critical to vProfile's success. A margin that is too small can result in more false positives and a margin that is too large can cause additional false negatives. For messages from ECUs in the training dataset, vProfile can also determine the attack's origin from the predicted cluster. Algorithm 3 summarizes the detection procedure.

Algorithm 1: vProfile edge set extraction procedure.

```
1  EXTRACT(msg) begin
2      bitValues = Array[0]; // decoded message bits
3      prevBit = 0; // keep track of the previous bit
4      sameBitCount = 0; // number of consecutive bits of the same value
5      bitCount = 0; // number of bits excluding stuffed bits
6      pos = FINDSOF(msg) + (bitWidth / 2); // go to middle of first bit
7      bitValues.append(GETBITVALUE(msg[pos]));
8      while (pos + bitWidth) < msg.size do
9          pos += bitWidth;
10         bit = GETBITVALUE(msg[pos]);
11         if bit ≠ prevBit then
12             pos = ALIGNTOEDGECENTER(msg, pos) + (bitWidth / 2);
13             prevBit = bit;
14             sameBitCount = 1;
15         else
16             sameBitCount += 1;
17         end
18         // ignore stuffed bits
19         if sameBitCount == 5 then
20             sameBitCount = 1;
21             continue;
22         end
23         bitValues.append(bit);
24         bitCount += 1;
25         if bitCount == 31 then
26             sa = EXTRACTSA(bitValues);
27         else if bitCount == 33 then
28             edgeSet = EXTRACTEDGESET(msg, pos);
29             break;
30         end
31     end
32     return sa, edgeSet;
33 end
```

```
33 GetBitValue(voltageValue) begin
34 |  if voltageValue ≥ threshold then
35 |  |  return 0;
36 |  end
37 |  return 1;
38 end
39 ExtractEdgeSet(msg, pos) begin
40 |  edgeSet = Array[0];
41 |  // find the next rising edge
42 |  while msg[pos] < threshold do
43 |  |  pos += 1;
44 |  end
45 |  while msg[pos] > threshold do
46 |  |  pos += 1;
47 |  end
48 |  for i from (pos − prefixLen) to (pos + suffixLen) do
49 |  |  edgeSet.append(msg[i]);
50 |  end
51 |  pos += (bitWidth / 2);
52 |  while msg[pos] < threshold do
53 |  |  pos += 1;
54 |  end
55 |  for i from (pos − prefixLen) to (pos + suffixLen) do
56 |  |  edgeSet.append(msg[i]);
57 |  end
58 |  return edgeSet;
59 end
```

Algorithm 2: vProfile training procedure.

```
1  TRAIN(edgeSets) begin
2      if fortunate then
3          clustSaLut = CLUSTERBYLUT(edgeSets);
4      else
5          saLut = GROUPBYSA(edgeSets);
6          saMeans = GETMEANS(saLut);
7          clustSaLut = CLUSTERBYDIST(saLut, saMeans);
8      end
9      clustMeans = GETMEANS(clustSaLut);
10     clustMaxDists = Array[clustSaLut.size];
11     foreach clust ∈ clustSaLut do
12         clustMaxDists[clust] = max(CALCDISTANCE(clust, edgeSets, clustMeans));
13     end
14     model = (clustSaLut, clustMeans, clustMaxDists);
15     return model;
16 end
```

Algorithm 3: vProfile detection procedure.

```
1  DETECT(model, testEdgeSet) begin
2      sa = testEdgeSet.sa;
3      if sa ∉ model.clustSaLut then
4          return ANOMALY;
5      end
6      expClust = GETCLUSTER(model.clustSaLut, sa);
7      minDist = ∞;
8      foreach clust ∈ model.clustSaLut do
9          dist = CALCDISTANCE(clust, testEdgeSet, model.clustMeans);
10         if dist < minDist then
11             predClust = clust;
12             minDist = dist;
13         end
14     end
15     if (expClust ≠ predClust) ||
        (minDist > (model.clustMaxDists[predClust] + margin)) then
16         return ANOMALY;
17     end
18     return OK;
19 end
```

Chapter 4

Experiments

This chapter begins with our testing setup on two commercial vehicles and explains the three types of experiments we use to assess vProfile's detection capabilities. Next, we present results from the three experiment types using Euclidean and Mahalanobis distance and with downsampled and lower resolution data. Finally, we show the effects of vehicle temperature and battery voltage on vProfile.

4.1 Experimental Setup

We performed experiments on two vehicles, a 2016 Peterbilt 579 truck shown in Figure 4.1 (henceforth known as *Vehicle A*) and a vehicle belonging to an industry partner (henceforth known as *Vehicle B*). Due to our partner's confidentiality requirements, we omit vehicle-specific details for Vehicle B and are limited in what we can disclose besides the final test results. Both vehicles have a 250 kb/s CAN bus using the SAE J1939 protocol. For test repeatability, we recorded the CAN bus traffic of each vehicle and replayed it into vProfile. Vehicle A was stationary during the capture and Vehicle B's driver performed various maneuvers, such as hard acceleration, sudden braking, gear shifting, and steering.

We ran three types of experiments:

- *False positive test*
- *Hijack imitation test*
- *Foreign device imitation test*

Figure 4.1: 2016 Peterbilt 579 truck that we subjected to experimentation [27].

For false positive tests, we train vProfile on messages from all of the ECUs and then replay the CAN bus captures as-is into vProfile. For hijack imitation tests, we also train vProfile on messages from all of the ECUs but, when we replay the data, we change each message's SA in software to one that belongs to another cluster with a 20% chance. This experiment simulates a test where every ECU can imitate every other ECU. For foreign device imitation tests, we pick two ECUs with the most similar voltage profiles and remove the former's messages from the training set and then replay data into vProfile while having it imitate the latter. Figure 4.2 shows each ECU's voltage waveforms for Vehicle A. We ignore the case where a message has an unknown SA since detection is trivial.

To sample the CAN bus voltage, we use a board that our colleagues developed for vehicular side-channel projects. Figure 4.3 shows the board's block diagram where the input is connected to the vehicle's OBD-II port.

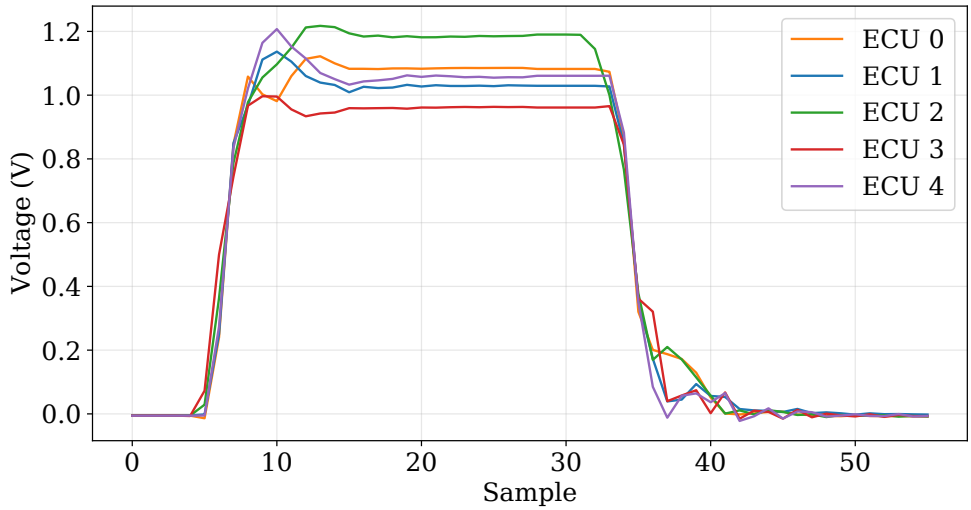


Figure 4.2: Voltage profiles for Vehicle A's ECUs.

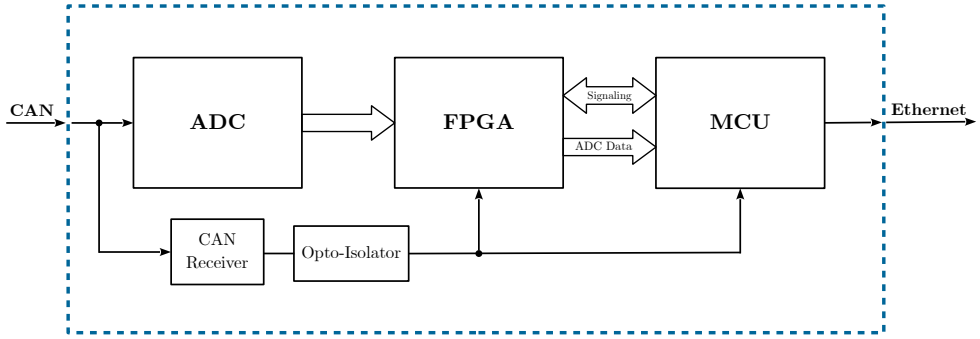


Figure 4.3: Custom board we used to sample CAN voltages.

4.2 Distance Metric

First, we chose a distance metric for vProfile. We explored using Euclidean and Mahalanobis distances. We captured data from Vehicle A at 20 MS/s and 16 bits and data from Vehicle B at 10 MS/s and 12 bits. This discrepancy exists because we used an AlazarTech PCI digitizer card on Vehicle A and our custom hardware on Vehicle B since we did not have access to the latter during the preliminary stages of vProfile.

4.2.1 Euclidean Distance

We tried Euclidean distance first because we thought we could achieve accurate clustering with Vehicle A's visually distinct voltage profiles. For the foreign device imitation test, we selected the two ECUs with the smallest Euclidean distance to imitate each other. For Vehicle A, these are ECUs 1 and 4 with a Euclidean distance of 3634.96. The next smallest distance is 6671.10 between ECUs 0 and 1. Table 4.1 shows the confusion matrices for each of the three types of experiments on Vehicle A. We selected the margin to maximize the accuracy for the false positive test and the F-score for the other two tests.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	0	0
	Normal	51	841190

(a) False positive test.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	168014	0
	Normal	36	673191

(b) Hijack imitation test.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	128	394240
	Normal	49	446824

(c) Foreign device imitation test.

Table 4.1: Confusion matrices for experiments on Vehicle A using Euclidean distance.

The false positive test yields an accuracy of 0.99994, the hijack test has an F-score of 0.99989, and the foreign device test's F-score is 0.00065. If we increase the margin to remove all false positives, resulting in perfect accuracy and F-score for the first two tests, the F-score for the foreign device test drops to 0.00013. Table 4.2 shows the results when we run the same tests on Vehicle B, where we selected the margin the same way as for Vehicle A.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	0	0
	Normal	38813	301833

(a) False positive test.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	66841	1041
	Normal	31059	241705

(b) Hijack imitation test.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	33500	90455
	Normal	1293	215398

(c) Foreign device imitation test.

Table 4.2: Confusion matrices for experiments on Vehicle B using Euclidean distance.

Vehicle B exhibits considerably more false positives overall, with an accuracy of 0.88606 for the false positive test and F-scores of 0.80637 and 0.42205 for the hijack and foreign device tests, respectively. We could not find a margin that removed all false positives. This

performance degradation is likely due to Vehicle B having more ECUs with less distinct voltage profiles.

Next Steps

While experimenting with methods to improve vProfile's performance, we noticed that the variance of the edges was much greater than that of the overshoot and steady state. However, their contribution to the voltage profile is less significant than the latter two. Figure 4.4 shows the standard deviation of several sample indices for ECU 0 in Figure 2.5.

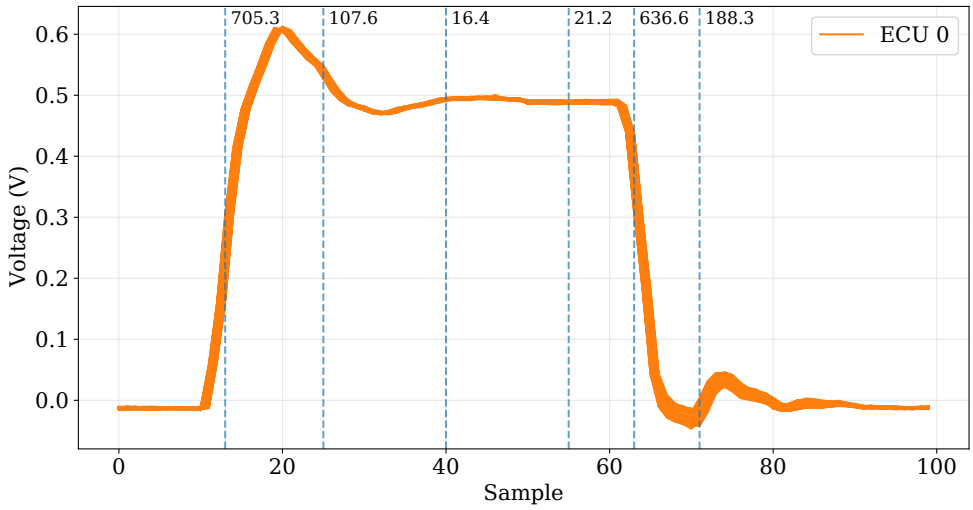


Figure 4.4: Standard deviation of some sample indices indicated by the dashed, vertical, blue lines for ECU 0 in Figure 2.5.

We see that the two edges have significantly higher standard deviations, despite their limited contribution to ECU 0's voltage profile, as they are very similar to the edges of ECU 1 in Figure 2.5. We can also observe this in Figure 4.2. Given our findings, we wanted to try a weighted distance metric that accounts for the inconsistent variance across an edge set, which led us to Mahalanobis distance.

4.2.2 Mahalanobis Distance

Similar to the Euclidean distance tests, we selected the two ECUs with the smallest Mahalanobis distance for the foreign device imitation test. For Vehicle A, these are ECUs

1 and 4 with a Mahalanobis distance of 115.17. The next smallest distance is 165.70 between ECUs 1 and 3. Table 4.3 shows the confusion matrices for each of the three types of experiments on Vehicle A. We selected the margin using the same way as the Euclidean distance tests.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	0	0
	Normal	2	841239

(a) False positive test.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	168116	0
	Normal	2	673123

(b) Hijack imitation test.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	394368	0
	Normal	1	446872

(c) Foreign device imitation test.

Table 4.3: Confusion matrices for experiments on Vehicle A using Mahalanobis distance.

We observe better performance with Mahalanobis distance where the false positive test received an accuracy of 1.00000 and the hijack and foreign device tests achieved F-scores of 0.99999 and 1.00000, respectively. If we increase the margin to eliminate false positives, the F-score of the foreign device test decreases to 0.84289. Table 4.4 shows the results when we run the same tests on Vehicle B. The performance improvement is more drastic,

with an accuracy of 1.00000 for the false positive test and F-scores of 0.99999 and 1.00000 for the hijack and foreign device tests, respectively. If we increase the margin to remove the single false positive, the F-score for the foreign device test drops to 0.61521.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	0	0
	Normal	1	340644

(a) False positive test.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	68591	0
	Normal	1	272053

(b) Hijack imitation test.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	123955	0
	Normal	1	216689

(c) Foreign device imitation test.

Table 4.4: Confusion matrices for experiments on Vehicle B using Mahalanobis distance.

The advantage of using the Mahalanobis distance for CAN voltage traces over other distance metrics comes from its use of a covariance matrix. Although each waveform from the same ECU has similar overall characteristics, they are not identical. As seen in Figure 4.4, the variance can differ significantly across samples, even for neighboring samples and especially at the rising and falling edges. The covariance matrix captures the correlation between a sample and its neighbors, which plays a vital role in distinguishing

traces from different sources. This results in a more accurate clustering metric for this use case. To visualize the effect of the matrix, we make a deeper comparison between Mahalanobis and Euclidean distance.

Consider Figure 4.5, which includes the cluster means of Figure 2.5 and a test edge set originating from ECU 0 (henceforth referred to as E_{test}). Table 4.5 shows the Euclidean and Mahalanobis distances from E_{test} to the means of both ECUs. We observe that both metrics correctly indicate that E_{test} is more similar to edge sets from ECU 0. However, the Mahalanobis distances' quotient is an order of magnitude larger than the Euclidean quotient. This difference signifies that Mahalanobis distance sees E_{test} as considerably more similar to ECU 0 than ECU 1. In contrast, Euclidean distance sees that it is only slightly more similar to ECU 0.

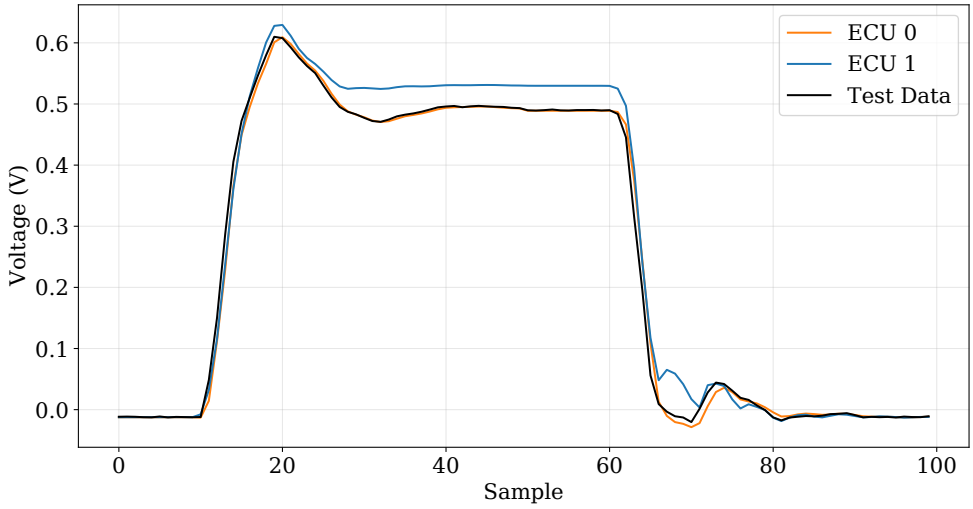


Figure 4.5: Cluster means of two different ECUs and a test edge set from ECU 0.

Updates to vProfile

Because we want to use Mahalanobis distance, we must calculate and store the data's covariance matrices where appropriate. In the training algorithm, this occurs when we cluster SAs. We must also include each cluster's covariance matrix in the model for use during detection.

Metric	Distance to ECU 0	Distance to ECU 1	Quotient
Euclidean	232740	514284	2.21
Mahalanobis	9.90	182.94	18.48

Table 4.5: Mahalanobis and Euclidean distance from a edge set belonging to ECU 0 to the means of ECUs 0 and 1 in Figure 4.5.

4.3 Sampling Rate and Resolution

Next, we determine an appropriate sampling rate and resolution for our vehicles. We want to balance classification performance with computational load.

4.3.1 Vehicle A

We downsampled and reduced the resolution of Vehicle A's 20 MS/s and 16-bit data in software and then ran the three tests. We selected a margin that maximized the accuracy of the false positive test and F-score of the hijack and foreign device tests for each sampling rate and resolution combination. We tried 20 MS/s, 10 MS/s, 5 MS/s, and 2.5 MS/s sampling rates with 16-bit, 14-bit, 12-bit, and 10-bit resolutions. We could not reduce the resolution past 10 bits since it resulted in singular covariance matrices. Table 4.6 shows the results for every combination of sampling rate and resolution.

Though very slight, we do see a drop in performance with lower sampling rates and resolutions, but more so for sampling rates, likely due to the voltage profiles not changing significantly until resolutions lower than 8 bits, per Figure 3.1. These findings suggest that Vehicle A can tolerate much lower sampling rates and resolutions. However, we had to lower the margin during each experiment to maintain the high scores, starting with a margin of 25.0 for the 20 MS/s tests and ending with a margin of 1.0 for the 2.5 MS/s tests. Since we do not consider negative margins, it leaves little room for fine-tuning. We decided to use 10 MS/s at 12 bits because it provides ample flexibility and does not impact vProfile's detection rate.

4.3.2 Vehicle B

As mentioned at the beginning of Section 4.2, we used only our custom hardware on Vehicle B, meaning that we do not have data at sampling rates and resolutions above 10 MS/s and 12 bits. We also get singular covariance matrices when we use resolutions 10 bits and lower. Therefore, Table 4.7 shows the effects of downsampling to 5 MS/s and 2.5 MS/s. As with Vehicle A, we adjust the margin to maximize the accuracy of the false positive tests and the F-scores of the hijack and foreign device tests.

We see a more pronounced performance drop on Vehicle B with lower sampling rates, but vProfile still performs very well in all cases with scores over 0.999. These results suggest that Vehicle B can tolerate lower sampling rates and confirm that our selection of 10 MS/s at 12 bits works well, even in more challenging conditions.

4.4 Effect of Environmental Variability

Prior research suggests that environmental changes, such as temperature and battery voltage, can affect the CAN bus voltage and other analog message characteristics [7,9,13,34,36]. Our tests determine how severe the impact is on vProfile. We performed the experiments on Vehicle A.

4.4.1 Temperature

During this test, the battery voltage should be as stable as possible. Because the alternator supplies power when the engine is running, the voltage should not experience much fluctuation while the vehicle is idling and, indeed, the battery stayed at $13.60\,\mathrm{V} \pm 0.03\,\mathrm{V}$ for all three trials.

Procedure

We let Vehicle A idle and checked the temperature of the ECM (corresponding to ECU 0) every two minutes while recording the bus traffic. We chose the ECM because it is mounted to the engine block and is easily accessible. We tested temperatures from $-5\,^{\circ}\mathrm{C}$ to $25\,^{\circ}\mathrm{C}$. We then separated the data from all trials into groups of $5\,^{\circ}\mathrm{C}$ increments and trained a model using just the data from $-5\,^{\circ}\mathrm{C}$ to $0\,^{\circ}\mathrm{C}$. Next, we replayed the unmodified data from $0\,^{\circ}\mathrm{C}$ to $25\,^{\circ}\mathrm{C}$ into vProfile.

Results

Table 4.8 shows the confusion matrix for the described experiment. All four false positives occurred between 20 °C and 25 °C. If we add data collected at 20 °C during a fourth trial to the training set, all false positives disappear.

To show the impact of temperature on the waveforms, we use the −5 °C to 0 °C model and then calculate the mean Mahalanobis distance and 99% confidence interval between it and each 5 °C group. Figure 4.6 visualizes the percent change from the training data for the five ECUs. The distance increases with increasing temperature for all ECUs, with a drastic increase for ECUs 0 and 2 and more subtle increases for the others. We theorize that the temperature of some ECUs did not rise much throughout the experiments, resulting in minimal changes in their distances.

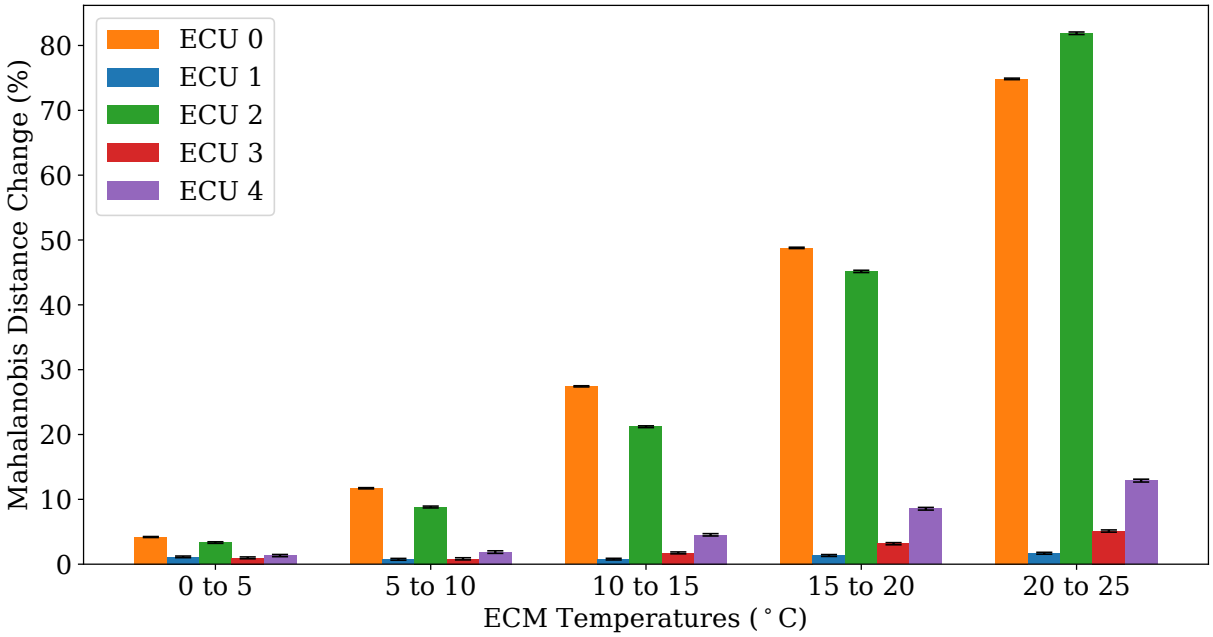


Figure 4.6: Percent delta of Mahalanobis distance means and 99% confidence intervals between training data from −5 °C to 0 °C and test data from 0 °C to 25 °C.

4.4.2 Voltage

During this test, the vehicle's temperature should be as stable as possible. To minimize temperature fluctuations, all tests were performed in a shaded area within an hour on the same day. Measuring the temperature of the vehicle before and after each of the five trials shows that we maintained $28.4\,^{\circ}\text{C} \pm 0.4\,^{\circ}\text{C}$.

Ideally, we would control the battery voltage and observe its effect on the bus voltage. To accomplish this, we wanted to switch the batteries with an adjustable power supply, but our equipment could not meet the current demands of Vehicle A. Thus, our experiment determines if using vehicle functions that draw a lot of power will affect the bus voltage. As mentioned in Section 4.4.1, the alternator supplies power when the engine is running, so we performed testing when Vehicle A is in accessory mode and is operating entirely off of the batteries. We also consider whether starting the engine affects the bus voltage.

Procedure

Data capture starts with Vehicle A in accessory mode. We then turned on and off all of the interior and exterior lights, the air conditioning (A/C), and then both together. We recorded the battery voltage before and after each trial.

Next, we separated the data by event and trained a model using data captured when Vehicle A was in accessory mode with nothing else turned on. We then replayed the unmodified data from the other events into vProfile.

Results

The battery voltage before each trial was $12.61\,\text{V} \pm 0.02\,\text{V}$ and $12.54\,\text{V} \pm 0.01\,\text{V}$ after each trial, not including when we ran the engine which measured $13.60\,\text{V} \pm 0.02\,\text{V}$. Table 4.9 shows the confusion matrix for the described experiment. Any changes to the bus voltage did not affect vProfile as it exhibited a perfect detection rate.

We also calculate the percent change of the Mahalanobis distances for this experiment. Figure 4.7 shows the percent delta when we trained the model on accessory mode data from each trial. We see that the high-power functions have a minimal impact on the bus voltage, even resulting in a decreased distance for some instances. We note that the most significant increase in distance occurred during and after the most current-consuming event when both the lights and A/C were on, suggesting that very high loads can affect the CAN bus voltage.

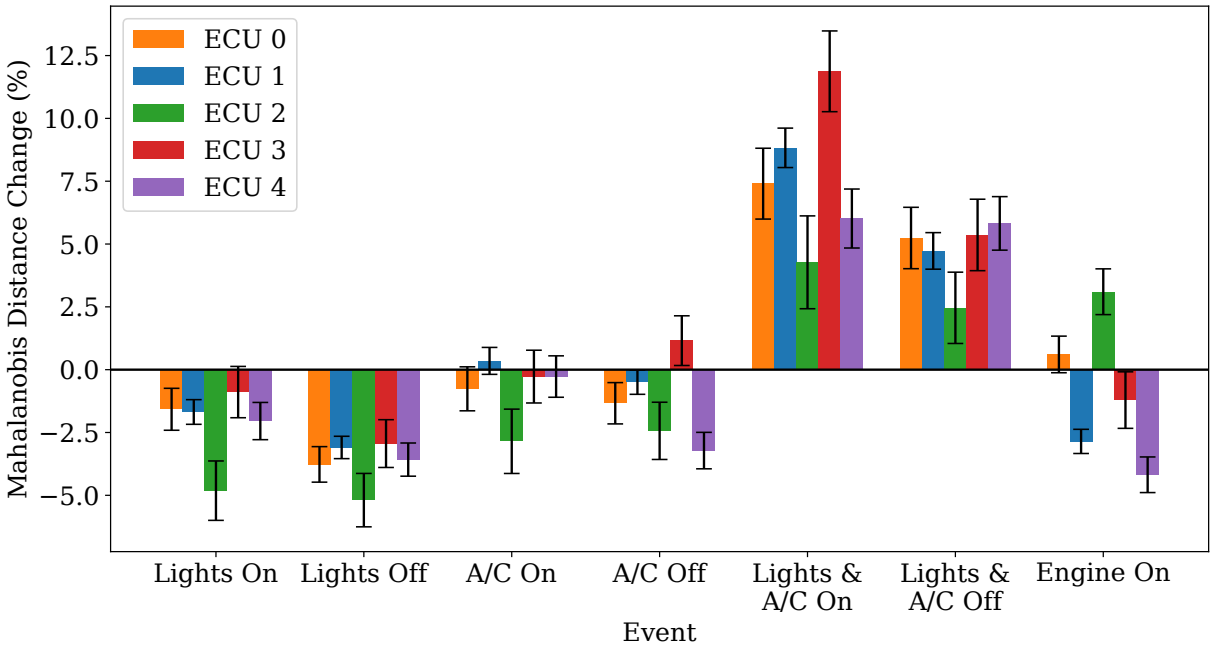


Figure 4.7: Percent delta of Mahalanobis distance means and 99% confidence intervals between training data from accessory mode and test data from high-power vehicular functions.

We observe an interesting phenomenon when running the experiment with a model trained on accessory mode data from the first trial. Figure 4.8 shows the Mahalanobis distance percent delta for this case using the accessory mode event of each trial, as the other events exhibit similar behavior. There is an overall increase in distance over time. We believe that a rise in temperature might have caused this. Despite our measurements showing minimal deviation in the vehicular temperature during the experiment, it is possible that the bus, itself, or other wiring experienced a temperature increase.

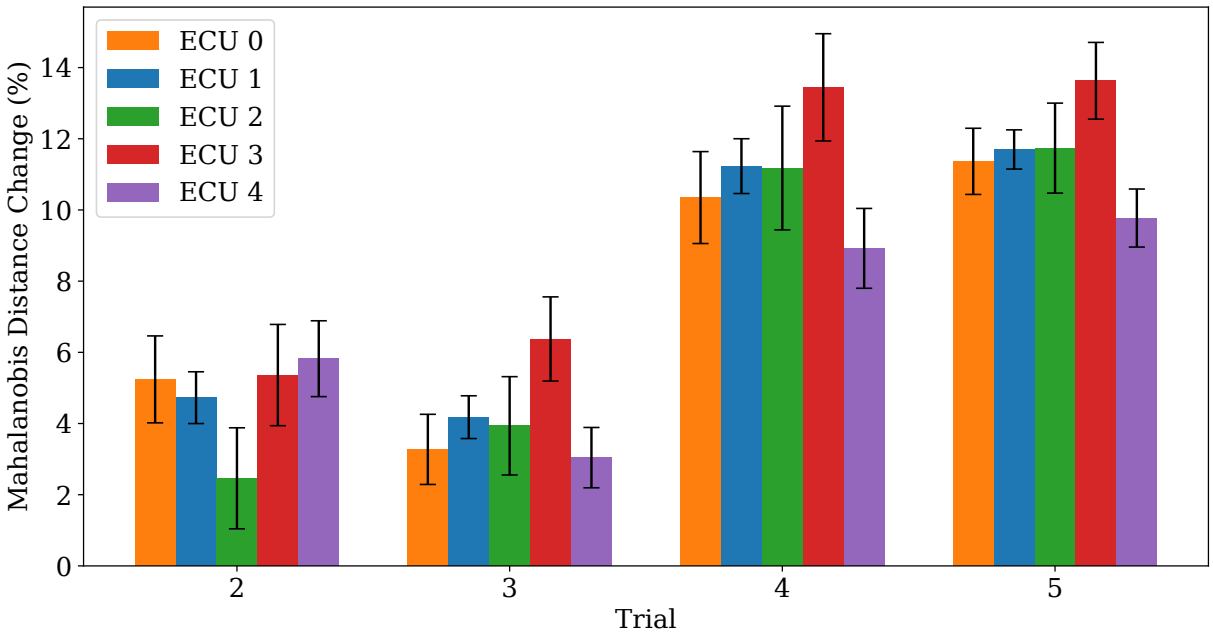


Figure 4.8: Percent delta of Mahalanobis distance means and 99% confidence intervals between training data from accessory mode of the first trial and test data from accessoy mode of the other four trials.

Resolution (bits) \ Sampling Rate (MS/s)	2.5	5	10	20
10	0.99998	1.00000	1.00000	1.00000
12	0.99999	1.00000	1.00000	1.00000
14	0.99999	1.00000	1.00000	1.00000
16	0.99999	1.00000	1.00000	1.00000

(a) False positive test accuracies.

Resolution (bits) \ Sampling Rate (MS/s)	2.5	5	10	20
10	0.99997	1.00000	1.00000	0.99999
12	0.99999	1.00000	1.00000	0.99999
14	0.99998	1.00000	1.00000	1.00000
16	0.99998	1.00000	0.99999	0.99999

(b) Hijack test F-scores

Resolution (bits) \ Sampling Rate (MS/s)	2.5	5	10	20
10	0.99996	1.00000	1.00000	0.99999
12	0.99999	1.00000	1.00000	1.00000
14	0.99999	1.00000	1.00000	1.00000
16	0.99999	1.00000	1.00000	1.00000

(c) Foreign device test F-scores.

Table 4.6: Results with downsampled and low-resolution data on Vehicle A.

Sampling Rate (MS/s)	Accuracy
2.5	0.99961
5	0.99999
10	1.00000

(a) False positive test accuracies.

Sampling Rate (MS/s)	F-Score
2.5	0.99928
5	0.99999
10	1.00000

(b) Hijack test F-scores.

Sampling Rate (MS/s)	F-Score
2.5	0.99904
5	0.99999
10	1.00000

(c) Foreign device test F-scores.

Table 4.7: Results with downsampled data on Vehicle B.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	0	0
	Normal	4	5775553

Table 4.8: Temperature variance confusion matrix for Vehicle A.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	0	0
	Normal	0	840625

Table 4.9: High-power vehicle functions confusion matrix for Vehicle A.

Chapter 5

vProfile Enhancements

This chapter describes three enhancements to the base vProfile algorithm that can improve its detection rate. The goal of the first two methods is to reduce the intra-cluster variance and maximum distances to increase the perceived similarity of traces in the same cluster. The third method is an online model update algorithm that allows vProfile to adapt to bus voltage changes that may not warrant training a new model.

5.1 Edge Set Extraction Threshold

The edge set extraction algorithm in Section 3.2.1 uses a constant threshold for the center of each message's rising edge. We found that having a different threshold for each cluster can positively impact the statistics of the resulting traces. To show this, we train models using both a fixed threshold and cluster thresholds. We used data from Vehicle A. For demonstration purposes, we use a simple calculation for each ECUs threshold where we find the mean of the maximum and minimum values from the first half of a message. We use only the first half because the voltage level of the ACK bit can deviate significantly from the rest of the message [7].

Table 5.1 shows the standard deviation and maximum distance of a trace to its ECU's mean for both model types. For standard deviation, our results show an improvement when using cluster thresholds for ECUs 2 and 4 but a degradation for the others. For maximum distance, we see an improvement for ECUs 1, 2, and 4. Though these differences do not affect vProfile's performance for our vehicles, the cluster thresholds did impact the values. A better algorithm for threshold finding could improve vProfile's performance for different applications.

ECU	Standard Deviation		Maximum Distance	
	Static Threshold	Cluster Threshold	Static Threshold	Cluster Threshold
0	168.059	168.065	11.142	11.171
1	171.865	171.879	21.149	21.089
2	176.562	176.542	11.007	10.716
3	152.875	152.972	11.253	12.011
4	190.560	190.558	10.490	10.421

Table 5.1: Standard deviation and maximum distance from the mean using fixed and cluster thresholds with Vehicle A data.

5.2 Number of Extracted Edge Sets

The edge set extraction algorithm in Section 3.2.1 returns only the first edge after the arbitration field. We can extract more edges from the same message at the cost of additional latency and then take their mean. This can reduce the impact of minor inconsistencies between edge sets from the same source on our distance measurements. Table 5.2 contains the intra-cluster standard deviation and maximum distance between a trace to its ECU's mean for one edge set and three edge sets. For the three edge set case, we extract them 250 samples apart, which is a simple way of sampling multiple message sections.

The results show lower standard deviations for every cluster and lower maximum distances for all but ECU 1. These improved statistics did not improve detection rates for our vehicles, but this enhancement could have a more significant impact on other systems.

5.3 Online Model Update

As described in Section 4.4, we confirmed that fluctuations in temperature and battery voltage can impact the CAN bus voltage. Instead of training a new model, we can update the existing one with new data. We can update the covariance Σ_{ij} of samples E_i and E_j (i.e., the i^{th} and j^{th} samples of an edge set and the element at index $[i, j]$ of the covariance matrix) with the n^{th} edge set using

ECU	Standard Deviation		Maximum Distance	
	1 Edge Set	3 Edge Sets	1 Edge Set	3 Edge Sets
0	168.065	168.002	11.136	9.671
1	171.879	171.716	21.089	25.472
2	176.542	176.019	10.716	10.463
3	152.972	152.857	12.011	9.619
4	190.558	190.218	10.428	9.329

Table 5.2: Standard deviation and maximum distance from the mean using one and three edge sets with Vehicle A data.

$$\Sigma_{ij,n} = \frac{[(E_{i,n} - \overline{E}_{i,n-1}) \cdot (E_{j,n} - \overline{E}_{j,n})] - \Sigma_{ij,n-1}}{N_n} \tag{5.1}$$

where $\overline{E}_{i,m}$ and $\overline{E}_{j,m}$ are the means up to and including the m^{th} edge set for indices i and j, respectively, and N_n is the new number of edge sets. In this case, $N_n = N_{n-1} + 1$, since we add only one new edge set. The algorithm requires that we store N_n in the model for each cluster.

The model update algorithm is as follows, assuming that no new SAs exist. We separate the new edge sets into clusters using the cluster-SA LUT in the existing model. Next, we update the edge set count (N_n above), mean, covariance matrix, and maximum distance for each cluster. Algorithm 4 summarizes the model update procedure.

We caution that a model should not be updated too often using this algorithm because new data will decrease the update's impact on the model as N_n increases. Therefore, we recommend training a new model after N_n reaches some upper bound M. The threshold can be applied to individual clusters since our findings show that some ECUs transmit more often than others.

Algorithm 4: vProfile Online Model Update procedure.

```
1  UpdateModel(model, newEdgeSets) begin
2      clustNewEdgeSetsLut = GroupByCluster(model.clustSaLut, edgeSets);
3      foreach clust ∈ clustNewEdgeSetsLut do
4          foreach edgeSet ∈ clustNewEdgeSetsLut[clust] do
5              prevNumEdgeSets = model.clustNumEdgeSetsLut[clust];
6              prevMean = model.clustMeans[clust];
7              model.clustNumEdgeSetsLut[clust] += 1;
8              UpdateMean(model.clustMeans[clust], edgeSet,
                model.clustNumEdgeSetsLut[clust], prevNumEdgeSets);
9              UpdateInvCov(model.clustInvCovs[cluster], edgeSet,
                model.clustMeans[clust], prevMean);
10             model.clustMaxDists[clust] =
                max(MahDistance(model.clustMeans[clust], edgeSet,
                model.clustInvCovs[clust]), model.clustMaxDists[clust]);
11         end
12     end
13 end
```

Chapter 6

Conclusion

In this thesis, we explored vProfile, a sender identification system for CAN messages that exploits the unique voltage characteristics of ECUs. Compared to the existing methods mentioned in Section 1.2, we proposed a lightweight, single-feature system that exhibits high detection rates for both hijack and foreign intruders despite having low latency and using lower sampling rates and resolutions. These characteristics suggest that vProfile has a higher potential to be implemented in real systems on less expensive embedded hardware. We also suggested two modifications to vProfile's preprocessing stage that can increase its detection rate along with an online model update algorithm.

Our method is similar to [13], but we distinguish ourselves by leveraging the information possessed by rising and falling edges. We also apply Mahalanobis distance directly to the raw voltage trace of the first stable edge set without any additional transformations. This further simplifies the algorithm and reduces latency while still exhibiting excellent performance.

6.1 Limitations and Future Work

Despite vProfile's demonstrated detection capabilities, there exist some limitations. As shown in Section 4.4, variances in battery voltage and ECU temperatures can affect the CAN bus voltage. However, we can reduce their effect on vProfile with curated training data from a wide variety of operating conditions and using the online update algorithm described in Section 5.3. Also, the current implementation of vProfile cannot detect when a hijacked ECU sends messages with SAs that are within its normal operating set. For

additional coverage, we recommend using vProfile in an IDS that can detect anomalies based on other message properties, such as the period and payload.

Though our voltage test produced conclusive results, it was not ideal, so a more thorough experiment with a variable power supply and a way to measure the bus resistance would be worth investigating. Furthermore, we conducted experiments on vehicles that use only extended CAN frames, whereas most consumer vehicles use the standard format. We want to investigate adapting vProfile for standard frames, though we do not anticipate many required changes.

References

[1] ARILOU. Solutions. [Online]. Available: https://ariloutech.com/solutions/. Accessed: Feb 10, 2021.

[2] BOSCH. CAN specification, 1991. [Online]. Available: http://esd.cs.ucr.edu/webres/can20.pdf. Accessed: Feb 1, 2021.

[3] Mehmet Bozdal, Mohammad Samie, and Ian Jennions. A survey on CAN bus protocol: attacks, challenges, and potential solutions. In *2018 International Conference on Computing, Electronics Communications Engineering*, pages 201–205, 2018.

[4] Ismail Butun, Salvatore D. Morgera, and Ravi Sankar. A survey of intrusion detection systems in wireless sensor networks. *IEEE Communications Surveys Tutorials*, 16(1):266–282, 2014.

[5] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *20th USENIX Security Symposium*, 2011.

[6] Kyong-Tak Cho and Kang G. Shin. Fingerprinting electronic control units for vehicle intrusion detection. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 911–927, Austin, TX, 2016. USENIX Association.

[7] Kyong-Tak Cho and Kang G. Shin. Viden: attacker identification on in-vehicle networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1109–1123, New York, NY, USA, 2017. Association for Computing Machinery.

[8] Wonsuk Choi, Hyo Jin Jo, Samuel Woo, Ji Young Chun, Jooyoung Park, and Dong Hoon Lee. Identifying ECUs using inimitable characteristics of signals in con-

troller area networks. *IEEE Transactions on Vehicular Technology*, 67(6):4757–4770, 2018.

[9] Wonsuk Choi, Kyungho Joo, Hyo Jin Jo, Moon Chan Park, and Dong Hoon Lee. VoltageIDS: Low-level communication characteristics for automotive intrusion detection system. *IEEE Transactions on Information Forensics and Security*, 13(8):2114–2129, 2018.

[10] United States Congress. Security and privacy in your car act of 2015, 2015. [Online]. Available: https://www.congress.gov/bill/114th-congress/senate-bill/1806?overview=closed. Accessed: Apr 28, 2021.

[11] embitel. Electronic control unit is at the core of all automotive innovations: Know how the story unfolded, 2017. [Online]. Available: https://www.embitel.com/blog/embedded-blog/automotive-control-units-development-innovations-mechanical-to-electronics. Accessed: Apr 28, 2021.

[12] ESCRYPT. Solutions overview. [Online]. Available: https://www.escrypt.com/en/solutions-overview. Accessed: Feb 10, 2021.

[13] Mahsa Foruhandeh, Yanmao Man, Ryan Gerdes, Ming Li, and Thidapat Chantem. SIMPLE: Single-frame based physical layer identification for intrusion detection and prevention on in-vehicle networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, pages 229–244, New York, NY, USA, 2019. Association for Computing Machinery.

[14] Ryan M. Gerdes, Mani Mina, Steve F. Russell, and Thomas E. Daniels. Physical-layer identification of wired ethernet devices. *IEEE Transactions on Information Forensics and Security*, 7(4):1339–1353, 2012.

[15] David Gessner, Manuel Barranco, Alberto Ballesteros, and Julián Proenza. Designing sfiCAN: A star-based physical fault injector for CAN. In *ETFA2011*, pages 1–4, 2011.

[16] Christopher Gutierrez, Marcio Juliato, Shabbir Ahmed, and Manoj Sastry. Detecting attacks against safety-critical ADAS based on in-vehicle network message patterns. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – Industry Track*, pages 9–12, 2019.

[17] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Security threats to automotive CAN networks - practical examples and selected short-term countermeasures. In *Computer Safety, Reliability, and Security, 27th International Conference*, pages 235–248, 2008.

[18] CAN in Automation. History of CAN technology. [Online]. Available: https://www.can-cia.org/can-knowledge/can/can-history/. Accessed: Feb 1, 2021.

[19] Marcel Kneib and Christopher Huth. 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*, CCS '18, pages 787–800, New York, NY, USA, 2018. ACM.

[20] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental security analysis of a modern automobile. In *2010 IEEE Symposium on Security and Privacy*, pages 447–462, 2010.

[21] Kvaser. J1939 introduction. [Online]. Available: https://www.kvaser.com/about-can/higher-layer-protocols/j1939-introduction/. Accessed: Feb 1, 2021.

[22] Nathan Liu, Carlos Moreno, Murray Dunne, and Sebastian Fischmeister. vProfile: Voltage-based anomaly detection in controller area networks. Design, Automation and Test in Europe Conference, 2021.

[23] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L. Massart. The Mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1–18, 2000.

[24] Charlie Miller and Chris Valasek. Remote exploitation of an unaltered passenger vehicle, 2015. [Online]. Available: http://illmatics.com/Remote%20Car%20Hacking.pdf. Accessed: Feb 3, 2021.

[25] Charlie Miller and Chris Valasek. Advanced CAN injection techniques for vehicle networks, 2016. [Video]. Available: https://infocon.org/cons/Black%20Hat/Black%20Hat%20USA/Black%20Hat%20USA%202016/Advanced%20CAN%20Injection%20Techniques%20for%20Vehicle%20Networks.mp4. Accessed: Jul 8, 2021.

[26] Carlos Moreno and Sebastian Fischmeister. Sender authentication for automotive in-vehicle networks through dual analog measurements to determine the location of the transmitter. In *5th International Conference on Information Systems Security and Privacy (ICISSP)*, Prague, Czech Republic, 2019.

[27] Jack Morgan. 2016 Peterbilt 579 truck, 2020.

[28] Pal-Stefan Murvay and Bogdan Groza. Source identification using signal characteristics in controller area networks. *IEEE Signal Processing Letters*, 21(4):395–399, 2014.

[29] Marco Di Natale. Understanding and using the controller area network, 2008. [Online]. Available: https://inst.cs.berkeley.edu/~ee249/fa08/Lectures/handout_canbus2.pdf. Accessed: Feb 1, 2021.

[30] United States Government Accountability Office. Report to Congress on vehicle cybersecurity, 2016. [Online]. Available: https://www.gao.gov/assets/gao-16-350.pdf. Accessed: Apr 28, 2021.

[31] H. F. Othman, Y. R. Aji, F. T. Fakhreddin, and A. R. Al-Ali. Controller area networks: Evolution and applications. In *2006 2nd International Conference on Information Communication Technologies*, volume 2, pages 3088–3093, 2006.

[32] Andrea Palanca, Eric Evenchick, Federico Maggi, and Stefano Zanero. A stealth, selective, link-layer denial-of-service attack against automotive networks. In Michalis Polychronakis and Michael Meier, editors, *Proceedings of the 14th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, pages 185–206. Springer International Publishing, 07 2017.

[33] Stephen Stachowski, Ron Gaynier, and David LeBlanc. An assessment method for automotive intrusion detection system performance, 2019. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/41006/dot_41006_DS1.pdf. Accessed: Feb 3, 2021.

[34] Miaoqing Tian, Ruobing Jiang, Chaoqun Xing, Haipeng Qu, Qian Lu, and Xiaoyun Zhou. Exploiting temperature-varied ECU fingerprints for source identification in in-vehicle network intrusion detection. In *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8, 2019.

[35] Jyothsna Veeramreddy, V. Prasad, and Koneti Prasad. A review of anomaly based intrusion detection systems. *International Journal of Computer Applications*, 28:26–35, 08 2011.

[36] Xiaoyun Zhou, Ruobing Jiang, Miaoqing Tian, Haipeng Qu, and He Zhang. Temperature-sensitive fingerprinting on ECU clock offset for CAN intrusion detection and source identification. In *Proceedings of the ACM Turing Celebration Conference - China*, ACM TURC'20, page 89–94, New York, NY, USA, 2020. Association for Computing Machinery.