2020

Techniques for utilizing classification towards securing automotive controller area network and machine learning towards the reverse engineering of CAN messages

Clinton Young
Iowa State University

Techniques for utilizing classification towards securing automotive controller area network and machine learning towards the reverse engineering of CAN messages

by

Clinton Young

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering (Secure and Reliable Computing)

Program of Study Committee:
Joseph Zambreno, Major Professor
Phillip Jones
Yong Guan
Liang Dong
Gary Tuttle

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2020

DEDICATION

To my loving parents, Shih-Yih Ryan Young and Chih-Ming Liau. You provided your loving guidance and support during my work.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to take the opportunity to thank my advisor, Dr. Joseph Zambreno, for providing the great opportunity to pursue my Ph.D. I would like to thank him for giving me the freedom to explore my ideas while providing his guidance, patience, and constructive feedback throughout this research.

I would like to thank my committee members Dr. Phillip Jones, Dr. Yong Guan, Dr. Neil Gong, Dr. Gary Tuttle, and Dr. Liang Dong for their guidance and feedback during the years of my Ph.D. I would also like to thank all the members of the Reconfigurable Computing Lab for their support and good times in the lab.

Lastly, I want to thank my parents and brother for all their support and encouragement. Without them, all of this would not have been possible.

ABSTRACT

The vehicle industry is quickly becoming more connected and growing. This growth is due to advancements in cyber physical systems (CPSs) that enhance the safety and automation in vehicle. The modern automobile consists of more than 70 electronic control units (ECUs) that communicate and interact with each other over automotive bus systems. Passenger comforts, infotainment features, and connectivity continue to progress through the growth and integration of Internet-of-Things (IoT) technologies. Common networks include the Controller Area Network (CAN), Local Interconnect Network (LIN), and FlexRay. However, the benefits of increased connectivity and features comes with the penalty of increased vulnerabilities. Security is lacking in preventing attacks on safety-critical control systems. I will explore the state of the art methods and approaches researchers have taken to identify threats and how to address them with intrusion detection. I discuss the development of a hybrid based intrusion detection approach that combines anomaly and signature based detection methods.

Machine learning is a hot topic in security as it is a method of learning and classifying system behavior and can detect intrusions that alter normal behavior. In this paper, we discuss utilizing machine learning algorithms to assist in classifying CAN messages. I present work that focuses on the reverse engineering and classification of CAN messages. The problem is that even though CAN is standardized, the implementation may vary for different manufacturers and vehicle models. These implementations are kept secret, therefore CAN messages for every vehicle needs to be analyzed and reverse engineered in order to get information. Due to the lack of publicly available CAN specifications, attackers and researchers need to reverse engineer messages to pinpoint which messages will have the desired impact. The reverse engineering process is needed by researchers and hackers for all manufacturers and their respective vehicles to understand what the vehicle is

doing and what each CAN message means. The knowledge of the specifications of CAN messages can improve the effectiveness of security mechanisms applied to CAN.

CHAPTER 1. INTRODUCTION

The vehicle industry is quickly becoming more connected and growing. This growth is due to advancements in cyber physical systems (CPSs) that enhance the safety and automation in vehicle. The modern automobile consists of more than 70 electronic control units (ECUs) that communicate and interact with each other over automotive bus systems [8]. Passenger comforts, infotainment features, and connectivity continue to progress through the growth and integration of Internet-of-Things (IoT) technologies. Common networks include the Controller Area Network (CAN), Local Interconnect Network (LIN), and FlexRay. I focus specifically on CAN, which is the most commonly used bus system.

However, the benefits of increased connectivity and features comes with the penalty of increased vulnerabilities [99, 96], as demonstrated in Figure 1.1. Security is lacking in preventing attacks against comfort and infotainment features from compromising safety-critical control systems [9]. Cyber attacks on automotive in-vehicle networks have risen [9, 7, 97] and this has driven a need for security in vehicles. Koscher et al. [38] demonstrated the ability of malicious agents to control several vehicle operations from unlocking the doors to stopping the engine. Miller and Valasek demonstrated their ability to control a Jeep Cherokee remotely through a cellular network, leading to the recall of over a million vehicles [52].

As an communication network develop in 1983 without cybersecurity in mind, the CAN bus has become an enticing target for hackers. The CAN protocol does not have origin and destination for messages. Instead, each node in network can send and receive messages based on the pre-defined ECU configuration. As the standard in modern vehicles, CAN has become an enticing target for hackers. An adversary may physically access the On-Board Diagnostic (OBD) port standard in all modern vehicles. Or they may remotely access the vehicle through a variety of wireless or cellular

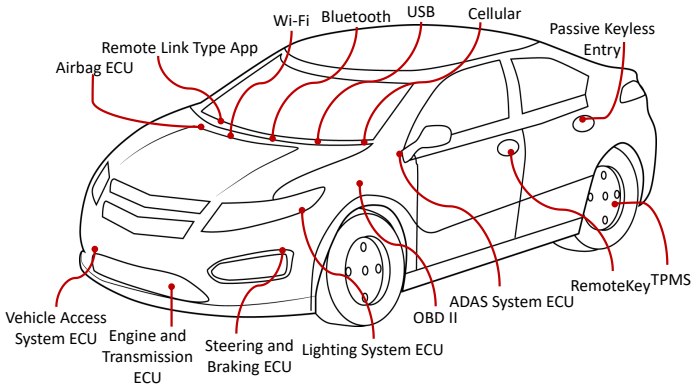

Figure 1.1 Automotive attack surfaces.

connections. Thus, a major security concern is in protecting the CAN bus that facilitates the communication and interactions of ECUs.

One option to enhance the security of in-vehicle networks is to adopt intrusion detection and prevention techniques. Intrusion detection systems (IDS) are used to mitigate intrusions in computer network systems. However, many traditional techniques in network security cannot be directly applied to vehicular networks. The limited computational, memory, and power resources in ECUs hinder the implementation of complex security mechanisms. Thus, an effective and efficient IDS that can work for in-vehicle networks is an important need.

I will explore the state of the art methods and approaches researchers have taken to identify threats against vehicles and how to address them with IDS approaches. I discuss the development of a hybrid based intrusion detection approach that combines anomaly and signature based detection methods. I will detail the individual components of each approach and how they were developed and evaluated.

Machine learning is a hot topic in security as it is a method of learning and classifying system behavior and can detect intrusions that alter normal behavior. In this paper, I discuss utilizing machine learning algorithms to assist in classifying CAN messages. The proposed approach employs the k-means and agglomerative clustering to categorize CAN messages. The distance between CAN message IDs are used to cluster the CAN messages by relevance to each other. The machine learning algorithm, combined with previously labeled CAN IDs, classifies CAN messages. The goal is to

determine the function of unknown CAN messages using machine learning and labels. To the best of our knowledge, this is the first paper toward utilizing machine learning in reverse engineering CAN messages without access to a physical vehicle nor having a DBC file.

I present work that focuses on the reverse engineering and classification of CAN messages. The problem is that even though CAN is standardized, the implementation may vary for different manufacturers and vehicle models. These implementations are kept secret, therefore CAN messages for every vehicle needs to be analyzed and reverse engineered in order to get information. Due to the lack of publicly available CAN specifications, attackers and researchers need to reverse engineer messages to pinpoint which messages will have the desired impact. The reverse engineering process is needed by researchers and hackers for all manufacturers and their respective vehicles to understand what the vehicle is doing and what each CAN message means. The knowledge of the specifications of CAN messages can improve the effectiveness of security mechanisms applied to CAN.

1.1 Objectives

This paper aims to identify the constraints and requirements of developing intrusion detection systems for modern vehicles. Presented are findings to contribute to a deeper understanding of the challenges of implementing IDS for in-vehicle networks. The research questions that drive this research are:

- What are the considerations that need to be made in designing an IDS?
- What vulnerabilities exist in vehicles and what can be done to secure them?
- How can an attack be detected and how does the automotive system respond?
- Can machine learning be utilized towards reverse engineer CAN messages?
- What features exist in raw CAN messages and what features can be extracted?

- What metrics should be used to evaluate the success and shortcomings of our detection and classification approaches?

1.2 Contributions

There is numerous research that highlight the importance of integrating security into in-vehicle networks. However, due to the complex nature of these approaches, they cannot be easily implemented. To further advance vehicle security, a greater understanding is required of the vehicle traffic. Provided are the following contributions to address these issues.

- Frequency-based anomaly detector. The design and implementation of a frequency-based IDS that can identify message injection attacks using the frequencies of CAN messages to capture normal behavior and then detect violations of the normal as signs of an attack.
- HAIDS:Hybrid Automotive Intrusion Detection System. The design and implementation of an IDS that combines anomaly and signature-based detection methods for a more comprehensive detection system.
- Machine learning messages classifier. An approach towards classifying unknown controller area network messages utilizing hierarchical clustering.

This dissertation is organized as follows. Chapter 2 gives background information on the controller are network and its vulnerabilities. I discuss the security challenges and attacker motivations. Chapter 3 provides a comprehensive survey on intrusion detection and related work on automotive intrusion detection systems. Chapter 4 outlines the details of our approach towards automotive intrusion detection. I break down the different message timing features I examined and utilized. Chapter 5 details the hybrid automotive intrusion detection system, called HAIDS. I outline the details of the anomaly and signature based detectors for intrusion detection. Additionally, the evaluation and analysis of HAIDS is discussed. Chapter 6 delves into the CAN reverse engineering process using machine learning. This chapter provides background on reverse engineering and machine learning, to help get the reader up to speed. Also, related work on novel approaches of reverse

engineering CAN and utilizing machine learning towards CAN are investigated. This chapter also details the reverse engineering process using hierarchical clustering and reports the evaluation and analysis of our message classification. Chapter 7 outlines the future work of using machine learning towards automotive security. Finally, Chapter 8 concludes the dissertation.

CHAPTER 2. BACKGROUND

In this chapter, provided are details into in-vehicle networks, their vulnerabilities, and how attackers are exploiting these openings.

A network is a system of interconnected devices where information is shared. While a standard local area network consists of a group of computers, printers, and other devices,communicating and sharing resources with each other, an in-vehicle network consists of several control units. These control units include engine control systems, transmission control systems, electronic control systems. These units communicate via the data bus in real-time to manage vehicle operation. Electronic systems in vehicles are connected to each other. The demand for enhanced safety, comfort, and infotainment have increased the number of electronic systems in vehicles.

An in-vehicle network consists of nodes, gateways, and buses. A node is an ECU, which is connected to the bus. Together these buses and nodes form a network. Gateways connect different in-vehicle networks together. There are many different types of network bus domains in an vehicle, with the most common one being the controller area network.

2.1 Controller Area Network

The controller area network (CAN) was created to reduce the network complexity and wiring costs by simplifying current mechanical wiring in vehicles with a two-wire bus. CAN is an asynchronous, serial, multi-master communication network protocol that connects ECUs [14]. Vehicles, airplanes, and industrial machinery utilize CAN as a serial communication protocol. CAN supports bit rates in the range of 1Kbps to 1Mbps. Low speed CAN normally has data rates up to 125Kbps while the high speed CAN has data rates up to 1Mbps.

CAN is a message-based protocol that uses a lossless bitwise arbitration to transmit binary signal data. It uses the term dominant bits to represent the logical 0 and recessive bits for the

logical 1 signals. When the voltage difference between the two wires is large, the state is dominant. The state is recessive when the voltage difference is small between the two wires. Dominant state overwrites the recessive state when two or more ECUs send messages at the same time.

The CAN bus communication follows the Open Systems Interconnection (OSI) model for different layers: application, object, transfer, and physical. The broadcast nature of the CAN bus means that transmitted data are received by all ECUs on the bus. Data is transmitted between ECUs in the data frame of CAN packets, which consists of an ID field, data field, cyclic redundancy check (CRC), and other fields seen in Figure 2.1. CAN efficiently implements static fixed priority non-preemptive scheduling of messages through bus arbitration. CAN messages may be periodic, sporadic, or aperiodic. Periodic messages arrive at regular intervals with fixed lengths called periods. Sporadic messages recur with a minimum inter-arrival time between successive messages, while aperiodic messages occur at arbitrary times. The period of a messages is the fixed time interval after which a message releases another message. Each transmitted message goes through the arbitration process to determine which wins the bus. When a message wins arbitration and starts transmitting, it becomes non-preemptable. Messages win arbitration according to their priority, which is determined by the messages identifier (ID). A message with a lower ID has higher priority.

An important feature of CAN is the MAC protocol Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). It manages and avoids communication collisions, in the case of several nodes trying to access and send frames on the bus at the same time. When two or more ECUs start transmitting messages simultaneously, arbitration is done and the messages with the lowest message ID wins. The ECU which is attempting to transmit a recessive bit cancels its transmission, because it recognizes the other ECU's attempt to transmit a message with higher priority.

The CAN bus does not implement security mechanism such as encryption, authorization or authentication. The CAN architecture was envisioned to be lightweight and robust and designed to be unsegmented, unencrypted, and lacking authentication so that CAN messages could flow freely to and from each ECU. However, these properties directly lead to CAN's security vulnerabilities.

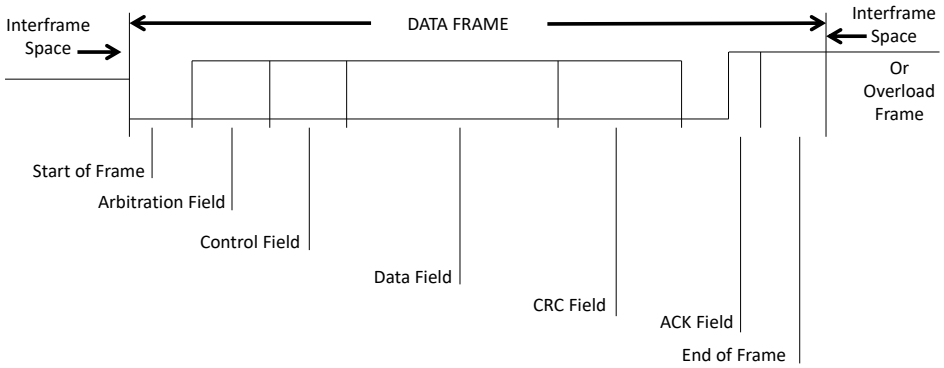

Figure 2.1 Standard CAN frame format.

2.2 CAN Messages

Message transfer is established and controlled by four different frame types: data, remote, error, and overload frame. The data frame transfers data between the nodes on the bus. A unit on the bus transmits the remote frame to request the transmission of the data frame with the same identifier. On bus error detection, an error frame is transmitted by any unit in the bus. The overload frame is used to provide an extra delay between the leading and succeeding data or remote frames.

Controller area messages can be sniffed and logged from the CAN bus. It is standard for recorded CAN data to consist of a time stamp for the message, the message ID of each message, and the individual 8 bytes of data that make up the Data field. CAN ID and data fields are presented in hexadecimal form. Below is an example of a single CAN message. The first field is the time stamp of when the message was captured on the bus. The 2nd field is the CAN message ID. The last 8 fields are the 8 bytes that make up the data field for the CAN message.

0.000248134, 202, 0C, EE, 72, 6D, 60, 00, 00, 00

It can be seen that not every field is recorded by some CAN snooping software. Depending on how the CAN messages are captured, more or less information may be available. What I present are the standard fields that useful software will present.

2.3 Vulnerabilities of CAN

There have been several publicized attacks on the vehicular network by researchers [4, 7, 27]. Securing automotive in-vehicle networks is challenging because of the resource constraints and the need to maintain predictable performance of the ECUs. Due to their low-cost nature and low data rates, most ECUs have limited computing power and memory to implement security mechanisms. Automotive networks have significantly lower transmission capability compared to network systems. CAN is susceptible to eavesdropping, denial-of-service, message injection, and impersonation attacks because of its broadcast nature, lack of authentication, and encryption.

2.3.1 Lack of Message Authentication

Each ECU broadcasts and receives all data on the CAN bus then decides whether messages are meant for them. CAN by design is unable to prevent unauthorized devices from joining the bus and broadcasting malicious messages to all the ECUs. By accessing the bus, hackers can send spoofed messages to any ECU on the network. Security in this context is provided only through a lack of open documentation. A hacker needs to dedicate time and resources to reverse engineer the CAN protocol before being able to launch malicious attacks on a particular vehicle.

2.3.2 Unsegmented Network

All ECUs are connected to a common network. This is a major reason CAN was adopted in automotive networks, to reduce the need of point-to-point connections between automotive systems. However, this reduction means a system component dealing with infotainment can communicate to safety critical vehicle systems. While some manufacturers utilize different networks for safety-critical systems, there is still cross-communication between safety critical and non-critical systems.

2.3.3 Unencrypted Messages

CAN was designed to be lightweight and robust back in the 1980s, when car hacking was not a reality. Addition of encryption would only slow down CAN messages and clog the network.

However, because CAN traffic is unencrypted, it can be easily sniffed, spoofed, modified, and replayed. There is a large area of research in applying encryption to automotive networks [5, 58, 44, 3, 64, 75].

Knowing these vulnerabilities, I examine how attackers could take advantage of them and to determine how to secure these vulnerabilities.

2.4 Threats and Attacks

To address the security concerns of modern vehicles, security analysis is needed in the development process. Threat modeling is a technique for security analysis using a structured approach that identifies security threats. A good threat model captures the possible attacks, motivations of attackers, and their capabilities to achieve their goals. With a threat model, a system designer can evaluate threats to the system and identify the vulnerabilities and attack-able vectors.

Threat modeling is a whole sub-field in automotive security. Beyond identifying vulnerabilities that can be exploited, a good understanding of potential attackers, their capabilities, and their objectives is needed. In this section I give an examination of attackers and attack vectors in vehicles.

2.4.1 Attackers and Motivations

Past attacker taxonomies [66] are based at least partially on the intentions of the attacker. Beginner attackers are amateurs or script kiddies that have limited security knowledge but know just enough to exploit security flaws with tools made by more knowledgeable hackers. These knowledgeable hackers can be known as crackers and vandals. These hackers have the security knowledge to seek out specific data utilizing security vulnerabilities. Lastly, the category of career criminals are individuals with the knowledge, skill set, and focused objective for monetary gain.

Modern vehicles contain multiple interfaces that expose the vehicle's systems to cyber attacks through physical and wireless access.

- Physical access: The attacker has direct connection to the OBD port in the vehicle that is connected to the CAN bus and all ECUs. This port can be accessed with the right tools and opportunity. An attacker can plug a small dongle to the OBD port to gather information or inject messages. An attacker can also plug a wireless device to the port and access it remotely [9].

- Wireless access: The more critical of the two attack surfaces. An attacker does not need physical access to the vehicle. These attack surfaces include Bluetooth and the telematics common in modern vehicles for wireless and cellular connectivity. Miller and Valasek demonstrated unauthorized CAN bus access to a Jeep Cherokee through its WiFi network that exploited a weakness in its password generation protocol [52].

Further research needs to be done in determining the objectives of different attackers and what methods they can use to accomplish their goals. Understanding the motivations, goals and methods of attackers helps with developing methods to prevent their attacks.

Some primary motivations for attacking vehicles include:

- Research: Attacks can be performed by security experts; this process is known as penetration testing.

- Illegal Profit: Attacker aims to profit by stealing a vehicle, stealing information on the vehicle, or other illegal activities.

- Illegal Falsification: Attacker aims to alter actual vehicle information such as odometer readings to increase the vehicle's value.

- Remote Hijacking: Attacker aims to take control and operate a vehicle while it is in use.

The attacks demonstrated by these works [38, 9, 52, 1, 18, 96, 90] can be broken down into the following categories:

- Sniffing: Because CAN does not support confidentiality and every CAN message is broadcast to every node on the network, an attacker can sniff and eavesdrop all CAN messages.

- Unauthorized Data Injection: An attacker can execute a replay attack easily because they can sniff all messages transmitted on the CAN. An attacker can also impersonate an ECU and transmit arbitrary messages on the network through message injection.

- Denial of Service: It is trivial for an attacker to continuously send high-priority messages over the OBD-II port, thereby keeping the network busy and unavailable to the rest of the ECUs on the network.

Each of these attack vectors tie into one another. Typically attackers start by sniffing CAN messages on the network and then reverse engineer them. After understanding specific CAN messages, they can then inject specific messages onto the network to make the vehicle misbehave. Through message injection, an attacker can launch a Denial of Service (DoS) attack and disable the network and the vehicle [78]. A core component in most modern attacks is unauthorized data injection.

Unauthorized data injection can be broken down into several categories:

- Injecting a single CAN message ID: Inject a single CAN packet repeatedly. The goal is to make the vehicle do what the message is commanding.

- Injecting multiple messages: Similar to single CAN message injection but injecting multiple CAN IDs onto the network to make the vehicle misbehave.

- Massive message injection to perform a DoS attack: This is also known as message flooding, where there is an influx of messages on the network.

- Bus-off attack: By iteratively injecting attack messages, the adversary coerces the transmit error counter (TEC) of an un-compromised victim ECU to continuously increase, deceiving it to think it is defective, which triggers the CAN fault confinement to force the victim or even the entire network to shutdown [11, 29].

A defining characteristic of injection attacks is how they affect the timing intervals of CAN messages. The interval between messages decreases because the original ECU continues to transmit its

messages as attackers are injecting their own malicious messages. I will focus on different attack scenarios, all with an adversary whose goal is to control the vehicle.

CHAPTER 3. RELATED WORK

3.1 Intrusion Detection Systems

Intrusion Detection Systems (IDS) are software or hardware systems that automate the attack detection process, usually through use of sensors and reporting systems. Most modern IDS monitor either the host computers or networks to capture intrusion related data [70, 79, 94, 101, 68, 73, 69, 64]. The sensors are monitoring agents on the network. They collect real time network or host data to detect malicious activities. I examine approaches and implementations of traditional IDS and how these principles can be applied to automotive security.

3.1.1 Host-based

A Host-based Intrusion Detection System (HIDS) resides in and monitors the host system. The IDS monitors activity such as process events and system calls. In automobiles, a host-based IDS would reside in individual ECUs, where it monitors the traffic packets entering and leaving, and check to ensure packets are not malicious. HIDS also monitors the ECU itself to detect behavior indicative of an intrusion. The issue with host-based IDS is that some ECUs lack the processing power required to support a HIDS. Implementing a HIDS in automobiles would require re-work on the part of manufacturers on their ECUs.

3.1.2 Network-based

A Network-based Intrusion Detection System (NIDS) is part of the communication system and monitors all traffic traversing the network. Information monitored include header and content of each message or packet. An automotive NIDS monitors all traffic on the network with the NIDS acting as an ECU, so that it can receive and monitor all messages broadcast.

3.2 Intrusion Detection Methods

Intrusion detection methods can be classified under two main categories: signature and anomaly-based. IDSs can only detect an intrusion, they are not able to offer protections for networks or information systems. Preventative and reactive measures against intrusions have to be implemented in order to secure a network.

3.2.1 Signature-Based

Signature-based approaches detect attacks using a pre-defined knowledge base of attack signatures that is captured and created, and current network traffic is monitored for these signatures. This detection mechanism is effective in detecting known attacks with high accuracy and low error rates. However, signature-based IDSs cannot detect any attack not defined in the database, and therefore are unable to detect new attacks, nor any deviation from known attacks. It is critical to maintain the knowledge base and update it frequently for accurate detection.

3.2.2 Anomaly-Based

Anomaly-based intrusion detection typically starts with a training or normal model of the system's activity. To obtain best accuracy in detection, the normal model must be thorough. The IDS then compares current system's activity to past captured normal model to detect variations in behavior and label those deviations as anomalies. Any deviation not captured in the normal profile could be correctly or mistakenly identified as an intrusion. It is important to have the most complete normal profile, so the system does not suffer from high rates of false positives. The main advantage of anomaly detection is its ability to identify new and previously unknown attacks.

Although anomaly detection can be effective in detecting previously unknown attacks, the use of anomalies to infer intrusions suffers from a few key challenges:

- Defining a normal model that encompasses all possible expected behavior is extremely difficult. The boundary between normal and anomalous behavior is often blurred.

- Normal continues to evolve, and what is normal currently may not be so in the future.
- Anomalies may not be a result of malicious actions. Data often contains noise which may produce anomalies that can be difficult to distinguish from attacks.
- Attackers try to mask their actions to appear normal, therefore making the task of defining a tight boundary around normal more important, and also more difficult.

To address these challenges, researchers have used different types of intrusion detection systems and methods, and will detail these.

IDSs vary in data sources and the specific techniques used to analyze this data. Signature-based detect attacks are effective in detecting known attacks with low error rates, but they cannot detect any attack not in the database of signatures, and therefore are unable to detect newly created attacks nor any deviation of known attacks. Anomaly-based detection can infer an attack is occurring if network traffic deviates from the normal model, which enables detecting new attacks, but can incur high false-positive rates. As attacks are always evolving, so must detection, and this requires updates to detection methods to detect new attacks.

3.3 Intrusion Detection Systems for Automotive Security

I investigate how researchers are applying traditional intrusion detection approaches to securing automotive networks in this related works section. I summarize some of the cutting edge work on automotive intrusion detection in Table 3.1 and discuss their advantages and drawbacks.

3.3.1 Message Timing

In normal vehicle operation, each message ID generated by an ECU has a regular frequency. When attackers inject messages to execute a command to an ECU, this frequency will unexpectedly change. Even when an attacker is injecting messages, the ECUs still send their messages periodically. Eventually, rate of messages on the network will be increased by a factor of more than 2 to 100 times, depending on the attacker's injection speed. Miller and Valasek reported that they

needed to inject at a rate of at least 20 times faster than normal for their attack to be successful [52]. Because the original ECU is still transmitting its message, an attacker needs to send in messages at a fast enough rate to overwrite the normal message with the same ID.

Detection is based on the following principles.

1. When a new message is transmitted on the CAN bus, the IDS will check the ID and compute the time interval from the arrival time of the latest message.

2. If the time interval of the new message is shorter than the normal model, the IDS indicates message as an anomalous message due to this message is arriving sooner than expected.

A conceptual diagram on the effects of message injection attacks on normal traffic is given in Figure 3.1. Injected messages shorten the inter-arrival times of those specific messages.

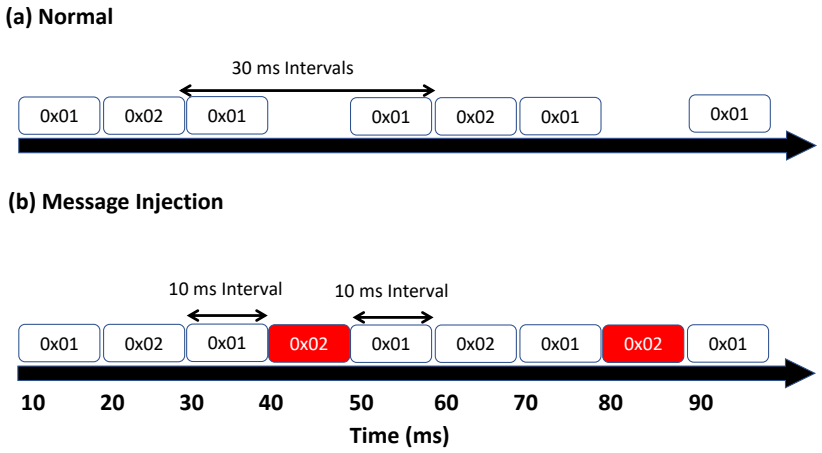

Figure 3.1 Diagram about transmitted messages on CAN bus on (a) normal status and (b) under message injection attack. The time interval of message CAN ID 0x02 is shortened by the injection of attack messages.

Miller and Valasek [51] introduced a concept of analyzing the rate of messages for in-vehicle network intrusion detection. The number of messages on CAN bus is the sum of the number of normal messages and attack messages. By analyzing the distribution rate of messages, it should be possible to detect anomalous messages.

Researchers have explored utilizing message timing features for intrusion detection. These works have shown good results in using message intervals for detecting a significant threat to automotive security, message injection.

Gmiden et al. [21] proposed a simple intrusion detection method for CAN bus. Their proposed algorithm does not require any modification to the CAN bus, which would mitigate changes to the native system and computational overhead, and is based on the analysis of time intervals of CAN messages. Their future work involves implementing and evaluating their proposed detection method.

Moore et al. [57] proposed an anomaly detector based on the regularity of CAN message frequency. Similar to the detection method proposed by Gmiden [21], Moore's detector relies on the time intervals of CAN messages. They observed regularity in the signal frequencies, and hypothesize that a simple anomaly detection system monitoring the inter-signal wait times of CAN bus traffic will provide accurate detection of a regular-frequency signal injection attacks. To test their detector, they defined and executed three signal injection attacks. In their investigations they identified that many signal injection attacks require repeated injections to be effective and the regularity of normal signals has an exceptionally accurate anomaly detection capability. They conclude that their approach is a promising avenue for accurate detection of an important class of CAN bus attacks.

Song et al. [76] also proposed a lightweight intrusion detection algorithm that examines the time interval of CAN messages. They evaluated how three different types of message injection attacks affect the unique time interval of each CAN ID. They combined 100 one-second sample of normal and attack data logs and then applied their IDS to determine which logs were of attacks. They determined that the time interval is a feature capable in detecting message injection attacks in CAN bus traffic by showing there was a clear difference between time intervals of messages in normal status and attack status. The strength of their proposed detection algorithm is that it is simple and efficient to use.

Taylor et al. [80] proposed an algorithm that measures inter-packet timing over a sliding window. Adapted from industrial control systems, the authors applied a flow based method to the CAN bus to measure changes in the data content and their frequencies then compare them to historical values for anomaly detection. Communications between the endpoints of the CAN networks are contained in the flow statistics which are trained using the One Class Support Vector Machines for flow classification. The result of this approach was evaluated using time test statistics of the flow detector and consistent detection improvement is observed with OCSVM as the number of packet insertions increased with time.

Tomlinson et al. [82] presents an analysis of CAN broadcasts and consequent testing of statistical methods to detect timing changes in the CAN traffic indicative of some predicted attacks. The detection is implemented in time-defined windows. The generation of simulated attack data, and the determination of positive detections, are also considered.

Utilizing the CAN message timing intervals show good detection capabilities with minimal change to the vehicle's native network. This approach using CAN message timing features has shown the most success in detecting known attacks. However, the methods are very simple and is currently limited to detecting attacks that inject numerous messages onto the CAN bus. While the majority of demonstrated attacks have been message injection, it is conceivable that other methods of attacks exist. I examine alternative detection methods in the following sections.

3.3.2 Signature-Based

A signature-based detection method relies on signatures that describe the behavior of the system and attacks. These behaviors of the system are described by it functionality. The monitoring of the system involves detecting these signatures or deviations from the signature. Larson et al. [41] proposed a specification-based attack detection approach that has a detector placed in each ECU. The incoming and outgoing network traffic can be analyzed based on information from the protocol stack and object directory of the CAN-protocol at the expected ECU. They show that potential attacks can be detected from the trace of extracted information through theoretical simulation.

The authors inferred that a likely target for attackers is the gateway ECU because a variety of attacks can be accomplished when it is compromised.

Mitchell and Chen [53, 54, 55] proposed a behavior-rule signature based IDS for medical CPS and unmanned aircraft systems. In their approach, they used a binary failure threshold to classify a node as normal or malicious based on the node's compliance threshold. Esquivel-Vargas et al. [16] proposed an approach to automatically deploy a signature-based IDS to monitor automation systems. They developed an approach to generate rules that represent valid device behavior in BACnet networks that are used by the IDS to detect violations in the network traffic. This approach was implemented in a passive way and with network- wide coverage. Fauri et al. [19] proposed an approach to combine formal specification with anomaly-based monitoring to overcome the semantic gap between network anomalies and actionable alerts by leveraging the lightweight logical system specification.

3.3.3 Anomaly-Based

Researchers also explored other avenues applying intrusion detection to automotive networks beyond the simple examination of message timing features. However, a limiting factor of implementing complex intrusion detection systems is the computing power of ECUs. ECUs come in varying complexity and sophistication from a simple seat control unit that adjusts seat height and angle to complex engine control units that control a variety of engine functions. Some of the following techniques are computationally heavy and implementing them onto automotive networks may require major rework of the automotive system.

3.3.3.1 Cyber-Physical

The following works define different ECU characteristics to authenticate individual vehicle ECUs. Each IDS proposes using a specific characteristic that is unique to every ECU on the vehicle. Similar to message timing anomaly detection, when these properties vary from the cap-

tured normal, an anomaly is detected. The works here describe alternative features to message timing that can be captured and examined to detect certain attacks.

Cho and Shin [12] introduced a clock-based IDS that uses clock skew (timing error) to authenticate ECUs. The IDS records communications on the CAN bus and creates fingerprints of every ECU on the network. Each ECU is assigned a fingerprint based on their specific clock skew and this is used to distinguish them. The authors proposed that by analyzing the CPU clocks behaviors, spoofing attacks can be detected in the network.

Ji et al. [30] investigates a detection method based on clock drift. Their approach considers that every ECU has a fixed clock skew and it is possible to establish a normal model of ECU's clock behaviors to detect abnormal measurements. They evaluate the effectiveness of the method against injection and suspension attacks. The analysis results demonstrated that the proposed detection method can detect small-scale change of packets transmitted in CAN networks.

Choi et al. [13] proposed a novel automotive intrusion detection system, VoltageIDS. This system leverages the electrical CAN signal characteristics as a fingerprint of the ECUs. The VoltageIDS does not require any modification of the vehicular system and can distinguish between errors and bus-off attacks. They evaluated their IDS on moving as well as idling vehicles. The method is shown to be capable of detecting the recently introduced bus-off attack.

Lee et al. [42] proposed an IDS called OTIDS that measures the response performance of network nodes based on the offset ratio and the time interval between request and response. The authors claimed that each node has a fixed response offset ratio and time interval in a normal operation mode which varies significantly during attack. This difference in the offset ratio and time intervals is used to detect attacks in the network.

3.3.3.2 Entropy

Entropy-based intrusion detection has been applied to traditional network-based systems, but typically has a high rate of false positives [59] due to typical traffic variance. As automotive network traffic tends to be more periodic, entropy-based detection has been shown it can detect anomalies

with a low rate of false-positives. Müter et al. [59], using data recorded from the in-vehicle network communication during normal operation, calculated the Shannon entropy value. Deviations from that entropy are identified as potential intrusions. Marchetti et al. [47] proposed an entropy-based algorithm for detecting anomalies in CAN messages in an unmodified vehicle. They conducted extensive evaluations based on several hours of CAN traffic captured during driving sessions on public motorways. Their experimental evaluations show that entropy-based anomaly detectors are a viable approach for identifying CAN bus anomalies caused by attackers injecting messages.

3.3.3.3 Message Rate

Very similar to the message timing detection, Hoppe et al. [27] proposed an anomaly-based IDS that is placed on the CAN bus so that it can listen to network traffic. Their IDS examines the rate of transmission of specific messages and compares it to what is normal to detect additional or missing messages. This approach differs from the previously examined works as it counts rate of transmission of packets as opposed to the timing intervals of the packets. Deviations from the expected normal number of messages transmitted are identified as anomalies. Their future work involves implementing and evaluating their proposed detection method.

3.3.3.4 CAN-Fields

Several works utilize the makeup and data fields of CAN messages for anomaly detection. Matsumoto et al. [50] proposed a method of preventing unauthorized data transmission in CAN. Each ECU monitors all the data on the bus, and broadcasts an error message if it recognizes spoofed messages with its own ID, before the unauthorized message is completely transmitted. Boudguiga et al. [89] proposed an intrusion detection method that makes each legitimate ECU monitor the data frames on the CAN bus to detect whether another ECU is sending frame on its behalf. The authors used a hardware security module, a security processor dedicated to cryptographic computation and secure key storage, in the ECU to enforce security in the CAN bus.

Van Herrewege et al. [84] suggested adding message authentication protocol to the CAN messages. They showed that the standard authentication protocols are not suitable to the CAN bus, and presented CANAuth. This is a new backward compatible and lightweight message authentication protocol for the CAN bus. However, this required all ECUs to know a pre-shared key to be able to verify messages.

Taylor et al. [81] proposed an anomaly detector using Long Short-Term Memory recurrent neural network for CAN bus anomaly detection. This approach trained the neural network to predict the payload of the next message. The prediction errors are then used as signals for identifying anomalies in the messages sequence.

Martinelli et al. [49] proposed a detection method to identify attacks on CAN packets using fuzzy classification algorithms. The authors developed a fuzzy rough nearest neighbor classification technique to classify legitimate CAN messages generated by the human driver and the injected ones. Kuwahara et al. [39] proposed a statistical anomaly detection approach based on supervised and unsupervised learning of message patterns. The authors assume that there is likely a malicious message in a sequence if the number of messages is higher than normal.

Markovitz et al. [48] proposed a novel domain-aware anomaly detection system for CAN bus traffic. They discovered semantically meaningful fields through the inspection of real CAN traffic. They developed a greedy algorithm to split CAN messages into fields and classify these fields into specific types they observed. Their anomaly detection system uses classifiers to characterize the fields and build a model for the messages, based on their field types in the learning phase. In the enforcement phase, the system detects deviations from the model. They evaluated their system on simulated and real CAN traffic and achieved near zero false positives. These methods require a deeper understanding of CAN messages and reverse engineering of the messages and their data fields.

Tyree et al. [83] present a transformational approach to CAN IDS that exploits the geometric properties of CAN data to inform two novel detectors, one based on distance from a learned, lower dimensional manifold and the other on discontinuities of the manifold over time. Proof-of-concept

tests are presented by implementing a potential attack approach on a driving vehicle. The initial results suggest that (1) the first detector requires additional refinement but does hold promise; (2) the second detector gives a clear, strong indicator of the attack; and (3) the algorithms keep pace with high-speed CAN messages. As their approach is data-driven it provides a vehicle-agnostic IDS that eliminates the need to reverse engineer CAN messages and can be ported to an after-market plugin.

3.3.4 Other Works

Müter et al. [60] introduced an approach for anomaly detection using sensors to recognize attacks on in-vehicle networks during normal vehicle operation. The authors discussed the design and the application criteria for attack detection in the network, especially the CAN bus, without causing false positives. This detection scheme consists of eight sensors for detecting an attack. The sensors serve as a criteria for recognizing a threat to the automobile by monitoring different aspects of the network. In their proposed approach, the applicability of these sensors is based on different criteria such as the type and number of messages, the number of buses they need to access, and if the payload of the message needs inspection. The authors showed sensor data results can be evaluated and how to integrate the approach into an holistic IDS concept.

Kang and Kang [32] proposed a machine learning based IDS approach using deep neural network structure to monitor CAN packets. Their IDS consists of two modules. A monitoring module that decides a type of CAN packet based on trained features of known attacks. Once the monitoring module identifies a new attack, a profiling module records the attack model and updates the system for an upcoming packet. These two modules would be embedded in each ECU to analyze CAN packets. They used an unsupervised Deep Belief Net to capture underlying statistical features of CAN data and used them to classify messages as benign or anomalous. They reported a 99 percent detection ratio while keeping false positives under 1 to 2 percent through the use of software simulation. However, the authors did not discuss the overhead to implement their machine learning approach on modern vehicles.

Studnia et al. [77] proposed a language based IDS that uses language theory to define a set of attack signatures from an ECU behavioral model. This idea involves automatic generation of forbidden sequences from patterns characterizing the expected behavior of ECU based on language theory. Formal language is used here to describe a set of attacks which vehicles may be subjected to. This approach describes an algorithm that can be used to overcome the excessive use of memory which may be necessitated by an increase in states caused by the language.

These works show that there are multiple CAN and vehicle ECU characteristics that can be leveraged for intrusion detection in vehicles. Some works [12, 30, 13] capture specific characteristics without requiring changes to the native vehicular system to detect attacks. There are works [50, 48, 3, 23] that require reverse engineering of the CAN system and its messages to implement an intrusion detection system. While it is difficult to determine whether which approach is better, as some have not been evaluated, the best approach to detect the most comprehensive range of attacks may be a combination of some of these works.

The prior research in automotive security lacks a unified notion of anomalies and baselines. I have attempted to clarify and unify the concepts of anomalies and intrusion detection regarding automotive security. This begins with identifying threat models and threats that effect all vehicles and not just one specific model. From a technical perspective, IDSs can work well for detecting intrusions on the CAN bus. Different implementations of anomaly detection methods can detect different types of anomalies. Current approaches have a focus on message injection attack detection because it is the main attack vector for hackers trying to manipulate a vehicle to misbehave. The link to the next step after detection is to enable prevention; an effective IDS for cyber-physical systems should have an active response to cyberattacks.

As research in this field continues to progress, so will the attackers and their attacks. This progression requires continual updates to threat models to identify new vulnerabilities and attacks, and subsequent adjustments to IDS to counter them. The fundamental issue remains that CAN, while inherently insecure is a modern day vehicle standard, exemplifying the need for security should be addressed throughout the design process.

Table 3.1 Comparison of Proposed IDSs for In-Vehicle Networks

Detection Feature	**Proposed System**	**Intrusions Detected**	**Evaluation**
Message Frequency	Miller and Valasek (2016) [51]	Message Injection	Live Road Tests
	Hoppe (2008) [27]	Message Injection and Deletion	Testbench Simulation
Message Interval	Gmiden (2016) [21]	N/A	No Evaluation
	Song (2016) [76]	Message Injection	Live Road Tests
	Moore (2017) [57]	Message Injection	Real Vehicle Simulation
Signatures	Larson (2008) [41]	Known Attacks with Defined Signatures	Theoretical Simulation
Cyber-Physical	Cho and Shin (2016) [12]	Spoofing	Real Vehicle Simulation
	Ji (2018) [30]	Injection and Suspension Attacks	Testbench Simulation
	Choi (2018) [13]	Bus-Off Attack	Real Vehicle Simulation
Entropy	Marchetti (2016) [47]	Message Injection	Real Vehicle Simulation
	Müter (2011) [59]	Various Attacks that Effect System Entropy	Real Vehicle Simulation
CAN Fields	Matsumoto (2012) [50]	Message Spoofing	No Evaluation
	Markovitz (2017) [48]	N/A	Real and Simulated CAN Traffic
Sensor Data	Müter (2010) [60]	Message Injection	No Evaluation
Deep Neural Network	Kang and Kang (2016) [32]	Attacks Based off Statistical Features	SW Simulation with OCTANE

CHAPTER 4. OUR APPROACH TOWARDS AUTOMOTIVE INTRUSION DETECTION

I showed that numerous works utilize CAN message timing intervals for intrusion detection [76, 57]. In this section of the paper, I present my adaptation of current methods, their limitations, and how I improved upon current methods while developing a new anomaly detection method.

4.1 Introduction

This chapter investigates the performance of an anomaly detector based on message timing features. The key idea of this detection approach is to model CAN message timings for every message ID to describe normal vehicle behavior. The goal is to design a detector that detects changes in message timing from normal as an indicator of an anomaly.

This chapter examines the performance of different message timing based anomaly detectors. I evaluate the effectiveness of my approach using real-world CAN data. The main contributions of this chapter are:

1. I investigate multiple message timing features and model normal CAN message timings using message interval and frequency.

2. I develop an intrusion detector based on message frequencies across different driving modes. The model characterizes message timings for detecting message injection attacks.

3. I prototype and evaluate the performance of the detection system using simulated CAN logs and logs generated from a real vehicle.

4.2 Message Timing Detection

I investigate the most common type of attack, message injection, in this paper. The attack can take control of a vehicle's driving operations. The basic idea is that an attacker transmits a packet with the same ID as a legitimate packet as soon as the legitimate packet is transmitted. This will cause the other ECUs on the CAN bus to use the data from the attacker instead of the legitimate one. Even when an attacker is injecting messages, the ECUs still send their messages periodically. Eventually, the rate of messages on the network will be increased by a factor of more than 2 to 100 times, depending on the attacker's injection speed. Miller and Valasek reported that they needed to inject at a rate of at least 20 times faster than normal for their attack to be successful [52]. Because the original ECU is still transmitting its message, an attacker needs to send messages at a fast enough to overwrite the normal message with the same ID.

Detection is based on the following two principles.

1. When a new message is transmitted on the CAN bus, the IDS will check the CAN ID and compute the time interval from the arrival time of the latest message with the same CAN ID.

2. If the time interval of the new message is shorter than the normal model, the IDS indicates new message as an anomalous message since this message is arriving sooner than expected.

These detection principles are possible due to the predictable periodic nature of CAN. A conceptual diagram on the effects of message injection attacks on normal traffic is given in Figure 3.1.

The detection process is detailed in Figure 4.1. Every CAN message is read and its ID captured. The time interval from current to previous message with the same ID is calculated and compared to the normal model timing interval for that message. If the interval time is less than normal, the IDS will indicate an anomaly and the IDS sends an alert. The anomaly detector usually has two modes - one mode to capture the normal timing intervals for every CAN message and another intrusion detection mode that observes current CAN traffic and compares its message timings to the normal model to detect anomalies. To train the normal model, a few seconds of normal, no anomalies CAN traffic, is used to train the IDS. For each CAN ID, the IDS will calculate the interval time that is

defined as the difference from the previous message time to the current message time, as shown in Equation (4.1). The average interval time is determined in Equation (4.2).

$$\Delta_1 = t_1 - t_0, .., \Delta_n = t_n - t_{n-1} \tag{4.1}$$

$$\mu = (\sum_{i=1}^{n} \Delta_i)/n \tag{4.2}$$

My normal model consists of the average message interval times for every CAN ID. Anomalous messages are detected when the interval time is less than the normal time. Shorter interval times indicate additional messages on the bus.

4.3 Message Interval Evaluation

To demonstrate the utilization of message intervals as the detection feature, the traffic capture of normal operation of two test vehicles was analyzed and the normal message interval timings for all CAN messages were calculated. The data set for our first test vehicle is the live vehicle capture data published by Miller and Valasek [51]. This data set contained the CAN traffic of their test vehicle during normal operations. The message injection attacks are simulated by injecting specific messages to create data sets of malicious CAN traffic. During simulations, I can control the rate of injection to allow multi-rate injection attacks and evaluation of the various injection rates.

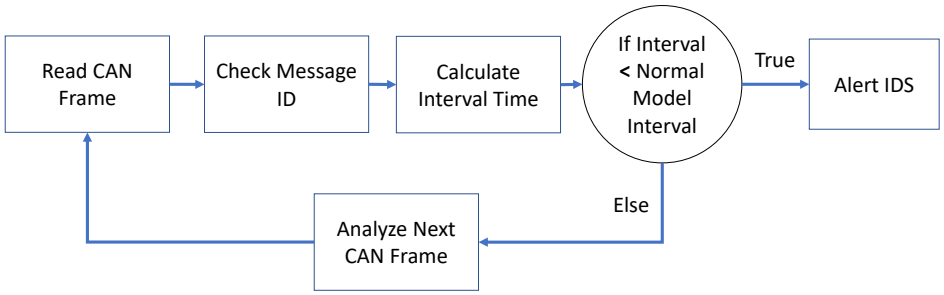

Figure 4.1 Every CAN message is read and its ID captured. The time interval from current to previous message with the same ID is calculated and compared to the normal model timing interval for that message. If the interval time is less than normal, the IDS will indicate an anomaly and the IDS sends an alert.

Additionally, I simulate single CAN ID injection along with multiple CAN message ID injection at multiple rates. Figure 4.2 illustrates the message injection on the message time interval.

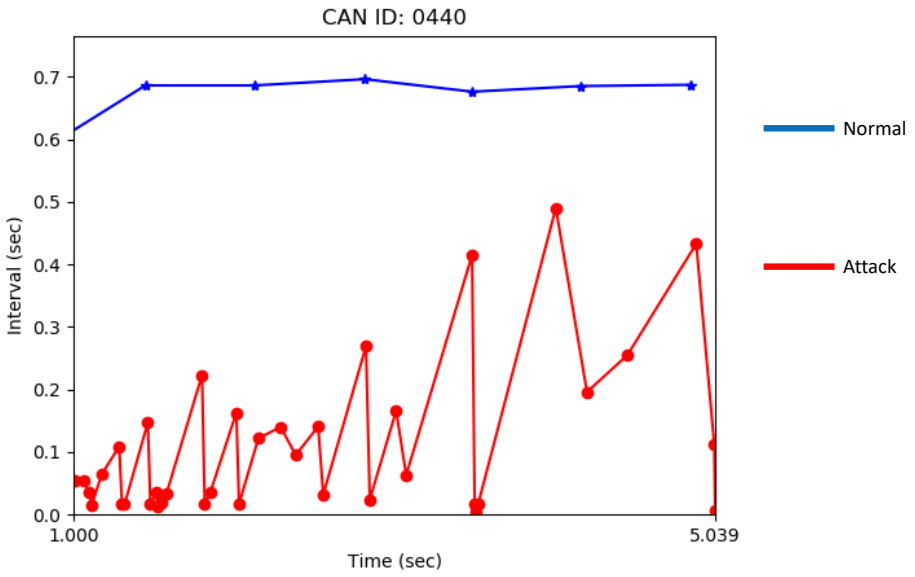

Figure 4.2 Demonstrating message injection attacks affecting the timings of messages. The simulated attack messages have fluctuating message timings and are shorter than normal depending on rate of injection.

To evaluate the IDS based on constant interval timings, the IDS is trained with normal operation CAN traffic to build a normal model for all CAN message interval timings. Each of our vehicle data sets had over 100 different CAN IDs. Then I applied this IDS against the simulated data sets of message injection attacks. It was tested against multiple rates of injection: 2, 5, and 20 times faster than normal values. It was reported that at least 20 faster rate was required for successful attack, but I tested for slower rates because it had been demonstrated it is possible to have a successful attack if rate of injection was 2 times faster, depending on vehicle and type of attack.

Table 4.1 reports the detection accuracy and false positive rates across the variety of message injection speeds and attacks. Detection accuracy is defined as the number of CAN messages that were correctly identified as having anomalous timings and the false positive rate is the number of incorrectly identified CAN ID messages out of the total messages. The results show that the faster the rate of injection, the higher the detection rate of anomalies. The false positive rate varies

Table 4.1 Detection Accuracy for Single and Multiple CAN ID Injection Attacks

Attack Type	**Message ID**	**Injection Speed**	**Detection Accuracy**	**False Positive**
Single Message Injection	0440	2×	75%	0%
		5×	94%	11%
		20×	99%	5%
Multiple Message Injection	0440 and 03D3	2×	0440: 75%	0%
			03D3: 68.75%	0%
		5×	0440: 94%	0%
			03D3: 97.5%	0%
		20×	0440: 99.5%	6.6%
			03D3: 99.4%	6.6%

depending on the rate of message injection and which messages get overwritten by the injected messages. There is an increase of false positives in both 5 and 20 times injection speeds, however it is seen that the false positives is higher in 5 times faster injection. This can be explained by how false positives is defined, when a message has anomalous timings but was not being injected, this is a false positive message. However, when an attacker is injecting messages, depending on which messages are being over written, this can increase or decrease the false positive rate because different messages have different sensitivity in timings. A message that is transmitted often has timing that is not as affected by message injection, however if a message that is rarely transmitted and misses a transmission, then this greatly alters the timing of said message.

The data set for the second test vehicle was obtained from a vehicle at Oak Ridge National Laboratory. The data set was constructed by logging CAN traffic through the ODB-II port of a real sedan while driving on a dynamometer system. The dynamometer system simulates real road operation. Attacks were performed by injection malicious messages at high rate to override normal vehicle operations. The malicious messages were constructed by sniffing and spoofing legitimate messages transmitted on the bus. I discovered there were two major issues in the message timing intervals for this second test vehicle.

1. There were several messages that demonstrated multiple timing intervals. It was assumed that CAN messages have a singular consistent message interval. Numerous previous works stated that all CAN messages have consistent timing intervals [9, 57, 52, 7].

2. A few CAN messages had message timing intervals vary depending on the vehicle's behavior. I expected some variation in the interval but the change that was seen was at times greater than twice the normal rate.

When the IDS was applied to the data set from vehicle 2, it generated 30 percent false positives, with 41 other message IDs detected as anomalous. These poor results were caused by the inconsistencies in the message timing intervals.

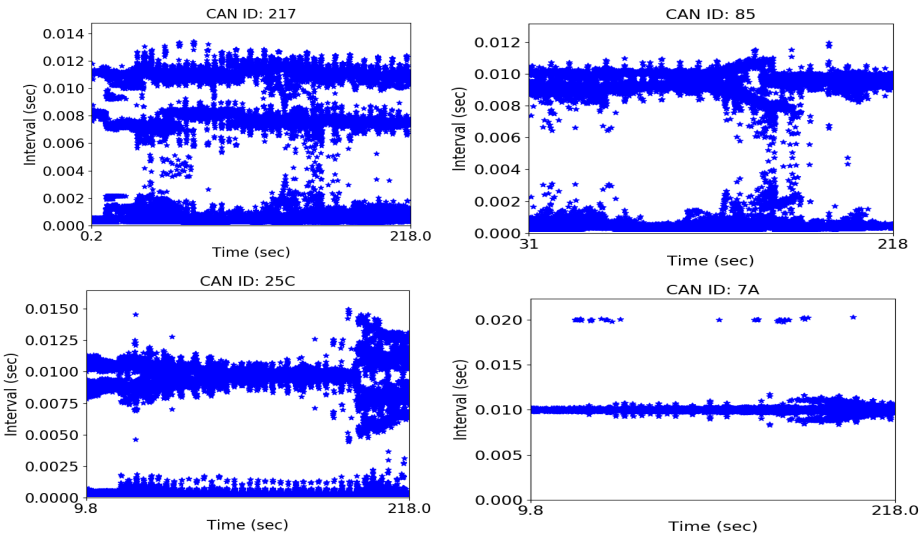

Figure 4.3 These plots depict the varying message intervals seen by multiple CAN messages. Some CAN messages show multiple distinct intervals while some vary due to changes in vehicle operations.

For example, in the CAN data set for the second test vehicle, several CAN messages showed non-periodic and multiple timing intervals. Figure 4.3 illustrates that multiple CAN IDs have varying message intervals. The data set for vehicle 2 had 137 different CAN messages, and about 30 of these messages showed the inconsistencies in the message timing intervals. The differences in CAN message timing behavior and required injection attack rates between the test vehicles may

indicated that CAN may differ depending on manufacturer and vehicle. This inconsistent behavior is problematic for anomaly detection algorithms based on the assumption of constant message timing intervals.

4.4 Message Frequency Detection

To address the aforementioned problem, it had been observed that the frequency of the messages remained relatively consistent, even while the CAN message intervals varied. Additionally, I noticed that the number of messages stays consistent even as the vehicle changes modes of operation. Thus, it was hypothesized the rate of messages is a better indicator for anomaly detection compared to message interval.

I proposed an anomaly detection method based on the message frequency - frequency (f) is equal to the rate of messages (m) transmitted in a set time interval (t) - as shown in Equation (f=m/t). When the frequency of messages increases by a factor of more than twice the normal value, an anomaly is indicated. As reported earlier, for an attack to be successful, the rate of injection must be at least twice the normal value. I implemented the frequency-based algorithm in our IDS. The IDS obtains a normal model for every CAN ID with the use of a few seconds of training data. The IDS then scans current CAN traffic and calculates the frequency of messages for every CAN ID. If the frequency of messages deviates at a rate of greater than 2 times normal, the IDS will indicate an anomaly for said CAN ID.

It should be noted that this approach is relatively simple in complexity as to be implemented in a simple ODB-II dongle. The dongle is likely to connect to the vehicle's OBD port and acts as an additional ECU on the bus. In addition, this proposed method is cost-effective since it does not require any modification to the native CAN bus.

4.4.1 Vehicle Modes of Operation

I examined the two causes of false positives and message timing irregularities: 1) Vehicle mode of operation and 2) CAN message behavior. CAN message timing intervals were affected by normal

Table 4.2 Different Driving Modes Message Count

Driving Mode	Time Duration (Seconds)	Message Rate (Msgs/Sec)
Key ON	10	4448
Key ON to Start	10	4432
Shift to Reverse	15	4356
Accelerate Reverse	10	4351
Decelerate Reverse	10	4353
Shift to Drive	10	4352
Accelerate	15	4413
Maintain Speed	10	4513

changes in the vehicle and generated false positives in anomaly detection. I chose to use message frequency as opposed to message interval to handle these normal behaviors and changes in the vehicle and vehicles themselves.

As the vehicle changed modes of operation, Table 4.2 shows the rate of messages varies at a rate less than 20 percent. The vehicle had an average transmission rate of 4402 messages per second through all modes of operation. It was demonstrated that a successful message injection attack will increase the rate of the injected message by more than a factor of 2. Therefore, the variability in rate of messages during normal operation is negligible when compared to the change caused by injection attacks.

Additionally, I examined the messages that had varying message intervals. As discussed above, the timing intervals of these messages demonstrate inconsistent behavior. There are certain points where the interval doubles its normal or there are multiple intervals. However, the frequency of these messages never varies more than 20 percent. This variation is within normal parameters. The dips in message rate may be attributed to the delay caused by higher priority messages on the bus as the vehicle is in operation and are not indicative of malicious anomalies.

4.4.2 Message Injection Attack

Knowing that during a message injection attack, the number of messages will increase and therefore the frequency of certain messages will increase. To demonstrate the change in frequency

during a message injection attack, I simulated an injection attack with the backup light message with ID of 202. Logging the CAN traffic of the test vehicle as it was driving on a dynamometer to simulate real road driving. The attacks were simulated by injecting malicious messages at a high enough rate to override the normal vehicle operations through the OBD-II port. It required a rate of at least 2 times faster than normal for the desired attack to be successful, as in the backup light turned on.

Figure 4.4 shows the captured normal traffic is represented in the blue. The frequency of this particular message was 200 messages per second. The captured message frequency when injecting messages is represented in red. Starting with 15 seconds of normal traffic, then injected messages for 15 seconds at a 2 times faster rate. Repeated this procedure of 15 seconds of normal traffic, followed by 15 seconds of attack traffic. Figure 4.4 shows that the frequency of messages doubles to 400 message per second when wmessages are injected.

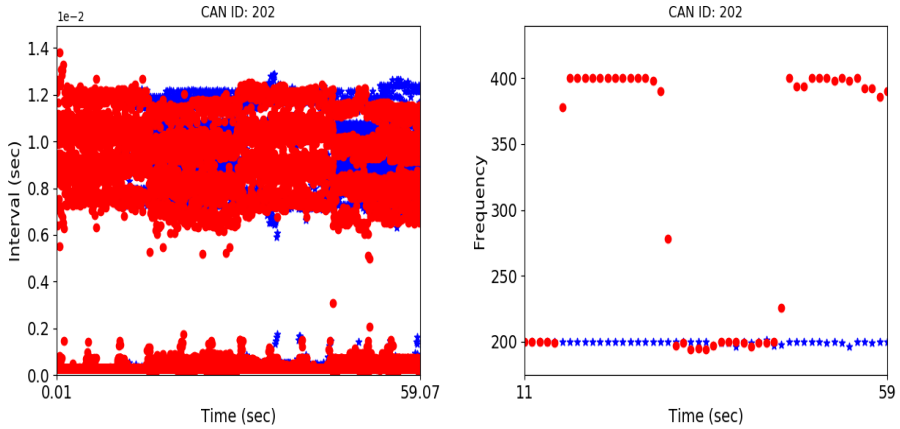

Figure 4.4 A comparison of the effect message injection attack has on the interval and frequency of CAN ID: 0x202. Blue is normal operation and Red is attack traffic. Left: Injection attack alters the interval of messages by shortening them. Right: It is clear when the message injection attack is occurring. The frequency doubles in rate during attacks.

Evaluating the message frequency-based IDS against the message injection attack for both test vehicles. Using normal traffic data as the training data, I captured the normal frequency for every CAN ID. The IDS then compared current captured traffic and calculated the frequency for every

message. If the frequency increased by more than 50 percent, an anomaly was indicated. The results are summarized in Table 4.3 that shows the frequency-based detection has better detection rates and maintains low rates of false positives compared to current interval-based detection approaches.

Table 4.3 Frequency Analysis of Message Injection Attack

Vehicle	Attack Type	Detection Type	Detection Accuracy	False Positive
1	Message Injection	Interval	75%	0%
		Frequency	100%	0.7%
2	Message Injection	Interval	96.9%	30%
		Frequency	100%	1.4%
Moore et al.	Message Injection	Interval	99.9%	0.29%
Seo et al.	Message Injection	Generative Adversarial Nets	96.5%	3.8%
Kang et al.	Message Injection	Deep Neural Network	97.8%	1.6%

I showed that the basic premise that all CAN messages have consistent timing intervals across different vehicles and modes is not always true. Thus, IDS approaches based on constant timing intervals are not reliable. I propose to address these issues with frequency based approaches. I showed my method can resolve the issues encountered by interval-based approaches and reduces false positive. Besides just detecting message injection attacks, the approach is able to handle timing variations from normal driving behavior and can be implemented on multiple vehicles. The method is light-weight as it does not require changes to the native CAN.

4.4.3 Frequency Analysis with Fourier Transform

I also performed a frequency domain analysis to assess the frequency components of the time series CAN data. If the CAN messages have a constant frequency, it should show a singular peak in the frequency domain. This frequency should change during an attack if it affects the CAN message transmission rates. The discrete Fourier transform is used to convert a sequence of discrete-time data to discrete data in the frequency domain. The Fast Fourier Transform is often used in completing a discrete Fourier transform.

The equations for Fourier Transform are given below, Equation 4.3 and 4.4. Fourier Transform takes a time-based pattern, measures every possible cycle, and then returns the overall cycles that

make up a signal. In this case, a Fourier Transform takes the CAN message signals and finds the speeds, amplitudes, and phases of each individual CAN message.

$$X_k = \sum_{n=0}^{N-1} x_n * e^{-i2\pi * kn/N} \tag{4.3}$$

$$x_n = 1/N \sum_{k=0}^{N-1} X_k * e^{i2\pi * kn/N} \tag{4.4}$$

Equation 4.3, adds up all the time spikes from a CAN message signal to determine the frequency of the message. Equation 4.4, determines the time point by adding up each frequency. I am breaking down the transmission signal of the CAN messages into time spikes.

The time series data for test vehicle 2 was converted into the frequency domain with the Fast Fourier Transform and investigated the effects of different driving modes and message injection attacks. Some data processing was needed to get the discrete CAN time signals into a state which could apply the Fourier Transform. The CAN time stamps were converted into a periodic square wave. The signal was held high during transmission of the specific CAN message and low when the message was not transmitting. The sampling frequency was (fs) as $fs = D/t$ with the number of data points (D) and the time interval (t). In the datasets, the amount of data points varies depending on the CAN message. Typically, it had anywhere from 2000 to 8000 data points depending on CAN message in the logs. With the timing interval, its average sampling frequency for CAN messages was 200 samples per second. One limitation in using the FFT function is that the number of data points must be a number that is a power of two. Some of the samples were padded with zeros to meet this requirement.

Converting the time series data into periodic square waves allowed us to apply FFT to the data. I chose this representation of the data to simulate the transmission and hold times of each CAN message being transmitted on the bus. However, perfect square waves with equal magnitude correspond to a FFT output with multiple frequency peaks. Figure 4.5 shows that a square wave decomposes into multiple sine waves at varying magnitudes. Messages that have a single consistent message interval showed this ideal FFT behavior. However, messages that demonstrated multiple timing intervals had more complicated FFT output, as seen in Figure 4.6. It shows that multiple

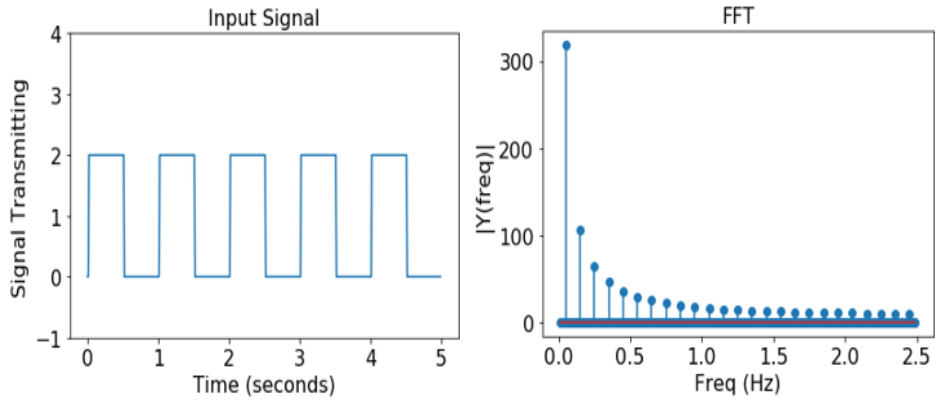

Figure 4.5 Left plot depicts an ideal square wave, which is representative our converted CAN message data. The messages get transmitted at a fixed interval and have a certain transmission time represented by the duty cycle of the wave. The right plot shows the Fourier transform of the square wave.

factors of CAN messages may affect the output of the Fourier Transform, including rate of transmission, interval behavior of messages, changes in driving mode, and normal variations in normal timing.

To conduct a frequency analysis against an attack, the same injection data from vehicle 2 was used. During a message injection attack, the frequency of messages is increased. However, it was observed during an attack, the frequency peaks stayed at the same frequencies while the magnitude of the peaks changed. An attack is just a duplication of the normal message signal but with a phase shift. Examining the frequency plots of normal and attack traffic in Figure 4.7, the magnitude of the frequencies are increased in the attack as there is now an an additional attack signal.

Additionally, I created a spectrogram as a visual representation of the frequencies as they vary of time. Creating a spectrogram using FFT is a digital process. By digitally sampling the data, in the time domain, the data is split into chunks and Fourier transformed to calculate the magnitude of the frequency spectrum for each chunk. Each chunk corresponds to a vertical line in the image. By examining the normal and attack plots, it can be seen that the lower frequency ranges, 200 and 500 Hz, seem to lose strength during an attack.

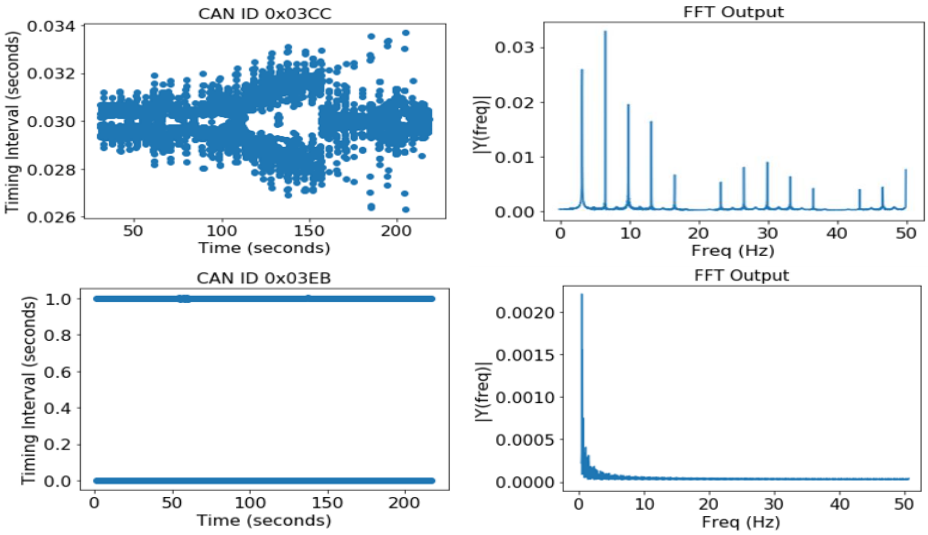

Figure 4.6 On the left side there are two example plots of CAN messages with irregular timing intervals. The right side shows their respective FFT outputs

These results open the possibility that frequency domain analysis is able to discern hidden features that are not present in simple timing analysis. Further investigation is needed to determine the applicability of frequency domain analysis towards intrusion detection.

4.5 Summary

In this chapter, I presented an anomaly intrusion detection system to identify message injection attacks on the CAN bus. The approach is based on modeling CAN message timings using message intervals and frequency to detect timing changes and intrusions in the CAN bus. Experimental results have shown that the detection system can efficiently detect injection attacks with high detection accuracy.

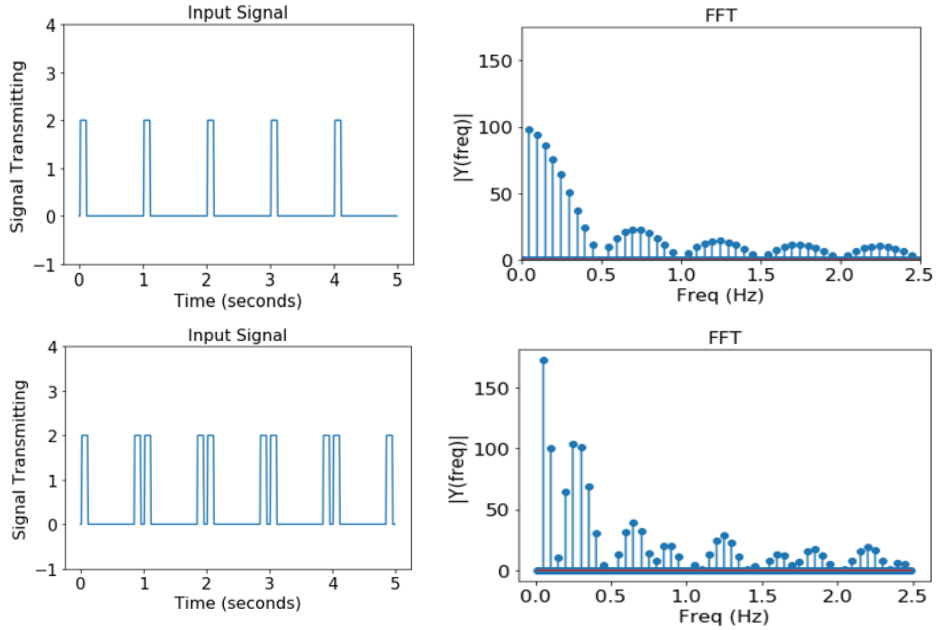

Figure 4.7 These plots show the change in the input signal and FFT output during a message injection attack. It can be seen as the message rate doubles, the magnitudes of the frequency peaks increase. The increase in magnitude is caused by the doubling of singles with the same frequency.

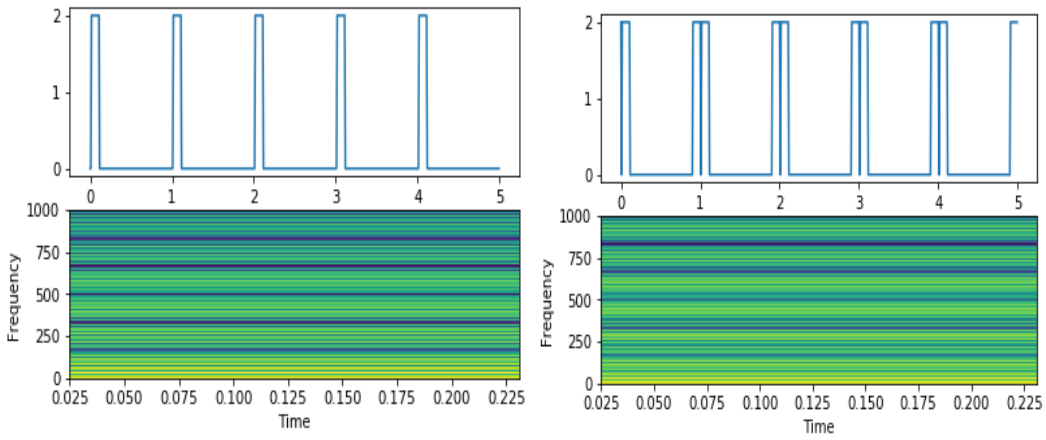

Figure 4.8 Spectrogram Plots. Left side plots normal message signal, and the right side plots a 2x message rate injection attack.

CHAPTER 5. HYBRID AUTOMOTIVE INTRUSION DETECTION SYSTEM

This chapter introduces a novel intrusion detection system called HAIDS. This stands for Hybrid Automotive Intrusion Detection System, an IDS that uses both anomaly and signature based detection. A limitation of my anomaly detector is its limited ability to only detect attacks that alter CAN message timing. In traditional network security, different anomaly detection approaches are combined together to form a combinational system that provides increased attack coverage. Building on this idea and propose a hybrid automotive intrusion detection system that combines anomaly with signature based detectors. Portions of this chapter have been published in [95].

The contributions of this chapter are:

1. An anomaly detector that models CAN message timings to detect anomalies in the CAN message traffic.

2. A signature detector that complements the anomaly detector. This detector uses different observers to detect attack signatures observed on the CAN bus.

3. A hybrid automotive intrusion detection system that combines the anomaly detector and signature detector to detect various types of attacks and intrusions.

4. Evaluation and comparison of HAIDS against different types of anomaly detection using real CAN logs generated from sedan vehicles.

5.1 Anomaly Detector

This detection method is used to detect new and previously unknown attacks by capturing a normal model of the system and identifying variances from normal. However this method can suffer from high rates of false positives. As having demonstrated with the message timing-based anomaly

detector, it can detect new attacks that affect CAN message timings, but can suffer from false positives. This type of detector will provide attack coverage of attacks that have not been defined by signatures. To supplement and increase detection accuracy, I propose combining the anomaly detector with a signature-based detector.

5.2 Signature Detector

Signature-based intrusion detection has been shown to be effective in detecting known attacks with high true positive rates while maintaining near zero false positives. This detector works with a knowledge base of signatures or rules that it scans the system for. When the detector identifies a signature, it will alert an anomaly. However, this type of detection mechanism is unable to detect any anomalies outside of the pre-defined database of attack signatures. Thus, by combining it with anomaly detection, HAIDS will be able to detect both unknown and known attacks. I propose using behavior-based detection techniques to identify anomalies difficult to detect using anomaly detection. These anomalies do not alter message timings, and as such will be missed by the message timing-based detector. I divide behavior-based signature detection into two classes.

5.2.1 Event-based Observers

Classifying these observers to detect point anomalies, which do not have a sense of time, and are typically binary in character (True or False). An example of a rule would be defining a characteristic of a vehicle, such as engine temperature, and if the characteristic's value falls outside a certain range, then an intrusion is indicated. While some of these rules will apply to all vehicles, the multitude of options in vehicles makes it difficult to predict what rules will apply to which vehicles. This complexity of vehicles means the rules will need to be tailored for every vehicle make and model. I try to focus on universal vehicle characteristics in creating the rules for the signature detector.

5.2.2 Context-based Observers

These observers can detect contextual and collective anomalies by taking in multiple inputs. Contextual anomalies are anomalies that occur in the context of a given scenario that by themselves are not considered an anomaly, but given certain circumstances are anomalous. Collective anomalies are multiple events that are not anomalies individually but when they occur together are anomalous. These detectors look into the past to determine what is occurring and predict what should happen.

The more up-to-date, specific, and well-defined the rules are, the greater detection rate and accuracy, while maintaining low false positives. It must be acknowledged that there is still much unknown in regards to signature detection. Vehicles have unique CAN signals, rule set need to be defined to separate acceptable driver behavior from attack behavior. As I delve deeper into the attack and threat models, the rule generation will continue to develop to better detect and classify intrusions.

5.3 Multiple Detection Approach

Our IDS uses multiple detectors to detect different types of intrusions. The use of timing-based detection will detect anything anomalous effecting message timings on the CAN bus, whether they be a true anomaly or not. The behavior-based intrusion detection will provide accurate detection of attacks that have been pre-defined into the rule database. The goal is to use multiple signature detectors to detect a variety of intrusions that have been identified, creating a comprehensive intrusion detection system.

These three components: timing signatures, behavior signatures, and logs of CAN traffic, work together to form our IDS as illustrated in Figure 5.1. The aim is to show that a multi-faceted detection system is better than any singular solution in detecting a wider range of anomalies. The goal is to use the IDS to mitigate the vulnerabilities of CAN. While we cannot change the topology of CAN, our IDS identifies anomalous packets even without message authenticators. Due to the unencrypted nature of CAN, its messages can be sniffed and spoofed. The IDS aims to identify these

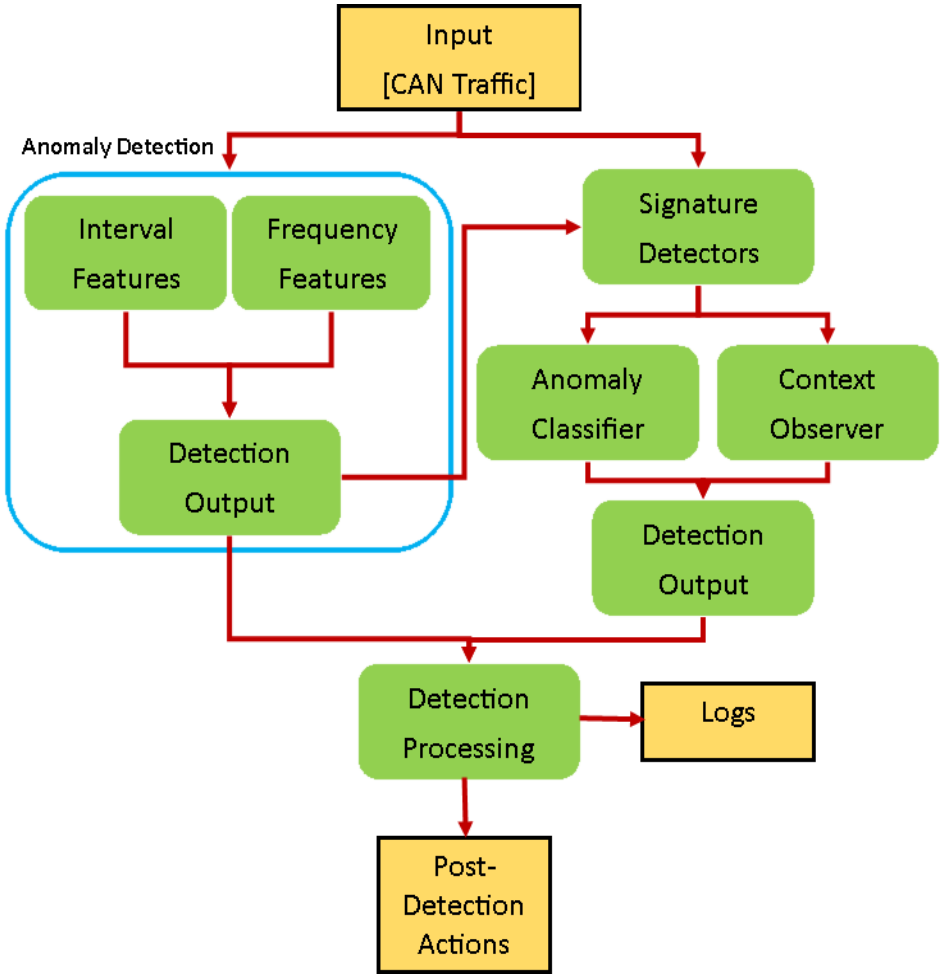

Figure 5.1 Automotive Intrusion Detection System

spoofed messages by detecting message injection. The strength of our security IDS is dependent on its detection efficiency and attack coverage, which is evaluated in the following section.

5.4 Evaluation

I propose and describe an evaluation criterion for our IDS to illustrate its effectiveness and performance of each component individually and combined.

5.4.1 Evaluation Setup

The IDS analyzes traffic capture during normal operation, starting with the live vehicle capture published by Miller and Valasek, to calculate the normal message timings for every CAN message ID. Next simulating a message injection attack by adding injected messages into a normal traffic capture. Simulating multiple injection attacks, from single and multiple CAN ID injection to specific messages specified by their CAN IDs. HAIDS would calculate and compare the current messages timings for every CAN message ID. If current message rate varied outside a range from normal, an intrusion is declared and logged. I had a second dataset from Oak Ridge National Labs. They captured normal driving and injection attack data from a live vehicle in a lab. The results of the anomaly detector are presented in sections 7 and 8.

To simulate the attack signatures for the signature detector, I utilized the data set from Miller and Valasek. They had reversed engineered the CAN IDs to correspond to their respective vehicle operations. Using this knowledge to create data sets that simulated the desired attacks. I created two attack data sets to simulate diagnostic message injection and the denominational attack of vehicle accelerating and car door opening.

5.4.2 Signature Anomaly Detection

To evaluate event-based signature anomaly detection, the initial step is to define the rules that guide observers in recognizing anomalies. I identify which CAN packets are important for detection and what they do in a vehicle. In addition, besides just identifying anomalies, I also identify sequential anomalies. Knowing that diagnostic message should not occur during normal operation, so checking that none of the packets had message ID of 07, which indicates a diagnostic message, to evaluate the event-based observers. The IDS had 100 percent accuracy in detecting event-based attacks that was injected during evaluation. To evaluate the context-based observers, I defined anomalies in this category. Deciding that vehicle doors should not open while vehicle is accelerating. I combined the detection methods to identify the signatures of car door opening and acceleration. The CAN IDs for both actions were identified and when they occurred together, this

as determined to be anomalous. The IDS was successful in detecting when the door opened while the car was accelerating. The signature detector was perfect in its detection and with zero false positives. This is due to the fact that detection is trivial given appropriate signature definition. As further exploration of the threat model, and as attacks develop, researchers will be able to generate more attack signatures, further enhancing event-based observers. Other possible attack signatures could include: limits on top speed, rapid changes in steering wheel angle, and other vehicle metrics.

5.4.3 Hybrid Anomaly Detection

My hybrid intrusion detection system stands out due to how it integrates anomaly detection with signature detection. HAIDS goes beyond just using different detection approaches separately, it feeds the results from the anomaly detector into the signature detector so that it can differentiate anomalies from intrusions. There are anomalies that occur in systems normally, by combining the detectors, it can identify when these anomalies are malicious and classify as intrusions.

I had reported earlier that the anomaly detector based on message timings struggles with messages that are not consistently periodic, and this behavior has multiple causes. These inconsistencies caused false positives in our anomaly detection. However, because I was able to determine the causes of some of these irregularities and attack features, it is possible to write signatures to wrap around the anomaly detector. This reduces the false positives from normal anomalies and is better able to identify intrusions, as seen in 5.1.

The difference between normal system anomalies versus malicious message injection anomalies lies in the difference in message behavior. While both alter message timing features, attacks have a defined duration of messages with altered timings. While a normal system anomaly will alter some message timings irregularly. An attacker needs to inject messages for a minimum duration and rate for the vehicle to behave maliciously. Knowing this fact, the signature detector is able to look for multiple and a series of anomalies to determine when an anomaly is an intrusion. Also by ignoring normal system anomalies, this is able to decrease the false positive rate of the anomaly detector without decreasing accuracy of detecting intrusions.

Table 5.1 HAIDS

IDS	Detection Accuracy	False Positive
Frequency	98.945%	40.58%
HAIDS	100%	1.4%

HAIDS uses the anomaly detector to detect any anomalous timings that alter the frequency of CAN messages, the signature detector is able to determine which of these anomalies are intrusions. By defining certain attacker intrusion characteristics that HAIDS will look for when examining the anomalies detected to identify attacks. This system is unique in how each detector complements each other by working together to detect attacks.

5.4.4 Implementation

The plan is to implement the hybrid intrusion detection system as a hardware security module, such as a OBD-II dongle that directly interfaces with the in-vehicle network. OBD port is standard in all modern vehicles, this ensures compatibility of the solution to a wide range of vehicles, while minimizing changes to the in-vehicle network. HAIDS acts as another ECU on the CAN bus, and therefore receives all messages transmitted on the bus. HAIDS can then monitor and analyze all traffic to detect any anomalies.

5.4.5 Summary

In summary, I present HAIDS, an approach for detecting intrusion in in-vehicle networks using a combination of anomaly and signature detection. The anomaly detection is developed through observing message timings and creating a normal model of message frequencies. Detection of anomalies is achieved when an attack alters these frequencies. The signature detection is based on a variety of different signatures captured by our observers. The key strength to the approach is that by combining different detection approaches, creates a more thorough and comprehensive detector.

CHAPTER 6. MACHINE LEARNING TOWARDS REVERSE ENGINEERING CAN MESSAGES

In this chapter, I present work that focuses on the reverse engineering and classification of CAN messages. The problem is that even though CAN is standardized, the implementation may vary for different manufacturers and vehicle models. These implementations are kept secret, therefore CAN messages for every vehicle needs to be analyzed and reverse engineered in order to get information. Due to the lack of publicly available CAN specifications, attackers and researchers need to reverse engineer messages to pinpoint which messages will have the desired impact. The reverse engineering process is needed by researchers and hackers for all manufacturers and their respective vehicles to understand what the vehicle is doing and what each CAN messages means. The knowledge of the specifications of CAN messages can improve the effectiveness of security mechanisms applied to CAN.

6.1 Background

This section introduces a high-level description of reverse engineering and machine learning as applied towards the controller area network.

6.1.1 Reverse Engineering

Reverse engineering CAN is a relatively simple process, but is extremely time consuming and labor intensive. Current techniques rely on manually analyzing the traffic and require knowledge of the inner workings of the vehicle and its various systems. Hermans et al. [25] and Huybrechts et al. [28] outline a technique for locating features in source code and applying it to CAN data. The proposed reverse engineering process consists of five steps.

1. Research: 1st step is investigate how the car is built and what functionality it has, ABS, traction control, etc. It is also recommended to look up which network these functions are located and the interconnections between these networks.

2. Interfacing: 2nd step is interfacing with the CAN bus. In this step, one needs to collect as much data as possible. Storage of these messages is needed for further analysis. Interfacing with the CAN bus is typically done through physically connecting with the OBD port or another indirect connection.

3. Identifiers: 3rd step is to link the found CAN IDs to their corresponding ECUs. Combining this step with the previous step provides information on which signal corresponds to which message. 2 main methods are proposed for this step. 1st is to manually unplug each ECU one by one and note which IDs disappear from the network. However, this may change the behavior of the other ECUs. The other method is to insert gateway nodes between target ECU and rest of network. This allows for observation of the data flow.

4. Analysis: 4th step is finding the essential data. In this step, correlation between data points and features are found. This is a tedious step of monitoring all data to find its correlation to messages.

5. Unknown data: The final step is analyzing the unknown messages. Combining IDs with possible messages, while reducing the combinations as more IDs get linked.

The effort here is to automate these steps as much as possible.

6.1.2 Machine Learning

Machine learning algorithms excel in prediction and classification. They are generally used to investigate the correlation between different inputs, also known as features, to approximate an output or discover interesting connections. Machine learning algorithms can be broken up into two main categories:

- Supervised learning: Machine learning algorithm that has a training set that includes both the input data and output data. The algorithm is trained to learn a mapping of inputs and outputs.
- Unsupervised learning: Machine learning algorithm has a training set that only has input data but not the expected output.

The classification of CAN messages is a time series classification problem. There exists many classification techniques, but most of them are designed for problems with multiple input features and an corresponding class. The issue with time series data is that these features are not independent from each each other. However, it is still possible to use classification techniques by extracting different features based on the series and using those for classification.

6.2 Related Work

Cyber attacks towards modern vehicles executed by injecting spoofed messages to the CAN bus has spawned numerous research efforts towards improving the security of modern vehicles. Koscher and Checkoway et al. [38, 9] were the first to demonstrate physical and remote security breaches in automotive systems. Both research groups injected malicious messages to take control of various vehicle systems such as brakes and engines. Valasek and Miller [51] demonstrated real-world attacks on multiple vehicles via remote hacking to cause a Jeep to brake and crash into a ditch. Their work lead to a Chrysler recall of 1.4 million vehicles.

To detect attacks, some works use machine learning approaches. Kang et al. [32] trained a deep neural network structure to classify normal vs attack messages using probability-based features of the message bits. Using normal and attack data, the system was able to be trained to recognize specific attacks. Taylor et al. [81] used a long short-term memory networks to detect attacks on the CAN bus. Their approach was applied to the identifier, and learned to predict the next message ID. This method assumes repetitive periodic sequences of messages and highly surprising message bits were flagged. Narayanan et al. [61] proposed building a Hidden Markov Model of normal behavior of the vehicle using sensor values. Their work showed it is possible to detect

data manipulation attacks. They focused on the signal changes rather than the signal values. The result was a model that served to detect signal jump types of anomalies. Kang et al. [33] proposed an automated process for correlating CAN messages with its ECU by creating a machine learning classifier trained on multiple vehicles. Their approached utilized the time stamp, CAN ID, and data fields and determined that a nearest neighbor approach would have the best success as a classifier. They evaluated multiple machine learning approaches using the same training data to draw their conclusion.

Lestyan et al. [43] proposed a method of identifying and extracting vehicle sensors from raw CAN data to infer personal driving behavior. Their approach examined each message data bit for the probability that it was 1. Using random forest, their algorithm output a binary value indicating whether the message belonged to a certain class or not.

These detection approaches require the analysis of raw CAN messages by manually inspecting high volumes of data to reverse engineer message syntax and semantics to reconstruct the vehicle operations and contextualize the messages. Literature on network traffic analysis and automatic recognition of the nature of given network packets already exists [100, 98, 92, 93, 91, 88, 71, 73, 70, 67, 56, 53, 26, 20, 63]. However, these works are inapplicable to the automotive network as CAN has different protocols. CAN messages do not include source nor destination addresses nor port numbers. IT networks also have a clear separation between network and application layers which is not the case with CAN.

There is research in the reverse engineering and translation of CAN messages. Marchetti and Stabili [46] proposed READ, a novel algorithm for automatic reverse engineering of CAN data frames. READ analyzes traffic traces containing unknown CAN messages to identify and label the signal type based on their data frames. Their method isolates counters and cyclic redundancy checks (CRCs), among other values, to label the signals. Verma et al. [86], proposed a simple algorithm to extract CAN message signals and label them using OBD-II PIDs. Their algorithm, ACTT: Automotive CAN Tokenization and Translation, leverages diagnostic information to parse CAN by breaking messages into tokens and then learning the translation from bits to vehicle function.

Their signal extraction only identifies signals that do not have a contiguous set of bits. Pese et al. [65], developed a tool called LibreCAN, which translates CAN messages. This tool captures the bit-flip rate of messages and uses them along with body data from a phone to classify messages.

The main limitation of these works is the few features that can be extracted and analyzed from CAN traffic. This is due to the fact that a vehicle is typically required to manually reverse engineer the CAN IDs and to validate results. This issue can be mitigated if access to the complete specifications of CAN messages were available. However, as CAN IDs and their functions are manufacturer trade secrets, it is difficult to reverse engineer CAN messages. Additionally, manufacturers have an incentive to prevent reverse engineering of their CAN messages, as it is the main deterrent for hackers. This paper proposes the novel contribution of reverse engineering CAN messages without the need for a physical vehicle.

6.2.1 Ground Truth Data

6.2.1.1 DBC file

A .dbc file (database for CAN) contains the translation for all CAN messages in a particular vehicle. This file is typically held secret by manufacturers and varies per make and model. DBCs contain the signal definitions, segment position (start and end bits), binary to decimal encoding scheme, and the conversion information for translating the decimal to meaningful physical value. The DBC also includes message timing information such as frequency and whether the message timing is constant or triggered by an event, and the corresponding ECU(s) responsible for a message. Nearly all research requires reverse engineering some of the information of DBCs.

6.2.1.2 J1939

CAN data can follow the Society of Automotive Engineers standard SAE J1939 [72], which is the vehicle bus recommended practice for vehicle component communication and diagnostics. SAE J1939 has been widely adopted by diesel engine manufacturers for use in large tractors and trucks. This standard defines that all J1939 packets, except for the request packet, should contain eight

bytes of data and a standard header which contains a Parameter Group Number (PGN), that is embedded in the message's 29-bit identifier. A PGN identifies a message's function and associated data. J1939 attempts to define standard PGNs to encompass a wide range of automotive and other vehicle purposes.

6.3 Proposed Work

In this section, I propose to reverse engineer CAN messages without manually testing on a physical vehicle and without the need of a CAN (DBC) file. I describe the process of extracting CAN message features and their utilization in machine learning techniques. I utilize machine learning techniques to reduce the obstacle of reverse engineering CAN messages by identifying important CAN messages and their functions. The contribution of the work can be broken down into:

- A supervised learning approach that extracts and identifies CAN data changes and labels specific IDs corresponding to vehicular functions.
- An unsupervised clustering approach that classifies unknown CAN messages using the labels learned with the supervised learning approach. Unknown CAN functions are extrapolated from known CAN messages.

The work provides an early demonstration that by utilizing machine learning techniques, it is possible to reverse engineer CAN message functions without needing to manually test against a physical test vehicle. The work indicates that the absence of a DBC file does not necessarily prevent a researcher or attacker from deriving the meaning of individual CAN messages.

6.4 Classifying Controller Area Network Messages

6.4.1 Feature Extraction

6.4.1.1 Message Feature Extraction

First needed was to extract the CAN fields and features required for the approach, as machine learning approaches require specific input feature formatting to properly generate classification. A simple a CAN message parser that extracts CAN IDs, time stamps, and data fields for every message was developed. These CAN fields are used to create ordered dictionaries and lists of CAN messages with the same ID. For the feature engineering process, it was determined it was pertinent to choose features that are standard to all vehicles. The CAN fields the parser extracts are in all CAN messages regardless of vehicle's make or model. Additionally, these CAN fields are used to extract other features such as message timings and distances.

6.4.1.2 CAN Function Extraction

Without the full CAN specification, researchers need to reverse engineer CAN messages to determine their functions and respective data details. The typical method of reverse engineering this information is to manually test individual CAN messages on a physical vehicle. I propose an alternative method to reverse engineer CAN messages with the use of labeled time frames that correspond to specific vehicular functions and a data change algorithm that detects CAN messages with changing data bits. I hypothesize that there are specific CAN messages that correspond to specific vehicular functions. For example, when a vehicle is braking, the brake controller is active, therefore a certain ECU is transmitting braking signal to the brake from the brake pedal. The data field of this certain CAN ID is changing as the brake pedal pressure is increasing and the brake is being engaged. I utilize a CAN data bit change detector in combination with labeled CAN logs to reverse engineer a few CAN messages.

The supervised learning approach takes as input a labeled CAN data set that detailed what the vehicle was doing in certain time frames. The approach monitors all the CAN messages in

the time frame for data bit changes. The reasoning is that data changes in CAN messages correspond to vehicular functions. Identifying which CAN messages have data changes during different vehicle functions can identify the CAN messages that relate to specific vehicular functions. The contribution here is that given minimal knowledge on what the vehicle is doing in certain time frames and by detecting CAN messages with changing data bits, it is possible to reverse engineer some CAN message functions. The current method of reverse engineering CAN messages requires physical testing of a vehicle or obtaining the full CAN message specification, our approach provides an alternative method of deriving CAN message functions. The data change detection approach is described in Algorithm 1.

Algorithm 1: Obtaining data values for CAN data frame and checking for data bit changes

```
Data: CAN Data Raw
Result: List of CAN IDs and training features
training features = ["CAN ID":[], "Time Stamp":[], "Data":[], "Vehicle Function:[]]
for Every message in CAN Log do
    Parser(message): To obtain individual CAN fields
    for Every CAN ID do
        Create dictionary of CAN fields and data
        Continuously monitor data field bits for changes
    if message data bit changes then
        Record CAN ID and vehicular function
    else
        pass
Return CAN data fields and training features
```

The algorithm associates labels with CAN messages with respect to their vehicular function. CAN function extraction is important in the next step of the approach as it provides CAN message labels to help classify unknown CAN messages. Given that the data sets are unlabeled, this approach provides the labeling needed for the machine learning approaches.

6.4.2 Clustering

The goal of clustering is to identify structure in an unlabeled data set by organizing the data into groups where within-group-object similarity is minimized and the between-group-object dissimilarity is maximized. Most clustering analysis has been performed on static data. Data is static if all their feature values do not change with time. Han and Kamber [24] classified clustering methods developed for handling various static data into five categories: partitioning, model-based, grid-based, density-based, and hierarchical methods.

Partitioning methods build k partitions from the data. Each partition represents a cluster containing at least one object. The partition is crisp if each object belongs to exactly one cluster, or fuzzy if one object is allowed to be in multiple clusters. Two known methods for partitioning are the k-means algorithm [45], where each cluster is represented by the mean value of the objects in the cluster and the k-medoids algorithm [35], where each cluster is represented by the most centrally located object in the cluster.

Model-based methods assume a model for each of the clusters and attempts to best fit the data into the assumed model. There are two major approaches of model-based methods: statistical and neural network approaches. An example of a statistical approach is AutoClass [10]. This uses Bayesian statistical analysis to estimate the number of clusters. Two methods of neural network approaches are competitive learning [6] and self-organizing feature maps [37].

Grid-based methods quantize the object space into a finite number of cells that form a grid structure, this is where the clustering is performed. STING is a typical example of a grid-based approach, it uses several levels of rectangular cells corresponding to different levels of resolution. Statistical information regarding the cell attributes are pre-computed and stored. A query process usually starts at a high level of the hierarchical structure. Each cell in the current level computes its confidence interval to the given query. Irrelevant cells are removed from further consideration. This query process continues to the next lower level until the bottom layer is reached.

Density-based methods, such as DBSCAN [17], grow a cluster as long as the number of objects(density) in the "neighborhood" exceeds some threshold. Another approach, OPTICS [2],

computes an augmented cluster ordering for automatic and interactive clustering analysis. The ordering contains information that is equivalent to density-based clustering.

Hierarchical clustering method works by grouping data into a tree of clusters. There are generally two types of hierarchical clustering: agglomerative and divisive. Agglomerative methods start by placing each object in its own cluster, and then merges clusters into larger and larger clusters. This process ends when all objects are in a single cluster or until specific end conditions, such as desired number of clusters, are met. Divisive methods do the opposite of agglomerative. Divisive methods start with a single large cluster that contains all the objects, and then splits clusters into separate clusters until a specific end condition is met. Hierarchical clustering is unable to make adjustments once a merge has happened. To improve the clustering, there is a trend to combine hierarchical clustering with other methods. Chameleon [34] and CURE [22] both perform careful analysis of the object "linkages" at each split or merge.

CAN data is not static however, but is an example of time series data that has values that change with time. Similar to static data, time series clustering requires an algorithm that forms clusters given a set of unlabeled data. Most methods try to modify existing approaches for clustering static data in a way that converts time series into a form of static data. I propose utilizing clustering towards classifying unknown CAN messages. Clustering machine learning techniques assume that instances of a particular class have data profiles that cluster into centroids. Each new data point can then be classified according to its distance from that centroid. Clustering approaches can be applied to unsupervised learning.

6.4.2.1 Hierarchical Clustering

I utilize hierarchical clustering towards reverse engineering CAN message functions. A basic understanding of how K-means clustering works is needed to explain hierarchical clustering. K-means can be broken down into these sections:

1. Decide number of clusters (k)

2. Select random points from the data as centroids

3. Assign all points nearest cluster centroid

4. Calculate centroid of new clusters

5. Repeat steps 3 and 4

This is an iterative process, it keeps running until the centroids of newly formed clusters do not change or the max number of iterations is reached. The issue with K-means clustering is that it always tries to make the clusters the same size. Additionally, the number of clusters needs to be defined at the beginning of the algorithm. Realistically, the number of clusters is unknown. Hierarchical clustering aims to address these issues. This is an example of unsupervised learning. I implement agglomerative clustering, a type of hierarchical clustering, to cluster CAN messages together by related vehicular functions. Each CAN ID is assigned as a single cluster and I iterate through the IDs and their respective distance measurements, then CAN ID clusters are merged starting with the nearest neighbor. By grouping CAN IDs together, related CAN IDs are clustered together by related function. Therefore, if certain CAN IDs have a known function, the functions of unknown CAN IDs can be determined.

I combine a supervised learning approach that reverse engineers some CAN ID message functions and then using these extracted labels towards classifying other unknown CAN message functions with the use of agglomerative clustering.

6.5 Reverse Engineering using Machine Learning

The current method of reverse engineering CAN messages requires physical testing of a vehicle or obtaining the full CAN messages specification. Manufacturers treat the full message specification as trade secrets, so this is an approach that is difficult to take. The other approach requires manually injecting individual messages and observing the changes in the vehicle to determine each message function. My contribution is a process that simulates the physical vehicle message injection, given minimal knowledge on what the vehicle is doing at certain time frames and by detecting which CAN messages have changing data bits. I describe the approach in Algorithm 2.

Algorithm 2: Get data values for CAN data frame and check for data bit changes

```
Data: CAN Data Raw
Result: List of CAN IDs and training features
training features = ["CAN ID":[], "Time Stamp":[], "Data":[], "Vehicle Function":[]]
for Every message in CAN Log do
    Parser(message): To obtain individual CAN fields
    Time Stamp = msg.TS
    Message ID = msg.ID
    Data Fields = [msg.D1, msg.D2, msg.D3, msg.D4, msg.D5, msg.D6, msg.D7, msg.D8]
    for Every CAN ID do
        Create or update dictionary of CAN fields and data
        ["CAN ID": msg.ID, "Time Stamp":msg.TS, "Data":Data Fields]
        Continuously monitor data field bits for changes
    for Every Data Field in CAN ID message do
        if new Data Field not equal to past Data field then
            Record CAN ID and vehicular function
            training features.update("CAN ID" = msg.ID, "Time Stamp" = msg.TS,
             "Data" = Data Field, "Vehicle Function" = Current Vehicle Action Label)
        else
            pass
Return CAN data fields and training features
```

The core of the work is to reverse engineer CAN messages with respect to their vehicular function. Raw CAN is unlabeled and requires proprietary information from the manufacturer to reverse engineer. This is a major impediment to researchers and attackers, and the aim is to address this obstacle. The work utilizes raw CAN data, captured from a variety of vehicles to derive CAN clusters. The aim is to circumvent the need for a physical test vehicle and manufacturer specific data towards reverse engineering CAN messages.

The process involves reading CAN data, calculating the distances between all CAN IDs, merging nearest IDs together to form clusters, and using these clusters to classify unknown CAN IDs. I utilize Euclidean distance as the distance metric between CAN messages.

$$Message1 = m1 = (x_1, y_1)$$

$$Message2 = m2 = (x_2, y_2)$$

$$Euclideandistance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

The CAN IDs with the smallest distance are merged and this process iteratively continues until there are k clusters left. The clusters left are grouping the different CAN messages together by their respective functions.

The work is two-fold, the labeling of the clusters and the visualization of clusters. A tool utilized for the clustering is a dendrogram, Algorithm 3. This is a tree-like structure that records the sequences of merges. It is used to visualize hierarchical clustering and determines the optimal number of k clusters. The tree-like structure records the sequences of merges of the leaves, in this case the CAN messages. In addition to depicting how the CAN IDs merge, the dendrogram also details the distances between IDs and clusters.

The clustering approach creates clusters of CAN messages that are related by their vehicular function. Unfortunately, without labels for these CAN IDs, it is not possible to determine what these clusters represent, as they are just groupings of unknown CAN messages. To solve this issue, the CAN message labels previously derived are utilized in the supervised learning approach. Using these labels, which relate specific vehicular functions to certain CAN IDs, are applied labels to unknown CAN messages.

Algorithm 3: Pseudocode for creating a dendrogram and returning predicted data clusters

```
Data: Distance values
Result: Dendrogram and cluster labels
#Create distance matrix;
matrix = DataFrame(squareform(distance values));
#Create dendrogram;
dendrogram = scripy.dendrogram(linkage(distance values));
#Create clusters;
cluster = AgglomerativeClustering();
cluster.predict(matrix);
Return dendrogram and clusterings
```

Algorithm 4: Clustering CAN IDs and labeling of messages

```
Data: CAN Data Raw
Result: CAN Message Clusters
["Cluster":"Label", "CAN IDs":[]]
for Every message in CAN Log do
    Parser(message): To obtain individual CAN fields
    for Every CAN ID do
        Calculate the distance of this message from every other message
    for Every Message do
        Merge closest CAN IDs
        Continue merges til K clusters are created
Return CAN data fields and training features
```

In the following section, I will analyze the CAN labels and the clusters that our approaches generated. I detail the process of verification and validation of the reverse engineering process. I demonstrate that the algorithms are correlating CAN messages to specific vehicular functions, and that these correlations can create labels for the CAN messages. I am also able to cluster CAN messages by their relative and related functions. Combining the clusters and labels enable reverse engineering of CAN message functions.

6.6 Evaluation and Analysis

I discuss the evaluation criterion for the CAN message classification using clustering machine learning. I analyze the CAN labels and the clusters that the approach generated and detail the process of verification and validation of the reverse engineering process. I demonstrate that the algorithms are correlating CAN messages to specific vehicular functions, and that these correlations can create labels for the CAN messages. I am also able to cluster CAN messages by their relative and related functions. Combining the clusters and labels enable reverse engineering of CAN message functions.

6.6.1 Data

For training machine learning algorithms, CAN data was required for the research, we obtained CAN traffic logs from Oak Ridge National Labs. They had a vehicle set up on a dynamometer, a treadmill system that simulates real road operation. CAN traffic was captured by connecting through the On-Board Diagnostic port under the dash of the vehicle. The data captured represents the vehicle under-going different driving modes, from accelerating to braking. The CAN data was captured and logged into comma separated values in an excel file. While this format was good for human readability, it had to be converted into a data frame understood by the machine learning algorithms and visualization functions.

Additionally, we had captured CAN data from a John Deere tractor has it tillage various locations in Iowa. This CAN data followed the J1939 standard, which gives more detailed information

on the CAN data that was not present in the Oak Ridge data set. The difference between J1939 standard data and the data captured by Oak Ridge, is that the CAN IDs are unknown in the Oak Ridge data, while the J1939 data details what each ID represents. This J1939 data was used as a ground truth in our evaluation procedure because this data had the reverse engineered information we were trying to extract from raw data.

6.6.2 Simulation

I created a simulated data set of 10 different CAN IDs, simulating a driving mode and a braking mode. This data log had messages with different frequencies and message intervals to simulate real data. There were 5 IDs specifically related to driving and another 4 IDs related to braking, with 1 message that was transmitted during both modes. This data was used to verify and validate the algorithms as a control group. Given that raw CAN data is dependent on the vehicle it was captured from, the simulated data gave a baseline to understand the workings of the algorithms when applied towards raw captured CAN data. Running the simulated data through the clustering algorithm to classify the CAN IDs into their vehicular mode clusters.

The clustering algorithm and dendrogram required a distance matrix to be created. This matrix represents all the distance metrics for the CAN IDs, Figure 6.1. The matrix was passed into our dendrogram creator and clustering algorithm to cluster the CAN messages. The dendrogram, Figure 6.2, is created from the leaves first to create the cluster tree. The dendrogram shows that there are two clusters of the CAN IDs, which is expected since I simulated two separate driving modes of braking and accelerating. Our clustering algorithm confirmed the dendrogram results. CAN IDs 1-6 were designated as accelerating and were clustered into one group, and CAN IDs 7-10 were designated as braking and clustered into another group. The simulated data was used to confirm the hypothesis and approach before applying them to real CAN data.

	10	1	3	2	5	4	7	6	9	8
10	0.00	0.75	0.65	0.70	0.50	0.55	0.20	0.25	0.05	0.15
1	0.75	0.00	0.10	0.05	0.25	0.20	0.55	0.50	0.70	0.60
3	0.65	0.10	0.00	0.05	0.15	0.10	0.45	0.40	0.60	0.50
2	0.70	0.05	0.05	0.00	0.20	0.15	0.50	0.45	0.65	0.55
5	0.50	0.25	0.15	0.20	0.00	0.05	0.30	0.25	0.45	0.35
4	0.55	0.20	0.10	0.15	0.05	0.00	0.35	0.30	0.50	0.40
7	0.20	0.55	0.45	0.50	0.30	0.35	0.00	0.05	0.15	0.05
6	0.25	0.50	0.40	0.45	0.25	0.30	0.05	0.00	0.20	0.10
9	0.05	0.70	0.60	0.65	0.45	0.50	0.15	0.20	0.00	0.10
8	0.15	0.60	0.50	0.55	0.35	0.40	0.05	0.10	0.10	0.00

Figure 6.1 Simulated distance matrix representing all distances between each CAN ID.

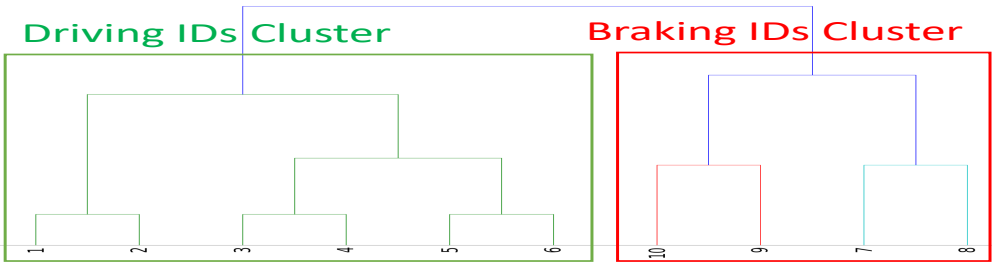

Figure 6.2 Dendrogram for simulated data. This figure shows how each CAN ID clustered with the other IDs. It also shows the two major groupings for the CAN IDs.

6.6.3 Oak Ridge Real CAN Data

The 2nd data set was CAN traffic logs from Oak Ridge National Labs [62]. They had a vehicle set up on a dynamometer, a treadmill system that simulates real road operation. The CAN traffic was captured by connecting through the On-Board Diagnostic port under the dash of the vehicle. The data was not reverse engineered, as the CAN messages are unknown, but it captured the vehicle driving in different modes, from accelerating to braking. This data was labeled based on the vehicle operations. Typically CAN data captured is not reverse engineered, and to reverse engineer this data requires manual injection. Each captured CAN message is injected into the vehicle to see how the vehicle responds. Our supervised learning approach simulated this process by detecting CAN messages with changing data bits due to different vehicular driving modes. Upon examining our data set, it was determined there were 133 unique CAN IDs. Applying the approach, I aimed to identify the significant CAN IDs and correlate them to specific vehicular functions.

Using Algorithm 1 and 2, these CAN IDs were identified, Table 6.1, and reasoned that they have these specific functions. This conclusion was derived from a combination of knowing what the

Table 6.1 Reverse engineered CAN IDs and their corresponding vehicular function

Vehicular Function	CAN IDs
Drive	17C
Brake	467, 3C3
Shifter	171, 230
Windows and door lock	331, 332, 333

vehicle was doing at specific times in the log and utilizing the data change algorithm to detect which IDs were changing during these times. By identifying these CAN IDs, it enabled assignment of labels for the clusters generated with our clustering algorithm. With more data, the algorithms would improve and be able to identify more CAN ID labels. The methodology applied here to derive CAN message functions can be applied to various different vehicles as it does not require any vehicle specific information on its messages.

The machine learning portion of the work clustered all the CAN IDs together. To accomplish this, a distance matrix was created for this data, this returns the respective distances between all the CAN IDs. The distance matrix was then used to create a dendrogram that visualizes the clusters and determines K for the number of ideal cluster labels. The dendrogram, Figure 6.3, showed 4 major clusters. When applying the clustering algorithm to the data set, it returned 4 clusters, determined by the dendrogram. The dendrogram and clustering algorithms clustered the driving and braking CAN IDs in different clusters. CAN ID 17C for driving and IDs 230 and 171 for gear shifter are in the green cluster, whereas braking 467, 3C3 and other uncommon functions such as door lock were clustered in red.

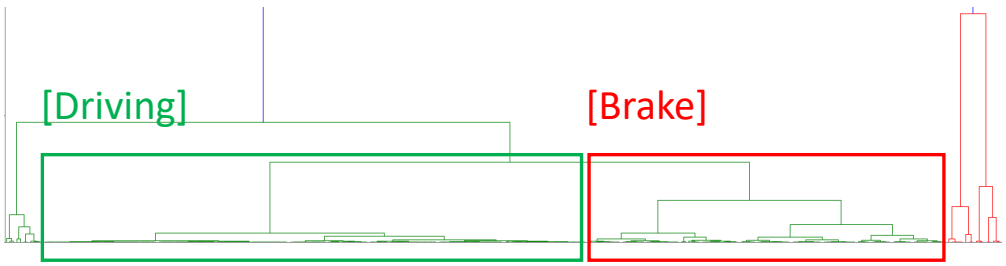

Figure 6.3 Captured raw data dendrogram, clustered the CAN IDs into 4 major groupings.

Cluster Label (Vehicle Function)	CAN IDs
0 (Driving)	'216', '217', '214', '213', '37B', '3CC', '92', '326', '336', '40A', '483', '366', '265', '422', 415', '416', 3B6', '410', '411', '596', '312', '43D', 43E', '82', '83', '81', '85', '3B3', '24A', '3B7', '24C', '24B', '3B8', '368', '369', '367', '423', '365', '581', '42C', '440', '42D', '440', '42D', '242', '59E', '2A1', '434', '430', '453', '455', '333', '332', '179', '178', '175', '171', '4B0', '185', '78', '91', '165', '166', '167', '25B', '25C', '25A', '17C', '33B', '35E', '156', '3AB', '3AA', '230', '7D', '7A', '5B3', '47', '42', '41', '20A', '471', '14B', '352', '200', '202', '204', '350', '77', '76', '485', 484', '4B', '4C', '4A', 486', '3A8', '476', '477', '474', '475', '2EC']
1 (Brake)	'439', '454', '3C7', '3C3', '43C', '3EB', '3EA', '84', '2F1', '447', '446', '386', '431', '331', '482', '465', '467', '466', '488', '38D', '3B4', '472', '473']
2	'5A5', '5B5'
3	'86', '87', '4BE', '4BF', '600'

Figure 6.4 Cluster labels from Oak Ridge

Using these labels, it classified the functions of unknown CAN IDs. To check the clusters, a few other CAN IDs had to be reverse engineered manually. CAN ID 430, was identified as shifting to drive. This ID was clustered together with with shifter (171, 230) and drive (17C). CAN ID 465 was identified as shifting to reverse. This ID was clustered together with brake (467, 3C3). This results demonstrates that even given raw unlabeled CAN data, it is possible to reverse engineer some CAN messages and using these known IDs, extrapolate other unknown IDs.

While these results are promising, the lack of reverse engineered data made verifying the results challenging. To address this issue, CAN data that followed the J1939 standard was obtained. This standard means that the PGNs are well-defined and therefore could validate the clustering results. J1939 messages have an identifier, PGN (parameter group number), that defines the message function. Where automotive CAN messages identify the message and the sending ECU, J1939 messages define the message function and the source of the message.

Table 6.2 Reverse engineered J1939 CAN IDs and their corresponding vehicular function

Cluster Label	CAN Function
0	Driving Functions (Engine and brake controllers)
1	Proprietary IDs
2	Memory Access/Data Transfer IDs
3	Fuel IDs
4	Electrical Power and Lighting

6.6.4 J1939 Data

The 3rd data set was CAN data captured from a John Deere tractor as it performed tillage operations at different sites. This data set is raw, has no labels, and contains over 300 CAN IDs. However, because this data followed the J1939 standard it allowed for greater detailed analysis. I were able to evaluate the following:

- Accuracy measurements: Determine the percentage of correctly and incorrectly classified CAN messages
- K clusters: Determine the effect that increasing K has on the CAN message classifications
- Message correlations: Determine how the messages are correlated together

Starting with the same algorithms and processes detailed above for the first 2 data sets, a distance matrix was created for this data. This distance matrix was then used to create a dendrogram to visualize the CAN clustering and determine the number of cluster labels. The algorithm returned five clusters, Table 6.2. The largest cluster contained all the driving critical functions, such as braking and engine controller. The other four clusters were much smaller and contained proprietary IDs, electrical power/lighting messages, memory accesses, and data transfer IDs in separate clusters.

I further examined the accuracy of classification for each cluster and the overall clustering of all messages.

- Cluster 0 contained driving function, engine and brake controller, messages. In this cluster there were 217 messages. 50 messages relating to engine/speed/brake, 15 messages relating to

the aux valve, 15 messages relating to aftertreament and fluids, 87 proprietary messages, 24 miscellaneous messages such as diagnostics, data and time, and air pressure, and 29 unknown messages.

- Cluster 1 contained only proprietary messages. There were 11 messages all label proprietary.
- Cluster 2 contained memory access messages. 8 messages in this cluster all relating to memory access and data transfer.
- Cluster 3 contained fuel related messages. 7 of 8 messages in this cluster related to fuel, with 1 unknown message.
- Cluster 4 contained messages related to electrical/power/lighting. In this cluster 13 messages related to light and electrical power, 14 proprietary messages, 6 miscellaneous messages such as diagnostic, turbo, switch, and 10 unknown messages.

Overall, after removing all unknown and proprietary messages from the cluster, resulted in 102 out of 135 correctly classified messages.

Using CANoe, a J1939 CAN analyzer, we were able to check the correlations between different CAN signals to verify the message classifications. CANoe enabled the extraction of specific signals; such as the accelerator pedal position, engine rpm and fuel rate, to analyze their correlations to each other. Seen in Figure 6.5, that as the accelerator pedal was pushed, the engine rpm and fuel rate increases. This is expected as this is real world vehicle behavior. Checking on each individual classification, it is seen that accelerator pedal was in cluster X, engine rpm in cluster X, and fuel consumption in cluster X. This shows that the algorithm was clustering related messages together. CANoe enabled further signal analysis, to examine more CAN messages and evaluate their relationships to each other.

Additionally, to further investigate whether increasing the number of clusters would give further insight and details on the relationships between messages. By increasing K, I aimed to identify more specific clusters. By going down the levels in the dendrogram, it determined that the next level of clusters were at K equals to 8 and 11. Analysis of messages when K equals 8, Table 6.3.

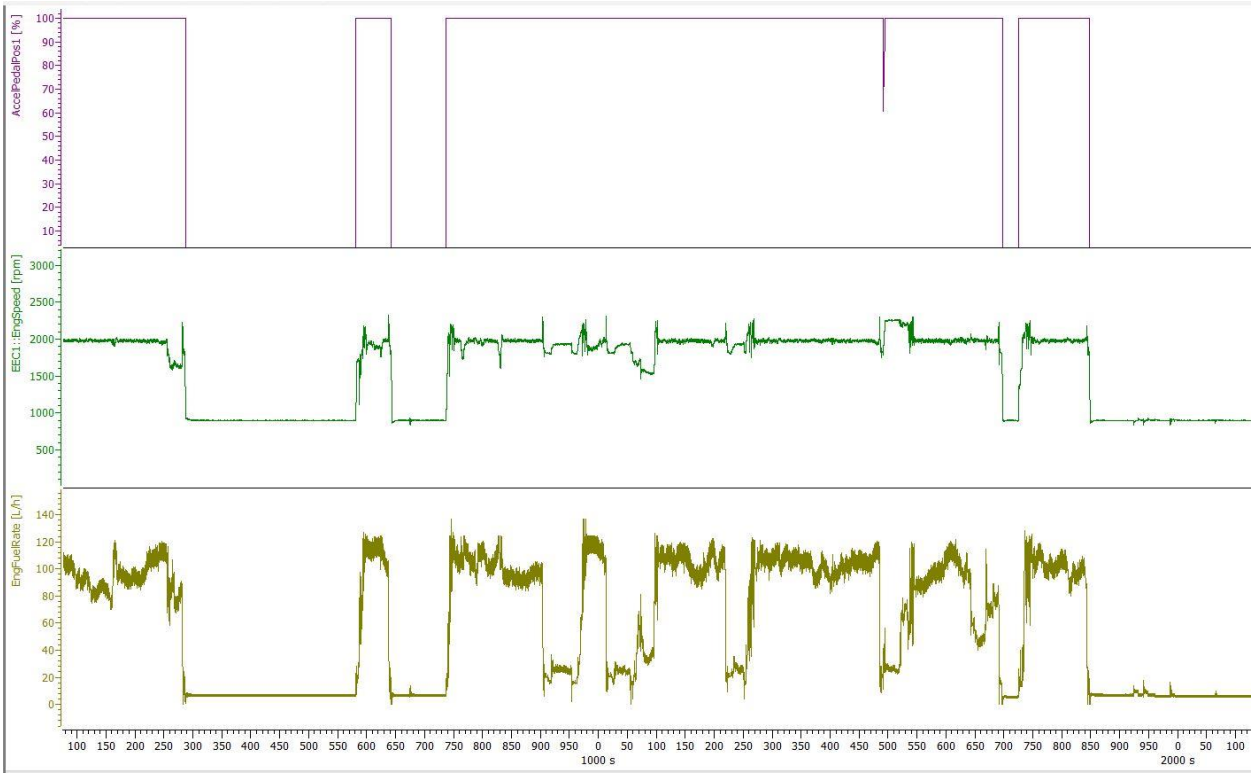

Figure 6.5 CANoe plots with 3 different yet related signals. Showing the relationship between accelerator pedal and engine rpm and fuel rate.

This resulted in 93 out of 133 messages classified correctly and 30 incorrectly classified messages.

What was seen was that non-important message clusters got split whereas the major vehicle function clusters remained consistent. These results also show that even having the full database file for CAN and knowing the full specification of J1939, there are still unknown messages that aren't defined in J1939 and there are numerous proprietary messages. These undefined messages make it difficult to fully reverse engineer all CAN messages. These complications increase in automobiles that do not follow the J1939 standard.

However, these results show that even with different CAN logs from different types of vehicles and following different standards, our algorithms were able to classify, identify, and label specific CAN IDs by their respective functions. The algorithms do not require insider knowledge nor reverse engineered data to classify CAN IDs. The only input required was raw CAN data logs. The results

Table 6.3 Reverse engineered J1939 CAN IDs and their corresponding vehicular function, with k = 8

Cluster Label	Message Breakdown
0:Engine and brake controllers	72 Messages: 36 vehicle control messages, 14 aux vale messages, 9 aftertreatment and fluids messages, 24 miscellaneous (diagnostics, data and time, air pressure) messages, 3 vehicle power messages
1:Proprietary IDs	8 Messages: 8 proprietary messages
2:Data Transfer IDs	5 Messages: 5 data transfer messages
3:Fuel IDs	2 Messages: 2 fuel messages
4:Electrical Power and Lighting	18 Messages: 13 electrical light messages, 5 miscellaneous (diagnostic, turbo, switch) messages, 10 unknown messages
5:Fuel IDs	3 Messages: 3 fuel messages
6:Memory Access IDs	3 Messages: 3 memory access messages
7:Mix Messages	27 Messages: 13 fluid and gas messages, 4 speed control messages, 10 electric and power messages

indicate that machine learning algorithms are able to classify CAN messages using raw CAN logs as the input.

6.7 Summary

Through the analysis it was determined that there are 3 different levels of data abstraction:

1. Raw data: unlabeled and undefined CAN messages
2. Message definition: known message IDs and corresponding functions
3. Vehicle definition: known vehicle actions respective to time

Raw data is the most simple level of data to obtain. Researchers and attackers can easily obtain raw data through online shared log files or sniff vehicle CAN traffic. The next level is to have the message definition for the messages in raw data. This is difficult to obtain as there are many different message definition depending on a variety of factors such as CAN standard, vehicle make, model,

and manufacturer. To obtain this level of abstraction typically requires reverse engineering of each message. Lastly, vehicle definition can be directly obtain by observing and controlling what the vehicle is doing when the CAN data is collected or may be inferred by analyzing and understanding the CAN messages that are being sent at given time frames. To infer vehicle definition using message would require having the message definitions. What the work does is assist in abstracting out these different levels from CAN data. The function of unknown CAN messages is determined through a combination of supervised and unsupervised machine learning techniques to determine message and vehicle definitions.

CHAPTER 7. MACHINE LEARNING TOWARDS AUTOMOTIVE SECURITY

Intrusion detection systems detect attacks by looking for suspicious system activity. Tradition IDSs commonly detect known threats based on defined rules or behavioral analysis through base-lining the network. However, as attackers become more sophisticated, they can bypass these techniques, so there is a need for more intelligent intrusion detection. One way to address this issue researchers are exploring is to apply machine learning towards intrusion detection. In this section of the paper, a background on the different types of machine learning techniques will be detailed.

Machine learning techniques can be categorized as either supervised or unsupervised algorithms. These two categories of algorithms can be broken down into further sub-categories. Supervised methods are dependent on having lot of labeled data to train the algorithm what inputs correspond to what outputs. Two subgroups of supervised learning is regression and classification. These algorithms are very accurate in their predictions of the output given enough labeled input data.

- Regression: These algorithms predict a continuous quantity. Labeled data sets give "correct" input and output pairs. The algorithm then identifies what input features predict the desired output.

- Classification: These algorithms also require labeled data sets. These data sets assign observations into discrete categories. If input x has label y then the output is z.

Un-supervised methods do not require pre-labeled data. The input data is clustered or grouped together naturally. Clustering is the main approach of un-supervised learning. The input data is grouped together in clusters depending on similarity of input and distance of variables. The main advantage of these methods is that it does not require pre-labeled data, which is difficult to come by and very time consuming to obtain.

The goal for applying machine learning toward vehicle security is to learn features that can model a vehicle's or driver's normal behavior. The anomaly detector then detects deviations from this normal behavior and classifies it as an attack. The aim is to apply these principles into HAIDS. HAIDS is essentially a classifier that currently classifies CAN messages by their timings. With the use of machine learning techniques the aim is to model vehicle driving modes by classifying CAN IDs and grouping them together by their respective vehicular function. Then the anomaly detector detects attacks that alter normal vehicle and driver behavior.

7.1 Related Work

7.1.1 Modeling Driver Behavior

Automotive driver modeling and fingerprinting is a popular area of research with numerous works reporting the application of machine learning towards identifying the driver of vehicles. Vehicle theft is a growing problem, and now with the integration of computer systems in vehicles, this is opening up new avenues of vulnerabilities for attackers. Driver identification is important to determine if the vehicle has been stolen. The following works use various methods to detect and determine the driver of the vehicle [36, 87, 85, 40, 31, 11].

Choi et al. [36] researched driver distraction detection and driver identification by examining driver actions using both Gaussian Mixture Model and Hidden Markov Model frameworks on CAN traffic. The authors evaluated the potential to identify when a driver was distracted among 9 drivers and achieved an accuracy of 31.45% using HMM. Their analysis showed that the average vehicle speed was lower when driver was distracted.

Wahab et al. [87] modeled individual driving behaviors and identified features that were effective at profiling drivers. They conducted feature extraction based on Gaussian Mixture Model and wavelet transform and showed that accelerator and braking were most effective at profiling drivers.

Van Ly et al. [85] proposed using inertial sensors to classify different drivers. Their goal was to use the inertial sensors in smart phones to identify drivers by driving features. The authors derived features from braking, accelerating, and turning to compare braking and turning actions

using k-means and SVM algorithms. They used a k of 2 and a simple distance metric. Braking was the most important feature to differentiate drivers.

Kwak et al. [40] modeled driver behavior to detect theft of vehicle using the various features such as: fuel trim, oil temp, wheel velocity, and fuel consumption. The authors examined various algorithms and determined that Random Forest had the best driver detection. The most important features were long term fuel bank trim and transmission oil temperature.

Enev et at. [15] classified drivers with driving data collected on live roads. They divided the range into two driving sections, running and parking. They derived brake pedal position, steering wheel angle, acceleration, turning speed, driving speed, current gear, engine speed, and throttle position. This work enriched the features derived from driving and parking. This is useful in SVM, Random Forest, Naive Bayes, and k-nearest neighbor algorithms.

Authentication based on driver driving patterns has been found to be effective in identifying and differentiating drivers. While these works have focused on characterizing driving behaviors, the aim is to show it can further be expanded toward security in connected cars. Features derived most were accelerator, brake, and steering wheel. I aim to enrich the feature set by examining CAN data from multiple vehicles and utilize them towards securing the vehicles from malicious attackers.

7.1.2 Machine Learning for Automotive Intrusion Detection

Kang and Kang [32] proposed a machine learning based IDS approach using deep neural network structure to monitor CAN packets. Their IDS consists of two modules. A monitoring module that decides a type of CAN packet based on trained features of known attacks. Once the monitoring module identifies a new attack, a profiling module records the attack model and updates the system for an upcoming packet. These two modules would be embedded in each ECU to analyze CAN packets. They used an unsupervised Deep Belief Net to capture underlying statistical features of CAN data and used them to classify messages as benign or anomalous. The CAN data feature extracted was the bit fields in the data field of CAN. Their approach counted the occurrences of the bit "1" in the data field. They reported a 99 percent detection ratio while keeping false positives

under 1 to 2 percent through the use of software simulation. However, the authors evaluated their approach using a simulator and not real normal and attack data.

Seo et al. [74] proposed GIDS (Generative Adversarial Nets based Intrusion Detection System), GAN (Generative Adversarial Net) based intrusion detection system, using deep-learning model, generative adversarial nets. Their proposed system extracted the patterns of CAN IDs from CAN data by converting the bits into images. GIDS has two discriminators to estimate the probability the message is real or fake. The authors showed that GIDS can learn to detect 4 unknown attacks using only normal data.

These two works extracted the bits from the CAN ID and data fields and modeled the behavior by statistical probability and patterns of the bits. When the bits demonstrated unexpected behavior caused by different attacks, an anomaly was indicated.

7.2 Threat Model

Modern vehicles have many embedded systems and technologies that connect externally. Once attackers get access to the internal system, they can manipulate the vehicle. Our threat model assumes the attacker has already accessed the vehicle's system by exploiting a vulnerability and is trying to make the vehicle misbehave. I do not consider sniffing of the bus as a malicious attack, as there are numerous 3rd party applications that connect through the vehicle's OBD-II port for system analysis. Therefore, the focus is on changes to the behavior of the vehicle caused by an attacker.

The work focuses on being able to detect when an attacker is attacking a vehicle. I do this by defining normal behavior of the vehicle and driver and differentiating attacker behavior from normal. It has been demonstrated that each driver has specific driving characteristics unique to them. It is reasoned that when an attacker is attacking a vehicle and trying to make it mis-behave, this will deviate from normal driver behavior. An attacker has numerous attack avenues, but each has similar end results of causing the vehicle to mis-behave.

7.3 Feature Engineering

By having a couple of raw captured CAN data sets from Oak Ridge National Labs and some labeled data sets from the Hacking and Countermeasures Research Lab. The different data sets required different pre-processing. Raw CAN is presented in hexadecimal form and needs to be reversed engineered to better understand what vehicular actions each CAN ID corresponds to and what their data fields are saying about these vehicle actions. I had to pre-process the CAN data to extract the features desired for the models. Also by defining when vehicle actions were occurring, as an example the braking action starts when the CAN ID for brake light is transmitted. Acceleration starts when accelerator pedal is pushed. These definitions are important to define such that it is possible to model normal driving behavior. This data is used for the normal modeling of driver behavior without fully labeled CAN data.

The HRCL data set was already reversed engineered, so the vehicle action and its respective data value were already converted to human read-able form. With this data I had to split it up for training and testing data. With data recorded from 10 different drivers I used 10-fold cross-validation. I used 9 drivers for training data and 1 driver for testing data. This data set will be used to test our anomaly detection. The reasoning being that an attacker essentially act as a different driver, hence if it is possible to differentiate different drivers, it should be able to differentiate when an attacker is attacking the vehicle.

Additionally, CAN values had to be normalized, Equation 7.1, for certain machine learning algorithms such as clustering. Vehicle values have a variety of ranges, by normalizing them it enabled consistency for distance measurements by making them all to be ranged between 0 and 1.

$$X_i = (x_i - min(x_i))/(max(x_i) - min(x_i)) \tag{7.1}$$

In choosing the features, because different vehicles may have a multitude of different sensors, the aim is to select common sensors in all vehicles, e.g. brake, accelerator, and wheel angle.

7.4 Clustering

Clustering is a popular approach that does not require a lot of pre-labeled data. Given that CAN signatures are dependent on the manufacturer and model, there is much variability in CAN and would require a lot of reverse engineering of CAN to obtain pre-labeled data. Also there is very little examples of live captures of attack data. Given these constraints on the data, it was chosen to use this unsupervised learning technique for the work. The aim was to answer the question do certain CAN messages cluster together and what extent can clustering be used to model normal traffic and detect attacks?

Clustering is the grouping of objects so that objects belonging in the same cluster are more similar to each other than those in other clusters. There are multiple types of clustering methods, but will detail a few of the most common ones.

- K-means clustering is one of the most common methods due to its simplicity and performance. It works by defining spherical clusters that are separable in a way so that the mean values converge towards cluster centers. The cluster centers are selected randomly in the beginning and recursively move until it is in the center of a its respective cluster. The K variable represents the number of clusters in the end.

- Agglomerative clustering is similar to k-means cluster but this time each node starts as its own center. Then pairs of clusters recursively merge together in a way that minimally increases a given link distance.

- K-Nearest Neighbor falls under the category of supervised learning as a classification algorithm. With a labeled training set, this algorithm can predict the class of new objects based on its distance from neighbors. The K variable represents the number of neighbors who have to say the new object is close.

7.5 Future Work

Driving consists of a small set of maneuvers that are made up of various actions. These maneuvers make up a sequence to take driver from point A to B. These maneuvers are defined and constrained by the laws and rules of the road. Related works have shown that each driver has a unique way of driving and this is reflected by driving features that can be extracted. Related works have also shown that these driving characteristics is distinct enough to differentiate drivers.

While CAN messages are easily sniffed, the differences in CAN messages between vehicles and manufacturers make understanding what is being transmitted difficult. A major obstacle in automotive research is reverse engineering CAN messages, from understanding what vehicle sensor each CAN ID corresponds to and to deciphering what the data fields feedback about these sensors. Manufacturers secure their vehicles through this obscurity, by keeping their implementation of vehicular CAN secret, this hinders mass attacks upon their vehicles. This requires researchers to reverse engineer CAN to create the required labels needed for machine learning evaluation and validation.

The proposed work is two-fold. First, proposed is to model normal vehicle and driver behavior with machine learning algorithms by classifying CAN IDs. Current automotive CAN work require manual reverse engineering of the raw hexadecimal CAN data specific to each individual vehicle. This is extremely time consuming and requires access to a test vehicle and closed track. To simplify this process by classifying CAN IDs to their vehicular function, broken up into 3 major categories: braking, acceleration, and turning. I detail how to utilize clustering techniques to accomplish this. There are numerous CAN messages being transmitted on the bus, many of these messages are noise and non-driving related. Related works have shown that only some specific features are important in modeling driver behavior. The work targets identifying these specific features.

Secondly, the aim is to utilize machine learning algorithms to detect variation in driving behavior to identify malicious intrusions. A motivation of attackers is to make the vehicle misbehave. The first part of the work models normal driving behavior using CAN features. Related works have shown that there are detectable variations in driving features that can differentiate drivers. Building

on this idea, and hypothesize that there are detectable variations in driving features during an attack. The aim is to extract driving features and detect variations in these features using machine learning algorithms for intrusion detection.

7.5.1 Intrusion Detection

An extension of our work in modeling normal driver behavior is detecting deviations in this behavior. The goal of attacker is to make the vehicle misbehave, whether it be by injecting messages, replaying messages, or taking over the vehicle. This deviation in driver behavior can be detected using machine learning algorithms. Utilizing the normal driver behavior model built in the previous section, this will enable the detection when the driver behavior changes due to malicious attackers.

We will build a model of CAN IDs respective to their vehicular function. Knowing when certain CAN IDs occur during a vehicle, it should be possible to detect when CAN IDs deviate from their normal clusters. Mathematically this may be represented by a distance measurement for a CAN ID being greater than a certain threshold from its cluster. Using Equation (4), each CAN ID should be within a certain distance from the center of its respective vehicular action. When this distance is greater than a threshold value then the IDS will detect an anomaly. We will use the mean and standard deviation of the feature values from the HCRL data set for clustering and intrusion detection. This anomaly is seen when the data for a certain CAN ID is changing with a cluster of CAN IDs that it normally doesn't change with. For example a set of CAN IDs correspond to braking, and the IDS detects a new CAN ID that is changing while braking is occurring.

To evaluate the detection of malicious attacks using our machine learning-based detection, the HCRL data set will be used. Knowing HAIDS can detect message injection attacks with great accuracy and low false positives. But what it the attacker has control of the vehicle and does not need to massively inject messages? With the HCRL data set, it has the driving profile of 10 different drivers. Using 9 of them to train the normal profile and use the data from the last driver to simulate an attacker. It is reasoned that an attacker essentially acts as a different driver, by

trying to make the vehicle mis-behave. We will evaluate how accurately the approach can identify when it is another driver acting upon the vehicle. To be continued.

CHAPTER 8. CONCLUSION

In this paper, I examined methods for applying IDSs to securing automotive systems with an overview of the techniques and a discussion of their advantages and disadvantages. I attempted to clarify and unify the concept of anomalies and intrusion detection regarding automotive security. This begins with identifying threat models for automotive security and identifying threats that effect all vehicles and not just one specific model. From a technical perspective, IDSs can work well for detecting intrusions on the CAN bus. Different implementations of anomaly detection methods can detect different types of anomalies. Current approaches have a focus on message injection attack detection because it is the main attack vector for hackers trying to make a vehicle misbehave.

I proposed a hybrid intrusion detection system that combines two approaches of anomaly and signature-based detectors. The anomaly detector is the wide-net that catches any anomaly. Where the signature detector is the comb that can filter through the detected anomalies and detect intrusions. I implement and demonstrate a message timing-based anomaly detector that utilizes interval and frequency features of CAN messages to detect message injection attacks. I also implement and demonstrate a signature-based detector for detecting other attacks that do not alter message timings. Lastly, I combine these to detectors to complement each other for better detection accuracy and false positive rates.

I also presented a novel application of machine learning towards the reverse engineering of CAN messages. The proposed approach does not require prior insider knowledge about the CAN messages. I proposed a supervised method to extract CAN message function by analyzing change in their data fields respective to vehicular function. The results of this method are used to apply labels in the unsupervised learning method.

I proposed utilizing the machine learning approach of clustering to cluster CAN messages together based on their function. This approach results in numerous clusters of CAN IDs, and with our previous labels we are able to classify the function for these unknown CAN messages. The work is an initial step towards automating reverse engineering of CAN message features without the need to physically and manually test messages on a live vehicle.

About the Author

Clinton Young (cwyoung@iastate.edu) is a graduate student pursuing a Ph.D. in Computer Engineering at Iowa State University. He is currently a research assistant working on automotive security and full-time system engineer at Collins Aerospace.

Acknowledgment

This material is supported by the National Science Foundation under Grant No. CNS 1646317 and CNS 1645987.

BIBLIOGRAPHY

[1] Robert Altschaffel, Tobias Hoppe, Sven Kuhlmann, and Jana Dittmann. Simulation of Automotive Security Threat Warnings to Analyze Driver Interpretations and Emotional Transitions. In Floor Koornneef and Coen van Gulijk, editors, *Computer Safety, Reliability, and Security*, number 9337 in Lecture Notes in Computer Science, pages 47–58. Springer International Publishing, sep 2015. DOI: 10.1007/978-3-319-24255-2_5.

[2] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *SIGMOD Rec.*, 28(2):49–60, June 1999.

[3] Mohammad Raashid Ansari, Shucheng Yu, and Qiaoyan Yu. Intellican: Attack-resilient controller area network (CAN) for secure automobiles. In *Proc. of International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, 2015.

[4] Chris Blommendaal. Information Security Risks for Car Manufacturers based on the in-Vehicle Network, Sep. 2015.

[5] B. Carnevale, F. Falaschi, L. Crocetti, H. Hunjan, S. Bisase, and L. Fanucci. An implementation of the 802.1AE MAC Security Standard for in-car networks. In *Proc. of 2nd World Forum on Internet of Things (WF-IoT)*, Dec. 2015.

[6] Gail A. Carpenter and Stephen Grossberg. *Pattern Recognition by Self-Organizing Neural Networks*. MIT Press, Cambridge, MA, USA, 1st edition, 1991.

[7] Paul Carsten, Todd R Andel, Mark Yampolskiy, and Jeffrey T McDonald. In-vehicle networks: Attacks, vulnerabilities, and proposed solutions. In *Proc. of the 10th Annual Cyber and Information Security Research Conference*, 2015.

[8] Robert N Charette. This car runs on code. *IEEE Spectrum*, 46(3), 2009.

[9] 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 *Proc. of USENIX Security Symposium*, 2011.

[10] Peter C. Cheeseman and John C. Stutz. Bayesian classification (autoclass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*, 1996.

[11] Kyong-Tak Cho and Kang Shin. Error handling of in-vehicle networks makes them vulnerable. In *Proc. of Computer and Communication Security*, 2016.

[12] Kyong-Tak Cho and Kang G Shin. Fingerprinting electronic control units for vehicle intrusion detection. In *Proc. of USENIX Security Symposium*, 2016.

[13] Wonsuk Choi, Hyo Jin Jo, Moon Chan Park, and Dong Hoon Lee. Voltageids: Low-level communication characteristics for automotive intrusion detection system. In *Proc. of IEEE Transactions on Information Forensics and Security*, 2018.

[14] Steve Corrigan. Introduction to the Controller Area Network (CAN). *Texas Instruments Application Report*, 2016.

[15] Miro Enev, Alex Takakuwa, Karl Koscher, and Tadayoshi Kohno. Automobile driver fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2016, 01 2016.

[16] Herson Esquivel-Vargas, Marco Caselli, and Andreas Peter. Automatic deployment of specification-based intrusion detection in the bacnet protocol. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy*, CPS '17, page 25–36, New York, NY, USA, 2017. Association for Computing Machinery.

[17] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[18] Takaya Ezaki, Tomohiro Date, and Hiroyuki Inoue. An Analysis Platform for the Information Security of In-Vehicle Networks Connected with External Networks. In Keisuke Tanaka and Yuji Suga, editors, *Advances in Information and Computer Security*, number 9241 in Lecture Notes in Computer Science, pages 301–315. Springer International Publishing, aug 2015. DOI: 10.1007/978-3-319-22425-1_18.

[19] Davide Fauri, Daniel Ricardo dos Santos, Elisa Costante, Jerry den Hartog, Sandro Etalle, and Stefano Tonetta. From system specification to anomaly detection (and back). In *CPS '17*, 2017.

[20] Hamid Reza Ghaeini and Nils Ole Tippenhauer. Hamids: Hierarchical monitoring intrusion detection system for industrial control systems. In *Proceedings of the 2Nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, CPS-SPC '16, pages 103–111, New York, NY, USA, 2016. ACM.

[21] M. Gmiden, H. Mohamed, and H. Trabelsi. An intrusion detection method for securing in-vehicle CAN bus. In *Proc. of Sciences and Techniques of Automatic Control and Computer Engineering*, 2016.

[22] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: An efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on*

Management of Data, SIGMOD '98, page 73–84, New York, NY, USA, 1998. Association for Computing Machinery.

[23] Gang Han, Haibo Zeng, Yaping Li, and Wenhua Dou. Safe: Security-aware flexray scheduling engine. In *Proc. of Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 1–4. IEEE, 2014.

[24] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[25] Tim Hermans, Joachim Denil, Paul De Meulenaere, and Jan Anthonis. Decoding of data on a can powertrain network. In *IEEE*, 2009.

[26] J. Hong, C. C. Liu, and M. Govindarasu. Integrated Anomaly Detection for Cyber Security of the Substations. *IEEE Transactions on Smart Grid*, 5(4):1643–1653, jul 2014.

[27] T. Hoppe, S. Kiltz, and J. Dittmann. Security threats to automotive can networks - practical examples and selected short-term countermeasures. *SAFECOMP*, 2008.

[28] Thomas Huybrechts, Yon Vanommeslaeghe, Dries Blontrock, Gregory Van Barel, and Peter Hellinckx. Automatic reverse engineering of can bus data using machine learning techniques. In *Advances on P2P, Parallel, Grid, Cloud and Internet COmputing*, pages 751–761, 01 2018.

[29] K. Iehira, H. Inoue, and K. Ishida. Spoofing attack using bus-off attacks against a specific ecu of the can bus. In *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1–4, Jan 2018.

[30] Haojie Ji, Yunpeng Wang, Hongmao Qin, Xinkai Wu, and Guizhen Yu. Investigating the effects of attack detection for in-vehicle networks based on clock drift of ecus. In *Proc. of IEEE Access*. IEEE, 2018.

[31] D. A. Johnson and M. M. Trivedi. Driving style recognition using a smartphone as a sensor platform. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1609–1615, Oct 2011.

[32] Min-Joo Kang and Je-Won Kang. Intrusion detection system using deep neural network for in-vehicle network security. *PloS one*, 11(6), 2016.

[33] T. U. Kang, H. M. Song, S. Jeong, and H. K. Kim. Automated reverse engineering and attack for can using obd-ii. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–7, Aug 2018.

[34] George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, August 1999.

[35] Leonard Kaufman and Peter Rousseeuw. Finding groups in data: An introduction to cluster analysis. In *Wiley-Interscience*, 1990.

[36] Whui Kim, Hyun-Kyun Choi, Byung-Tae Jang, and Jinsu Lim. Driver distraction detection using single convolutional neural network. *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1203–1205, 2017.

[37] T. Kohonen, M. R. Schroeder, and T. S. Huang. *Self-Organizing Maps*. Springer-Verlag, Berlin, Heidelberg, 3rd edition, 2001.

[38] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, and Hovav Shacham. Experimental security analysis of a modern automobile. In *Proc. of IEEE Symposium on Security and Privacy (SP)*, 2010.

[39] Takuya Kuwahara, Yukino Baba, Hisashi Kashima, Takeshi Kishikawa, Junichi Tsurumi, Tomoyuki Haga, Yoshihiro Ujiie, Takamitsu Sasaki, and Hideki Matsushima. Supervised and unsupervised intrusion detection based on can message frequencies for in-vehicle network. *Journal of Information Processing*, 26:306–313, 01 2018.

[40] Byung-Il Kwak, Jiyoung Woo, and Huy Kang Kim. Know your master: Driver profiling-based anti-theft method. In *International Conference on Privacy, Security and Trust*, pages 211–218, 12 2016.

[41] Ulf E Larson, Dennis K Nilsson, and Erland Jonsson. An approach to specification-based attack detection for in-vehicle networks. In *Proc. of Intelligent Vehicles Symposium*, pages 220–225. IEEE, 2008.

[42] Hyunsung Lee, Seong Hoon Jeong, and Huy Kang Kim. Otids: A novel intrusion detection system for in-vehicle network by using remote frame. *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 57–5709, 2017.

[43] Szilvia Lestyan, Gergely Ács, Gergely Biczók, and Zsolt Szalay. Extracting vehicle sensor signals from CAN logs for driver re-identification. *CoRR*, abs/1902.08956, 2019.

[44] Chung-Wei Lin and Alberto Sangiovanni-Vincentelli. Cyber-security for the controller area network (CAN) communication protocol. In *Proc. of International Conference on Cyber Security (CyberSecurity)*, 2012.

[45] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[46] Mirco Marchetti and Dario Stabili. Read: Reverse engineering of automotive data frames. *IEEE Transactions on Information Forensics and Security*, PP:1–1, 09 2018.

[47] Mirco Marchetti, Dario Stabili, Alessandro Guido, and Michele Colajanni. Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms. In *Proc. of International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*. IEEE, 2016.

[48] Moti Markovitz and Avishai Wool. Field classification, modeling and anomaly detection in unknown can bus networks. In *Proc. of Vehicular Communications)*, 2017.

[49] F. Martinelli, F. Mercaldo, V. Nardone, and A. Santone. Car hacking identification through fuzzy logic algorithms. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–7, July 2017.

[50] Tsutomu Matsumoto, Masato Hata, Masato Tanabe, Katsunari Yoshioka, and Kazuomi Oishi. A method of preventing unauthorized data transmission in controller area network. In *Proc. of Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2012.

[51] C. Miller and C. Valasek. A survey of remote automotive attack surfaces. *Unknown Journal*, 2014.

[52] C. Miller and C. Valasek. Remote exploitation of an unaltered passenger vehicle. In *BlackHat USA*. BlackHat USA, 2015.

[53] R. Mitchell and I. R. Chen. Effect of Intrusion Detection and Response on Reliability of Cyber Physical Systems. *IEEE Transactions on Reliability*, 62(1):199–210, mar 2013.

[54] R. Mitchell and I. R. Chen. Adaptive Intrusion Detection of Malicious Unmanned Air Vehicles Using Behavior Rule Specifications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(5):593–604, may 2014.

[55] Robert Mitchell and Ing-Ray Chen. A Survey of Intrusion Detection Techniques for Cyber-physical Systems. *ACM Comput. Surv.*, 46(4):55:1–55:29, mar 2014.

[56] S. Mohan, S. Bak, E. Betti, H. Yun, L. Sha, and M. Caccamo. S3a: Secure system simplex architecture for enhanced security and robustness of cyber-physical systems. In *Proceddings of the 2nd ACM International Conference on High COnfidence Networked Systems*, 2013.

[57] M. Moore, R. Bridges, F. Combs, M. Starr, and S. Prowell. Modeling inter-signal arrival times for accurate detection of CAN bus signal injection attacks. *Proc. of 12th Annual Conference on Cyber and Information Security Research*, 2017.

[58] Philipp Mundhenk, Sebastian Steinhorst, Martin Lukasiewycz, Suhaib A Fahmy, and Samarjit Chakraborty. Lightweight authentication for secure automotive networks. In *Proc. of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, 2015.

[59] M. Müter and N. Asaj. Entropy-based anomaly detection for in-vehicle networks. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 1110–1115, Jun 2011.

[60] Michael Müter, André Groll, and Felix C Freiling. A structured approach to anomaly detection for in-vehicle networks. In *Proc. of Information Assurance and Security (IAS)*, 2010.

[61] Sandeep Nair Narayanan, Sudip Mittal, and Anupam Joshi. Using data analytics to detect anomalous states in vehicles. *arXiv preprint arXiv:1512.08048*, 2015.

[62] Oak ridge national labs. https://www.ornl.gov/facility/ntrc/research-areas/vehicle-systems.

[63] Depren Ozgur, Topalla Murat, Anarim Emin, and Ciliz M. An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Systems with Applications*, 2005.

[64] F. Pasqualetti and Q. Zhu. Design and Operation of Secure Cyber-Physical Systems. *IEEE Embedded Systems Letters*, 7(1):3–6, mar 2015.

[65] Mert D. Pesé, Troy Stacer, C. Andrés Campos, Eric Newberry, Dongyao Chen, and Kang G. Shin. Librecan: Automated can message translator. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, pages 2283–2300, New York, NY, USA, 2019. ACM.

[66] Charles Pfleeger, Shari Pfleeger, and Jonathan Margulies. *Security in Computing*. IEEE, 2015.

[67] S. Ponomarev and T. Atkison. Industrial Control System Network Intrusion Detection by Telemetry Analysis. *IEEE Transactions on Dependable and Secure Computing*, 13(2):252–260, mar 2016.

[68] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *Proc. of the ACM Workshop on Data Mining Applied to Security*, 2001.

[69] U. K. Premaratne, J. Samarabandu, T. S. Sidhu, R. Beresh, and J. C. Tan. An Intrusion Detection System for IEC61850 Automated Substations. *IEEE Transactions on Power Delivery*, 25(4):2376–2383, oct 2010.

[70] Deepak Puthal, Saraju Mohanty, Priyadarsi Nanda, and Uma Choppali. Building security perimeters to protect network systems against cyber threats. *IEEE Consumer Electronics Magazine*, 6(4), Oct. 2017.

[71] M. Roesch. Lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX Conference on System Administration*, 1999.

[72] SAE International. Serial control and communications heavy duty vehicle network, June 2012.

[73] R. Samdarshi, N. Sinha, and P. Tripathi. A triple layer intrusion detection system for SCADA security of electric utility. In *2015 Annual IEEE India Conference (INDICON)*, pages 1–5, dec 2015.

[74] E. Seo, H. M. Song, and H. K. Kim. Gids: Gan based intrusion detection system for in-vehicle network. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6, Aug 2018.

[75] S. Shreejith and S. A. Fahmy. Security aware network controllers for next generation automotive embedded systems. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, jun 2015.

[76] Hyun Min Song, Ha Rang Kim, and Huy Kang Kim. Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network. In *Proc. of International Conference on Information Networking (ICOIN)*, 2016.

[77] Ivan Studnia, Eric Alata, Vincent Nicomette, Mohamed Kaaniche, and Youssef Laarouchi. A language-based intrusion detection approach for automotive embedded networks. *International Journal of Embedded Systems*, 10:1, 01 2018.

[78] Ivan Studnia, Vincent Nicomette, Eric Alata, Yves Deswarte, Mohamed Kaâniche, and Youssef Laarouchi. Survey on security threats and protection mechanisms in embedded automotive networks. In *Proc. IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, 2013.

[79] Farid Molazem Tabrizi and Karthik Pattabiraman. Flexible intrusion detection systems for memory-constrained embedded systems. In *Proc. of Dependable Computing Conference (EDCC)*, pages 1–12. IEEE, 2015.

[80] A. Taylor, N. Japkowicz, and S. Leblanc. Frequency-based anomaly detection for the automotive can bus. In *2015 World Congress on Industrial Control Systems Security (WCICSS)*, pages 45–49, Dec 2015.

[81] A. Taylor, S. Leblanc, and N. Japkowicz. Anomaly detection in automobile control network data with long short-term memory networks. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 130–139, Oct 2016.

[82] A. Tomlinson, J. Bryans, S. A. Shaikh, and H. K. Kalutarage. Detection of automotive can cyber-attacks by identifying packet timing anomalies in time windows. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 231–238, June 2018.

[83] Zachariah Tyree, Robert Bridges, Frank Combs, and Michael Moore. Exploiting the shape of can data for in-vehicle intrusion detection. In *Proc. of Connected and Autonomous Vehicles Symposium*, 08 2018.

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

[85] M. Van Ly, S. Martin, and M. M. Trivedi. Driver classification and driving style recognition using inertial sensors. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 1040–1045, June 2013.

[86] Miki E. Verma, Robert A. Bridges, and Samuel C. Hollifield. Actt: Automotive can tokenization and translation. *ArXiv*, abs/1811.07897, 2018.

[87] A. Wahab, C. Quek, C. K. Tan, and K. Takeda. Driving profile modeling and recognition based on soft computing approach. *IEEE Transactions on Neural Networks*, 20(4):563–582, April 2009.

[88] Liang Wen, Wei Jiang, Ke Jiang, Xia Zhang, Xiong Pan, and Keran Zhou. Detecting fault injection attacks on embedded real-time applications: A system-level perspective. In *High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conference on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on*, pages 700–705. IEEE, 2015.

[89] Jimmy Wesolowski, Aymen Boudguiga, Anup Patel, Matthieu Donain, Julien Galbert, Witold Klaudel, and Guillaume Scigala. Xvisor virtio-can: Fast virtualized can. In *Proc. of Embedded Real Time Software and Systems*, 01 2016.

[90] S. Woo, H. J. Jo, and D. H. Lee. A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):993–1006, apr 2015.

[91] Y. Yang, K. McLaughlin, S. Sezer, T. Littler, E. G. Im, B. Pranggono, and H. F. Wang. Multiattribute SCADA-Specific Intrusion Detection System for Power Networks. *IEEE Transactions on Power Delivery*, 29(3):1092–1102, jun 2014.

[92] M. K. Yoon, S. Mohan, J. Choi, J. E. Kim, and L. Sha. SecureCore: A multicore-based intrusion detection architecture for real-time embedded systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2013 IEEE 19th*, pages 21–32, apr 2013.

[93] Man-Ki Yoon and Gabriela F Ciocarlie. Communication pattern monitoring: Improving the utility of anomaly detection for industrial control systems. In *NDSS Workshop on Security of Emerging Networking Technologies*, 2014.

[94] Man-Ki Yoon, Swati Mohan, Jaesik Choi, Jung-Eun Kim, and Lui Sha. Securecore: A multicore-based intrusion detection architecture for real-time embedded systems. In *Proc. of Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 21–32. IEEE, 2013.

[95] Clinton Young, Habeeb Olufowobi, Gedare Bloom, and Joseph Zambreno. Automotive intrusion detection based on constant can message frequencies across vehicle driving modes. In *Proceedings of the ACM Workshop on Automotive Cybersecurity*, AutoSec '19, pages 9–14, New York, NY, USA, 2019. ACM.

[96] W. Zeng, M. Khalid, and S. Chowdhury. In-vehicle networks outlook: Achievements and challenges. *IEEE Communications Surveys Tutorials*, PP(99):1–1, 2016.

[97] T. Zhang, H. Antunes, and S. Aggarwal. Defending Connected Vehicles Against Malware: Challenges and a Solution Framework. *IEEE Internet of Things Journal*, 1(1):10–21, Feb 2014.

[98] B. Zheng, P. Deng, R. Anguluri, Q. Zhu, and F. Pasqualetti. Cross-Layer Codesign for Secure Cyber-Physical Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(5):699–711, may 2016.

[99] Bowen Zheng, W. Li, P. Deng, L. Gérardy, Q. Zhu, and N. Shankar. Design and verification for transportation system security. In *Proc. of Design Automation Conference (DAC)*, June 2015.

[100] C. Zhou, S. Huang, N. Xiong, S. H. Yang, H. Li, Y. Qin, and X. Li. Design and Analysis of Multimodel-Based Anomaly Intrusion Detection Systems in Industrial Process Automation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(10):1345–1360, oct 2015.

[101] Christopher Zimmer, Balasubramanya Bhat, Frank Mueller, and Sibin Mohan. Time-based Intrusion Detection in Cyber-physical Systems. In *Proc. of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, 2010.