Physics-based Security Analysis of Controller Area Network Protocols

Sang Uk Sagong

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2019

Reading Committee:

Linda Bushnell, Chair

Radha Poovendran, Chair

James Ritcey

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

Physics-based Security Analysis of
Controller Area Network Protocols

Sang Uk Sagong

Co-Chairs of the Supervisory Committee:
Professor Linda Bushnell
Electrical and Computer Engineering

Professor Radha Poovendran
Electrical and Computer Engineering

A contemporary automobile is equipped with many sensors and actuators that are controlled by electronic units. These electronic units are connected via wired bus networks, and data for controlling the automobile is exchanged among the electronic units using in-vehicle network protocols. The in-vehicle network protocols were designed to be isolated from external networks, and they do not encrypt data or authenticate messages as a consequence. To provide a higher quality of service, such as traffic information or over-the-air software updates, electronic units that can be connected to the external networks are installed in the contemporary automobile. These electronic units with outward-facing interfaces act as attack surfaces to in-vehicle networks and other electronic units.

In this dissertation, we identify vulnerabilities of in-vehicle network protocols such as the Controller Area Network (CAN) and existing intrusion detection systems. In order to enhance the security of the CAN protocol from these vulnerabilities, we develop software and hardware-based defense mechanisms that can detect attacks by exploiting timing, voltage, and motion information, where the hardware-based defense mechanism can also mitigate these attacks. We implement the attacks and demonstrate that the proposed defense mechanisms can effectively detect and mitigate the attacks using a CAN bus testbed and a real vehicle.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

C_A	:	The time kept by clock A
ΔT	:	The amount of delay for launching the cloaking attack
I_{max}	:	The absolute maximum rating of the microcontroller
L^-	:	The lower control limit of the Cumulative Sum
L^+	:	The upper control limit of the Cumulative Sum
O_A	:	The clock offset of clock A
O_{acc}	:	The accumulated offset in the k-th batch
O_{avg}	:	The average offset in the k-th batch
P_d	:	The detection probability of MIDS
P_H	:	The analog pin of the microcontroller connected to CANH
P_L	:	The analog pin of the microcontroller connected to CANL
P_s	:	The probability of a successful cloaking attack
ρ	:	The vector of correlation coefficient between two data sets
S_A	:	The clock skew of clock A
σ_{lim}	:	The minimum standard deviation of $\bar{d}$, at which $P_d > 1 - \epsilon$
σ_{bypass}	:	The maximum standard deviation of $\bar{d}$, below which P_d=0
σ_{detect}	:	The minimum standard deviation of $\bar{d}$, above which P_d=1
τ_{bit}	:	The bit length time
$V_{attack,H}$	:	The voltage applied to CANH under the voltage-based attacks
$V_{attack,L}$	:	The voltage applied to CANL under the voltage-based attacks
V_{Diff}	:	The differential voltage between CANH and CANL

ACKNOWLEDGMENTS

My Ph.D. study has been a precious experience with respect to both research and teaching. I would like to thank my advisors Professor Linda Bushnell and Professor Radha Poovendran for their guidance and mentoring on the research over the years. This dissertation would not have been possible without their support and guidance. I would also like to thank Professor James Ritcey and Professor Brian Fabien for their time and helpful discussion on the research.

I thank my colleagues and friends in the Network Security Lab including Dr. Shana Moothedath, Dr. Bhaskar Ramasubramanian, Dinuka Sahabandu, Baicen Xiao, Shruti Misra, Joanna Mazer, Kalana Sahabandu, Dr. Xuhang Ying, and Professor Andrew Clark for discussion on research and friendship. I also thank Dr. Sukarno Mertoguno and Aman Kalia for their discussion on research.

This work was supported by ONR grants N00014-16-1-2710 and N00014-17-1-2946, NSF grants CNS-1446866 and CNS-1544173, and ARO grant W911NF-16-1-0485. DoE, General Motors Company, and Argonne National Laboratory sponsor the Advanced Vehicle Technology Competition.

I would like to thank my parents and sister for their unconditional love and support throughout my life. I can confidently say that I would not have made it through all this Ph.D. journey without them, and I am grateful for their help.

DEDICATION

to my parents and sister

Chapter 1

INTRODUCTION

Data for controlling a vehicle is exchanged among Electronic Control Units (ECUs) via in-vehicle network protocols such as the Controller Area Network (CAN) [6], FlexRay [5], and Local Interconnect Network [7]. Since the in-vehicle network protocols were designed for a closed network that is isolated from the external network and environment, a message is not encrypted or authenticated in these in-vehicle network protocols. In order to provide higher safety, more accurate navigation, and more entertainment options to a driver and passengers of vehicles, contemporary vehicles are equipped with more ECUs that have outward-facing interfaces such as CD players, cellular networks, WiFi networks, or Bluetooth. Due to these ECUs that have the outward-facing interfaces, the closed network assumption no longer holds, and these ECUs act as attack surfaces to the in-vehicle network, which consequently resulting in vulnerabilities to the in-vehicle network protocols and ECUs. An adversary compromises ECUs that have the outward-facing interfaces and gets access to the in-vehicle network such as the CAN bus. Then, the adversary may block message transmission from the compromised ECUs or inject spoofed messages such as steering or engine control to the CAN bus using the compromised ECUs [15, 29, 36, 37, 38]. These attacks on the in-vehicle network and ECUs can create false alarms and warnings to a driver, disable brakes, and cause the vehicle to accelerate uncontrollably in order to cause serious safety risks to passengers, pedestrians, and other vehicles [29, 36, 38].

It is, however, difficult to upgrade the existing in-vehicle network protocols to encrypt data or authenticate messages due to the lack of backward compatibility with legacy systems and the resource constraints of ECUs such as memory size and computing capability [17, 41]. As a consequence, Intrusion Detection Systems (IDSs) have been developed to detect attacks on the CAN bus, such as suspension attack, fabrication attack, and masquerade attack, by tracking abnormal

deviations in physical properties of the CAN bus and ECUs [18]. Commonly exploited physical properties are the frequency of message occurrence [26], clock skew of an ECU [18], the entropy of the CAN bus [42], and voltage levels of the CAN bus lines [19, 20, 41]. The goal of this dissertation is exploring vulnerabilities of the CAN protocol and existing IDSs in order to address challenges in security issues in the CAN protocol and also providing novel approaches to defend the CAN protocol, CAN bus network, and ECUs by mitigating these vulnerabilities.

1.1 CAN Background

The CAN protocol was first deployed on Mercedes-Benz W140 in 1991 as an isolated network, and the CAN protocol is now *de facto* one of the most widely used in-vehicle network protocols [6, 12]. The CAN protocol is a multi-master broadcast bus network in which any ECU can transmit messages and receive all ongoing messages through the CAN bus. An ECU that accesses the CAN bus first can transmit a message. If two or more ECUs attempt to transmit messages simultaneously, the message that has the smallest message ID is first transmitted through a content-based collision detection process called *arbitration*. For instance, consider two ECUs A and B that try to send their messages whose message IDs are 0x010 and 0x001, respectively. Because ECU B transmits a message 0x001 that is a smaller ID than ECU A's message, ECU A recognizes that a higher priority message (i.e., 0x001) is being transmitted. Then, ECU A stops transmitting its message (i.e., 0x010) through an arbitration process. When bits 0 and 1 are attempted to be transmitted simultaneously, bit 0 is transmitted through the CAN bus. Bits 0 and 1 are also called dominant and recessive bits, respectively.

A data frame in the CAN protocol consists of 7 fields, which are start of frame (SOF), arbitration, control, data, cyclic redundancy check (CRC), acknowledgment (ACK), and end of frame (EOF) fields, as illustrated in Fig. 1.1, and the length of the data field can be varied from 1 to 8 bytes. As shown in the structure of the data frame, the data field is not encrypted, and there does not exist a field for message authentication. If a message is not transmitted and received correctly because of external electromagnetic interference or malfunction of CAN transceivers, the ECU retransmits that message after an error frame is transmitted.

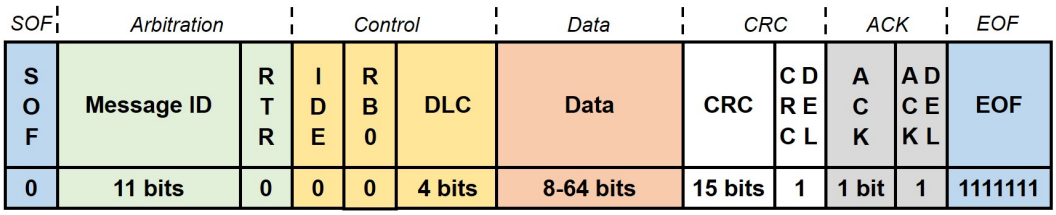


Figure 1.1: Structure of a data frame in the CAN protocol. A data frame of the CAN protocol consists of SOF field, arbitration field, control field, data field, CRC field, ACK field, and EOF field.

The CAN bus network is composed of two bus lines, which are CANH and CANL, terminated by two 120Ω resistors. In order to make the CAN bus more robust to the external electromagnetic interference, the CAN protocol uses the differential voltage between CANH and CANL to represent a bit [10]. When transmitting a recessive bit (i.e., bit 1), both CANH and CANL are set to the same value, which makes the differential voltage be 0.0V. When transmitting a dominant bit (i.e., bit 0), CANH and CANL are set to different voltages to make the differential voltage larger than a predetermined threshold, typically 0.9V [35, 44, 50]. Fig. 1.2 illustrates the voltage levels of CANH and CANL when transmitting a dominant bit and recessive bit using a CAN transceiver whose supply voltage is 5.0V. When transmitting a recessive bit, the voltage levels of both CANH and CANL are set to nominal 2.5V in order to make the differential voltage be 0.0V. When transmitting a dominant bit, the differential voltage is 2.0V since CANH and CANL are set to nominal 3.5V and 1.5V, respectively. For a recessive bit, the voltage levels of CANH and CANL are 2.3V when we use a CAN transceiver whose supply voltage is 3.3V, in which the differential voltage is still 0.0V. For a dominant bit, this CAN transceiver sets CANH and CANL to 3.0V and 1.0V, respectively. The differential voltage becomes 2.0V [51, 52]. Although the supply voltage of the CAN transceivers might be different, the CAN transceivers exploit the differential voltage between CANH and CANL to represent a bit [35, 44, 50, 51, 52, 54].

The voltages of CANH and CANL are controlled by two transistors in a CAN transceiver, as shown in Fig. 1.3. When transmitting a recessive bit, both transistors are in the off-state at which

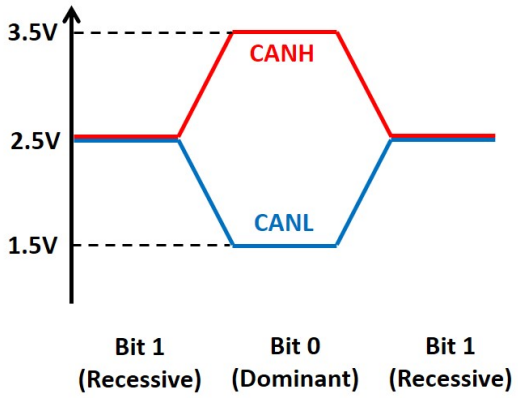


Figure 1.2: Voltage levels of CANH and CANL when dominant and recessive bits are transmitted by a CAN transceiver whose supply voltage is 5V. When transmitting a recessive bit, both CANH and CANL are set to 2.5V, which makes the differential voltage between CANH and CANL be 0.0V. When transmitting a dominant bit, CANH and CANL are set to 3.5V and 1.5V, respectively. As a result, the differential voltage becomes 2.0V.

the current cannot flow from the supply voltage (V_{DD}) to the ground. The reference voltage (e.g., nominal 2.5V in a 5V CAN transceiver) is applied to both CANH and CANL, and CANH and CANL are set to the reference voltage. As a result, the differential voltage becomes 0.0V. The current does not flow through the termination resistors because the electric potential across the termination resistors is zero.

When transmitting a dominant bit, the driver control of the CAN transceiver sets both transistors to the on-state at which the electric path is created between the emitter and the collector in bipolar junction transistors (BJTs) or the source and the drain in field effect transistors (FETs). Then, the current flows from the supply voltage to the ground. Due to the current flowing through the termination resistors, the voltage is dropped from CANH to CANL, and consequently, a non-zero differential voltage is induced. Since the operations of BJT and FET as switches are identical to each other, the analysis of the operation of Microchip MCP2551 CAN transceiver, which is composed of BJTs, can be applied to any CAN transceivers that are composed of FETs [44]. We

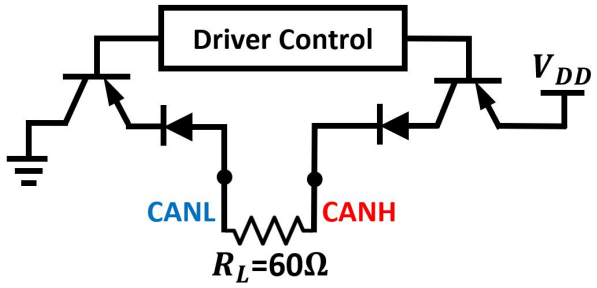


Figure 1.3: Schematics of the Microchip MCP2551 CAN transceiver, which is composed of two BJTs and two diodes. The driver control turns off the BJTs when transmitting a recessive bit and turns them on when transmitting a dominant bit. The impedance R_L between CANH and CANL is 60Ω since two 120Ω termination resistors are connected in parallel.

explain how the voltages of CANH and CANL are set to 3.5V and 1.5V, respectively, when a Microchip MCP2551 CAN transceiver is used. A Microchip MCP2551 CAN transceiver is operated by 5.0V (i.e., V_{DD} is set to 5.0V.). When the BJT is in the on-state, the voltage is dropped by 0.7V between the base and the emitter of the BJT, which makes the voltage at the collector set to 4.3V. The voltage is again dropped by 0.7V at the diode from 4.3V, and the voltage of CANH becomes 3.5-3.6V. Similarly, the voltage is increased by 0.7V from the ground at the BJT, and the voltage at the base becomes 0.7V. Then, the voltage is again increased by 0.7V from the cathode of the diode to the anode of the diode, which makes the voltage of CANL set to 1.4-1.5V.

A microprocessor measures voltage via its analog pin set to the input mode by using an Analog-to-Digital Converter (ADC) that is embedded in the microprocessor [8, 33, 43, 45, 53]. In addition to measuring the voltage, a microprocessor may generate electric signals such as a Pulse Width Modulation (PWM) signal using an analog pin if that analog pin is set to the output mode [8, 45, 53]. For example, Microchip ATmega328P used in an Arduino UNO Rev3 board, NXP MPC563 used in an ECU of Hyundai Sonata, and Renesas V850 used in an Engine Control Module of Toyota Camry provide analog pins that can be used to measure voltage or generate PWM signals [33, 43, 45].

1.2 Related Work

Recent works have demonstrated that contemporary cars are equipped with many ECUs that have outward-facing interfaces such as Bluetooth, and these cars are vulnerable to the cyber attacks because these ECUs act as attack surfaces to the CAN bus network and ECUs [15, 23, 29, 36, 37, 38]. Because the CAN protocol does not encrypt the data and authenticate messages, many works have proposed the various IDSs that detect the cyber attacks using abnormal deviations in the traffic through the CAN bus. Based on the fact that most of the messages in the CAN protocol are transmitted with a fixed length and frequency, the authors of [26] proposed an IDS that detects the existence of spoofed messages using the frequency of the message occurrence. If spoofed messages are injected while the legitimate messages are transmitted, the number of messages through the CAN bus increases. By exploiting the number of messages that are transmitted through the CAN bus within a unit time, an attack can be detected. In addition to the number of messages transmitted through the CAN bus, the coincidence among a set of messages is also exploited in an entropy-based IDS [42].

An adversary, however, can launch an attack that replicates the structure and pattern of the legitimate traffic, which also does not change the number of messages transmitted through the CAN bus [18]. This attack can bypass the IDSs that exploit the abnormal deviations in the traffic through the CAN bus. IDSs that exploit physical invariants of ECUs such as clock skew and voltage characteristics have been proposed in order to detect such an attack [18, 19, 20, 41]. These IDSs detect an attack by identifying the transmitting ECU of the messages. The authors of [18] proposed a clock-based IDS (CIDS) that exploits clock skew of ECUs to fingerprint each ECU. Each ECU on the CAN bus has a different hardware clock, which has a distinct clock speed due to variations in the clock's hardware crystal, a property referred to as clock skew [39, 40]. Since all process clocks in an ECU are derived from the hardware clock, they are affected by the clock skew as a consequence. In particular, the inter-transmission times of messages that are periodically transmitted by an ECU depend on its clock skew because the local clock of the ECU in the CAN bus is not synchronized globally with other ECUs in the same CAN bus. If an adversary injects

spoofed messages periodically using a compromised ECU that has different clock skew, the inter-transmission time of the spoofed messages is also different from that of the legitimate messages. An ECU can estimate the clock skew of an ECU by using periodic messages that are transmitted from that ECU based on inter-arrival times of the periodic messages.

In comparison to physical properties like message periodicity, traffic pattern, and clock skew, the voltage characteristics of an ECU are more difficult to be mimicked, because the voltage characteristics are determined by the CAN transceiver's hardware such as the internal impedance of the transistors and diodes which are not affected by the software of the microcontroller [19]. Despite a potential risk of connecting the microcontroller's analog pins to the CAN bus lines, the voltage-based IDSs (VIDSs) have been proposed in recent works [19, 20, 41]. The authors of [41] proposed a VIDS that exploits the mean squared error between the measured voltage and the reference voltage of each ECU that has been collected before the attack. All the reference voltage values of each ECU have to be collected by and saved at the VIDS prior to the implementation of the VIDS. In [20], the proposed VIDS exploits the voltage difference between the two CAN bus lines and transition time between bits 0 and 1 to detect the deviation from the normal voltage behavior using machine learning algorithms. The authors of [19] proposed a VIDS that measures the voltages of each CAN bus line and extracts features from the distribution of the measured voltage samples. The Viden measures the voltage and detects an attack in a more realistic environment, such as driving a car, although the voltage is measured at a slower rate and less accuracy compared to an oscilloscope.

These VIDSs have to measure the voltages of the CAN bus lines to extract the voltage characteristics of each ECU. An oscilloscope is used to measure the voltage in [41] and [20], which may not be possible in a car due to the limitations in supply power and space. Alternatively, a VIDS may be implemented in a microcontroller like the Viden, which can be operated at an automobile by using the car battery that can supply the power to the microcontroller (e.g., typically 12V) [19]. If the microcontroller or oscilloscope is compromised, the adversary may exploit the wires that are connected to the CAN bus lines to launch the attacks. The recent works on the VIDSs, however, did not discuss any potential cyber attacks in which these wires are maliciously used, nor did

the works propose a protection mechanism to mitigate the attacks [19, 20, 41]. We consider such attacks in this paper.

Although it is demonstrated that an IDS is effective in protecting the CAN bus, the IDS is limited to detecting cyber attacks and alerting an operator of the automobile. The reaction to the detected attacks remains to the operator of the automobile [14]. After detecting the attacks, an Intrusion Response System (IRS), however, can promptly minimize the consequence of the attacks or remove the attacks, which provides more time to the operator to stop the automobile safely [14, 26].

Many IDSs that detect cyber attacks by fingerprinting ECUs in the CAN bus have been proposed [18, 19, 20, 41]. These IDSs track abnormal deviations in the physical properties of ECUs to detect an attack. It has been demonstrated that these IDS can effectively detect an attack in which a spoofed message is transmitted by any compromised ECU that is not the original transmitter of the message [18, 19, 20]. An adversary, however, may compromise a legitimate ECU and launch an attack, in which the adversary injects spoofed messages that convey false sensor measurements or control signals using that compromised ECU. The IDSs that fingerprint ECUs cannot detect this attack because the transmitter of the spoofed messages remains the same after the attack, which does not induce any deviations in the physical properties. In a contemporary vehicle, multiple and many different types of sensors are implemented to measure a physical movement of the vehicle. The authors of [24] demonstrated that there is a correlation between measurements from these sensors under the normal operation of a vehicle because a physical movement of a car affects multiple sensors simultaneously.

1.3 Overview of Contributions

For timing-based attack and IDS: we analyze CIDS and observe that CIDS computes clock skew inaccurately. To address this, we develop the Network Time Protocol (NTP)-based IDS that computes clock skew as defined in the NTP specification [39]. A key observation is that an adversary who realizes that CIDS at the receiving ECU computes the clock skew using message inter-arrival times can manipulate the inter-transmission times to match the clock skew of the targeted ECU and

avoid detection. We refer to this attack as the cloaking attack. We verify the NTP-based IDS and cloaking attack on the CAN bus testbed and the University of Washington (UW) EcoCAR [4].

For voltage-based attack and IRS: the VIDS must use two additional wires to connect analog pins of the microcontroller (the VIDS) to the CAN bus lines to measure the voltages of CANH and CANL. These two extra wires may introduce vulnerabilities to the CAN bus and ECUs. We observe that if the VIDS is compromised, instead of passively measuring the voltage levels through the two wires, the adversary can actively manipulate the voltage levels applied to the analog pins in order to impede the CAN transceivers' operation and disable the VIDS through the two connected wires. We study an attack surface of the VIDSs and develop four new voltage-based attacks: 1) the overcurrent attack, 2) the denial-of-service (DoS) attack, 3) the forced retransmission attack, and 4) the pulse attack. We also develop two hardware-based IRSs that detect the voltage-based attacks and mitigate them by isolating the compromised ECU. We verify the effectiveness of the voltage-based attacks and hardware-based IRSs on the CAN bus testbed.

For motion-based IDS: we notice that IDSs that fingerprint ECUs in the CAN bus cannot detect an attack in which an adversary only manipulates data of messages but does not change the transmitting ECU of the spoofed messages. Based on the fact that measurements from sensors that monitor the same physical movement of a vehicle are correlated, we develop the motion-based IDS (MIDS) that exploits the correlation between messages that contain speed-related data. We also analyze the detection probability of the MIDS by deriving bounds of the detection probability under an attack. We verify the effectiveness of the MIDS using data from two real vehicles that are the UW EcoCAR and Toyota Camry.

1.4 Organization of Dissertation

This dissertation is organized as follows. Chapter 2 presents the NTP-based IDS that exploits the clock skew of ECUs in order to fingerprint each ECU and the cloaking attack that can bypass the NTP-based IDS as well as CIDS by manipulating the transmission time of spoofed messages. In Chapter 3, four voltage-based attacks that may damage the microcontroller of the compromised ECU with VIDS, block transmission of messages, and make messages to be retransmitted are

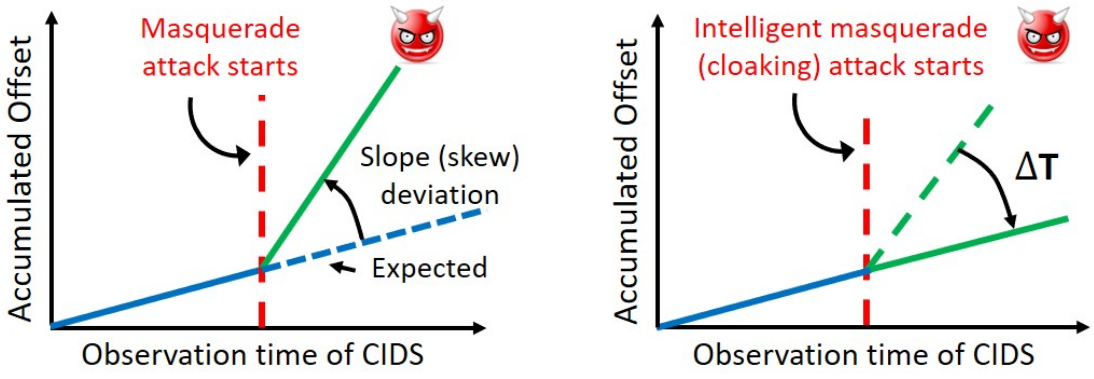
developed. Then, two hardware-based IRSs that can both detect and mitigate these voltage-based attacks are proposed. Chapter 4 presents the MIDS that exploits the correlation between messages that contain sensor measurements or control signals related to the physical movement of a vehicle such as speed.

Chapter 2

TIMING-BASED ATTACK AND INTRUSION DETECTION SYSTEM

2.1 Introduction

A CIDS detects an intrusion to the CAN bus by tracking abnormal deviations in the clock skew of the ECUs [18]. Since the CAN messages are transmitted according to the local clocks of the ECUs, the inter-arrival times of periodic messages depend on the clock skew of ECUs. Fig. 2.1 illustrates the accumulated offset of a periodic message over time, where an attack occurs after the red dash line. The slope of the accumulated offset curve indicates the clock skew computed by CIDS. As illustrated in Fig. 2.1 (a), the slope of the curve suddenly deviates from the expected



(a) (b)

Figure 2.1: Clock skew computation at CIDS. (a) CIDS detects deviations in the clock skew after the masquerade attack occurs. (b) An adversary adds a delay ΔT to message inter-transmission times in order to emulate the clock skew of the targeted ECU, and the cloaking attack bypasses CIDS.

slope after the masquerade attack occurs. CIDS computes the clock skew only using the inter-arrival time of the periodic messages because a CAN message does not have a transmit timestamp. An adversary may make the inter-arrival time of the spoofed messages to be the same as that of the legitimate messages by manipulating the inter-transmission time of the spoofed messages in which the adversary adds a delay ΔT to the inter-transmission time. The slope of the curve after the cloaking attack remains the same as that of the targeted message, as shown in Fig. 2.1 (b). As a result, the cloaking attack is not detected by CIDS.

In this chapter, we make the following contributions:

- We develop the NTP-based IDS that computes the clock skew as defined in the NTP specification.
- We develop the cloaking attack, in which an adversary adjusts message transmission time and cloaks its clock skew to match that of the targeted ECU.
- We introduce a new performance metric called the Maximum Slackness Index (MSI) to quantify how much leeway an attacker has to do the attack.
- We evaluate the cloaking attack on a CAN bus testbed and the UW EcoCar. The evaluations on the testbeds show that the cloaking attack is successful against both the CIDS and NTP-based IDS. We show that the NTP-based IDS has a smaller MSI than CIDS, which indicates the NTP-based IDS is more effective at detecting masquerade attacks.

2.2 Preliminaries

In this section, we explain the definitions of clock offset and clock skew, the adversary model, and CIDS.

2.2.1 Clock Offset and Clock Skew

We use clock-related terminologies as defined in the NTP specification [39]. Let C_{true} be the true clock (i.e., $C_{true}(t) = t$) and $C_A(t)$ be the time kept by clock A. The clock offset of C_A, denoted

by $O_A(t)$, is the difference between C_A and C_{true} as defined in Eq. (2.1).

$$O_A(t) = C_A(t) - C_{true}(t). \tag{2.1}$$

The frequency of C_A at time t is defined as the first derivative of C_A, which is denoted as $C'_A(t)$. The clock skew of C_A denoted by $S_A(t)$ is defined as Eq. (2.2).

$$S_A(t) = C'_A(t) - C'_{true}(t). \tag{2.2}$$

If the true clock does not exist or is not available, a local clock of the receiving ECU may be chosen as the reference clock. Then, the relative clock offset and the relative clock skew are computed according to the reference clock. Let us consider the clock skews of C_A and C_B, which are denoted as S_A and S_B, respectively. The clock skew of C_B relative to C_A, denoted as S_{BA}, is given by

$$S_{BA} = \frac{\Delta t_B - \Delta t_A}{\Delta t_A} = \frac{S_B - S_A}{1 + S_A}. \tag{2.3}$$

Similarly, S_{AB} can be computed as

$$S_{AB} = \frac{-S_{BA}}{1 + S_{BA}}. \tag{2.4}$$

2.2.2 Masquerade Attack

An adversary may compromise an ECU using various attack surfaces such as On-Board Diagnostics (OBD-II) port and telematics units [15]. As described in [18], we consider two types of attackers: 1) a weak attacker that can only suspend the transmission of messages from the weakly compromised ECU, and 2) a strong attacker that can both suspend the message transmission and inject spoofed messages from the fully compromised ECU. Using these two types of attackers, we discuss three possible attack scenarios, which are suspension attack, fabrication attack, and masquerade attack. In a suspension attack, a weakly compromised ECU is prohibited from transmitting certain messages by an adversary. In a fabrication, however, the adversary injects spoofed messages with legitimate message IDs using a fully compromised ECU. The suspension attack and fabrication attack significantly change the frequency of certain messages, and the existing IDSs easily detect these attacks [26, 36, 42].

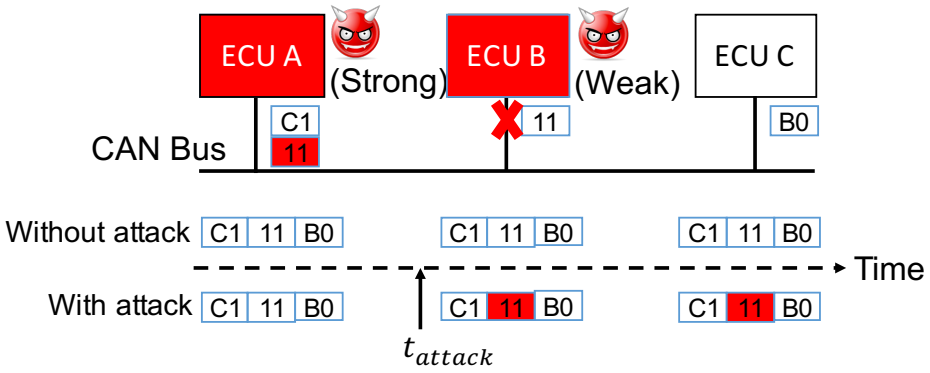


Figure 2.2: Illustration of a masquerade attack. ECU A is a fully compromised ECU, and ECU B is a weakly compromised ECU. Before the attack occurs, ECU B transmits message 0x11 every T seconds. At $t = t_{attack}$, the weak attacker suspends ECU B's transmission of message 0x11, and the strong attacker starts injecting spoofed messages 0x11 every T seconds using ECU A.

An adversary combines suspension attack and fabrication attack to launch a masquerade attack. In the masquerade attack, ECU A and ECU B are fully compromised and weakly compromised, as illustrated in Fig. 2.2. An ultimate goal of the masquerade attack is that the fully compromised ECU A impersonates ECU B. During the masquerade attack, the adversary blocks transmission of message 0x11 from ECU B, and ECU A transmits spoofed message 0x11. As shown in Fig. 2.2, the adversary transmits a spoofed message 0x11 from ECU A in the same period of the legitimate message 0x11. As a consequence, the masquerade attack does not change the traffic through the CAN bus. The IDSs that monitors and tracks the abnormal deviations in the traffic through the CAN bus cannot detect the masquerade attack.

2.2.3 Overview of CIDS

Based on the fact that an ECU transmits periodic messages according to its local clock, the clock skew can be exploited as a fingerprint of each ECU. Since a transmit timestamp is not available in a CAN message, clock skew computation methods in [27, 28, 57] are not applicable in the CAN protocol. In the CAN protocol, a CIDS uses message periodicity to extract the clock skew of each

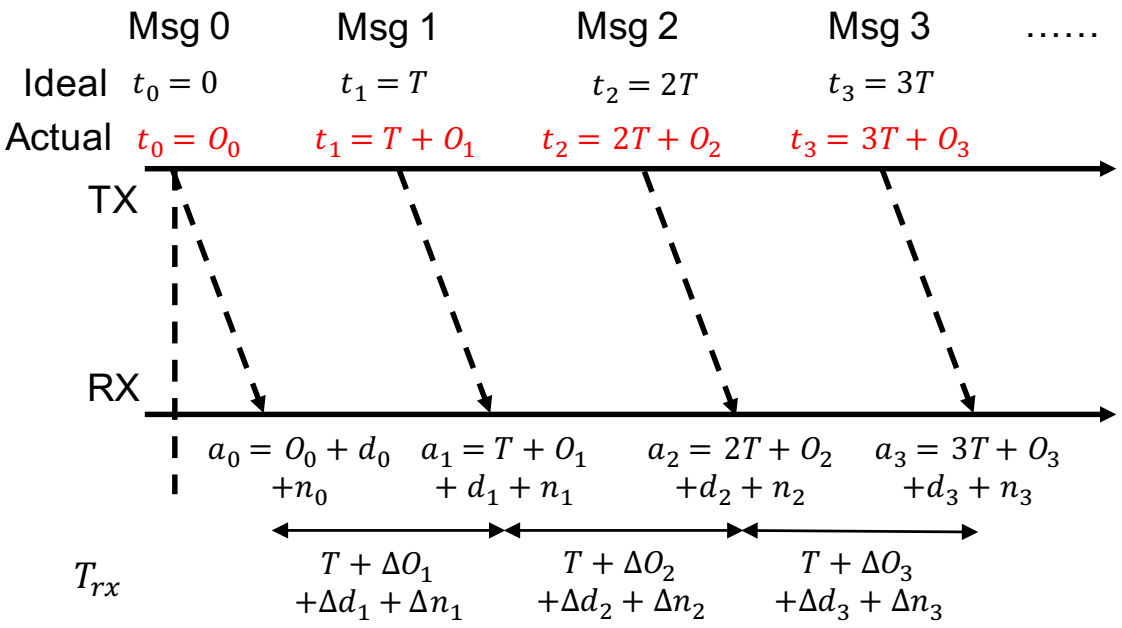


Figure 2.3: Timing model of the transmission and reception of periodic messages in the CAN protocol. Due to the clock skew of the transmitting ECU, an accumulated offset is added to the inter-arrival times of the periodic messages. In addition to the accumulated offset, the network delay and quantization error affect the inter-arrival time.

ECU in the CAN bus [18].

Timing model of CAN message transmission: Fig. 2.3 illustrates a timing model of the periodic message for the receiving ECU in the CAN protocol. Only receive timestamps are available at the receiving ECU denoted as ECU RX, we use ECU RX's clock as the reference clock. Based on this reference clock, we compute the relative clock offset and relative clock skew, respectively.

Consider that a transmitting ECU denoted as ECU TX sends a message every T seconds according to its local clock. In an ideal case where clocks of ECU TX and ECU RX are synchronized, message i will be transmitted at $t_i = iT$ according to the local clock of ECU RX. Due to the clock skew, however, the actual transmission time of message i is $t_i = iT + O_i$ in ECU RX's clock where O_i is an accumulate offset when message i is transmitted. For consistency, we adopt the

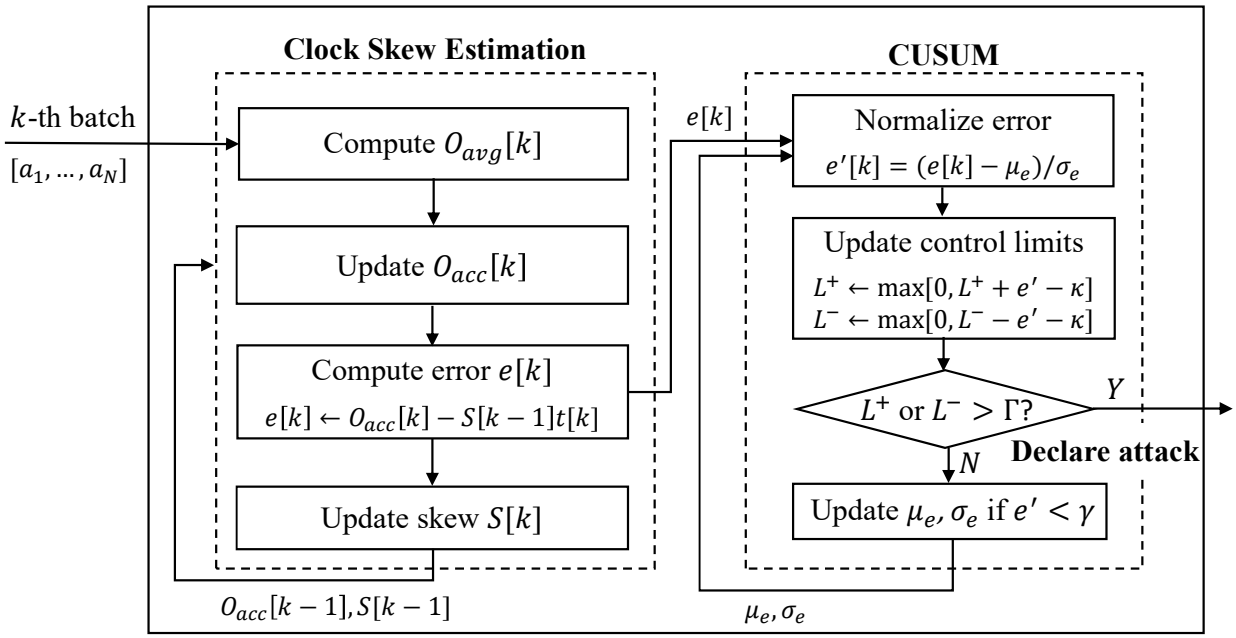


Figure 2.4: Workflow of CIDS. CIDS is composed of clock skew estimation block that estimates the clock skew and CUSUM block that tracks the estimated clock skew and detects abnormal behavior in the clock skew.

version of formula $t_i = iT + O_i$ from [18], instead of $t_i = iT - O_i$, which is the version that is consistent with the NTP specifications. Nevertheless, it does not affect our following analysis. We consider a network delay d_i due to the processing delay and propagation delay and a zero-mean noise n_i due to a quantization process [57]. Then, the receive timestamp of message i, a_i, becomes $iT + O_i + d_i + n_i$. The inter-arrival time between messages $(i-1)$ and i, $T_{rx,i}$, is given by

$$T_{rx,i} = T + (O_i - O_{i-1}) + (d_i - d_{i-1}) + (n_i - n_{i-1}) = T + \Delta O_i + \Delta d_i + \Delta n_i,$$

where ΔO_i is the clock offset in period i, and Δd_i and Δn_i are the differences in network delay and quantization noise between periods i and $(i-1)$, respectively. Since $\mathbb{E}[\Delta d_i] = 0$ and $\mathbb{E}[\Delta n_i] = 0$, $\mathbb{E}[T_{rx,i}] = T + \mathbb{E}[\Delta O_i]$ [18].

Workflow of CIDS: Fig. 2.4 describes the workflow of the CIDS. A CIDS consists of two blocks: clock skew estimation and Cumulative Sum (CUSUM). The clock skew estimation block

takes the timestamps of N newly arrived messages as the input. For the k-th batch, the CIDS computes the average offset $O_{avg}[k]$ and the accumulated offset $O_{acc}[k]$. The CIDS uses the absolute value of the average offset when it updates the accumulated offset, whereas the NTP-based IDS uses the original value of the average offset. Then, the identification error $e[k]$ is computed and used to obtain the updated clock skew $S[k]$ using the RLS algorithm.

The CUSUM block takes the identification error as an input. It logs the statistics of all past identification errors, especially the mean and standard deviation of the identification errors. As a new identification error is fed into the CUSUM block, it normalizes the identification error using the mean and standard deviation. Then, the CUSUM block updates the control limits using that normalized identification error. If either an upper control limit L^+ or lower control limit L^- exceeds the detection threshold Γ, the CIDS declares an attack. In order to be robust against noise, if the normalized error $e'[k]$ is less than the threshold γ, the mean and standard deviation will be updated using the new identification error and all past identification errors. Otherwise, the new identification error will be dropped, and the mean and standard deviation will not be updated.

Clock skew detector: In order to compute the clock skew from receive timestamps, the CIDS processes the received messages in a batch of size N and computes the average offset in the k-th batch as follows

$$O_{avg}[k] = \frac{1}{N-1} \sum_{i=2}^{N} \left[a_i - (a_1 + (i-1)\mu_T[k-1]) \right], \tag{2.5}$$

where $\mu_T[k-1]$ is the average inter-arrival time of the previous batch. Then, the accumulated offset in the k-th batch, $O_{acc}[k]$, is computed by adding an absolute value of the average offset to the accumulate offset in the $(k-1)$-th batch as in Eq. (2.6).

$$O_{acc}[k] = O_{acc}[k-1] + |O_{avg}[k]|. \tag{2.6}$$

The accumulated offset is modeled as $O_{acc}[k] = S[k]t[k] + e[k]$, where $S[k]$ is the regression parameter that indicates the clock skew, $t[k]$ is the elapsed time, and $e[k]$ is the identification error. The regression parameter S is estimated using the Recursive Least Squares (RLS) algorithm that minimizes the identification error [25].

The identification error is used as an indicator of an attack. In a masquerade attack, an impersonating ECU's clock skew is different from that of the targeted ECU. The identification error significantly increases due to a sudden change in the clock skew. CIDS tracks the mean μ_e and standard deviation σ_e of the identification errors, which are updated only if $|(e - \mu_e)/(\sigma_e)| < \gamma$ where γ is an update threshold. By letting $\theta_e = (e - \mu_e)/\sigma_e$, the upper and lower control limits of the CUSUM, L^+ and L^-, are updated for each new error sample as $L^+ = \max(0, L^+ + \theta_e - \kappa), L^- = \max(0, L^- - \theta_e - \kappa)$, where κ is a sensitivity parameter. If either the control limit exceeds a detection threshold Γ, the CIDS declares an attack. The values of γ, κ, and Γ chosen in [18] are 3, 5, and 5, respectively.

Correlation detector: The authors of [18] demonstrated that the average offsets of two different messages are likely to be equivalent and show a high correlation (i.e., the correlation coefficient close to 1) if those messages are transmitted from the same ECU. If two messages, however, are transmitted from two different ECUs, their average offsets would have a low correlation. The correlation detector exploits the correlation between the average offsets of two messages as an indicator of an attack. The correlation detector tracks the correlation coefficient of two highly correlated messages, which is known to the correlation detector in prior. The correlation detector declares an attack if the correlation coefficient ρ is less than a detection threshold. As a result, in cases where the impersonating ECU happens to have a similar clock skew with the targeted ECU, the masquerade attack may bypass the clock skew detector, but the attack would still be detected by the correlation detector. It is important to note that the clock skew detector can be applied to a periodic message, but the correlation detector is only applicable to a pair of periodic messages with highly correlated average offsets. We observe that a pair of messages does not show a high correlation even though those messages are transmitted from the same ECU. That is, a high correlation between the average offsets of two messages is more likely to exist when the messages are consecutively transmitted by the same ECU and also consecutively received by the receiving ECU.

2.3 NTP-based IDS

In this section, we present the NTP-based IDS that computes clock offsets and clock skews according to the NTP specification [39]. The motivation for developing the NTP-based IDS is two-fold. First, we note that the average offset is not computed in CIDS (i.e., Eq. (2.5)) is not consistent with the definition in the NTP specification as Eq. (2.1) because CIDS does not calculate the difference between the transmitting ECU's clock and the reference clock. Also, it is assumed that O_i is a random variable and $\mathbb{E}[\Delta O_i] = 0$, which implies that $\mathbb{E}[O_i] = \mathbb{E}[O_j]$ for $i \neq j$. This, however, does not hold in general because clock offset accumulates over time (e.g., if $i \gg j$, $\mathbb{E}[O_i] \gg \mathbb{E}[O_j]$). The second motivation is the widespread acceptance of the NTP as a timing mechanism for real-time systems, which raises the question of whether the NTP can be used for intrusion detection as well. The key difference between the CIDS and NTP-based IDS is clock skew estimation.

2.3.1 Clock Skew Estimation according to the NTP Specification

In the NTP-based IDS, the accumulated offset up to message i is modeled as a random variable $O_i = iO + \epsilon_i$, where O is the constant clock offset in each period T due to the constant clock skew and ϵ_i is the offset deviation due to ECU jitters. Since ϵ_i's are assumed to be independent and identically distributed, the expected accumulated offset increases linearly. Consider two consecutive receive timestamps a_{i-1} and a_i. When the message period is T in the transmitter's clock, the observed period in the receiver's clock is $T_{rx,i} = a_i - a_{i-1}$. According to Eq. (2.1), the observed clock offset becomes as follows

$$\hat{O}_i = T - (a_i - a_{i-1}) = -(O + \Delta\epsilon_i + \Delta d_i + \Delta n_i), \tag{2.7}$$

where $\Delta\epsilon_i = \epsilon_i - \epsilon_{i-1}$ and $\mathbb{E}[\Delta\epsilon_i] = 0$. The average offset of the k-th batch, $O_{avg}[k]$, is computed using a batch of N messages as Eq. (2.8).

$$O_{avg}[k] = \frac{1}{N}\sum_{i=1}^{N}\hat{O}_i = \frac{1}{N}\sum_{i=1}^{N}[T - (a_i - a_{i-1})] = T - \frac{a_N - a_0}{N}, \tag{2.8}$$

where a_0 is the receive timestamp of the last message in the previous batch. The accumulated offset up to the last message of the k-th batch, $O_{acc}[k]$, is given as Eq. (2.9).

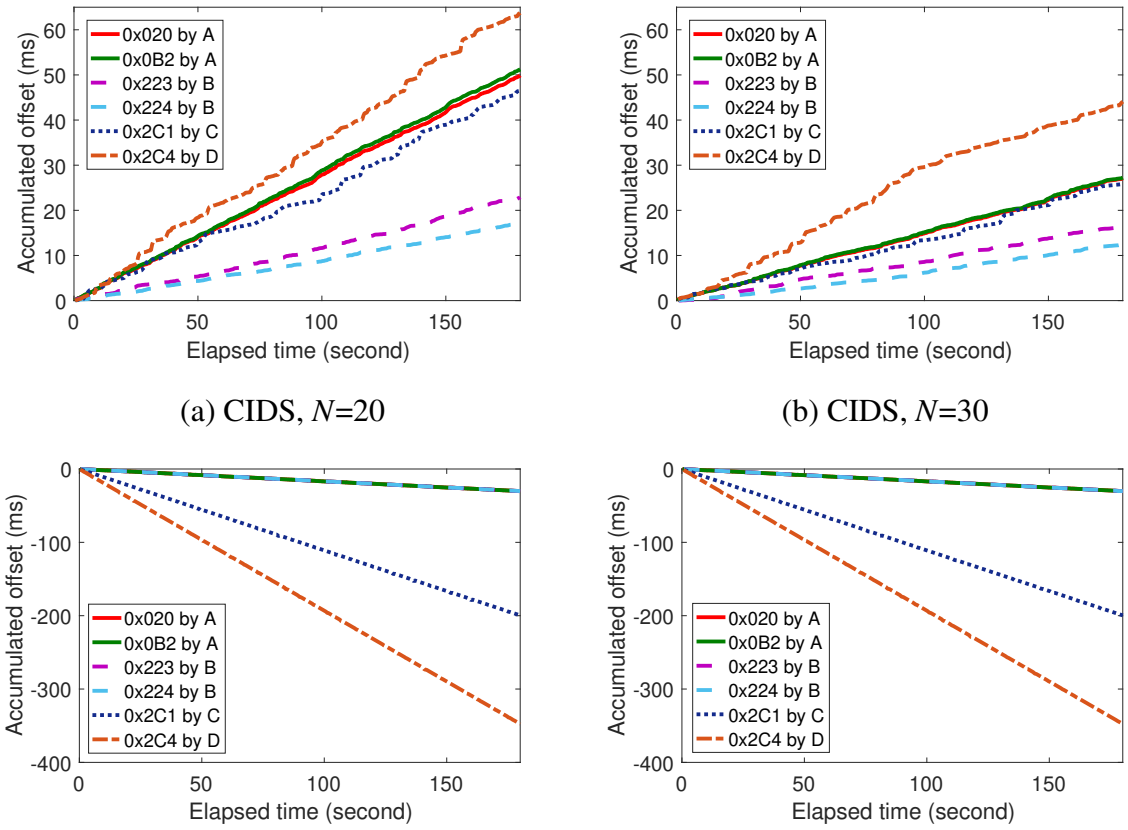
$$O_{acc}[k] = O_{acc}[k-1] + N \cdot O_{avg}[k]. \tag{2.9}$$

Note that the original value of the average offset is used in the NTP-based IDS instead of the absolute value of the average offset in CIDS, by which the NTP-based IDS computes the clock skew in a consistent definition as the NTP specification. The other components of the NTP-based IDS remain the same as CIDS.

2.3.2 Estimation Consistency in the NTP-based IDS

We experimentally demonstrate that the NTP-based IDS computes the clock skew of a periodic message consistently for various settings of the NTP-based IDS, while the clock skew computed by the CIDS varies for different settings of the CIDS. As a physical property of an ECU, a clock skew is considered to be constant over time, and thus the estimated clock skew should be consistent across 1) different batch sizes used by the CIDS and NTP-based IDS, 2) different portions of the same data trace, and 3) different traces of the same ECU. Using the Toyota Camry dataset [49], we demonstrate that the clock skew computed by the NTP-based IDS is consistent. Fig. 2.5 illustrates the accumulated offsets computed by the CIDS and NTP-based IDS with different batch sizes (i.e., N=20 and N=30). Due to the lack of ground truth, the authors in [18] empirically identified that 0x020, 0x0B2, 0x223 and 0x224 are transmitted by two different ECUs. Based on the NTP-based clock skew estimation results, however, we believe that the four messages come from the same ECU. A significant difference in slopes of the curves, which indicate the clock skew, for the same message are observed for CIDS as illustrated in Figs. 2.5(a) and 2.5(b). For example, the clock skew of message 0x020 (based on the endpoint of the curve) is computed to be around 273ppm with the batch size of 20, but the clock skew of the same message is dropped to around 151ppm when the batch size is 30. The NTP-based IDS, however, provides a consistent estimation of the clock skew for various values of the batch size.

In order to demonstrate the consistency in clock skew computation in the NTP-based IDS, we



(a) CIDS, N=20

(b) CIDS, N=30

(c) NTP-based IDS, N=20

(d) NTP-based IDS, N=30

Figure 2.5: Accumulated offsets computed by the CIDS and NTP-based IDS with batch sizes of 20 and 30. The same portion of the data trace (with the trace ID of 25) from the Toyota dataset is used. A significant difference in the slope of the curve, which indicates the clock skew, is observed for the same messages in CIDS as the batch size changes from 20 to 30. The clock skew estimated by the NTP-based IDS is almost identical with different batch sizes.

consider the following three cases: 1) use the same portion of the same trace, and vary N from 20 to 100 with a step of 20, 2) set N=20, and use different portions of the same trace by omitting the first m messages, where m is varied from 1 to 19, and 3) set N=20, and use 14 different traces from the Toyota Camry dataset. If the standard deviation of the estimated clock skews is smaller, the computed clock skew is more consistent for various settings of the CIDS and NTP-based IDS.

Table 2.1: Standard deviations (σ_1, σ_2, σ_3) of clock skews computed by the CIDS and NTP-based IDS using Toyota Camry dataset. The NTP-based IDS has a significantly smaller standard deviation than CIDS, which indicates its consistency in clock skew computation.

Message ID	CIDS			NTP-based IDS		
	σ_1	σ_2	σ_3	σ_1	σ_2	σ_3
0x020	92.3682	12.0589	20.1727	0.3706	0.2000	1.7716
0x0B2	94.4480	11.7543	19.4549	0.4252	0.2045	1.7929
0x223	41.6631	16.2060	25.4885	0.3083	0.4429	1.6437
0x224	29.0736	17.0442	32.6059	0.8348	0.5491	2.3660
0x2C1	85.3753	7.9963	26.2618	0.0866	1.2191	3.2977
0x2C4	116.5630	13.0896	53.7820	1.0763	1.1599	3.3602

As shown in Table 2.1, the NTP-based IDS has a smaller standard deviation than the CIDS for all six messages in all three cases.

2.4 Cloaking Attack

In this section, we develop the cloaking attack on the clock skew detector in which the adversary adjusts the inter-transmission time of spoofed messages to manipulate the estimated clock skew. Also, we develop the cloaking attack on the correlation detector, in which the average offset of spoofed messages is highly correlated with that of legitimate messages. As a result, the cloaking attack can bypass the CIDS and NTP-based IDS.

2.4.1 Cloaking Attack on Clock Skew Detector

For ease of the mathematical derivation, let us assume that the receiving ECU RX has the true clock. Without loss of generality, the same derivation can be applied to the case where ECU RX's

clock is not the true clock. In such a case, the clock skews become the relative clock skews with respect to ECU RX's clock. Consider that the targeted ECU B transmits a message every T seconds according to its local clock, which corresponds to every $\hat{T} = T/(1 + S_B)$ seconds in ECU RX's clock where S_B is ECU B's clock skew. For the ease of discussion, we ignore offset deviations and the noise in arrival timestamps due to network delay and quantization. Then, ECU B's clock skew estimated by ECU RX, $\hat{S}$, becomes $\hat{S} = (T - \hat{T})/\hat{T} = S_B$. In a masquerade attack, the adversary prevents ECU B from transmitting messages because ECU B is weakly compromised. The adversary lets ECU A that is fully compromised transmit the spoofed messages every T seconds, according to ECU A's local clock, as illustrated in Fig. 2.2. ECU RX receives the spoofed message every $\hat{T}' = T/(1 + S_A)$ seconds in ECU RX's local clock where S_A is the clock skew of ECU A. The clock skew computed by ECU RX becomes S_A after the masquerade attack occurs. A CIDS detects a sudden deviation in the clock skew since $S_A \neq S_B$ and declares the masquerade attack.

The key idea of the cloaking attack is that, although the clock skew is a physical property of an ECU, a clock skew estimation based entirely on the inter-arrival times of messages can be easily manipulated by an adversary (i.e., a fully compromised ECU A) if the adversary adjusts transmission times of the spoofed message. As a consequence, the adversary cloaks the skew of its hardware clock, which motivates the term cloaking attack. In the cloaking attack, instead of transmitting the spoofed messages every T seconds according to the local clock of the fully compromised ECU A, ECU A transmits the spoofed messages every $\tilde{T} = T + \Delta T$ seconds in order to match the inter-arrival times of the spoofed messages to that of the legitimate messages with respect to ECU RX.

The value of ΔT can be chosen as follows. In the cloaking attack, the inter-arrival time of the spoofed message observed by ECU RX is

$$\hat{T}'' = \frac{\tilde{T}}{1 + S_A} = \frac{T + \Delta T}{1 + S_A}, \tag{2.10}$$

and the transmitter's clock skew estimated by ECU RX is given as

$$\hat{S}'' = \frac{T - \hat{T}''}{\hat{T}''} = \frac{S_A \cdot T - \Delta T}{T + \Delta T}. \tag{2.11}$$

In order to bypass the CIDS and NTP-based IDS, the adversary needs to choose ΔT such that $\hat{S}'' = \hat{S}$, or equivalently $\hat{T}'' = \hat{T}$. As a result, ΔT can be represented as

$$\Delta T = \frac{(S_A - S_B)}{1 + S_B} \cdot T = S_{AB} \cdot T = \frac{-S_{BA}}{1 + S_{BA}} \cdot T, \tag{2.12}$$

where S_{AB} is the clock skew of ECU A relative to the clock skew of ECU B, and the last two equalities are due to Eq (2.3) and Eq. (2.4), respectively. Hence, the message inter-transmission time $\tilde{T}$ would be

$$\tilde{T} = T + \Delta T = T - \frac{S_{BA}}{1 + S_{BA}} T = \frac{T}{1 + S_{BA}}, \tag{2.13}$$

which is the period of the message from ECU B (i.e., a weakly compromised ECU) measured by the local clock of ECU A (i.e., a fully compromised ECU).

In summary, the cloaking attack is performed as follows. An adversary compromises two ECUs fully and weakly, respectively. A fully compromised ECU measured the period of the target message $\tilde{T}$ according to its local clock. While the cloaking attack is launched, the fully compromised ECU transmits spoofed messages every $\tilde{T}$ seconds. Although the preceding analysis ignores the noise in the CAN bus network, our analysis shows that the cloaking attack is effective in a realistic environment.

2.4.2 Maximum Slackness Index (MSI)

In practice, the adversary will not be able to precisely match its clock skew to the clock skew of the targeted ECU due to hardware limitations such as the resolution of the clock. Deviations between the clock skew of the fully compromised ECU (i.e., the adversary) and that of the targeted ECU, however, may still be mistaken for random delays and quantization errors by the CIDS and NTP-based IDS. These sources of randomness create an interval of ΔT that an adversary can introduce while the attack is not detected. A more effective the detector has a smaller interval of ΔT. We introduce a metric that formalizes this notion of an interval of ΔT as follows. Let $P_s(\Delta T)$ denote the probability of a successful cloaking attack when the added delay is ΔT. We define the upper

and lower limits of ΔT for a successful cloaking attack as

$$(\Delta T)_{\max}(\epsilon) = \max\{\Delta T : P_s(\Delta T) > 1 - \epsilon\}$$
$$(\Delta T)_{\min}(\epsilon) = \min\{\Delta T : P_s(\Delta T) > 1 - \epsilon\}$$

We define the ϵ-Maximum Slackness Index (ϵ-MSI) of the adversary against the CIDS and NTP-based IDS as

$$\epsilon\text{-MSI} = (\Delta T)_{max}(\epsilon) - (\Delta T)_{min}(\epsilon). \tag{2.14}$$

The normalized ϵ-MSI is defined as the ratio between of ϵ-MSI (in μs) and the message period (in second), and its unit is ppm. A smaller value of ϵ-MSI indicates a more effective detector, and the attacker has less freedom in choosing ΔT since the adversary's clock skew must closely match the targeted ECU's clock skew in order to bypass the CIDS and NTP-based IDS.

2.4.3 Cloaking Attack on Correlation Detector

It is common that an ECU transmits multiple messages that have different IDs with the same or different periods. We call the messages that are transmitted from the same ECU as sibling messages. We discuss why the average offsets of two messages that are transmitted and received consecutively are highly correlated. Although we analyzed the correlation between the average offset for the NTP-based IDS, the same analysis can be applied to CIDS.

Denote the i-th message in the k-th batch for messages v and w as $v_{k,i}$ and $w_{k,i}$, which are transmitted at $t_{k,i}^{(v)}$ and $t_{k,i}^{(w)}$, respectively. This is another requirement for two messages to be highly correlated: the two consecutively transmitted messages needs to be processed as simultaneously as the i-th message in the k-th batch. Without loss of generality, suppose that $w_{k,i}$ is transmitted right after $v_{k,i}$. Let Δt be the transmission duration of each message v, which is constant, given the fixed message length and CAN bus speed. As a result, we have $t_{k,i}^{(w)} = t_{k,i}^{(v)} + \Delta t$.

Consider the first case where $v_{k,i}$ and $w_{k,i}$ are received consecutively at $a_{k,i}^{(v)}$ and $a_{k,i}^{(w)}$, which means no other messages with higher priority IDs are received between $a_{k,i}^{(v)}$ and $a_{k,i}^{(w)}$ due to arbitration. For simplicity, we assume constant network delays for both messages (denoted as

d_v and d_w, respectively), and ignore quantization noise at the receiver. Then, we have $a_{k,i}^{(w)} = a_{k,i}^{(v)} + \Delta t + (d_w - d_v)$.

In the NTP-based IDS, the estimated average offset for messages v and w in the k-th batch are

$$\begin{aligned} O_{avg}^{(v)}[k] &= T - \frac{1}{N}\left(a_{k,N}^{(v)} - a_{k,0}^{(v)}\right) \\ &= -O^{(v)} - \frac{1}{N}\left(\epsilon_{k,N}^{(v)} - \epsilon_{k,0}^{(v)}\right) \\ O_{avg}^{(w)}[k] &= T - \frac{1}{N}\left(a_{k,N}^{(w)} - a_{k,0}^{(w)}\right) = O_{avg}^{(v)}[k]. \end{aligned} \tag{2.15}$$

Since $O_{avg}^{(v)}[k]$ and $O_{avg}^{(w)}[k]$ are the k-th realizations of the random variables $O_{avg}^{(v)}$ and $O_{avg}^{(w)}$, respectively, Eq. (2.15) implies $O_{avg}^{(w)} = O_{avg}^{(v)}$. As a consequence, their correlation coefficient ρ becomes as high as 1. In general, as along as the two messages are received with a constant delay, the average offsets of those two messages are highly correlated. In practice, however, the correlation would slightly decrease due to network delay variations and quantization noise at the receiver.

We examine the second case in which messages with higher priority IDs are received in between the two messages. Let the arbitration delay be $d_{k,i} \geq 0$, and thus $a_{k,i}^{(w)} = a_{k,i}^{(v)} + \Delta t + (d_w - d_v) + d_{k,i}$. Then, the average offset can be written as

$$O_{avg}^{(w)}[k] = O_{avg}^{(v)}[k] - \frac{1}{N}(d_{k,N} - d_{k,0}), \tag{2.16}$$

where the second term may be considered as the k-th realization of a random variable D, independent of $O_{avg}^{(v)}$ and $O_{avg}^{(w)}$. We have $O_{avg}^{(w)} = O_{avg}^{(v)} + D$, and

$$\rho\left(O_{avg}^{(v)}, O_{avg}^{(w)}\right) = \frac{\sqrt{Var(O_{avg}^{(v)})}}{\sqrt{Var(O_{avg}^{(v)}) + Var(D)}} < 1.$$

As a result, depending on the variance of arbitration delay, the correlation in the second case may be much smaller than 1. On the other hand, if two messages are transmitted from different ECUs, the average offset becomes

$$O_{avg}^{(w)}[k] = -O^{(w)} - \frac{1}{N}\left(\epsilon_{k,N}^{(w)} - \epsilon_{k,0}^{(w)}\right).$$

Since $\{\epsilon_{k,i}^{(v)}\}$ and $\{\epsilon_{k,i}^{(w)}\}$ are independent, $O_{avg}^{(w)}$ is also independent of $O_{avg}^{(v)}$, which implies $\rho \approx 0$. The above analysis is supported the experimental results.

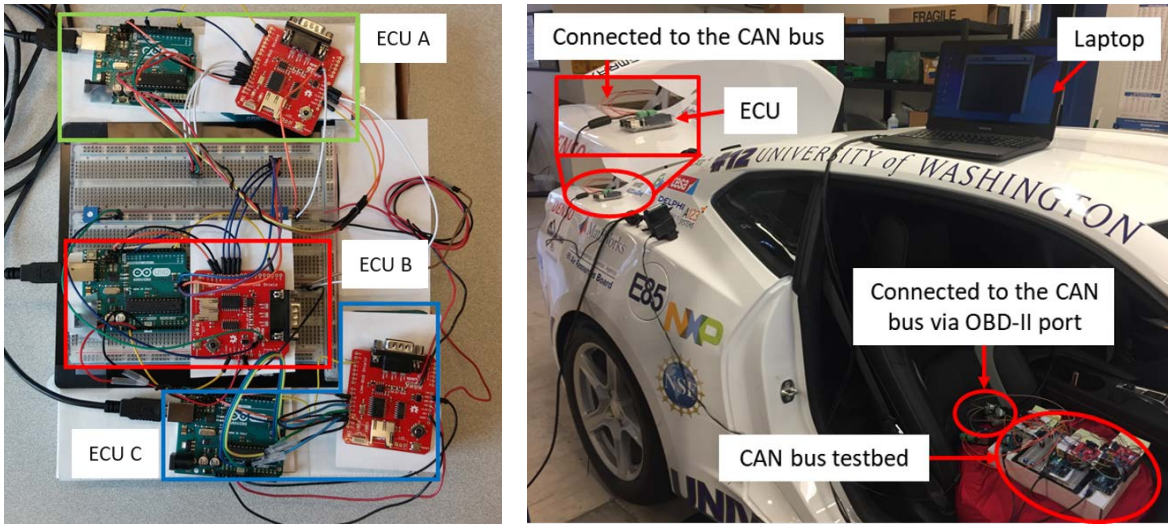
In order to bypass the correlation detector, the attacker adopts the following strategy. Before executing the attack, the attacker observes the targeted message over a period of time and identifies any sibling messages of the targeted message. During the cloaking attack on the correlation detector, the fully compromised ECU A transmits the spoofed message immediately after the sibling message is transmitted. Since the transmission of the spoofed messages from ECU A begins as soon as the sibling message is received, the average offset of the spoofed messages and sibling messages will be equivalent. Consequently, the average offsets of these two messages are highly correlated even though the two messages are not transmitted from the same ECU, as described by Eq. (2.15). Note that Eq. (2.15) also implies that their accumulated offsets, as well as the clock skews, are equivalent, which means the cloaking attack on the correlation detector can bypass the clock skew detector.

2.5 Experimental Result

In this section, we evaluate the performance of the cloaking attack on a CAN bus testbed and the UW EcoCAR and demonstrate that the cloaking attack can bypass both the CIDS and NTP-based IDS. We first describe our testbeds, followed by an illustration of a single trial run of the cloaking attack. We provide detailed results for the cloaking attack against both the clock skew and correlation detectors.

2.5.1 Testbed

Fig. 2.6(a) shows the CAN bus testbed that consists of three testbed ECUs. Each testbed ECU is composed of an Arduino UNO Rev3 board and a Sparkfun CAN bus shield. The CAN bus shield has a Microchip MCP2515 CAN controller and a Microchip MCP2551 CAN transceiver. The two bus lines of the CAN bus testbed are terminated by two 120Ω resistors. The CAN bus speed is set to 500kbps that is widely used in contemporary cars [4, 18]. ECU A is the receiving ECU that logs all messages. ECU B is the targeted ECU that is a weakly compromised ECU. ECU B transmits messages 0x11 every 100ms. ECU C, which is a fully compromised ECU, is used to launch the



(a) CAN bus testbed (b) UW EcoCar testbed

Figure 2.6: CAN bus testbed and UW EcoCAR testbed. The CAN bus testbed consists of three testbed ECUs. And the two bus lines of the CAN bus testbed are terminated with two 120Ω resistors. The CAN bus speed is set to 500kbps. We connect the CAN bus testbed to the CAN bus network of the UW EcoCAR through the OBD-II port. ECU C in the CAN bus testbed launches the cloaking attack by injecting spoofed message 0x180 in order to impersonate message 0x184.

cloaking attack to impersonate ECU B. ECU C transmits messages every 100ms.

The UW EcoCAR testbed is shown in 2.6(b). The UW EcoCAR provides real CAN bus network environment for evaluating and demonstrating the cloaking attack More than 2500 messages with 89 different message IDs are transmitted per second in the UW EcoCAR. The CAN bus testbed is connected to the CAN bus of the EcoCAR via the OBD-II port. During the experiments, the UW EcoCAR is in the park mode in an isolated and controlled environment, but all ECUs are functional and exchange CAN messages. The Arduino-based testbed ECU cannot log all messages on the CAN bus due to its computing limitation. We build a fourth testbed ECU that consists of a Raspberry Pi 3, whose clock speed is 1.4GHz, and a PiCAN 2 board, which has the same CAN controller and transceiver as in the CAN bus shield, as a receiving ECU. A stock ECU that is transmitting message 0x184 every 100ms is considered as the targeted ECU, and an Arduino-based

testbed ECU (i.e., ECU C) launches the cloaking attack by injecting spoofed messages whose message ID is 0x180.

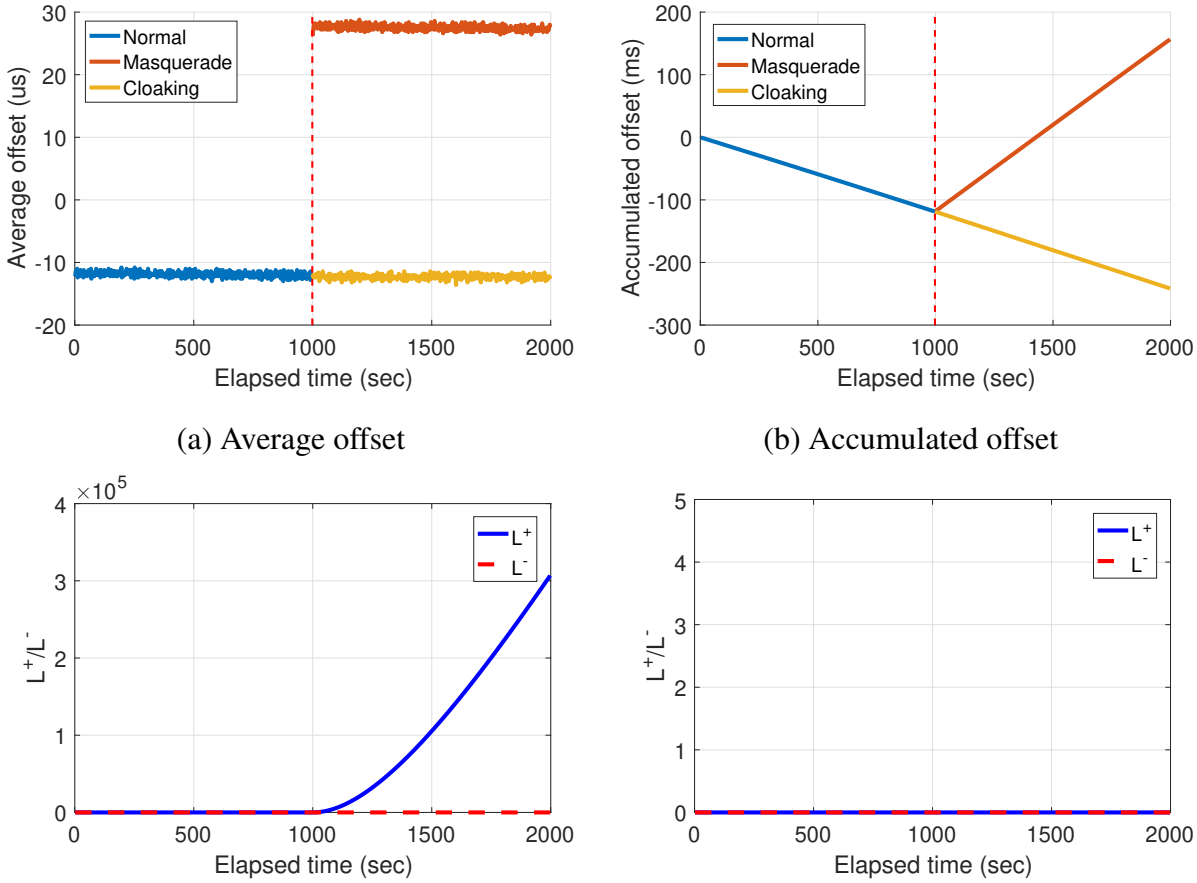
2.5.2 Cloaking attack on clock skew detector

We demonstrate one example of the masquerade attack and the cloaking attack on the NTP-based IDS using the CAN bus testbed. We compare the masquerade attack and the cloaking attack with respect to the average offset, accumulated offset, and the control limits of the CUSUM detector in the NTP-based IDS. In this example, we set the update threshold γ to 4, the detection threshold Γ to 5, and the sensitivity parameter κ to 5 in order to avoid a false alarm.

The NTP-based IDS tracks the clock skew of message 0x11 from the targeted ECU B for 1000s before the attack occurs. Then, the NTP-based IDS is fed with the timestamps of the attack messages. For the masquerade attack, the fully compromised ECU C transmits every T=100ms according to its local clock. For the cloaking attack, the fully compromised ECU C observes the inter-arrival time of message 0x11 to be around 100040μs according to its local clock and sets the message inter-transmission time to $\tilde{T}$=100040μs (i.e., ΔT=40μs).

As shown in Fig. 2.7(a), the average offset increases from -12μs to 28μs as soon as the masquerade attack happens. Also, the slope of the curve suddenly changes from -118ppm to 275ppm, as shown in Fig. 2.7(b) due to the clock skew difference between ECUs B and C. As a result, such deviations in the clock skew add up and cause the control limits of the NTP-based IDS to increase under the masquerade attack, as shown in Fig. 2.7(c). Under the cloaking attack, however, the average offset remains the same as that before the attack happens, as demonstrated in Fig. 2.7(a). Also, the slope of the curve stays almost the same as the original curve, as shown in Fig. 2.7(b). As a result, the control limits are always zero, and thus the NTP-based IDS cannot detect the cloaking attack as demonstrated in Fig. 2.7(d). The experimental results on the EcoCAR lead to a similar observation.

In order to launch the cloaking attack on the CAN bus testbed and the UW EcoCAR, ECU C transmits a spoofed message every 100040μs (ΔT=40μs) on the CAN bus testbed to spoof the 10Hz message 0x11 and every 99971μs (ΔT=-29μs) to spoof message 0x184 on the UW EcoCAR.



(a) Average offset

(b) Accumulated offset

(c) Control limits under the masquerade attack

(d) Control limits under the cloaking attack

Figure 2.7: Average offset, accumulated offset, and control limits of the NTP-based IDS under the masquerade attack and the cloaking attack on the CAN bus testbed. After the masquerade attack occurs, the average offset suddenly changes due to the clock skew difference between ECUs B and C. As a result, the masquerade attack is detected by the NTP-based IDS. Under the cloaking attack, the average offset remains close to that of ECU B, and the clock skew that is computed at the NTP-based IDS does not change after the attack. Consequently, the cloaking attack bypasses the NTP-based IDS.

We set ΔT to -28μs and changed it to -32μs every five messages so that $\Delta T \approx$ -29μs on average because the time resolution of the Arduino UNO Rev3 board is 4μs. We collected 8.5 hours of

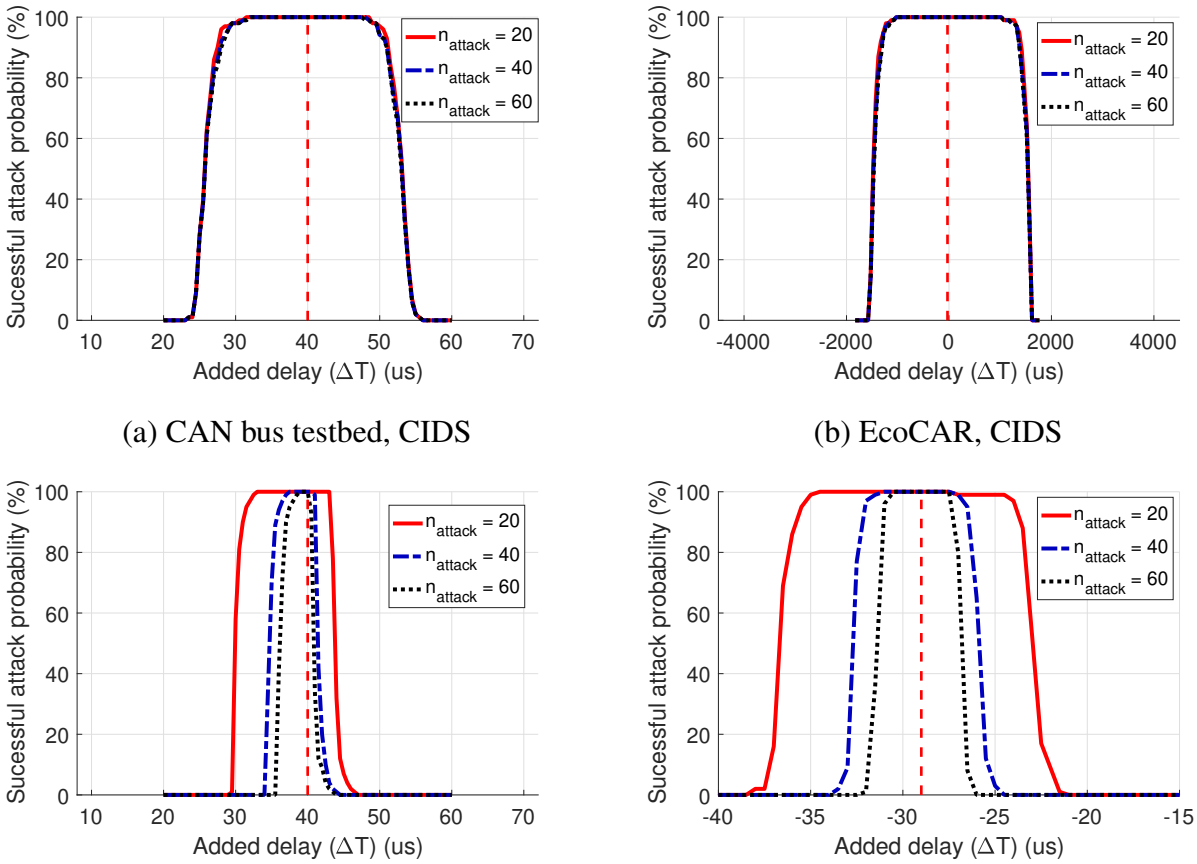
attack data from the CAN bus testbed and the UW EcoCAR, respectively.

Both the CIDS and NTP-based IDS are fed with 1000 batches of normal data, followed by n_{attack} batches of attack data in each experiment. We assume that the first attack message is received at the next expected time of the targeted message. The impact of mistiming on the cloaking attack is also studied, and the successful attack probability is analyzed. A total of 100 non-overlapping segments of size n_{attack} from the attack data are prepared in order to simulate 100 independent attack experiments. To measure the performance of the cloaking attack, we compute successful attack probability, P_s, which is the percentage of experiments where the attack is successful.

We set the batch size N to 20 for both the CIDS and NTP-based IDS. For CIDS, γ=3 and Γ=5 [18]. For the NTP-based IDS, we set γ=4 and Γ=5. For the data collected from the CAN bus testbed, κ is set to 5, whereas it is set to 8 for the data collected from the UW EcoCAR. For the value of ΔT achieved in the evaluation, P_s=1 for both the CIDS and NTP-based IDS, which is indicated by the red dash line in Fig. 2.8. In order to gain additional insight into the performance of each IDS under a cloaking attack, we generate additional data sets by adding different values of ΔT to the message inter-arrival times and analyze the new datasets.

On the CAN bus testbed, with n_{attack}=20 and ϵ=0.05, the ϵ-MSI value for CIDS is 22.5μs as shown in Fig. 2.8(a) and 11.5μs for the NTP-based IDS as shown in Fig. 2.8(c). This result indicates that it is much easier for the cloaking attack to bypass CIDS than the NTP-based IDS. We observed that increasing n_{attack} has little impact on ϵ-MSI in CIDS, which is 20.5μs for $n_{attack} = 40$ or 60 while it affects significantly ϵ-MSI of the NTP-based IDS, which varies from 11.5μs to 2.5μs as n_{attack} is increased from 20 to 60. This result shows that the performance of the NTP-based IDS improves as the attack duration increases.

ϵ-MSI for CIDS increases significantly in the UW EcoCAR, as shown in Fig. 2.8(b) due to the heavier CAN traffic than the CAN bus testbed. As an example, a cloaking attack with ΔT between -1029μs and 1021μs can bypass CIDS with 100% of the successful attack probability for n_{attack} is 20, 40, and 60. For the NTP-based IDS with ϵ=0.01, ϵ-MSI is 10.5μs for n_{attack}=20. This experimental result implies that the NTP-based IDS is more effective in detecting a masquerade



(a) CAN bus testbed, CIDS

(b) EcoCAR, CIDS

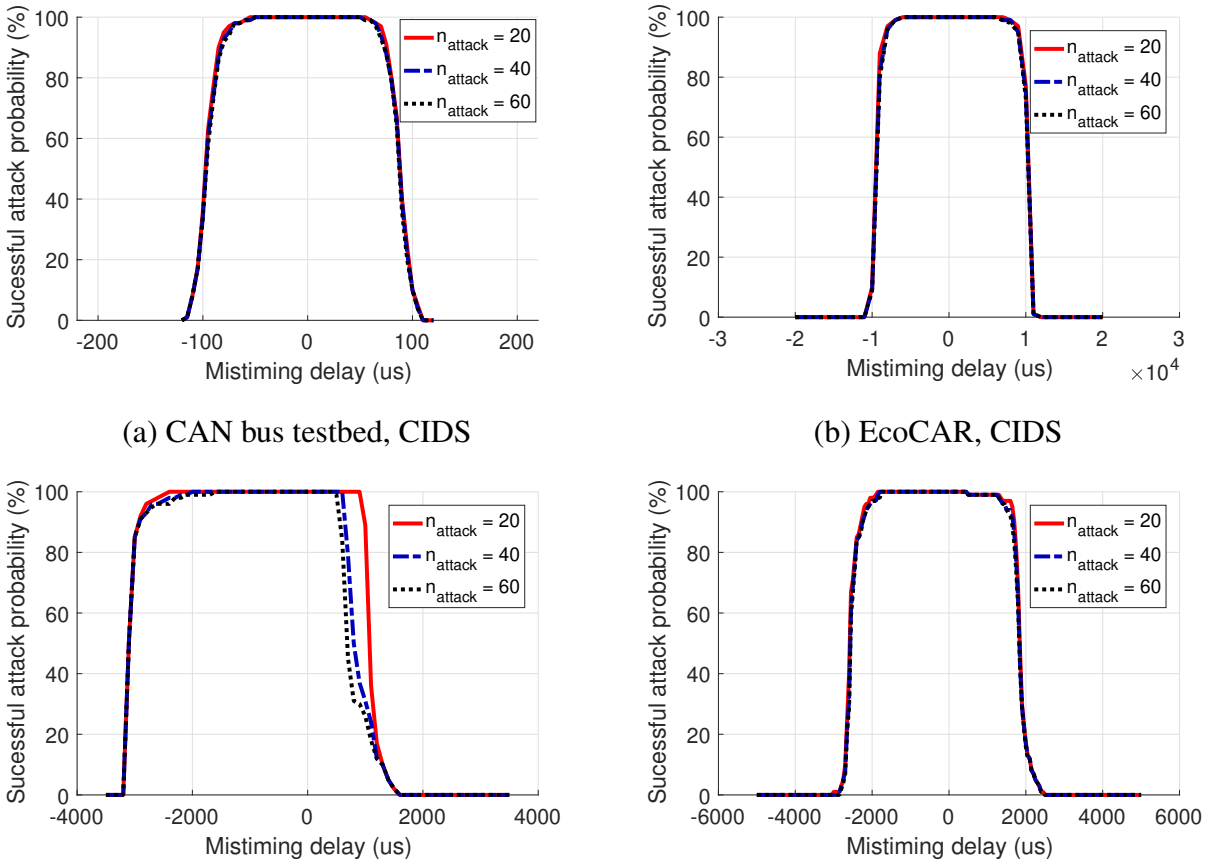
(c) CAN bus testbed, NTP-based IDS

(d) EcoCAR, NTP-based IDS

Figure 2.8: Successful attack probability of the cloaking attack on the CIDS and NTP-based IDS on both the CAN bus testbed and the UW EcoCar. When ΔT is perfectly matched, the cloaking attack can bypass both the CIDS and NTP-based IDS as indicated by the red dash line.

attack than CIDS on the UW EcoCAR as well. The cloaking attack can bypass CIDS and the NTP-based IDS when ΔT is between $(\Delta T)_{min}(\epsilon)$ and $(\Delta T)_{max}(\epsilon)$.

In the masquerade and cloaking attacks, the fully compromised ECU needs to start transmitting the spoofed message at the time constant at which the targeted message should have been transmitted if it had not been suspended. It naturally raises the question of whether mistiming affects the performance of the cloaking attack. In this experiment, we introduce a mistiming delay (either



(a) CAN bus testbed, CIDS

(b) EcoCAR, CIDS

(c) CAN bus testbed, NTP-based IDS

(d) EcoCAR, NTP-based IDS

Figure 2.9: Impact of the mistimed cloaking attack on the CIDS and NTP-based IDS.

positive or negative) between the last message of the normal data and the first message of attack data in addition to the message period.

We use exactly the same settings of normal data, attack data, batch size, γ, Γ, and κ as before. Fig. 2.9 shows the impact of the mistiming of the cloaking attack on the CIDS and NTP-based IDS. A larger mistiming causes attack performance to decrease. On the CAN bus testbed, any amount of mistiming between -55μs and 55μs does not affect the attack performance on CIDS, whereas the allowed mistiming is much larger for the NTP-based IDS due to the difference in clock skew estimation. Since the clock skew of the Arduino-based ECU slowly decreases due to

the temperature change in hardware as it warms up, the estimator tends to overestimate the clock skew and thus is more sensitive to larger positive mistiming.

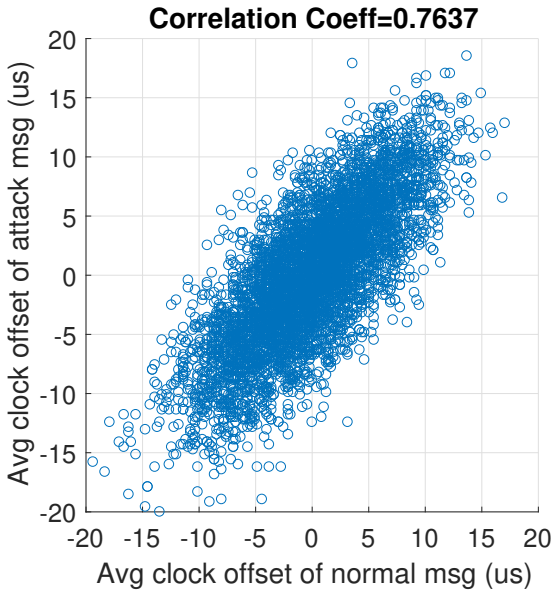
On the UW EcoCAR, the allowed mistiming is increased significantly, which is between -6ms to 7ms for CIDS and between -1.8ms and 0.5ms for the NTP-based IDS due to much heavier CAN traffic in a real vehicle. The above observations imply that the timing is hardly a strict requirement for the adversary to launch a clocking attack in a real vehicle.

2.5.3 Cloaking attack on correlation detector

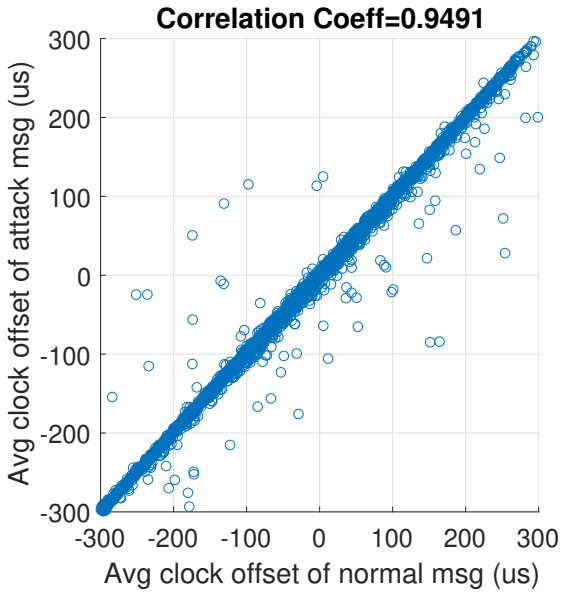
We first launch the cloaking attack on the correlation detector using the CAN bus testbed. ECU C transmits a spoofed message 0x11 after observing a sibling message of the targeted message, whose frequency is 5Hz, with a constant delay of 100ms. On the UW EcoCAR, messages 0xC1 and 0xC5, which are transmitted from the same ECU with a frequency of 10Hz, are identified to be highly correlated. We choose 0xC5 as the target and 0xC1 as a sibling message. We use the Raspberry-Pi-based testbed ECU to inject messages with a non-conflicting ID 0xC0, instead of 0xC5 in order to avoid any undesirable impact on the UW EcoCAR. Fourteen hours and 1.2 hours of attack data were collected from the CAN bus testbed and the UW EcoCAR, respectively. As a baseline, we collected 4.7 hours of normal data with one ECU transmitting two messages consecutively on the CAN bus testbed. For the UW EcoCAR, the data contains the normal data since the targeted message is not suspended.

Fig. 2.10 shows a scatter plot of average offsets of the sibling message and the attack message under the cloaking attack. For the CAN bus testbed, the correlation coefficient between the average offsets, ρ, is 0.76 and 0.90 for the CIDS and NTP-based IDS, respectively, because jitter and offset deviation are small whereas the quantization error is significant due to the time resolution of the Arduino UNO Rev3 board. On the UW EcoCAR, however, the cloaking attack can achieve a correlation of up to 0.95 and 0.92 for the CIDS and NTP-based IDS, respectively.

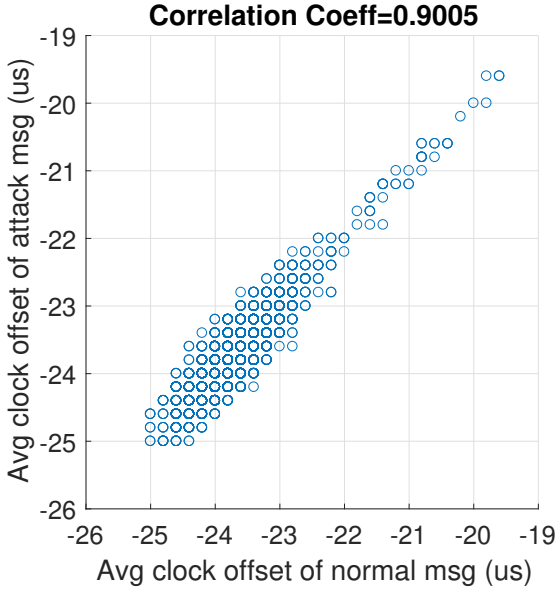
We examine 17 messages from five ECUs with periods of 10ms, 12ms, or 100ms based on the ground truth. All pairs of messages are classified into the following three categories: 1) from the same ECU and almost received consecutively, 2) from the same ECU but not received consecu-



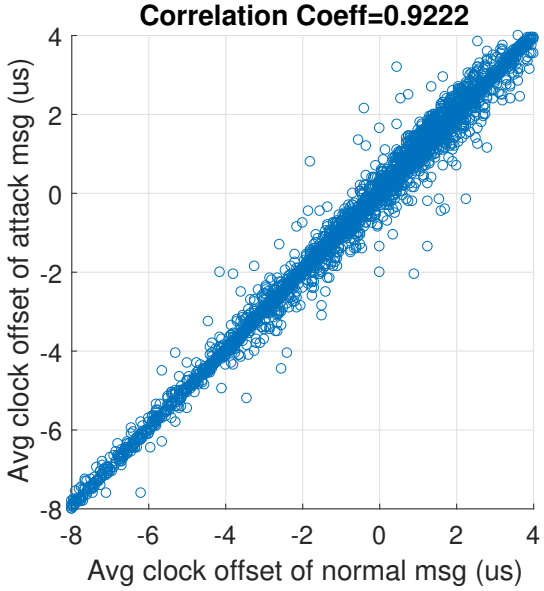
(a) CAN bus testbed, CIDS



(b) EcoCAR, CIDS



(c) CAN bus testbed, NTP-based IDS



(d) EcoCAR, NTP-based IDS

Figure 2.10: Scatter plot of the average offsets of sibling messages and attack messages under the cloaking attack.

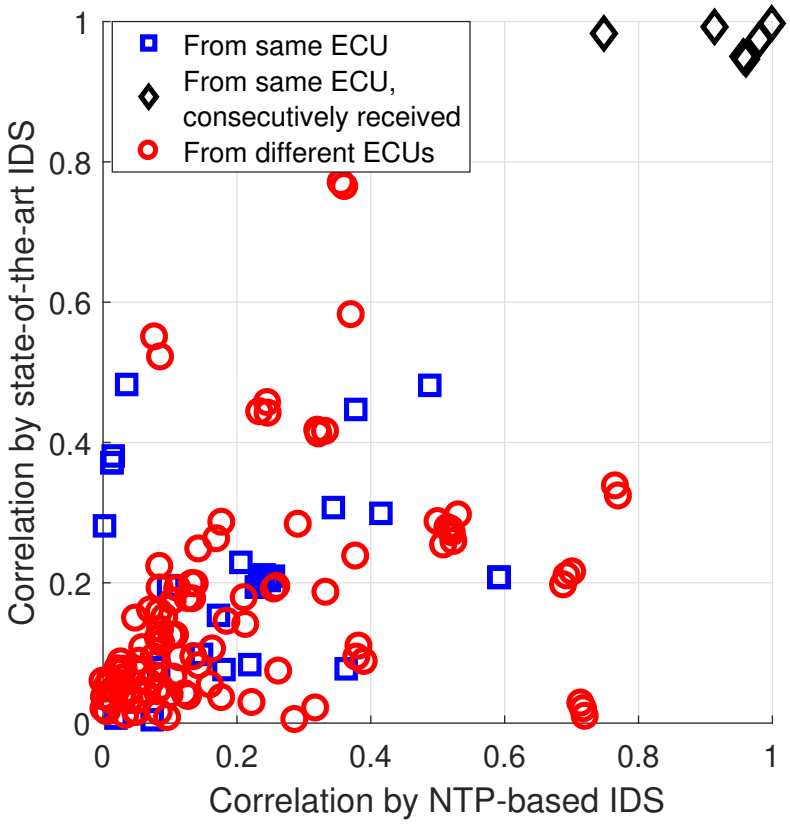


Figure 2.11: Correlation between pairwise messages on the UW EcoCar testbed. The messages that are transmitted consecutively from the same ECU show a high correlation in the average offsets while others are less correlated.

tively, or 3) from different ECUs. The correlation coefficients are computed using 200 batches. As illustrated in Fig. 2.11, for two messages from different ECUs, their correlation is generally low for the CIDS and NTP-based IDS. Not all pairs of messages from the same ECU have a high correlation: 81% of them correlate less than 0.6, and there are only five pairs with a correlation higher than 0.9 for both the CIDS and NTP-based IDS. The messages that are transmitted consecutively transmitted from the same ECU show a high correlation in the average offset. Otherwise, the average offsets of the two messages are less correlated.

2.6 Conclusion

In this chapter, we investigated masquerade attacks on in-vehicle networks and CIDS that exploits ECU's clock skew to fingerprint each ECU in the CAN bus network. We developed the NTP-based IDS that computes the clock skew according to the NTP specification. In addition, we developed the cloaking attack, in which an adversary manipulates the inter-transmission times of spoofed messages in order to match the clock skew of the targeted ECU. We evaluated the cloaking attack on the CAN bus testbed and the UW EcoCAR and demonstrated that the cloaking attack could bypass both the CIDS and NTP-based IDS. In order to quantify the effectiveness of the CIDS and NTP-based IDS, we introduced a new performance metric, the Maximum Slackness Index, which is the range of the added delay that the adversary can introduce without being detected. This work makes the case that an impact of coupling between cyber and physical components in the automobile security needs to be understood, especially when attempting to leverage physical invariants arising from physical components to provide security assurances.

Chapter 3

VOLTAGE-BASED ATTACK AND INTRUSION RESPONSE SYSTEM

3.1 Introduction

Voltage-based Intrusion Detection Systems (VIDSs) measure the voltages of CANH and CANL to extract the voltage characteristics of each ECU. The voltage characteristics are determined by the CAN transceiver's hardware and cannot be modified by the microcontroller's software, which makes mimicking the voltage characteristics through cyber attacks difficult for an adversary [19]. An oscilloscope is used to measure the voltage in [41] and [20], which may not be possible in a car due to limitations in power and space. Alternatively, a VIDS may be implemented in a

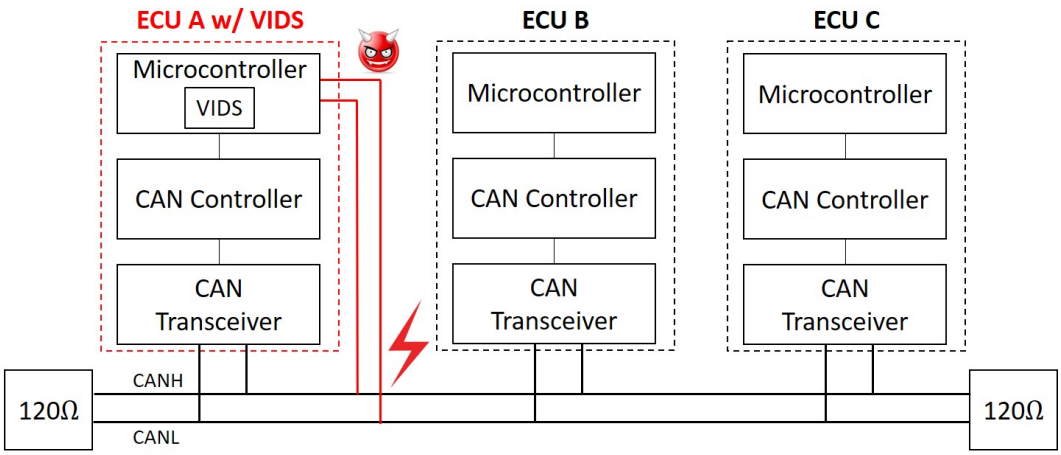


Figure 3.1: Architecture of VIDS. A VIDS is implemented on the microcontroller of ECU A. In order to measure voltages of the CAN bus lines, the microcontroller of ECU A is directly connected to the CAN bus lines via two extra wires (two red lines). If ECU A is compromised, the adversary may launch voltage-based attacks using the directly connected wires.

microcontroller, which requires 12V that can be supplied from a car battery [19].

In order to measure voltages of the CAN bus lines, the microcontroller is directly connected to CANH and CANL using extra wires, respectively, as illustrated in Fig. 3.1. These two extra wires introduce new attack surfaces on the CAN bus and ECUs. If the microcontroller is compromised, the adversary may exploit the wires connected to the CAN bus lines to launch attacks. The recent works on the VIDSs, however, did not discuss any potential cyber attacks in which these wires are maliciously used, nor did the works propose a protection mechanism to mitigate the attacks [19, 20, 41].

We develop four voltage-based attacks using the extra wires: 1) an overcurrent attack, 2) a denial-of-service (DoS) attack, 3) a forced retransmission attack, and 4) a pulse attack. In the overcurrent attack, the adversary makes the current that flows into the microcontroller with VIDS exceed the hardware limit of the microcontroller (i.e., the current absolute maximum rating), which may damage the microcontroller. In the DoS attack, the adversary keeps the CAN bus in an idle state by holding the voltages of the CAN bus lines at one level, by which messages cannot be transmitted. In the forced retransmission attack, the adversary violates the bit timing requirement of the CAN protocol to make an error, which makes a message be retransmitted. In the pulse attack, the adversary arbitrarily changes the voltages of the CAN bus lines by applying a pulse signal, which blocks message transmission. In this chapter, we make the following contributions:

- We develop four voltage-based attacks that are an overcurrent attack, a DoS attack, a forced retransmission attack, and a pulse attack by manipulating the voltage levels of the CAN bus lines using the extra wires.

- We develop two hardware-based Intrusion Response Systems (IRSs), which are a fuse-based IRS and a heat-based IRS, that detect and mitigate the voltage-based attacks by isolating the compromised ECU from the CAN bus.

- We demonstrate the developed voltage-based attacks and evaluate the effectiveness of the developed IRSs using a CAN bus testbed. The evaluation using the testbed shows that the

voltage-based attacks can be implemented in practice. We also demonstrate that the proposed IRSs mitigate the voltage-based attacks.

3.2 Adversary Model

In this section, we describe an adversary model for the voltage-based attacks on the CAN bus and VIDS. A VIDS is implemented as a software code on an ECU's microcontroller as shown in Fig. 3.1. We first consider how an ECU with VIDS can be compromised. VIDS can be compromised in the following ways. If an adversary accesses the CAN bus physically via an OBD-II port that is mandated for all cars in the US[13] and EU[55], the adversary can upload its malicious code to the ECU with VIDS using a pass-thru device such as Hyundai Global Diagnostic System [1], Ford Vehicle Communication Module [22], Volkswagen VAG-COM Diagnostic System [48], or Toyota Technical Information System [56]. It is also possible for an adversary to remotely compromise an ECU without physical access to the vehicle as demonstrated in [15, 38, 47]. The adversary can remotely access the operating system of an ECU that has a telematics unit in order to figure out the particular code that handles wireless connectivity such as Bluetooth. By exploiting that particular code, the adversary executes its malicious code on that ECU and compromises that ECU. Then, any ECU on the CAN bus, including the ECU with VIDS, can be compromised using the compromised ECU as a pass-thru device [15].

Although an adversary can compromise the microcontroller of an ECU with VIDS, the firmware of the CAN controller cannot be modified without a special equipment [9]. As a result, the compromised ECU cannot manipulate the output voltages of the CAN transceiver in order to violate the CAN protocol and launch the voltage-based attacks. The adversary, however, can manipulate the analog pin modes of the microcontroller once the ECU with VIDS is compromised. Hence, the adversary can apply electric signals to the CAN bus by controlling the analog pin modes of the microcontroller.

3.3 Voltage-based Attacks

In this section, we introduce four proposed voltage-based attacks that are an overcurrent attack, a DoS attack, a forced retransmission attack, and a pulse attack. Let us denote two analog pins of the microcontroller with VIDS that are directly connected to CANH and CANL as P_H and P_L, respectively. We explain combinations of P_H and P_L for the normal operation of a VIDS and voltage-based attacks. P_H and P_L can operate in one of the following four modes: 1) input for measuring the voltage, 2) high voltage output, 3) low voltage output, or 4) pulse output. We consider that the binary output voltage levels and a PWM signal can be generated by switching low and high voltage output modes in the pulse output mode. Depending on microcontrollers, an analog pin may generate multiple levels of the output voltage.

In the normal operation of a VIDS, both P_H and P_L are in the input mode to measure the voltage levels of CANH and CANL, respectively. If the adversary compromises an ECU with VIDS, the adversary may change the pin modes of P_H and P_L to either high voltage output, low voltage output, or pulse output by uploading its malicious code. Some combinations of the analog pin modes may damage the microcontroller with VIDS, block message transmission, or cause message retransmission as listed in Table 3.1.

3.3.1 Overcurrent Attack

A microcontroller can absorb the current below a threshold through an analog pin due to its hardware limitation, which is called the current absolute maximum rating, I_{max}. If the current larger than I_{max} flows into the analog pin, the microcontroller can be damaged because of an electric shock. The idea of the overcurrent attack is that the adversary makes the current larger than I_{max} flow into P_L of the compromised ECU with VIDS. The values of I_{max} differ depending on microcontrollers. For instance, values of I_{max} are 40mA for Microchip ATmega328P [8] and Renesas V850 [46] and 20mA for NXP MPC563 [43]. The datasheet of Renesas V850 does not provide the current absolute maximum rating. The voltage absolute maximum rating of Renesas V850 and operating conditions such as temperature are similar to that of Microchip ATmega328P. Hence, it

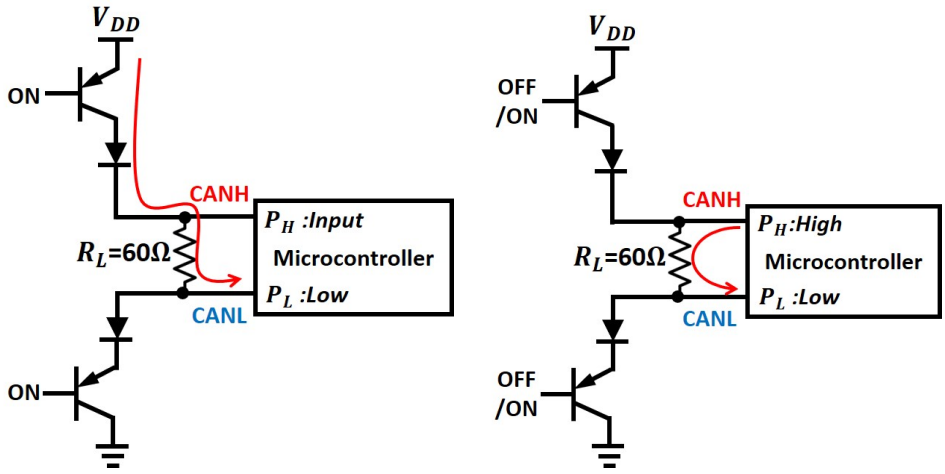
is reasonable to assume that the current absolute maximum rating of Renesas V850 is similar to Microchip ATmega328P, which is 40mA.

When a microcontroller measures the voltage, a small current flows through the analog pin of the microcontroller. The current drawn by the microcontroller is negligible, which is in the order of nA, due to the high impedance at the microcontroller. As a result, the current flowing through an analog pin during the voltage measurement is much smaller than I_{max} and does not damage the microcontroller.

The adversary may let the current flow into an analog pin by manipulating the pin mode as illustrated in Fig 3.2. Let us denote $V_{H,b}$ and $V_{L,b}$ as the voltages of CANH and CANL when bit b is transmitted, respectively, where b is either 0 or 1. For instance, the voltage of CANL when a recessive bit is transmitted is denoted as $V_{L,1}$. The condition of $V_{H,0} - V_{L,0}$ for the overcurrent

Table 3.1: Combinations of analog pin modes of P_H and P_L for measuring the voltage of the CAN bus lines in the normal operation of the VIDS and launching the voltage-based attacks.

P_H mode	P_L mode	Type of attack
Input	Input	Not an attack, setting for measuring voltage
Input	High	DoS attack
Input	Low	Passive overcurrent attack
High	Input	Forced retransmission attack
High	Low	Active overcurrent attack
Low	Input	DoS attack or passive overcurrent attack
Low	High	DoS attack or active overcurrent attack
Low	Low	DoS attack or passive overcurrent attack
Pulse	Input	Pulse attack
Input	Pulse	Pulse attack



(a) Passive overcurrent attack (b) Active overcurrent attack

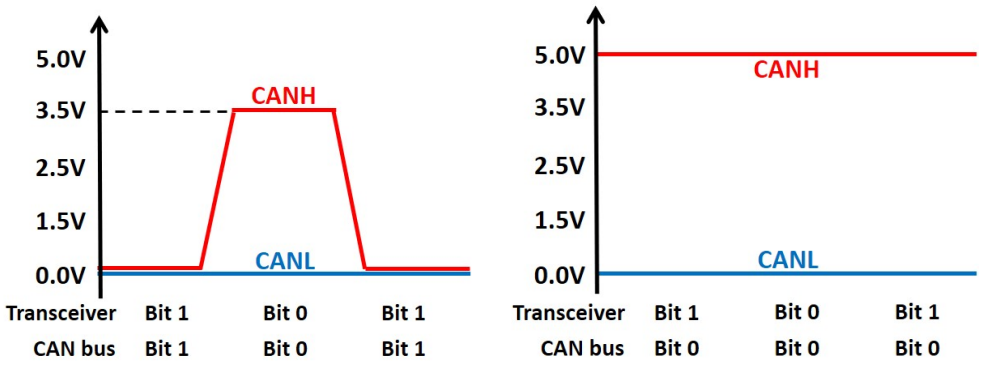
Figure 3.2: Circuit diagrams under the overcurrent attacks. The red curve indicates the current in each overcurrent attack. (a) In the passive overcurrent attack, the current flows from V_{DD} to P_L when a dominant bit is transmitted. (b) In the active overcurrent attack, the current flows from P_H to P_L regardless of the bits.

attack can be derived in terms of R_L and I_{max} as follows

$$\frac{(V_{H,0} - V_{L,0})}{R_L} > I_{max}, \tag{3.1}$$

where R_L is 60Ω and I_{max} depends on a hardware limit of the microcontroller. Depending on the voltage of $V_{H,0}$ with $V_{L,0}$=0.0V, we propose passive and active overcurrent attacks.

In the passive overcurrent attack, the adversary changes P_L to the low voltage output mode as illustrated in Fig. 3.2(a). Since $V_{H,0}$ is 3.5V as shown in Fig. 3.3(a), the current absorbed through P_L can be computed as $\frac{3.5V-0.0V}{60\Omega}$=58.3mA, which satisfies Eq. (3.1). In the passive overcurrent attack, the current is supplied from a car battery that can provide the current in the order of Ampere as illustrated in Fig. 3.2(a). In the active overcurrent attack, the adversary changes P_H and P_L to the high and the low voltage output modes, respectively, as illustrated in Fig. 3.2(b). $V_{H,0}$ and $V_{L,0}$ become 5.0V and 0.0V, respectively, as shown in Fig. 3.3(b). Then, the current flowing through P_L is computed as $\frac{5.0V-0.0V}{60\Omega}$=83.3mA that is also larger than I_{max}. In the active overcurrent attack, the



(a) Passive overcurrent attack (b) Active overcurrent attack

Figure 3.3: Voltages of CANH and CANL under the overcurrent attacks. (a) In the passive overcurrent attack, the maximum voltage difference between CANH and CANL is 3.5V when a dominant bit is transmitted. (b) In the active overcurrent attack, the voltage difference is always 5.0V.

current is supplied from the microcontroller as illustrated in Fig. 3.2(b).

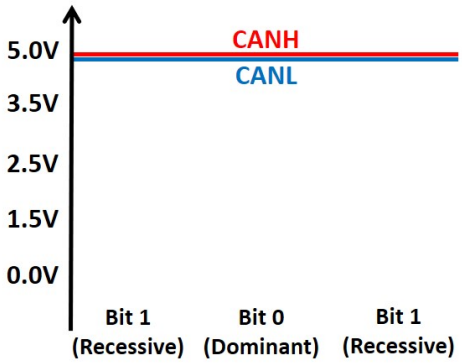


Figure 3.4: Voltages of the CAN bus under the DoS attack when $V_{attack,L}$ is 5.0V. Both CANH and CANL become 5.0V, which represents a recessive bit.

3.3.2 Denial-of-Service (DoS) Attack

In order to transmit a bit on the CAN bus, the differential voltage between CANH and CANL, V_{Diff}, must satisfy the CAN protocol. The idea of the DoS attack is that the adversary increases the voltage level of CANL such that V_{Diff} is always smaller than the decision threshold for determining a dominant bit. As a result, a recessive bit can only be transmitted, which means that the CAN bus is in the idle state. Also, messages cannot be transmitted. The adversary sets P_L from the input mode to the high voltage output mode. Let $V_{attack,L}$ denote the voltage applied to CANL by the adversary using P_L. Then, V_{Diff} can be computed as follows

$$V_{Diff} = V_{H,b} - V_{attack,L}. \tag{3.2}$$

When a recessive bit is transmitted, both CANH and CANL are set to $V_{attack,L}$ under the DoS attack because the current does not flow through the termination resistors. Using Eq. (3.2), V_{Diff} becomes 0.0V, which represents a recessive bit. Although the voltages of CANH and CANL are manipulated as illustrated in Fig. 3.4, the recessive bit can be transmitted under the attack. A CAN transceiver determines a dominant bit if $0.9\text{V}<V_{Diff}<5.0\text{V}$ with the differential input hysteresis between 100-200mV as shown in Fig. 3.5. When transmitting a dominant bit, the adversary may thwart the transmission of the dominant bit by setting $V_{attack,L}$ to make V_{Diff} smaller than 0.9V. If $V_{attack,L}$ is larger than 3.5V, $V_{H,0}$ also becomes $V_{attack,L}$ because current cannot flow through the termination resistors due to the diode in the CAN transceivers. Hence, V_{Diff} again becomes 0.0V, which indicates that the DoS attack is successfully launched. Also, the adversary may launch the DoS attack by setting P_H in the low voltage output mode since V_{Diff} is always equal to or less than 0.0V.

3.3.3 Forced Retransmission Attack

The duration of one bit must satisfy the timing requirement of the CAN protocol according to the CAN bus speed. The idea of the forced retransmission attack is that the adversary makes the transition time from a dominant bit to a recessive bit longer than the nominal value, typically

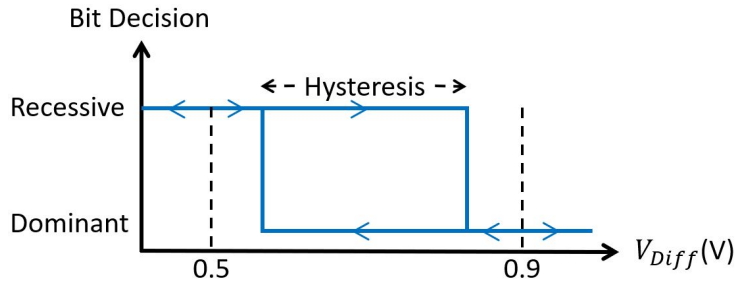


Figure 3.5: Bit decision criteria of Microchip MCP2551 CAN transceiver. A CAN transceiver can determine a dominant bit if V_{Diff} is larger than 0.9V and a recessive bit if V_{Diff} is less than 0.5V.

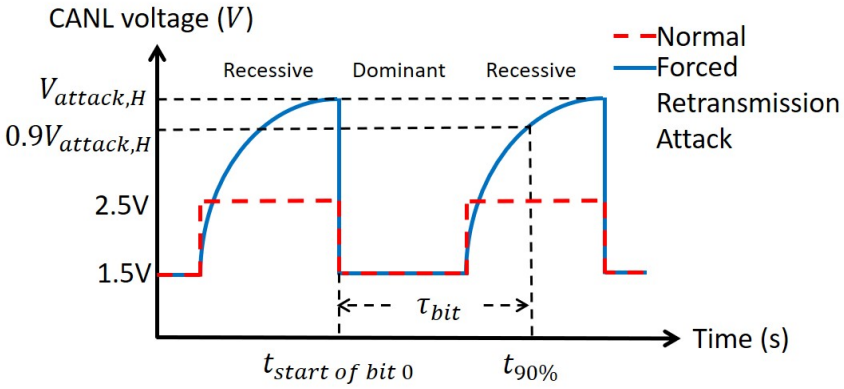


Figure 3.6: Voltage of CANL under the forced retransmission attack. Compared with the voltage level in the normal operation, the voltage of CANL is pulled up to $V_{attack,H}$, which increases the transition time.

70-130ns [35]. As a result, an error occurs while receiving a message, especially at the ACK delimiter position of a data frame as shown in Fig 1.1. The adversary increases the transition time, thus making a dominant bit at the ACK delimiter position that has to be a recessive bit. The adversary changes P_H from the input mode to the high voltage output mode to launch the forced retransmission attack.

In order to analyze the transition time from a dominant bit to a recessive bit quantitatively, we

define the *bit length time* τ_{bit} as follows

$$\tau_{bit} \triangleq t_{90\%} - t_{start\ of\ bit\ 0}, \tag{3.3}$$

where $t_{start\ of\ bit\ 0}$ and $t_{90\%}$ denote the time at which a dominant bit starts and the time at which the voltage of CANL reaches 90% of $V_{attack,H}$, respectively, as illustrated in Fig. 3.6. τ_{bit} also indicates the sum of the duration of the dominant bit and the transition time. For instance, if the CAN bus speed is set to 500kbps, τ_{bit} without the attack is nominal 2μs.

Let $V_{attack,H}$ denote the voltage applied to CANH by the adversary via P_H, and we consider $V_{attack,H}$>2.5V. Fig. 3.7 illustrates the voltages of CANH and CANL under the forced retransmission attack when $V_{attack,H}$ is 5.0V. When transmitting a recessive bit, both CANH and CANL become $V_{attack,H}$ since current does not flow through the termination resistors. A recessive bit can be transmitted under the attack. A dominant bit can also be transmitted because V_{Diff} is greater than 0.9V, where V_{Diff} increases as increasing $V_{attack,H}$. If $V_{attack,H}$ becomes greater than 3.5V, the adversary forcefully sets CANL higher than the normal operating range of the CAN transceiver. As a result, the voltage of CANL switches from 1.5V to $V_{attack,H}$ when a recessive bit is transmitted right after a dominant bit, which increases the transition time as illustrated in Fig. 3.6. Using a local clock of an ECU, the ECU decodes a bit by sampling the voltage levels of CANH and CANL with the fixed period that is determined by the CAN bus speed. If the duration of a bit is longer than the nominal value, the ECU cannot decode the bit correctly. Hence, an error occurs while receiving a message, and that message is retransmitted.

3.3.4 Pulse Attack

The idea of the pulse attack is blocking message transmission using a PWM signal. Consider that the PWM signal is applied to CANL using P_L. There are four possible cases for the voltage behavior of the CAN bus because the PWM signal could be in either the high voltage output mode or the low voltage output mode when a dominant or recessive bit is transmitted, respectively. If the PWM signal is in the high voltage output mode while a dominant bit is transmitted, both CANH and CANL become the same voltage, which represents a recessive bit. As a result, the dominant

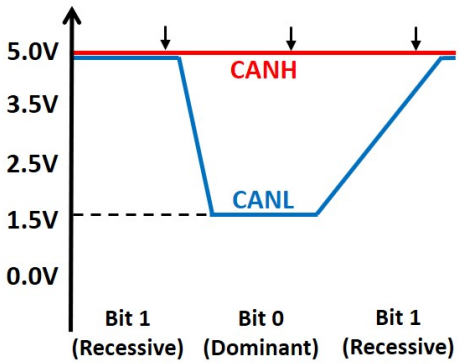


Figure 3.7: Voltages of the CAN bus under the forced retransmission attack when $V_{attack,H}$ is 5.0V. When a recessive bit is transmitted right after a dominant bit, the transition time of CANL's voltage level increases. Hence, an ECU may receive a dominant bit because V_{Diff} is greater than 0.9V when sampling the voltages of the CAN bus lines at the instances indicated with the black arrows facing downward.

bit cannot be transmitted through the CAN bus, and the message transmission is blocked. A bit can be transmitted correctly in the other three cases because V_{Diff} satisfies the CAN protocol.

A PWM signal and the start of the message are not perfectly synchronized in most of the time because the PWM signal is applied in arbitrary time without considering message transmission. As a result, the case in which a dominant bit is tried to be transmitted when the PWM signal is in the high voltage output always occurs. If the PWM signal is applied to CANH, a dominant bit cannot be transmitted if the PWM signal is in the low voltage output. The pulse attack occurs when a PWM signal is applied to either P_H or P_L where the other analog pin is in the input mode.

A CAN controller determines a received bit using the voltage signal from the CAN transceiver. In order to determine the bit, this voltage signal has to stay constant longer than a predetermined threshold, typically 300-350ns [34]. Otherwise, the CAN controller cannot decode the received bit. The impact of the PWM signal during message transmission can be negligible if the duration of blocking the transmission of a dominant bit is shorter than this predetermined threshold. For example, when the duty cycle is fixed at 50%, the period of the PWM signal has to be longer than

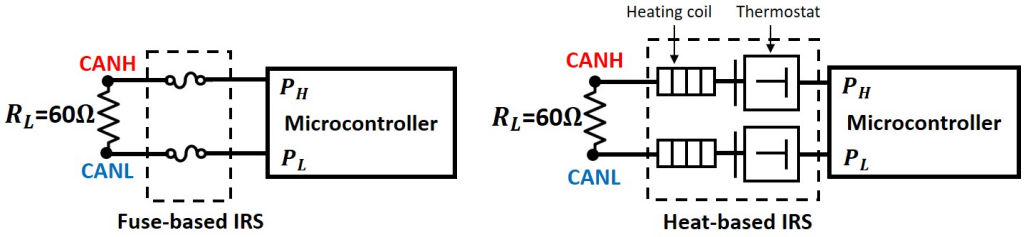
700ns to thwart the transmission of a dominant bit, which launches the pulse attack successfully. If the period of the PWM signal is twice longer than the message transmission time for the fixed duty cycle 50%, the PWM signal changes the voltage levels of the CAN bus lines for the entire message transmission time, which is identical to the DoS attack and forced retransmission attack.

3.4 Hardware-based IRSs

We develop two hardware-based IRSs, which are fuse-based IRS and heat-based IRS, and explain how these IRSs can detect and mitigate the voltage-based attacks. As shown in Fig. 3.8, the fuse-based IRS consists of fuses or circuit breakers that are attached between the microcontroller's analog pins and the CAN bus lines (Fig. 3.8(a)), while the heat-based IRS consists of heating coil and thermostats (Fig. 3.8(b)). The current flowing through the analog pins can be used as an indicator of the voltage-based attacks because the current flows through these analog pins under the voltage-based attacks. The hardware-based IRSs physically isolate VIDS from the CAN bus once anyone of the voltage-based attacks is detected. The hardware components such as fuses, circuit breakers, heating coil, or thermostats operate independently of the microcontroller. As a result, any cyber attacks on an automobile cannot disable these hardware components, which results in the developed IRSs cannot be disabled by the cyber attacks. We explain how each developed IRS can detect and mitigate the voltage-based attacks.

3.4.1 Overcurrent Attack

The overcurrent attack can be mitigated by limiting the current that flows into the microcontroller to be less than the current absolute maximum rating, I_{max}. A fuse may limit the current because it disconnects two parts of a circuit when the current that is larger than its current rating flows longer than its opening time. The typical opening time of very fast-acting fuses is in the order of μs to ms [31, 32]. Since it takes a longer time to damage a microcontroller by the overcurrent than to open a fuse [8], the fuse disconnects the wire before the microcontroller is burned. The overcurrent may damage the microcontrollers within a short amount of time in the order of ns and



(a) Fuse-based IRS (b) Heat-based IRS

Figure 3.8: Structures of the proposed hardware-based IRSs. The fuse-based IRS consists of fuses that are attached between the microcontroller's analog pins and the CAN bus lines. The heat-based IRS consists of heating coil and thermostats.

μs. Micro-electro-mechanical system (MEMS)-based fuses or thin film-based fuses can be used for the fuse-based IRS because their opening times are in the order of μs for the MEMS-based fuse [16] and in the order of ns for the thin-film based fuse [21]. A fuse does not thwart the voltage measurement at the microcontroller when the fuse is attached to an analog pin as illustrated in Fig. 3.8(a) because the fuse is a wire that does not induce any voltage drop ideally. A fuse with the current rating smaller than I_{max} can be used for the fuse-based IRS because 58.3mA and 83.3mA flow through P_L under the passive and active overcurrent attacks, respectively, as computed in the previous section. A fuse has to be replaced every time after it blows out. Though replacing a fuse is simple, a circuit breaker is reusable after it isolates a VIDS during the overcurrent attack. A system operator, however, still has to reset the circuit breaker manually. Since a fuse is lighter and cheaper, the fuse is a better component than a circuit breaker for an automobile application.

A thermostat also disconnects two parts of a circuit if it is heated above its temperature limiting threshold and connects them again if it is cooled down below the temperature limiting threshold. A heating coil is made of a metal composite such as a nichrome that generates heat when current flows through it. Under the passive overcurrent attack, heat is generated from the heating coil connected to CANL due to the current flowing through P_L. Both heating coils generate heat under the active overcurrent attack because the current flows from P_H to P_L. The thermostat is heated and isolates

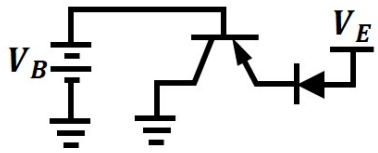


Figure 3.9: Test circuits for measuring current under the DoS attack. This test circuit that emulates the voltage levels of the transistor inside the CAN transceiver under the DoS attack.

a VIDS from the CAN bus. Consider that the malicious code is removed from the microcontroller. The current does not flow through the heating coil, thus cooling down the thermostat below the temperature limiting threshold. Then, the wires from the microcontroller to the CAN bus lines are connected again without replacing any hardware components. The heating coil and thermostat are a wire if the thermostat is closed. As a result, the heating coil and thermostat do not thwart the voltage measurement at the microcontroller when they are attached to an analog pin as illustrated in Fig. 3.8(b).

One might argue that a resettable fuse can be used instead of a thermostat because a system operator also does not need to replace the resettable fuse. Different from fuses or thermostats that completely disconnect a wire, the resettable fuse, however, allows a leakage current in the order of a hundred mA, which is large enough to damage the microcontroller when the resettable fuse is open [2]. Consequently, the resettable fuse should not be used in the proposed hardware-based IRSs.

3.4.2 DoS Attack, Forced Retransmission Attack, and Pulse Attack

The output voltage from an analog pin has to be limited in order to mitigate the DoS attack, forced retransmission attack, and pulse attack since an abnormal voltage is applied to either CANH or CANL under these attacks. Due to this voltage, current flows through the analog pins under these attacks. Under the DoS attack, current flows from P_L to the ground of the CAN transceiver when a dominant bit is transmitted. Also, current flows from P_H to the ground of the CAN transceiver under the forced retransmission attack. Under the pulse attack, current flows through either P_L or

P_H if the PWM signal is applied to CANL or CANH, respectively. As a result, fuses or circuit breakers can be used to mitigate these attacks.

In order to determine the current rating of the fuse, the current flowing through P_L and P_H under the DoS attack, forced retransmission attack, and pulse attack is analyzed, respectively. Fig. 3.9 illustrates a test circuit that emulates the voltage levels of the transistor inside the CAN transceiver under the DoS attack using a Motorola Solutions 2N2905A PNP BJT. When the BJT is in the on-state, we measure that 281mA flows into the ground using the test circuit. For the forced retransmission attack, we compute the current that flows through P_H as $\frac{(5.0\text{V}-1.5\text{V})}{60\Omega}$ =58.3mA. Since the voltage levels under the pulse attack via CANL is the same as that under the DoS attack, the current flowing through P_L becomes 281mA. For the pulse attack via CANH, the current flowing through P_H becomes 58.3mA as the forced retransmission attack. As a result, fuses or circuit breakers with a current rating of less than 58.3mA mitigate these attacks.

We analyze that current flows through the analog pins under the DoS attack, forced retransmission attack, and pulse attack. By exploiting this fact, the heating coil may heat the thermostat during these attacks, and the thermostat disconnects the extra wires. As a result, the heat-based IRS can also mitigate these attacks by isolating a VIDS from the CAN bus.

3.5 Experimental Result

In this section, we demonstrate the overcurrent attack, the DoS attack, the forced retransmission attack, and the pulse attack on the CAN bus testbed. Moreover, we demonstrate that the proposed IRSs are effective in defending the CAN bus against the voltage-based attacks.

3.5.1 Testbed and Equipment

We use the same CAN bus testbed that is explained in Chapter 2. We implement a CAN bus testbed that consists of three ECUs as shown in Fig. 3.10. Each testbed ECU is composed of an Arduino UNO Rev3 board and a Sparkfun CAN bus shield. The CAN bus shield uses a Microchip MCP2515 CAN controller and a Microchip MCP2551 CAN transceiver. We set the CAN bus

speed to 500kbps, which is widely used in many modern cars [18]. A key difference is that two analog pins (pin numbers A0 and A5) of the Arduino board in ECU A are connected to CANH and CANL as shown in Fig. 3.10, respectively, because the VIDS is assumed to be implemented at ECU A. ECU A is compromised and launches the proposed voltage-based attacks by changing the modes of these analog pins. ECU C transmits messages every 1s, and ECU B logs all the received messages.

We use a Tektronix TDS2004B oscilloscope to measure voltages of the CAN bus lines and bit duration in the normal message transmission and under the voltage-based attacks. A Keysight 34461A digital multimeter is used to measure voltage and current. The minimum period of a PWM signal that an Arduino board can generate is 1.02ms, and the Arduino board cannot generate various voltage levels from an analog pin [8]. We use a Tektronix AFG3021 function generator to apply PWM signals with various periods in order to determine the minimum period for a successful pulse attack. A Keysight U8031A power supply is used to apply various constant voltages to CANH and CANL to determine the minimum voltage levels for successful DoS attack and forced retransmission attack, respectively.

We use Littelfuse 0326 fuses with current rating 10mA to implement the fuse-based IRS. For the heat-based IRS, we use Sparkfun heating pad and Sensata Airpax 67L040 thermostats with temperature limiting threshold 40^{o}C. In the test circuit, as illustrated in Fig. 3.11, V_{DD} is 5.0V, and the analog pins are in the input mode. The microcontroller can correctly measure the voltages of CANH and CANL (i.e., 5.0V and 0.0V at A0 and A5, respectively), which indicates that the fuse does not thwart the voltage measurement at the microcontroller. By repeating the same experiments after replacing the fuses with circuit breakers and thermostats with heating coils, respectively, we verify that the proposed hardware-based IRSs do not thwart the voltage measurement. Consequently, the hardware-based IRSs can be implemented in a VIDS without bothering the operation of the VIDS.

Fig. 3.12 shows the voltage levels of CANH and CANL in the normal message transmission when the message ID is 0x01 and data is 0x01. Due to hardware characteristics of the CAN transceiver, the actual voltages of CANH and CANL are 2.4V when a recessive bit is transmitted.

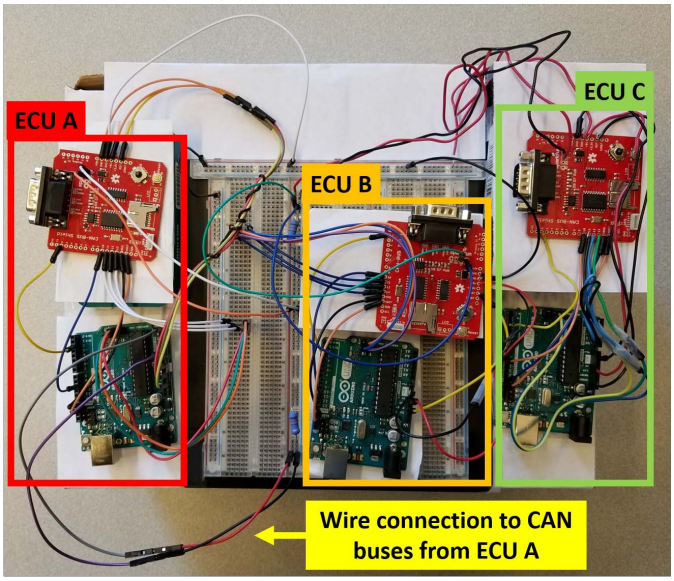


Figure 3.10: CAN bus testbed. The microcontroller of ECU A is connected to CANH and CANL via two wires. ECU A is compromised and launches the proposed voltage-based attacks. ECU C transmits messages every 1 second, and ECU B logs the received messages.

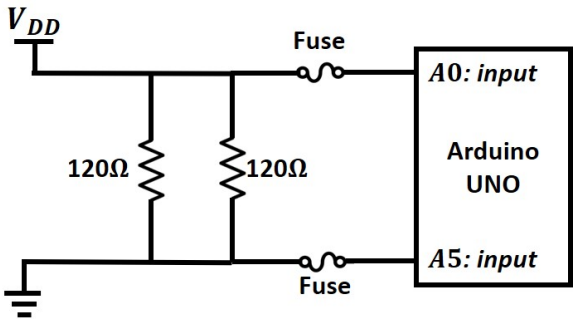


Figure 3.11: Test circuit for checking that the Arduino board can measure the voltage with the fuses. We verify that the proposed hardware-based IRSs do not thwart the voltage measurement at the Arduino board.

When transmitting a dominant bit, the actual voltages of CANH and CANL become 3.4V and 1.5V, respectively. The voltage levels for both dominant and recessive bits, however, meet the

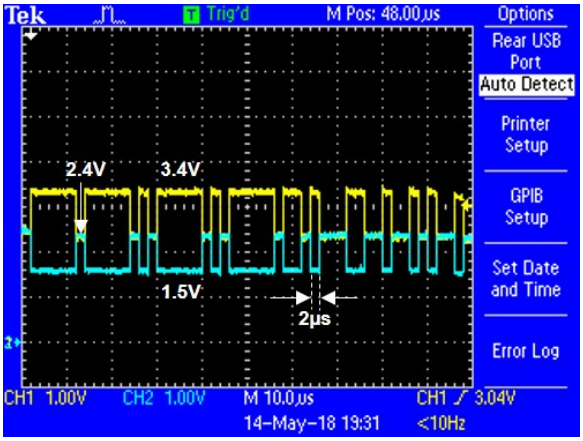


Figure 3.12: Voltages of CANH (in yellow) and CANL (in blue) in the normal message transmission. One bit is 2μs when the CAN bus data rate is 500kbps.

CAN protocol [35]. Since the CAN bus speed is set to 500kbps, one bit is 2μs long.

3.5.2 Voltage-based Attacks

We present the experimental results of the voltage-based attacks using the CAN bus testbed.

Overcurrent attack: Because we do not want to damage the testbed ECUs, we implement

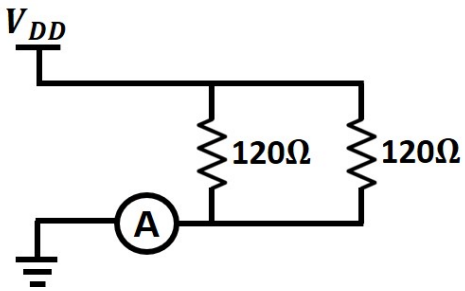


Figure 3.13: Test circuit to emulate the voltages of the CAN bus under the overcurrent attacks. We use the test circuit in order to avoid any damages on the microcontrollers of the testbed ECUs.

a test circuit as illustrated in Fig. 3.13 to emulate the CAN bus under the overcurrent attack. In

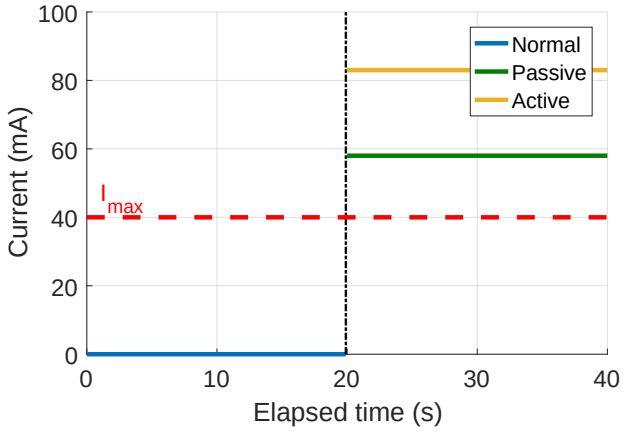


Figure 3.14: Current measured in the test circuit under the overcurrent attacks. Before the attacks occur, the current is 0mA. However, after the attacks, the current greater than I_{max} (i.e., 40mA.) flows to the ground in both attacks which may damage the microcontroller.

the test circuit, let the ground and V_{DD} represent P_L of the microcontroller and the voltage of CANH, respectively. To emulate the passive overcurrent attack, V_{DD} is set to 3.5V using the power supply, and the current flowing into the ground is measured to be 58mA. When emulating the active overcurrent attack, V_{DD} is set to 5.0V, and 83mA flows into the ground. These experimental values closely match the theoretical values. Since the current is supplied from the microcontroller in the active overcurrent attack, the Arduino board may not generate 83mA due to its hardware limitation. We measure that the maximum 52mA can be provided from an analog pin of the Arduino board, which is still large enough to damage many microprocessors including Microchip ATmega328P, Renesas V850, and NXP MPC563 [8, 43, 46].

Fig. 3.14 demonstrates the current flowing into the ground in the test circuit under the overcurrent attacks. Both passive and active overcurrent attacks occur at 20s, which is indicated by the black dashed line. The red dashed line represents I_{max} (i.e., 40mA) of the Arduino board. The current does not flow through the analog pins before the overcurrent attacks occur. After the attacks, however, the current larger than I_{max} flows through P_L, which may damage the microcontroller.

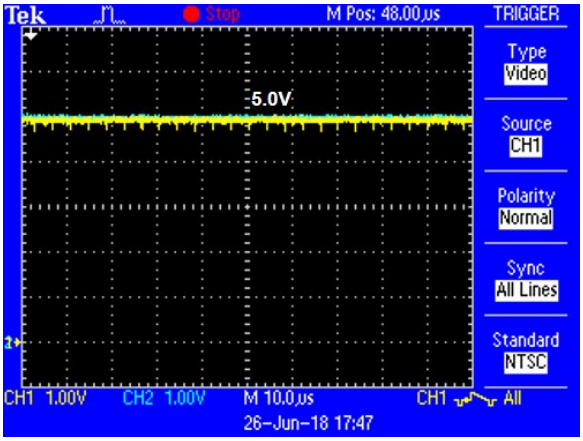


Figure 3.15: Voltages of CANH and CANL when the DoS attack is successfully launched by ECU A when $V_{attack,L}$ is set to 5.0V.

DoS attack: In order to demonstrate that the DoS attack is indeed feasible in the practical settings at an automobile, we design a DoS attack scenario using the CAN bus testbed. Initially, P_L of ECU A's Arduino board is set to the input mode for 11 seconds while the message is transmitted from ECU C every 1 second and logged at ECU B. Then, the DoS attack is launched using ECU A by setting P_L to the high voltage output mode that applies 5.0V to CANL. ECU A launches the DoS attack for 20 seconds and stops the attack by setting P_L back to the input mode. Using the oscilloscope, the voltages of each CAN bus line are observed under the attack.

Fig. 3.15 shows that the voltages of CANH and CANL become 5.0V when $V_{attack,L}$ is set to 5.0V. Because V_{Diff} is always 0.0V, a dominant bit cannot be transmitted, which demonstrates that the DoS attack is successfully launched using ECU A. We further investigate the DoS attack to determine the minimum voltage level that leads to a successful DoS attack. The power supply is connected to CANL since the analog pin of the Arduino board can only generate either 0.0V or 5.0V. The voltage from the power supply is increased from 0.1V to 5.0V, which covers the voltage range of most of the microprocessors [33, 43]. Fig. 3.16 demonstrates when the DoS attack is successful for various values of $V_{attack,L}$ where the attack indicator is 0 if the attack fails and 1 if the attack is successful. From Fig. 3.16, the DoS attack is successful if $V_{attack,L}$ is between

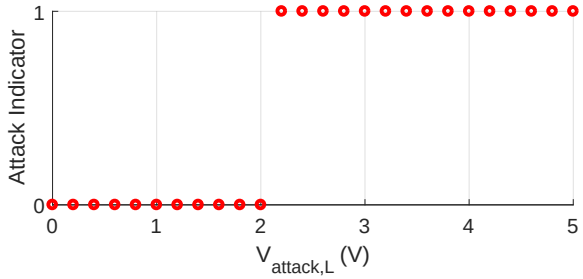
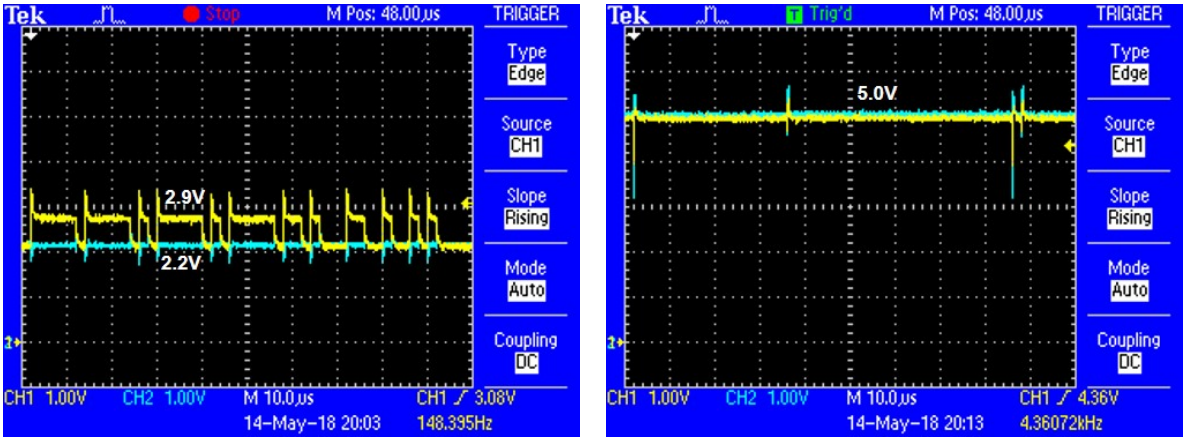


Figure 3.16: Minimum value of $V_{attack,L}$ that leads to a successful DoS attack. The attack indicator is either 0, meaning a failure of the attack or 1, meaning the success of the attack. The DoS attack becomes successful from $V_{attack,L}$=2.2V.

2.2V and 5.0V. When $V_{attack,L}$ is smaller than 3.5V, the current flows through the termination resistors, which induces a voltage drop between CANH and CANL even under the DoS attack. As demonstrated in Fig. 3.17(a), V_{Diff}, however, becomes less than the decision threshold for determining the dominant bit (i.e., 0.8V in this CAN transceiver.) when transmitting a dominant bit. If $V_{attack,L}$=5.0V, the voltages of CANH and CANL are the same as shown in Fig. 3.17(b), so the CAN bus represents a recessive bit all the time. As a result, the CAN bus is in the idle state regardless of transmitting dominant and recessive bits.

Fig. 3.18 demonstrates the message exchange between ECUs B and C when ECU A launches the DoS attack according to the attack scenario. When a message is received at t_0, the message indicator becomes 1 at t_0, and it is 0 if a message is not received. The message is received at ECU B every 1 second before the attack occurs at t=11 second. The messages, however, are not exchanged between 11-31 seconds due to the attack. As soon as ECU A stops the attack at t=31 second, the CAN bus returns to the normal state, and the messages are exchanged again. One thing to note is that the inter-arrival time between the first two messages right after stopping the attack is shorter than 1 second since the CAN transceiver transmits the message that was saved in its buffer due to the transmission failure during the attack.

Forced retransmission attack: We design an attack scenario that launches the forced retrans-



(a) $V_{attack,L}$=2.2V (b) $V_{attack,L}$=5.0V

Figure 3.17: Voltages of the CAN bus lines for $V_{attack,L}$=2.2V and 5.0V at which the DoS attack is successful. (a) When $V_{attack,L}$=2.2V, V_{Diff} is less than the threshold for determining the dominant bit (0.8V). (b) When $V_{attack,L}$=5.0V, V_{Diff} is always 0.0V since both CANH and CANL are set to 5.0V.

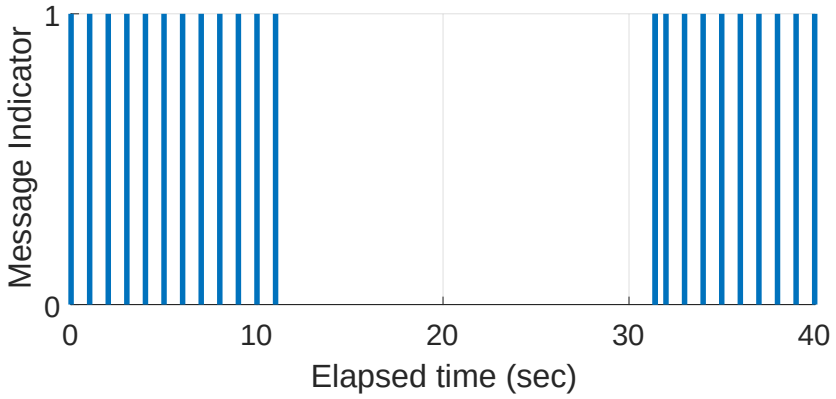


Figure 3.18: Message exchange through the CAN bus under the DoS attack. The DoS attack is launched from 11 to 31 seconds by ECU A. The message indicator at t_0 is 1 if the message is received at t_0 and 0 if a message is not received. The messages are not exchanged during the DoS attack.

mission attack using ECU A. ECU A initially sets P_H of its Arduino board to the input mode for 10 seconds and changes it to the high voltage output mode until t=30 second in order to launch the forced retransmission attack while keeping P_L in the input mode. Then, ECU A sets P_H back to the input mode to stop the attack. The settings of the CAN bus speed, the periodic message exchange between ECUs B and C, and the oscilloscope are identical to the settings used in the DoS attack.

Fig. 3.19 demonstrates the message retransmission under the forced retransmission attack. Although ECU C transmits a message every 1 second, two consecutive messages are spaced by about 30μs, which indicates the message retransmission. The voltage of CANH could not be maintained at 5.0V since the Arduino board cannot generate the current large enough to let the voltage difference between CANH and CANL be greater than 3.5V. Fig. 3.20 shows the message exchange between ECUs B and C under the forced retransmission attack according to the attack scenario. The message is received by ECU B every 1 second before the attack occurs. When the attack occurs at t=10 second, however, the message is retransmitted, and thus the interarrival time between two consecutive messages is about 132μs. As the forced retransmission attack is stopped at t=30 second, the message is exchanged every 1 second as normal.

In order to determine the minimum voltage level that successfully launches the forced retransmission attack, we connect the power supply to CANH. Since the nominal voltage of the CAN bus in the idle state is 2.5V, we increase $V_{attack,H}$ from 2.5V to 5.0V using the power supply. The forced retransmission attack becomes successful from $V_{attack,H}$=4.5V, and Fig. 3.21 shows the voltages of the CAN bus at $V_{attack,H}$=5.0V. The magnitude of V_{Diff} satisfies the CAN protocol when transmitting both dominant and recessive bits as demonstrated in Fig. 3.21. We, however, observe that the duration of a recessive bit after a dominant is significantly smaller than the nominal bit duration (2μs) since the transition time from a dominant bit to a recessive bit increases.

Fig. 3.22 compares the bit length time τ_{bit} in the normal message transmission and under the forced retransmission attack. In the normal message transmission, the transition time is almost negligible as shown in Fig. 3.22(a), and τ_{bit} is 2μs. Under the attack with $V_{attack,H}$=5.0V, however, it takes more than 1μs to change the voltage of CANL to represent a recessive bit after a dominant

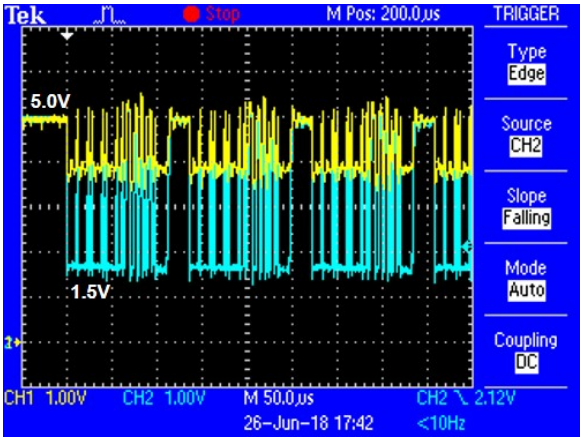


Figure 3.19: Voltages of the CAN bus for $V_{attack,H}$=5.0V. The forced retransmission attack is successfully launched by ECU A. The same voltage waveform is repeated every 132μs which indicates the message retransmission.

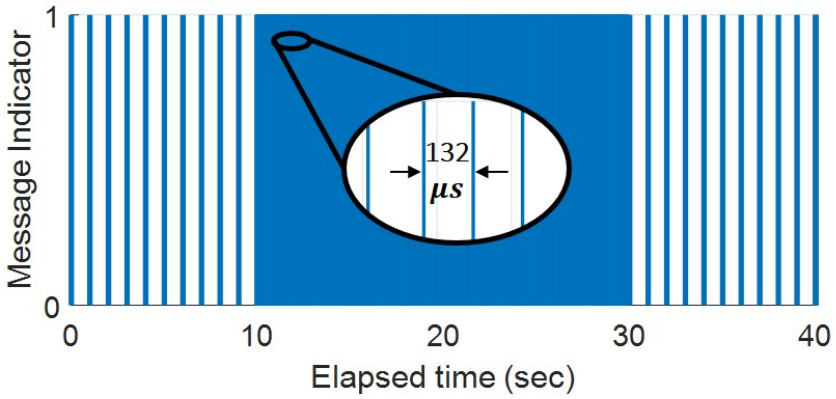


Figure 3.20: Message exchange through the CAN bus under the forced retransmission attack. The attack is launched from 10 to 30 seconds. The message is retransmitted during the forced retransmission attack.

bit is transmitted as shown in Fig. 3.22(b). The duration of a recessive bit is no longer 2μs under the attack since the transition time increases. Consequently, the duration of a recessive bit violates the CAN protocol. Since there are seven transitions from a dominant bit to a recessive bit in the

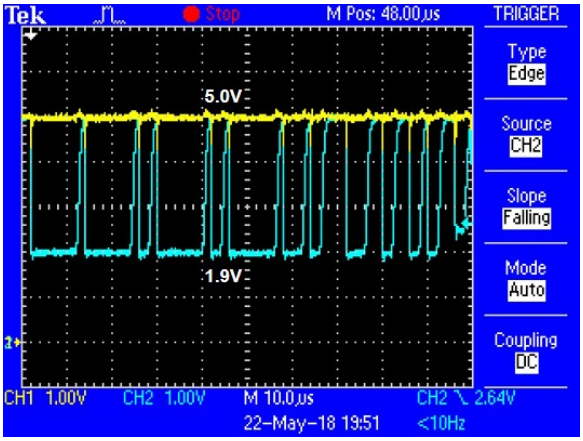
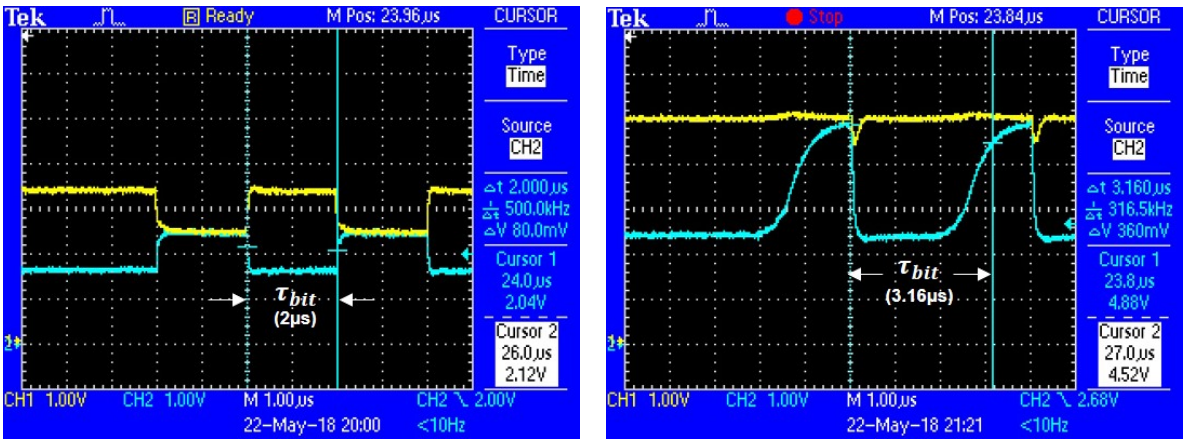


Figure 3.21: Voltages of the CAN bus for $V_{attack,H}$ =5.0V at which the forced retransmission attack becomes successful using the power supply.



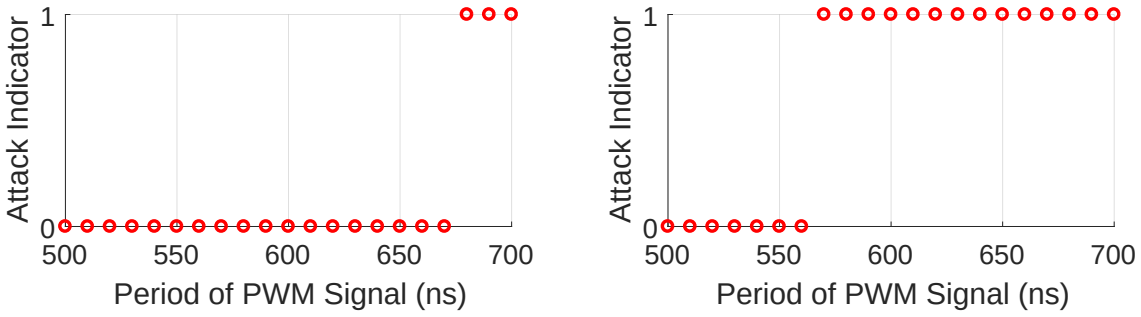
(a) Normal (b) $V_{attack,H}$=5.0V

Figure 3.22: Bit length time τ_{bit} in the normal message transmission and under the forced retransmission attack with $V_{attack,H}$=5.0V. (a) τ_{bit} is 2μs in the normal message transmission. (b) τ_{bit} is increased since the transition time of CANL from a dominant bit to a recessive bit increases under the attack.

message with ID 0x01 and data 0x01, we compute the average bit length time for various values of $V_{attack,H}$ from 2.5V to 5.0V as summarized in Table 3.2.

Table 3.2: Average bit length time for various values of $V_{attack,H}$ from 2.5V to 5.0V. A message with ID 0x01 and data 0x01 is transmitted when the CAN bus speed is 500kbps.

$V_{attack,H}$	2.5V	3.0V	3.5V	4.0V	4.5V	5.0V
Average τ_{bit}	2.00μs	2.24μs	2.86μs	2.98μs	3.07μs	3.16μs



(a) Pulse attack via CANL

(b) Pulse attack via CANH

Figure 3.23: Minimum period of the PWM signal that leads to a successful pulse attack. (a) The pulse attack is successful when the period is larger than 680ns if the pulse is applied to CANL. (b) The pulse attack becomes successful when the period is larger than 570ns if the pulse is applied to CANH.

Pulse attack: In order to determine the minimum period of the PWM signal that launches the pulse attack successfully, we apply the PWM signal with the duty cycle 50%, amplitude 5.0V, and offset 2.5V (i.e., the low output voltage is 0.0V, and the high output voltage is 5.0V) to CANL using the function generator while P_H is in the input mode. We increase the period of the PWM signal from 500ns to 700ns. Then, we repeat the same experiment after connecting the function generator to CANH, where P_L is set to the input mode. Fig. 3.23 demonstrates that the pulse attack becomes successful if the period is longer than 680ns and 570ns when the PWM signal is applied to CANL and CANH, respectively. The pulse attack via CANH becomes successful with a shorter period than that via CANL because the transition time increases when the voltage is applied to CANH.

3.5.3 Hardware-based IRSs

We demonstrate that a fuse-based IRS can mitigate all four voltage-based attacks using the CAN bus testbed. Also, the proof-of-concept of a heat-based IRS is verified using a test circuit.

Fuse-based IRS: We implement the fuse-based IRS as illustrated in Fig. 3.8(a). 58mA and 83mA flow through P_L under the passive and active overcurrent attacks, respectively. By using the fuse with current rating 10mA, both overcurrent attacks can be mitigated since P_L is disconnected from CANL. In order to verify that the fuse-based IRS can mitigate the DoS attack, forced retransmission attack, and pulse attack, we design an attack scenario in which ECU A launches an attack from 10s to 30s, while ECU C transmits messages every 1s. We repeat this experiment for each of the three attacks. For the DoS attack and forced retransmission attack, 5.0V is applied to CANL and CANH, respectively. The period of the PWM signal is set to 100μs with the duty cycle 50%, amplitude 5.0V, and offset 2.5V for the pulse attack. The message indicator is 1 if the message is received at ECU B and 0 if the message transmission fails. For all three voltage-based attacks, ECU B receives the messages every 1s as normal under each attack as shown in Fig. 3.24. This experimental result indicates that the fuse-based IRS may mitigate the voltage-based attacks.

Heat-based IRS: The maximum current that the Arduino board can supply is 52mA, by which the heat may not be generated high enough to let the thermostat disconnect the wires [8]. We implement a test circuit that emulates operations of the heat-based IRS, where the current to the heating coil is provided by the power supply instead of the Arduino board as shown in Fig. 3.25. The current does not flow through the heating coil when emulating the normal operation of a VIDS. Since current flows through either P_L or P_H under all four voltage-based attacks, we launch an attack by letting the current of 1A flow the heating coil from the power supply under the controlled laboratory setting for the safety. In the test circuit, A0 and A5 are in the input mode, while V_{DD} is set to 5.0V. Before emulating an attack, the microcontroller measures 5.0V via A0, which means the analog pin is connected to V_{DD}. After launching an emulated attack, however, the microcontroller cannot measure V_{DD} at A0, which indicates that the thermostat disconnects the wire. These experimental results demonstrate that the heat-based IRS may mitigate the voltage-based attacks.

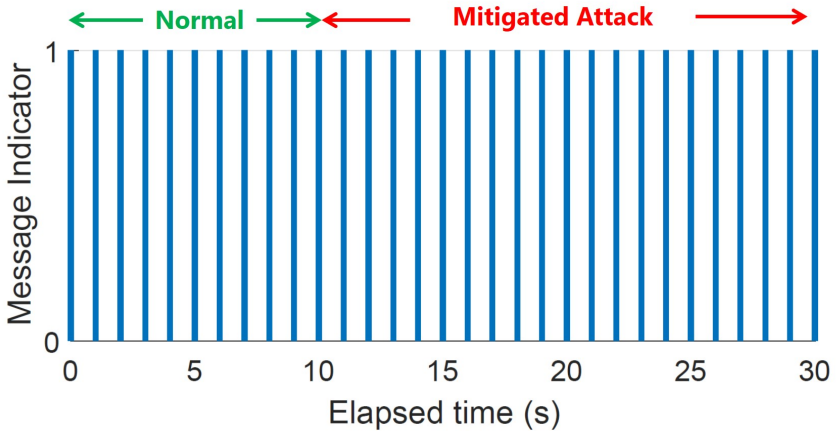


Figure 3.24: Message exchange through the CAN bus with a fuse-based IRS under the DoS attack, forced retransmission attack, and pulse attack. Each attack is launched from 10s to 30s. The fuse-based IRS mitigates all three voltage-based attacks.

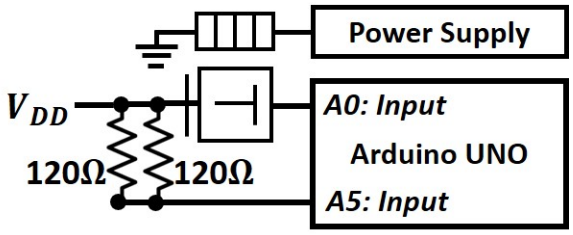


Figure 3.25: Test circuit to emulate the heat-based IRS. The power supply provides the current of 1A to the heating coil.

3.6 Conclusion

In this chapter, we investigated new vulnerabilities of the VIDS when an adversary maliciously exploits the extra wires given to the VIDS. We developed four voltage-based attacks (i.e., over-current attack, DoS attack, forced retransmission attack, and pulse attack) that may damage the microcontroller with VIDS, block message transmission, or make a message be retransmitted by manipulating the voltage levels of the CAN bus lines. In order to mitigate the voltage-based at-

tacks, we proposed two hardware-based IRSs (i.e., fuse-based IRS and heat-based IRS) that isolate the VIDS from the CAN bus once the voltage-based attacks are detected. We demonstrated the voltage-based attacks and hardware-based IRSs using our CAN bus testbed. This work implies that a new attack surface might be introduced to the CAN bus by the wires that directly connect the CAN bus lines and the VIDS if the VIDS is compromised. In order to provide security assurance to an automobile, defense systems based on hardware must be implemented together with the VIDS if the voltage characteristics are exploited as a fingerprint of ECUs.

Chapter 4

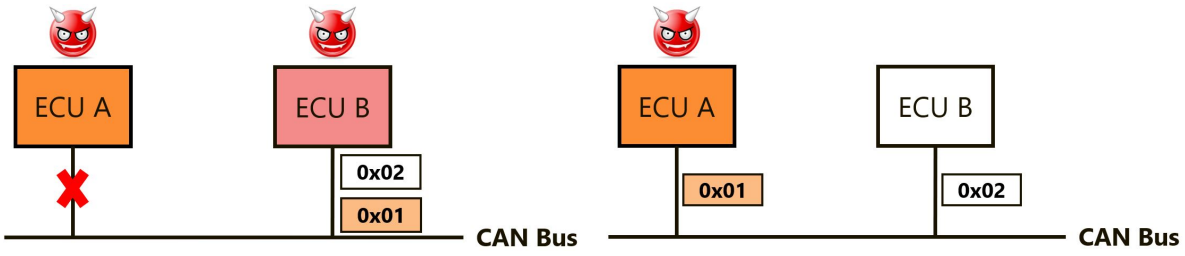
MOTION-BASED INTRUSION DETECTION SYSTEM

4.1 Introduction

Many IDSs that detect cyber attacks using abnormal deviations in traffic through the CAN bus, such as the number of messages transmitted through the CAN bus in a unit time, have been proposed [26, 42]. Because most of the messages in the CAN protocol are transmitted with a fixed length and period, when an adversary injects spoofed messages, an IDS that detects the existence of spoofed messages using a frequency of message occurrence is proposed [26]. A coincidence among a set of messages is also exploited to detect an attack in an entropy-based IDS [42]. The entropy-based IDS, however, can be bypassed if an adversary replicates the structure and pattern of legitimate traffic [18].

In order to detect attacks that replicate the legitimate traffic through the CAN bus, IDSs that fingerprint ECUs by exploiting physical properties of ECUs have been developed [18, 19, 20]. These IDSs detect spoofed messages by identifying and tracking the transmitter of each message. The authors of [18] proposed the CIDS that detects an attack by tracking a sudden change in the clock skew of ECUs. We developed the cloaking attack and demonstrated that the cloaking attack bypasses the CIDS by mimicking the clock skew in Chapter 2. Since a CAN transceiver's hardware determines voltage characteristics of ECUs, the voltage characteristics are unique to each ECU and challenging to replicate. The voltage-based IDSs have been proposed, which detect an attack by exploiting the voltage characteristics of each ECU [19, 20].

An adversary, however, may compromise a legitimate ECU and inject spoofed messages that convey false sensor measurements or control signals using that compromised ECU. These IDSs cannot detect this data falsification attack since the transmitter of the spoofed messages remains the same after the attack occurs, which does not induce any deviations in physical properties.



(a) Masquerade attack (b) Data falsification attack

Figure 4.1: Comparison between masquerade attack and data falsification attack. (a) In the masquerade attack, the transmitter of a spoofed message 0x01 is changed from ECU A to ECU B. Consequently, the physical properties such as clock skew and voltage characteristics that are tracked from message 0x01 changes after the masquerade attack occurs. (b) In the data falsification attack, however, the spoofed message 0x01 is transmitted from the compromised ECU A that is the original transmitter of message 0x01 after the data falsification attack occurs. As a result, the physical properties do not deviate from the normal behavior.

Consider two ECUs A and B that transmit messages with IDs 0x01 and 0x02, respectively. As illustrated in Fig 4.1(a), the compromised ECU B injects a spoofed message 0x01 instead of ECU A in a masquerade attack [18]. The IDSs that fingerprint each ECU can detect the masquerade attack because the transmitter of message 0x01 is changed, which causes deviations in physical properties. An adversary, however, may spoof data of message 0x01 using the compromised ECU A in a data falsification attack as shown in Fig. 4.1(b). The data falsification attack can bypass these anomaly-based IDSs since the transmitter of message 0x01 is not changed after the attack. In a contemporary automobile, multiple and different types of sensors are installed in order to measure the same movement of the automobile such as vehicle speed. The authors of [24] demonstrated that there is a correlation between measurements from different types of sensors under the normal operation of a vehicle because physical movement affects multiple sensors simultaneously. They, however, did not propose a concrete structure of an IDS or analyze the detection probability of the IDS under the data falsification attack. In order to detect the data falsification attack, we propose a

motion-based IDS (MIDS) that exploits the correlation between sensor measurements and control signals in the messages.

An ECU encapsulates multiple sensor measurements or control signals in a message after converting them to a bit sequence between 1 and 8 bytes. In order to compute correlation coefficients, these sensor measurements and control signals have to be extracted from the messages. We find the actual values of the sensor measurements and control signals as follows. First, we check a unit of the targeted data that we want to extract. Second, we check the message ID and bit location in the data field, where the targeted data is saved. This information can be found either by reverse engineering or in an interface control document of a vehicle. Third, we convert a bit sequence to a decimal number. If the Most Significant Bit (MSB) is a sign bit, the decimal number is negative when the MSB is 1. Fourth, we multiply a conversion ratio to that decimal number and then add an offset to get the actual value. For instance, Message 0x0B0 of a 2010 Toyota Camry contains measurements of a wheel speed sensor, which is located from the 1^{st} bit to 16^{th} bit [49]. Also, the wheel speed is measured in km/h, and the conversion ratio and offset for the wheel speed are 0.1 and 0, respectively.

4.2 Adversary Model

In this section, we describe an adversary model in which an adversary compromises a legitimate ECU and launches a data falsification attack. The adversary may have physical access to the CAN bus through the on-board diagnostics (OBD-II) port. Then, the adversary uploads its malicious code to a targeted ECU using a pass-thru device such as Hyundai Global Diagnostic System [1] and Volkswagen VAG-COM Diagnostic System [48] to compromise the ECU. The adversary can also remotely compromise an ECU without physical access to the CAN bus [15, 38]. We consider two cases: (1) the adversary compromises an ECU having a telematics unit, and (2) the adversary compromises an ECU without a telematics unit. In the first case, the adversary remotely accesses the operating system of an ECU having a telematics unit to figure out a particular code that handles wireless connectivity by reverse engineering. By exploiting that code, the adversary executes its malicious code on that ECU [15]. In the second case, the adversary lets a remotely compromised

ECU, which is compromised in the first case, upload the malicious code to a targeted ECU without a telematics unit as a pass-thru device does. As a result, the adversary can compromise any ECUs in the CAN bus [15].

Once an ECU is compromised, the adversary may manipulate periods and data fields of messages. Since an attack that changes periods of messages can be easily detected using the existing IDSs [18, 19, 20, 26, 42], we consider two types of the data falsification attacks. In the first type of attack, the adversary adds a disturbance to the legitimate data, where the disturbance is generated according to a zero-mean Gaussian distribution in order to analyze the effects of deviation of the disturbance. In the second type of attack, the adversary manipulates one of the wheel speed sensor values to be increased, in which a non-zero-mean disturbance is added, while other sensor values remain the same. The physical properties of ECUs do not deviate from the normal behavior after the data falsification attack, and the data falsification attack bypasses the existing IDSs.

4.3 Motion-based IDS

In this section, we propose MIDS, an IDS that exploits the correlation between messages. MIDS may exploit any pair of messages that contain data of the same physical movement of a vehicle. Without loss of generality, we use speed-related data for MIDS in this chapter.

4.3.1 Correlation Between Messages

Consider $\bar{a}$ and $\bar{b}$ with the same length of n, where a_i and b_i denote the i^{th} samples of $\bar{a}$ and $\bar{b}$, respectively. The correlation coefficient ρ between $\bar{a}$ and $\bar{b}$ can be computed as follows:

$$\rho = \frac{\sum_{i=1}^{n}(a_i - \mu_a)(b_i - \mu_b)}{\sqrt{\sum_{i=1}^{n}(a_i - \mu_a)^2}\sqrt{\sum_{i=1}^{n}(b_i - \mu_b)^2}}, \tag{4.1}$$

where μ_a and μ_b denote means of $\bar{a}$ and $\bar{b}$, respectively. We use Pearson correlation coefficient because sensor measurements of the same vehicle movement are in the linear relationship (Chapter 5.6 in [30]). Multiple sensors are implemented in a vehicle to measure the same movement such as vehicle speed, and measurements from these sensors are highly correlated with each other during normal operation [24]. We use data collected from a 2010 Toyota Camry, which contains

Table 4.1: Correlation coefficients of four wheel speeds, vehicle speed, and steering wheel angle. The wheel speeds and vehicle speed are highly correlated with each other, whereas the steering wheel angle is not correlated with wheel speed 1.

Data 1	Data 2	Correlation Coefficient
Wheel Speed 1	Wheel Speed 2	0.9999
Wheel Speed 1	Wheel Speed 3	0.9999
Wheel Speed 1	Wheel Speed 4	0.9999
Wheel Speed 2	Wheel Speed 3	0.9998
Wheel Speed 2	Wheel Speed 4	0.9999
Wheel Speed 3	Wheel Speed 4	1.0000
Wheel Speed 1	Vehicle Speed	0.9999
Wheel Speed 1	Steering Wheel Angle	-0.5324

four wheel speeds, vehicle speed, and steering wheel angle [49]. Table 4.1 shows correlation coefficients for possible pairs of four wheel speeds, vehicle speed, and steering wheel angle, using Eq. (4.1). All four wheel speeds and vehicle speed are highly correlated with each other (i.e., $\rho > 0.99$), while the steering wheel angle is not correlated with wheel speed 1 because the vehicle speed and steering wheel angle are independently controlled.

4.3.2 Proposed MIDS

MIDS consists of a correlation computer and CUSUM detector as illustrated in Fig. 4.2. In this section, we explain how MIDS computes correlation coefficients and tracks deviations in them to detect an attack.

Correlation Computer: The correlation computer calculates correlation coefficients between sensor measurements and control signals in messages. An ECU may not save all ongoing messages through the CAN bus due to its limited memory size and computation capability. Computing correlation coefficients using all past data, as defined in Eq. (4.1), is not a viable option in practice.

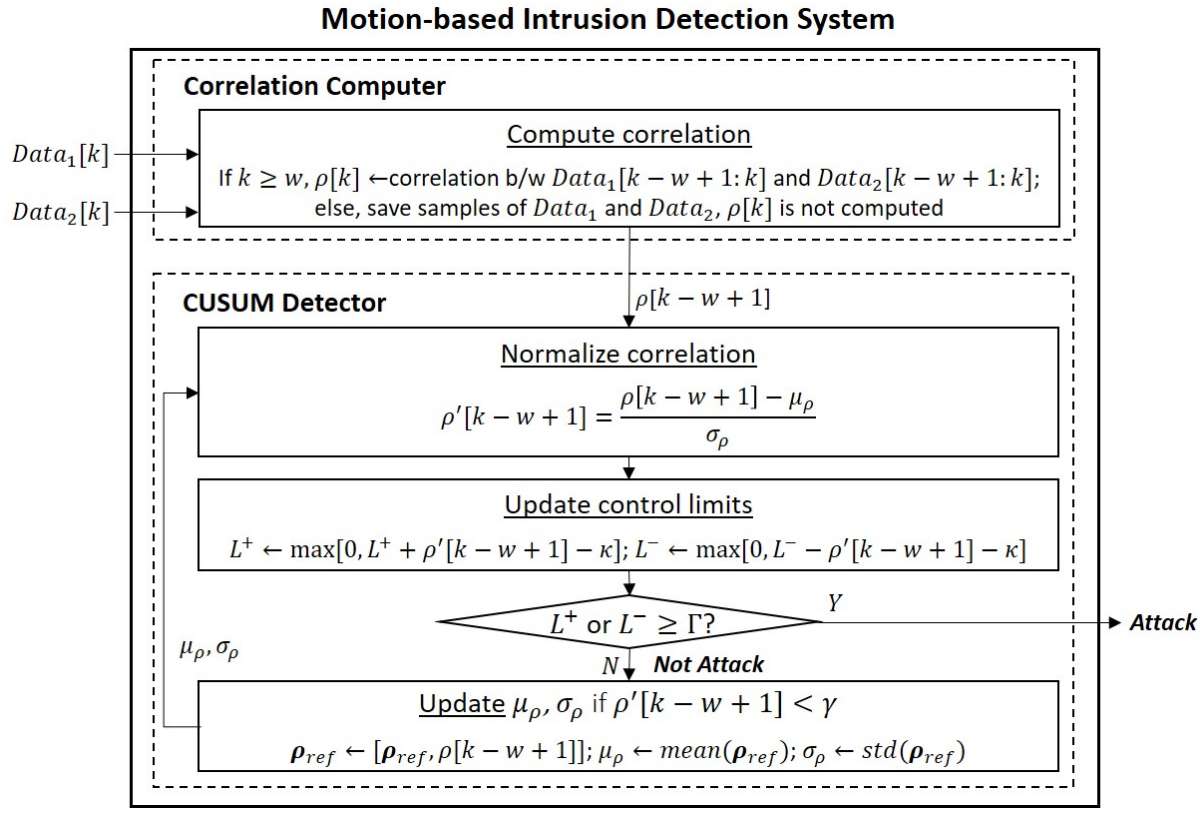


Figure 4.2: Structure of MIDS. The consists of a correlation computer and CUSUM detector. The correlation computer calculates correlation coefficients between sensor measurements and control signals in messages. The CUSUM detector tracks the correlation coefficients and detects abnormal deviation in the correlation coefficient, which indicates an occurrence of the attack.

MIDS computes correlation coefficients using the w most recent data samples in real-time, which is updated by a sliding window. The window size is determined by ECU's memory size and computation capability. If the number of data samples is less than w (i.e., $k < w$), MIDS saves the data samples in its memory but does not compute a correlation coefficient. When $k \geq w$, correlation coefficients are computed and fed to the CUSUM detector as described in Fig. 4.2.

CUSUM Detector: MIDS uses the CUSUM method that computes cumulative sums of deviations from the normal value in order to detect a sudden change. The CUSUM method is widely

used to track the drift of values that steadily increase or decrease (Chapters 2.1 and 2.2 in [11]). MIDS keeps the normal behavior of correlation coefficients by tracking the mean μ_ρ and standard deviation σ_ρ of ρ_{ref} that is a vector containing all previous ρ's from the normal data. For every incoming ρ, MIDS computes the normalized correlation coefficient $\rho' = \frac{\rho - \mu_\rho}{\sigma_\rho}$. Using ρ', the upper and lower control limits, L^+ and L^-, are updated as follows:

$$L^+ \leftarrow \max\left[0, L^+ + \rho' - \kappa\right], \; L^- \leftarrow \max\left[0, L^- - \rho' - \kappa\right],$$

where both L^+ and L^- are initially set to zero, and κ is the sensitivity threshold. MIDS declares an attack when either the control limit exceeds the detection threshold Γ. MIDS appends the current ρ to ρ_{ref} if $\rho' < \gamma$, where γ is the update threshold. Then, μ_ρ and σ_ρ are updated and used to normalize the next incoming ρ. We set γ, κ, and Γ to such values that make zero false-positive probability in the normal data [18].

4.4 Theoretical Analytics of MIDS

We derive standard deviations of a disturbance at which the attack is not detected by MIDS or always detected by MIDS. Then, we define a performance metric.

4.4.1 Bounds on Detection Probability of MIDS

Consider $\bar{a}$ and $\bar{b}$ that are measurements from two different sensors. An adversary spoofs $\bar{b}$ by adding a disturbance $\bar{d}$ to yield $\bar{b'} := \bar{b} + \bar{d}$, where $\bar{d}$ is generated according to a Gaussian distribution $N(0, \sigma^2)$. w and l denote the window size and index at which a spoofing attack starts in the window. $\bar{b'}$ can be approximated as $\bar{a} + \bar{d}$ because $\bar{b} \simeq \bar{a}$. If $\bar{b}$ is disturbed from the l^{th} element in the window with size w, using Eq. (4.1), the correlation coefficient ρ between $\bar{a}$ and $\bar{b'}$ can be approximated to be

$$\rho \simeq \frac{\sigma_a^2 + \sum_{i=l}^{w} a_i d_i}{\sigma_a \sqrt{\sigma_a^2 + 2\sum_{i=l}^{w} a_i d_i + \sum_{i=l}^{w} d_i^2}}. \tag{4.2}$$

where σ_a denotes the standard deviation of elements of $\bar{a}$. For the analysis, we assume that ρ linearly decreases as l decreases, which is experimentally observed in the experimental results.

Also, μ_ρ and σ_ρ are not updated because $\rho' \geq \gamma$ during the attack as explained in the previous section. Then, we derive σ_{bypass}, the maximum standard deviation of $\bar{d}$ below which MIDS cannot detect a spoofing attack in Theorem 1.

Theorem 1. *A data falsification attack is not detected by MIDS if $\sigma < \sigma_{bypass}$, where $\sigma_{bypass} = \sigma_a\sqrt{\frac{1}{(\rho_0-\kappa\sigma_\rho)^2}-1}$.*

Proof. When all elements of $\bar{b}'$ are disturbed (i.e., l=1), $\sum_{i=l}^{w} a_i d_i$ is zero because the mean of $\bar{d}$ is zero and $\bar{a}$ and $\bar{d}$ are independent. As a result, $\rho \simeq \frac{\sigma_a^2}{\sigma_a\sqrt{\sigma_a^2+\sigma^2}} = \frac{\sigma_a}{\sqrt{\sigma_a^2+\sigma^2}}$, which is the minimum value of ρ after the attack. Since ρ decreases linearly, a decrement of ρ per sample for σ_{bypass}, Δ_{bypass}, can be derived as

$$\Delta_{bypass} = \frac{\rho_0 - \frac{\sigma_a}{\sqrt{\sigma_a^2+\sigma^2}}}{w}. \tag{4.3}$$

We find the condition under which both control limits, L^+ and L^-, stay zero. The normalized correlation coefficient at the k^{th} data sample, $\rho'(k-w+1)$, can be derived in terms of Δ_{bypass} as

$$\rho'(k-w+1) = \frac{\rho_0 - (k-n_{normal})\Delta_{bypass} - \mu_\rho}{\sigma_\rho} = -\frac{(k-n_{normal})\Delta_{bypass}}{\sigma_\rho},$$

because $\rho_0 \simeq \mu_\rho$ when there is no attack. L^+ and L^- stay zero if $\frac{(k-n_{normal})\Delta_{bypass}}{\sigma_\rho}$ is less than κ for all $k \leq (n_{normal}+w)$, which indicates that the data falsification attack is not detected. At the last attack sample that corresponds to the minimum value of ρ (i.e., $k = n_{normal}+w$), Δ_{bypass} becomes

$$\Delta_{bypass} = \frac{\kappa\sigma_\rho}{w}. \tag{4.4}$$

Substituting Eq. (4.3) into Eq. (4.4) gives $\sigma_{bypass} = \sigma_a\sqrt{\frac{1}{(\rho_0-\kappa\sigma_\rho)^2}-1}$. □

We derive σ_{detect}, the minimum standard deviation of $\bar{d}$, above which the spoofing attack is always detected by MIDS, in the following theorem.

Theorem 2. *MIDS can always detect a data falsification attack if $\sigma > \sigma_{detect}$, where $\sigma_{detect} = \sigma_a\sqrt{\frac{1}{(\rho_0-j\sigma_\rho(\Gamma+\kappa))^2}-1}$.*

Proof. If ρ_{attack} is small enough to make L^- larger than Γ, the attack is always detected at the first attack sample when ρ_{attack} denotes the correlation coefficient right after an attack. Using Eq. (4.2), ρ decreases from ρ_0 to $\frac{\sigma_a}{\sqrt{\sigma_a^2+{\sigma_{detect}}^2}}$ at the j^{th} attack sample in the window if σ is set to σ_{detect}. $\frac{\sigma_a}{\sqrt{\sigma_a^2+{\sigma_{detect}}^2}} \simeq 0$ because $\sigma_{detect} >> \sigma_a$. As a consequence, j can be approximated as $\left\lceil \frac{\rho_0}{\rho_0-\rho_{attack}} \right\rceil$, where $\lceil\ \rceil$ denotes a ceiling operator. A decrement of ρ per sample for σ_{detect}, Δ_{detect}, can be derived as

$$\Delta_{detect} = \frac{\rho_0 - \frac{\sigma_a}{\sqrt{\sigma_a^2+{\sigma_{detect}}^2}}}{j}. \tag{4.5}$$

An attack is always detected if either L^+ and L^- becomes larger than Γ at the first attack sample. Notice that $L^- > \Gamma$ when $\rho_{attack} \leq \mu_\rho - \sigma_\rho(\Gamma + \kappa)$. As a result, Δ_{detect} can be also represented as follows:

$$\Delta_{detect} = \sigma_\rho(\Gamma + \kappa). \tag{4.6}$$

Substituting Eq. (4.5) into Eq. (4.6) gives $\sigma_{detect} = \sigma_a\sqrt{\frac{1}{(\rho_0 - j\sigma_\rho(\Gamma+\kappa))^2} - 1}$. □

4.4.2 Performance Metric

We introduce a metric that formalizes how the detection probability is close to σ_{bypass}. Let $P_d(\sigma)$ denote the detection probability of MIDS when the standard deviation of the disturbance is σ. We define σ_{lim} as:

$$\sigma_{lim}(\epsilon) = \min\left[\sigma : P_d(\sigma) > 1 - \epsilon\right].$$

We define the ϵ-Deviation Index (ϵ-DI) as ϵ-DI$= \sigma_{lim}(\epsilon) - \sigma_{bypass}$. A smaller value of ϵ-DI signifies a more effective detector and less freedom for the attacker at choosing a standard deviation of the disturbance.

4.5 Experimental Result

In this section, we demonstrate that MIDS is effective in detecting a data falsification attack by using data from two real vehicles.

4.5.1 Testbeds

UW EcoCAR: The UW EcoCAR is used in our experiment in a controlled environment as shown in Fig 4.3(a). A Vector GL1000 logger is connected to the CAN bus of the vehicle via the OBD-II port to collect all ongoing messages through the CAN bus as shown in Fig 4.3(b). We collect data while the vehicle is being driven on the road for 68 minutes in order to validate that MIDS detects a data falsification attack in a realistic environment. We also collect data for 24 minutes while the UW EcoCAR is on a chassis dynamometer. This experiment emulates an attack in which an adversary spoofs the rear wheel speed to large values because only the rear wheels rotate on the chassis dynamometer (i.e., adding non-zero-mean disturbance). We find messages that contain rear wheel speed, front wheel speed, odometer data, and battery voltage data using the interface control document. We assume that MIDS uses a pair of data that are transmitted from two different ECUs where only one ECU is compromised, thus data from only one ECU is spoofed. Since MIDS does not use odometer data and battery voltage data as an input pair in the evaluation, the fact that these data have come from the same ECU is not violating the assumption.

Toyota Camry: In order to demonstrate that MIDS applies to other vehicles, we use the CAN data logged from a 2010 Toyota Camry [49]. The data is collected using a Dearborn Group Gryphon S3 and the Hercules software through a wireless link. Messages 0x0B0 and 0x0B2 contain measurements from all four wheel speed sensors (i.e., two wheel speeds in each message). Because each wheel that corresponds to respective wheel speed sensor is not reverse engineered in [49], we indicate two wheel speeds in Message 0x0B0 as wheel speeds 1 (from 1^{st} bit to 16^{th} bit) and 2 (from 17^{th} bit to 32^{nd} bit) and the other two wheel speeds in Message 0x0B2 as wheel speeds 3 (from 1^{st} bit to 16^{th} bit) and 4 (from 17^{th} bit to 32^{nd} bit). Message 0x610 (from 17^{th} bit to 24^{th} bit) contains the vehicle speed [3].

CAN Data Preprocessing: In order to compute correlation coefficients using Eq. (4.2), the number of messages that contains sensor measurements or control signals has to be the same as that of the other message. The number of two different messages, however, can be different if their periods are different. For instance, periods of Messages 0x0B0 (wheel speed) and 0x610 (vehicle

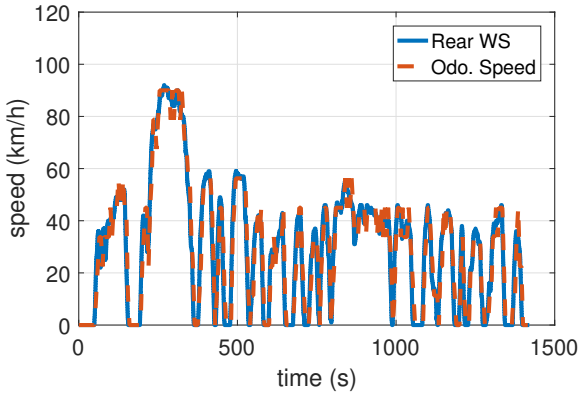
(a) UW EcoCAR (b) Vector GL1000 logger

Figure 4.3: UW EcoCAR is used to validate MIDS. We collect the data from the UW EcoCAR while the vehicle is driven on the road and on the chassis dynamometer. We connect a Vector GL1000 logger to the OBD-II port of the UW EcoCAR. All messages transmitted through the CAN bus are collected using the Vector GL1000 logger.

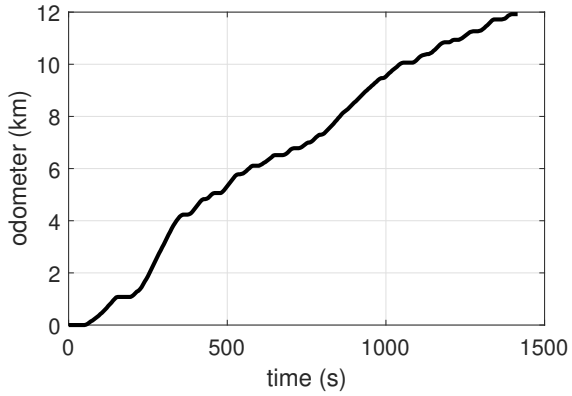
speed) in the Toyota Camry are 10ms and 500ms, respectively. As a result, the length of the wheel speed is 50 times longer than that of the vehicle speed. We generate more samples for the vehicle speed by interpolation to make its length be the same as that of the wheel speed when computing a correlation coefficient between the wheel speed and vehicle speed in Table 4.1.

Fig. 4.4 shows rear wheel speed, odometer-based vehicle speed that is computed using the odometer data, odometer, and battery voltage of the testbed vehicle on the dynamometer. Fig 4.4(a) contains rear wheel speed and odometer-based vehicle speed because the front wheels do not rotate on the dynamometer for a rear-wheel driven vehicle. We omit to show the front wheel speed. Fig 4.4(a) shows that the rear wheel speed matches closely to the odometer-based vehicle speed. For the odometer, we set the initial value to 0km to make Fig. 4.4(b) shows the distance that the UW EcoCAR is driven in that data collection. The battery voltage is within a normal range (12-14V) as shown in Fig. 4.4(c).

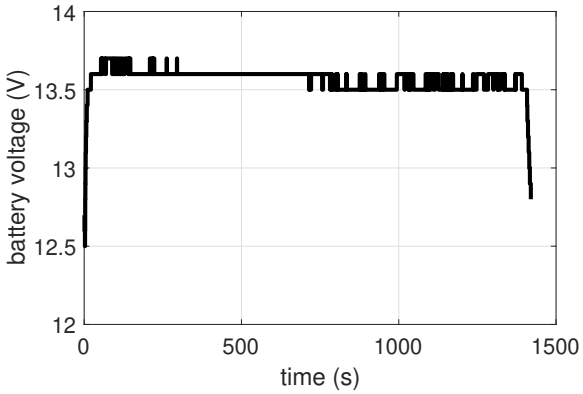
The rear wheel speed, front wheel speed, odometer, and battery voltage of the UW EcoCAR



(a) Wheel speed and odometer-based vehicle speed



(b) Odometer



(c) Battery voltage

Figure 4.4: Rear wheel speed and odometer-based vehicle speeds, odometer, and battery voltage of the UW EcoCAR when it is driven on the dynamometer. The odometer-based vehicle speed is computed by taking the first order derivative of the odometer data as shown in (b). Rear wheel speed and odometer-based vehicle speed are close to each other. The battery voltage stays within the normal operation range (i.e., 12-14V).

are presented in Fig. 4.5 when it is driven on the road. Fig. 4.5(a) shows the rear wheel speed, front wheel speed, and vehicle speed of the UW EcoCAR. For the odometer, we also set the initial value of the odometer data to 0km in Fig. 4.5(b) as Fig. 4.4(b). We compute the vehicle speed by differentiating odometer values and verify that wheel speeds closely match with the vehicle speed.

The battery voltage is within a normal range while the vehicle is driven on the road as shown in Fig. 4.5(c).

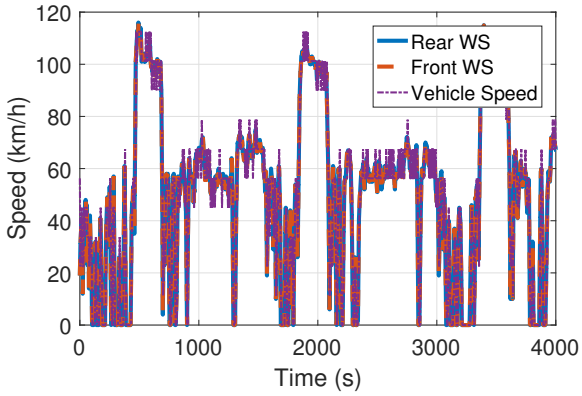
We also extract four wheel speeds and vehicle speed of the Toyota Camry as shown in Fig. 4.6. For the Toyota Camry, all four wheel speeds and vehicle speed are available, although we do not know which wheel speed corresponds to an actual wheel location in the vehicle. We denote four wheel speeds as WS1, WS2, WS3, and WS4. Fig. 4.6 demonstrates that all four wheel speeds and vehicle speed are close to each other. The experimental results show that we can decode the CAN messages of UW EcoCAR and Toyota Camry correctly.

4.5.2 Example of MIDS

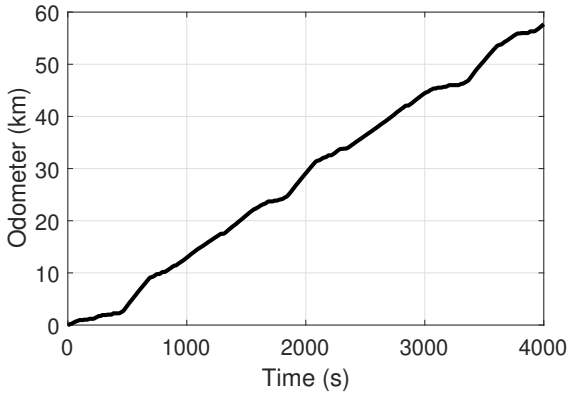
For an illustration of MIDS, we observe the correlation coefficients and control limits of the CUSUM detector under a single execution of a data falsification attack on the UW EcoCAR. We generate attack data by adding a disturbance to the front wheel speed, where the disturbance is generated according to a Gaussian distribution with a zero mean and standard deviation σ. In this experiment, we set the window size to 1500, update threshold γ to 4, sensitivity threshold κ to 7, and detection threshold Γ to 5 in order to avoid false alarm.

Fig. 4.7 shows scatter plots between the front and rear wheel speeds using 2500 samples. As demonstrated in Fig. 4.7(a), the front wheel speed is almost the same as the rear wheel speed without an attack. Figs. 4.7(b) and 4.7(c) present scatter plots under the attack when σ is 10 and 20, respectively. As increasing σ, points are scattered in a wider area, which indicates that the front wheel speed becomes less correlated with the rear wheel speed.

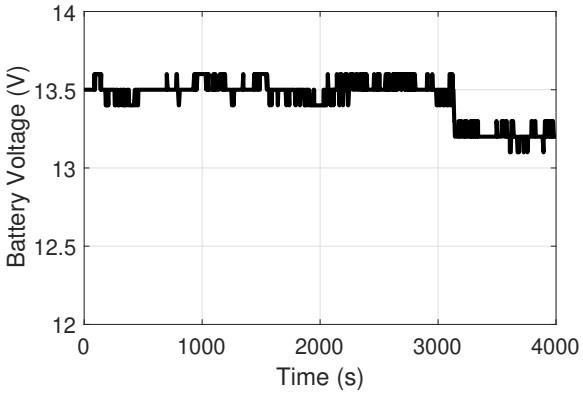
MIDS computes the correlation coefficients between the front and rear wheel speeds for 250 seconds to track the normal behavior of the correlation coefficients before a data falsification attack occurs. Then, the attack data is fed to MIDS. Fig. 4.8(a) demonstrates that the correlation coefficient ρ is greater than 0.99 before the attack, which indicates that the front wheel speed is highly correlated with the rear wheel speed. After the attack occurs, ρ decreases to 0.9566, 0.8514, and 0.7246 at the sample index of 2000 when σ is set to 10, 20, and 30, respectively. This result shows that ρ decreases more if a larger disturbance is added to the legitimate data. When a small



(a) Wheel speeds and vehicle speed



(b) Odometer

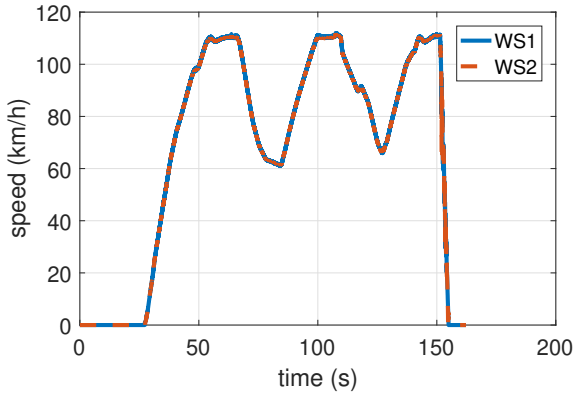


(c) Battery voltage

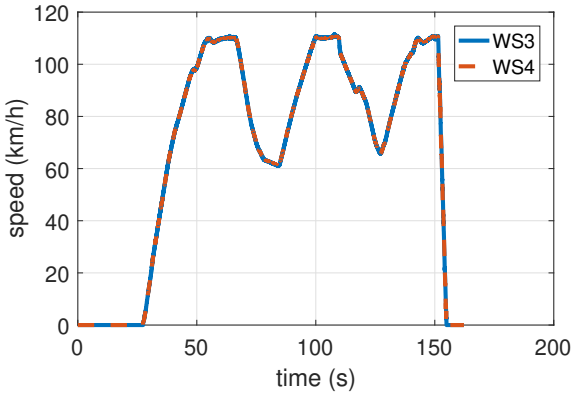
Figure 4.5: Front wheel, rear wheel, and vehicle speeds, odometer, and battery voltage of the UW EcoCAR when it is driven on the road. Both front wheel speed and rear wheel speed are extracted from the CAN messages, and those wheel speeds are close to the vehicle speed. Also, the battery voltage stays in the normal operation range in the data collection on the road.

disturbance is added (σ=1), the deviation in ρ is too small to increase either control limits. Consequently, L^+ and L^- stay zero as shown in Fig. 4.8(b). When σ is set to 20, ρ suddenly drops, which makes the lower control limit increase right after the attack as demonstrated in Fig. 4.8(c). Since the lower control limit becomes larger than Γ=5, MIDS detects the attack.

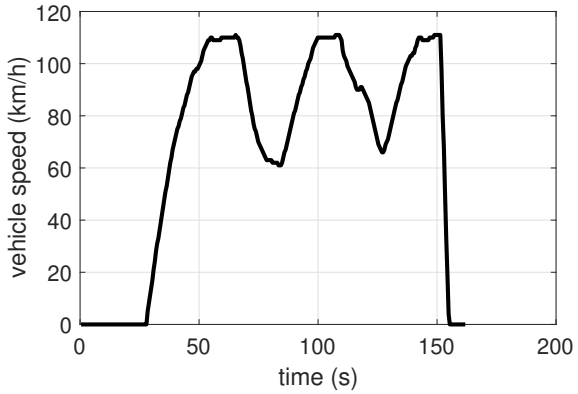
Using the data collected from the UW EcoCAR on the chassis dynamometer, Fig. 4.9(a) shows



(a) Wheel speeds 1 and 2



(b) Wheel speeds 3 and 4



(c) Vehicle speed

Figure 4.6: Four wheel speeds and vehicle speed of the Toyota Camry. Wheel speeds 1 and 2 are extracted from Message 0x0B0 and wheel speeds 3 and 4 are extracted from Message 0x0B2. Message 0x610 conveys the vehicle speed.

the front and rear wheel speeds of the UW EcoCAR when an adversary spoofs the rear wheel speed between 250s and 270s (between the sample indexes of 1000 and 1200) in order to cause a wheelspin. During the attack, the front wheel speed is set to 10km/h, while the rear wheel speed data is manipulated to be increased from 10km/h to 70km/h. In this experiment, we set the window size to 1500, γ to 4, κ to 7, and Γ to 5. As demonstrated in Fig. 4.9(b), the correlation coefficient decreases to 0.83 at the sample index of 1200 after the attack. Fig. 4.9(c) demonstrates that the

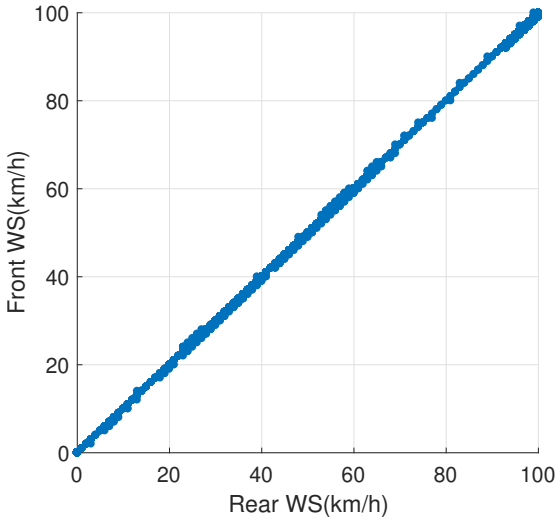
lower control limit increases above Γ=5, which demonstrates that MIDS successfully detects the data falsification attack.

4.5.3 Example of Detection Probability of MIDS

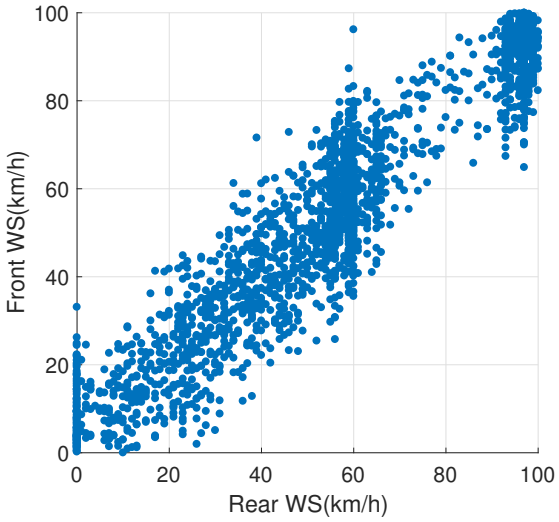
We select pairs of speed-related values for MIDS in each vehicle. The front and rear wheel speeds are used for the UW EcoCAR, and a disturbance is added to the front wheel speed. For the Toyota Camry, wheel speeds 1 and 3 are used where a disturbance is added to the wheel speed 3. MIDS is fed with 2500 samples of the normal data, followed by n_{attack} samples of the attack data in each experiment. MIDS is successful if it detects an attack and is failed otherwise. 100 and 95 non-overlapping segments of size n_{attack} are prepared from the attack data to emulate 100 and 95 independent attack experiments for the UW EcoCAR and Toyota Camry, respectively. We compute the detection probability of MIDS, which is the percentage of experiments where MIDS is successful. For this evaluation, we set γ to 4, κ to 7, and Γ to 5.

Fig. 4.10 demonstrates the detection probability of MIDS for the UW EcoCAR and Toyota Camry. We set n_{attack} to 60, 80, and 100 and window size to 750 and 1500. The black solid line and black dashed line indicate σ_{bypass} and σ_{detect}, respectively. Due to the limitation of the space, we present values of σ_{detect} with a black arrow in Fig. 4.10. For a window size of 1500 and ϵ=0.05, ϵ-DI decreases from 1.23 to 1.05 as increasing n_{attack} from 80 to 100 in the UW EcoCAR. The same trend is observed from the Toyota Camry that ϵ-DI decreases from 0.80 to 0.68 when n_{attack} is increased from 80 to 100. ρ decreases to a smaller value when MIDS exploits more spoofed messages per attack experiment, which makes MIDS more effective in detecting the attack.

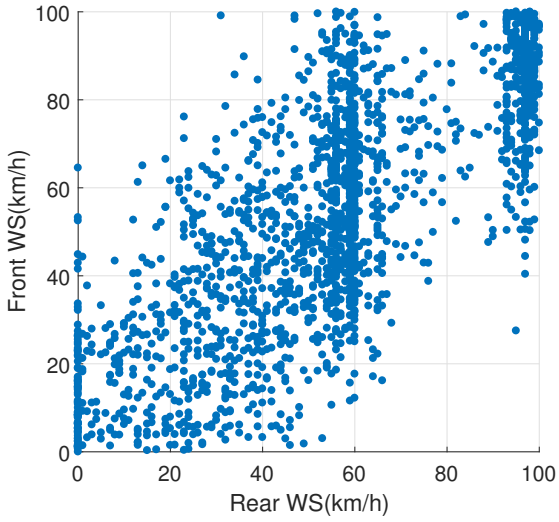
The correlation coefficient fluctuates less if more samples are used for computing ρ because the impact of an outlier is reduced. As a consequence, MIDS has more strict criteria (smaller σ_ρ) in detecting an anomaly in ρ if the larger window size is used. For ϵ=0.05 and n_{attack}=60, ϵ-DI reduces from 18 to 1.5 when the window size is increased from 750 to 1500 as shown in Figs. 4.10(a) and 4.10(b). For the Toyota Camry, Figs. 4.10(c) and 4.10(d), however, demonstrate that ϵ-DI increases from 1.05 to 1.10 as the window size increases from 750 to 1500. Even though MIDS has smaller σ_ρ when the window size is 1500, ρ drops more when the window size is 750,



(a) Scatter plot without attack

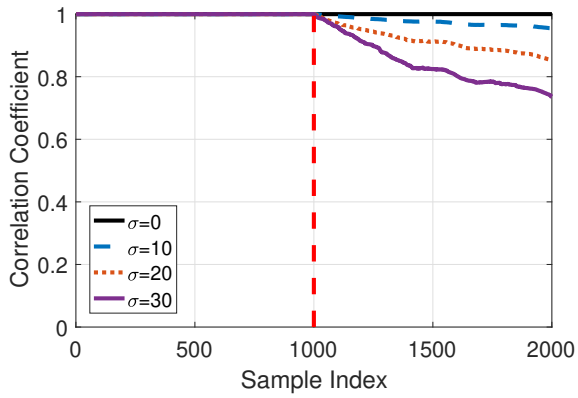


(b) Scatter plot for σ=10

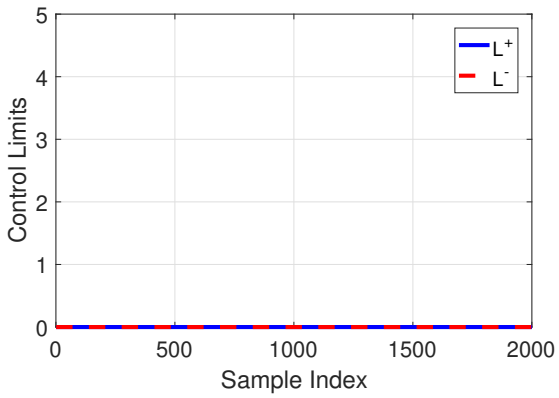


(c) Scatter plot for σ=20

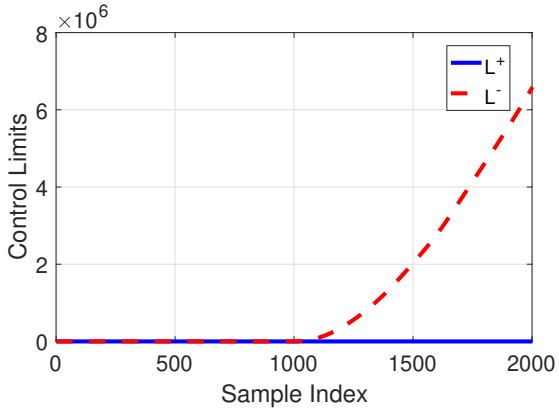
Figure 4.7: Scatter plots between the front and rear wheel speeds of the UW EcoCAR for various values of σ. When a disturbance is not added, the front wheel speed is almost the same as the rear wheel speed. As a bigger disturbance is added from σ=10 to σ=20, the front wheel speed deviates more from the rear wheel speed.



(a) Correlation coefficient for various values of σ



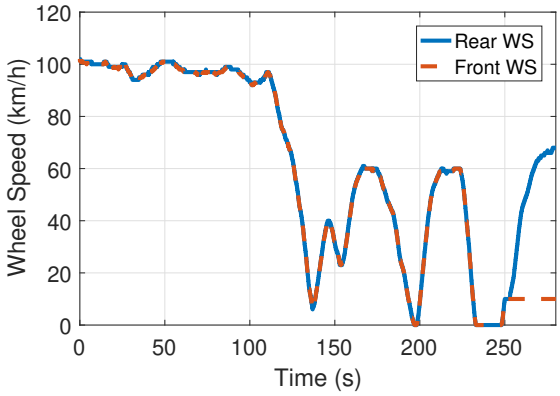
(b) Control limits when σ=1



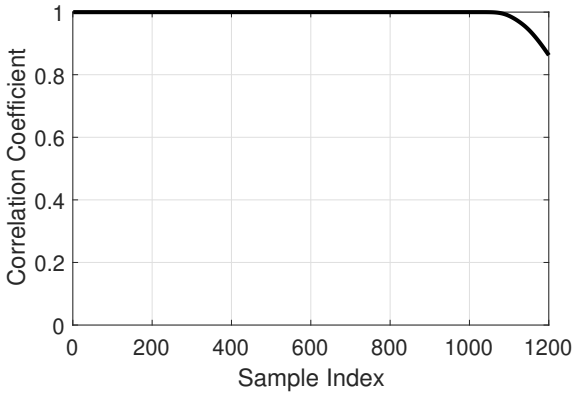
(c) Control limits when σ=20

Figure 4.8: Example of MIDS. Using the UW EcoCAR data, the correlation coefficient and control limits are demonstrated. The attack starts from the 1001st sample that is indicated by a red dashed line in (a). As a larger disturbance is added to the legitimate data, the correlation coefficient decreases more. When the added disturbance is small (i.e, σ=1), the front wheel speed less deviates from the rear wheel speed. As a result, the control limits are zero after the attack and MIDS cannot detect an attack. When the added disturbance is large (i.e., σ=20), the lower control limit increases after the attack occurs, and MIDS detects the attack.

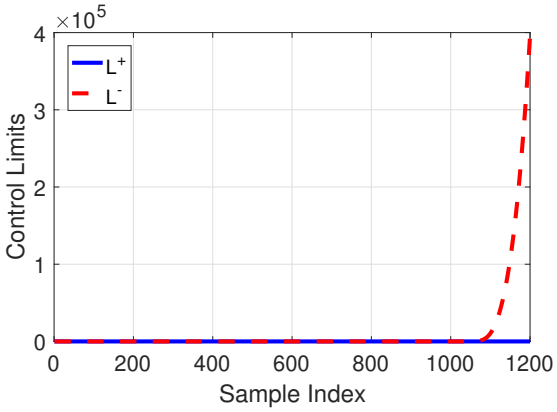
which dominates the impact of smaller σ_ρ. As a result, MIDS becomes more effective when the smaller window size is used in the Toyota Camry.



(a) Front and rear wheel speeds

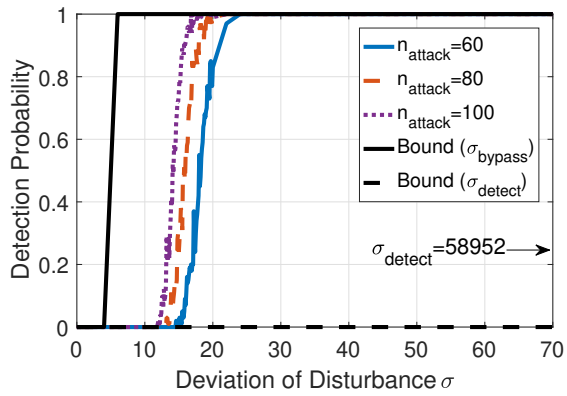


(b) Correlation coefficient

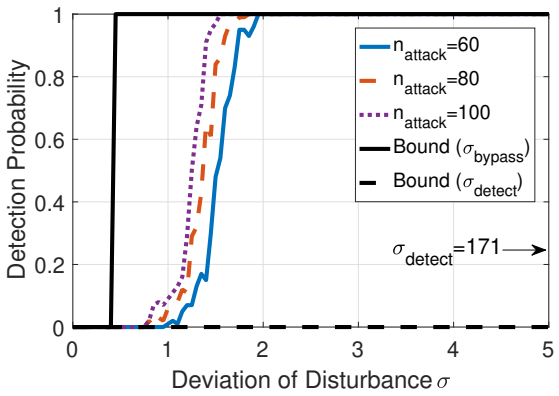


(c) Control limits

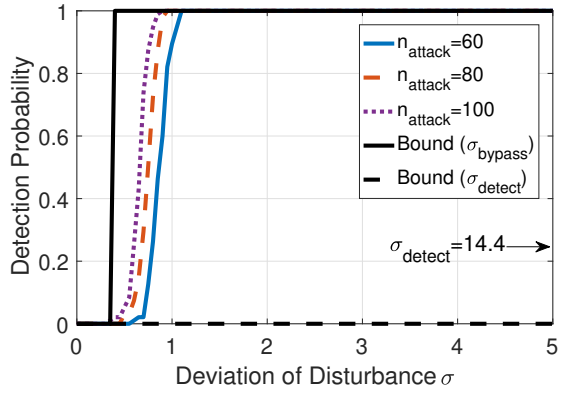
Figure 4.9: Example of MIDS under a data falsification attack. The adversary spoofs the rear wheel speed after 250s (after the sample index of 1000. An adversary spoofs the rear wheel speed between 250s and 270s (between the sample indexes of 1000 and 1200) in order to cause a wheelspin. As a result, the front wheel speed remains 10km/s whereas the rear wheel speed is manipulated to be increases. After the attack occurs, the correlation coefficient decreases, and MIDS detects the attack as the lower control limit increases higher than Γ=5.



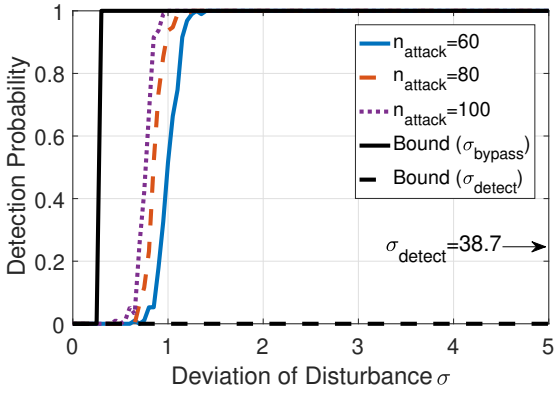
(a) Camaro on road with window size 750



(b) Camaro on road with window size 1500



(c) Toyota Camry with window size 750



(d) Toyota Camry with window size 1500

Figure 4.10: Detection probability of MIDS. The black solid line and black dashed line indicate σ_{bypass} and σ_{detect}, respectively. For UW EcoCAR, MIDS has more strict criteria (smaller σ_ρ) in detecting an anomaly in ρ if a larger window size is used. For Toyota Camry, ρ drops more when the window size is 750, which dominates the impact of smaller σ_ρ.

4.6 Conclusion

In this chapter, we investigated a limitation of the existing anomaly-based IDSs under a data falsification attack. We proposed a MIDS that exploits the correlation between messages. MIDS computes the correlation coefficients between two sensor measurements or control signals, and it

tracks sudden deviations in the correlation coefficients to detect the data falsification attack. We derived standard deviations of the disturbance, below which MIDS cannot detect an attack and above which MIDS always detects the attack. In order to quantify the effectiveness of MIDS, we presented the ϵ-DI that is a range of deviation of the disturbance that an adversary may introduce without being detected. We demonstrated that MIDS could detect the data falsification attack on the UW EcoCAR and Toyota Camry by using the wheel speeds. This work suggests that a defending mechanism exploiting the correlation between messages will increase security assurance to automobiles.

BIBLIOGRAPHY

[1] Hyundai Global Diagnostic System. https://hyundai.service-solutions.com/en-US/Pages/Home.aspx. Accessed: 2018-06-25.

[2] LittelFuse Fuses vs. PTCs. https://www.littelfuse.com/about-us/education-center/fuses-vs-ptcs.aspx/. Accessed: 2019-01-31.

[3] University of Tulsa Crash Reconstruction Research Consortium. http://tucrrc.utulsa.edu. Accessed: 2019-05-02.

[4] UW EcoCar. http://uwecocar.com/. Accessed: 2017-09-26.

[5] International Standard ISO 17458 Road Vehicles-FlexRay Communication System, Part 1 General Information and Use Case Definition, 2013.

[6] International Standard ISO 11898-1 Road Vehicles-Controller Area Network (CAN), Part 1 Data Link Layer and Physical Signaling, 2015.

[7] International Standard ISO 17987 Road Vehicles-Local Interconnect Network (LIN), Part 1 General Information and Use Case Definition, 2016.

[8] Arduino. Arduino UNO Rev3 Technical Specification. https://store.arduino.cc/usa/arduino-uno-rev3. Accessed: 2018-05-07.

[9] Atmel. The Atmel-ICE Debugger User Guide, 2016.

[10] Omid Avatefipour and Hafiz Malik. State-of-the-Art Survey on In-Vehicle Network Communication CAN-Bus Security and Vulnerabilities. *International Journal of Computer Science and Network*, pages 720–727, December 2017.

[11] Michèle Basseville and Igor Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice Hall Englewood Cliffs, 1993.

[12] Bosch. CAN Specification Version 2.0, 1991.

[13] California Air Resource Board. HD OBD Regulatory Documents, 2012.

[14] Curtis Carver, John Hill, John Surdu, and Udo Pooch. A Methodology for Using Intelligent Agents to Provide Automated Intrusion Response. In *IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, pages 110–116, 2000.

[15] 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 *USENIX Conference on Security*, pages 77–92, 2011.

[16] Yu-Shan Chiu, Kwan-Shi Chang, Robert Johnstone, and M. Parameswaran. Fuse-Tethers in MEMS: Theory and Operation. In *Canadian Conference on Electrical and Computer Engineering*, pages 1517–1520, May 2005.

[17] Kyong Cho and Kang Shin. Error Handling of In-vehicle Networks Makes Them Vulnerable. In *ACM Conference on Computer and Communications Security*, pages 1044–1055, 2016.

[18] Kyong Cho and Kang Shin. Fingerprinting Electronic Control Units for Vehicle Intrusion Detection. In *USENIX Conference on Security Symposium*, pages 911–927, 2016.

[19] Kyong Cho and Kang Shin. Viden: Attacker Identification on In-Vehicle Networks. In *ACM Conference on Computer and Communications Security*, pages 1109–1123, 2017.

[20] Wonsuk Choi, Kyungho Joo, Hyo Jin Jo, Moon Chan Park, and Dong Hoon Lee. VoltageIDS: Low-Level Communication Characteristics for Automotive Intrusion Detection System. *IEEE Transactions on Information Forensics and Security*, 13(8):2114–2129, 2018.

[21] Xuran Dinga, Wenzhong Loub, and Yue Feng. A Controllable IC-compatible Thin-Film Fuse Realized using Electro-Explosion. *AIP Advances*, 6, 2016.

[22] Ford. Ford Vehicle Communication Module. `https://www.fordtechservice.dealerconnection.com/VDIRS/wds/vcm_retail_renewal_Latest.asp`, June 2018. Accessed: 2018-06-25.

[23] Ian Foster, Andrew Prudhomme, Karl Koscher, and Stefan Savage. Fast and Vulnerable: A Story of Telematic Failures. In *USENIX Conference on Offensive Technologies*, 2015.

[24] Arun Ganesan, Jayanthi Rao, and Kang Shin. Exploiting Consistency Among Heterogeneous Sensors for Vehicle Anomaly Detection. In *SAE Technical Paper*, 2017.

[25] Simon Haykin. *Adaptive Filter Theory 2nd edition*. Prentice Hall, 2011.

[26] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Security Threats to Automotive CAN Networks - Practical Examples and Selected Short-Term Countermeasures. In *International Conference on Computer Safety, Reliability, and Security*, pages 235–248, 2008.

[27] Suman Jana and Sneha Kumar Kasera. On Fast and Accurate Detection of Unauthorized Wireless Access Points Using Clock Skews. In *ACM International Conference on Mobile Computing and Networking*, pages 104–115, 2008.

[28] Tadayoshi Kohno, Andre Broido, and Kimberly Claffy. Remote Physical Device Fingerprinting. In *IEEE Symposium on Security and Privacy*, pages 211–225, 2005.

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

[30] Alberto Leon-Garcia. *Probability, Statistics, and Random Processes For Electrical Engineering*. Pearson, 2011.

[31] LittelFuse. Fuse Characteristics, Terms and Condition Factors, 2014.

[32] LittelFuse. Radial Lead Fuses 272/273/274/278/279 Series Very Fast-Acting Fuses Datasheet, 2018.

[33] Microchip. ATmega 328P Automotive Datasheet, 2015.

[34] Microchip. MCP2515 Stand-Alone CAN Controller with SPI Interface, 2016.

[35] Microchip. MCP2551 CAN Transceiver Datasheet, 2016.

[36] Charlie Miller and Chris Valasek. Adventures in Automotive Networks and Control Units. In *DEF CON21*, 2013.

[37] Charlie Miller and Chris Valasek. A Survey of Remote Automotive Attack Surfaces. In *Black Hat USA*, 2014.

[38] Charlie Miller and Chris Valasek. Remote Exploitation of An Unaltered Passenger Vehicle. In *Black Hat USA*, 2015.

[39] David Mills. *Network Time Protocol (Version 3): Specification, Implementation and Analysis*, 1992.

[40] Sue Moon, Paul Skelly, and Don Towsley. Estimation and Removal of Clock Skew from Network Delay Measurements. In *18th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 227–234, March 1999.

[41] Pal-Stefan Murvay and Bogdan Groza. Source Identification Using Signal Characteristics in Controller Area Networks. *IEEE Signal Processing Letters*, 21(4):395–399, 2014.

[42] Michael Müter and Naim Asaj. Entropy-based Anomaly Detection for In-vehicle Networks. In *IEEE Intelligent Vehicles Symposium*, pages 1110–1115, 2011.

[43] NXP. MPC561/MPC563 Reference Manual, 2005.

[44] NXP. TJA1043 CAN Transceiver Datasheet, 2017.

[45] Renesas. V850/SA1 Application Note, 2000.

[46] Renesas. V850E2/FF4-G, 2014.

[47] Roland Rieke, Marc Seidemann, Elise Kengni Talla, Daniel Zelle, and Bernhard Seeger. Behavior Analysis for Safety and Security in Automotive Systems. In *Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pages 381–385, March 2017.

[48] Ross Tech. Volkswagen VAG-COM Diagnostic System. `http://www.ross-tech.com/vag-com/index.html`. Accessed: 2018-06-25.

[49] Richard Ruth, Wade Bartlett, and Jeremy Daily. Accuracy of Event Data in the 2010 and 2011 Toyota Camry during Steady State and Braking Conditions. *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, 5:358–372, 2012.

[50] Texas Instruments. Controller Area Network Physical Layer Requirements, 2008.

[51] Texas Instruments. Overview of 3.3V CAN (Controller Area Network) Transceiver, 2013.

[52] Texas Instruments. SN65HVD23x 3.3-V CAN Bus Transceivers, 2015.

[53] Texas Instruments. Am335x Sitara Processors Datasheet, 2016.

[54] Texas Instruments. Introduction to the Controller Area Network (CAN), 2016.

[55] The European Union. Directive 98/69/EC of the European Parliament and of the Council, 1998.

[56] Toyota. Toyota Technical Information System. https://techinfo.toyota.com/techInfoPortal, June 2018. Accessed: 2018-06-25.

[57] Sebastian Zander and Steven J. Murdoch. An Improved Clock-skew Measurement Technique for Revealing Hidden Services. In *Conference on Security Symposium*, pages 211–225, 2008.

Appendix A

DERIVATION OF σ_{BYPASS} AND σ_{DETECT}

This is a full derivation of σ_{bypass} in Theorem 1 and σ_{detect} in Theorem 2 in Chapter 4. Let us consider vectors $\bar{a}$ and $\bar{a'}$ as shown in Fig A, where w elements are used for computing the correlation coefficient. Fig A, where w elements are used for computing the correlation coefficient. We use the known result for computing correlation between two vectors $\bar{a}$ and $\bar{a'}$ as Eq.(4.1).

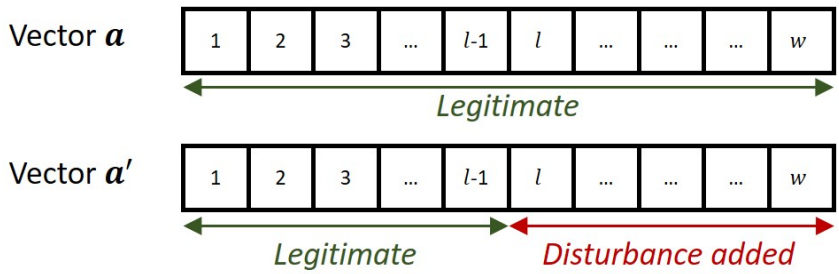


Figure A.1: Two vectors for correlation computation

The numerator of Eq.(4.1) can be simplified as follows.

$$\begin{aligned}
\sum_{i=1}^{w}(a_i-\mu_a)(a'_i-\mu_{a'}) &= \sum_{i=1}^{l-1}(a_i-\mu_a)(a'_i-\mu_{a'})+\sum_{i=l}^{w}(a_i-\mu_a)(a'_i-\mu_{a'}) \\
&= \sum_{i=1}^{l-1}(a_i-\mu_a)(a_i-\mu_a)+\sum_{i=l}^{w}(a_i-\mu_a)(a'_i-\mu_a) \\
&= \sum_{i=1}^{l-1}(a_i-\mu_a)(a_i-\mu_a)+\sum_{i=l}^{w}(a_i-\mu_a)(a_i+d_i-\mu_a) \\
&= \sum_{i=1}^{l-1}(a_i-\mu_a)(a_i-\mu_a)+\sum_{i=l}^{w}\{(a_i-\mu_a)(a_i-\mu_a)+(a_i-\mu_a)d_i\}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{l-1}(a_i-\mu_a)(a_i-\mu_a) + \sum_{i=l}^{w}(a_i-\mu_a)(a_i-\mu_a) + \sum_{i=l}^{w}(a_i-\mu_a)d_i \\
&= \sum_{i=1}^{w}(a_i-\mu_a)^2 + \sum_{i=l}^{w}(a_i-\mu_a)d_i \\
&= \sum_{i=1}^{w}(a_i-\mu_a)^2 + \sum_{i=l}^{w}a_i d_i - \sum_{i=l}^{w}\mu_a d_i \\
&= \sigma_a^2 + \sum_{i=l}^{w}a_i d_i.
\end{aligned}
$$

The last equation can be derived from the second last equation as $\sum_{i=l}^{w}\mu_a d_i = 0$ since d_i is a zero-mean Gaussian random variable.

The second term of the denominator in Eq.(4.1) can be simplified as follows.

$$
\begin{aligned}
\sqrt{\sum_{i=1}^{w}(a'_i-\mu_{a'})^2} &= \sqrt{\sum_{i=1}^{w}(a'_i-\mu_a)^2} \\
&= \sqrt{\sum_{i=1}^{l-1}(a_i-\mu_a)^2 + \sum_{i=l}^{w}(a'_i-\mu_a)^2} \\
&= \sqrt{\sum_{i=1}^{l-1}(a_i-\mu_a)^2 + \sum_{i=l}^{w}(a_i+d_i-\mu_a)^2} \\
&= \sqrt{\sum_{i=1}^{l-1}(a_i-\mu_a)^2 + \sum_{i=l}^{w}(a_i-\mu_a+d_i)^2} \\
&= \sqrt{\sum_{i=1}^{l-1}(a_i-\mu_a)^2 + \sum_{i=l}^{w}(a_i-\mu_a)^2 + 2\sum_{i=l}^{w}(a_i-\mu_a)d_i + \sum_{i=l}^{w}d_i^2} \\
&= \sqrt{\sum_{i=1}^{w}(a_i-\mu_a)^2 + 2\sum_{i=l}^{w}(a_i-\mu_a)d_i + \sum_{i=l}^{w}d_i^2} \\
&= \sqrt{\sum_{i=1}^{w}(a_i-\mu_a)^2 + 2\sum_{i=l}^{w}a_i d_i - 2\sum_{i=l}^{w}\mu_a d_i + \sum_{i=l}^{w}d_i^2} \\
&= \sqrt{\sigma_a^2 + 2\sum_{i=l}^{w}a_i d_i + \sum_{i=l}^{w}d_i^2}.
\end{aligned}
$$

We can write the correlation coefficient ρ as function of $\bar{a}$, $\bar{d}$, n, and l as follows.

$$\rho(\bar{a}, \bar{d}, w, l) = \frac{\sigma_a^2 + \sum_{i=l}^{w} a_i d_i}{\sigma_a \sqrt{\sigma_a^2 + 2\sum_{i=l}^{w} a_i d_i + \sum_{i=l}^{w} d_i^2}}. \tag{A.1}$$

We consider two extreme cases: (1) the disturbance is not added at all, and (2) the disturbance is added to all elements. For the first case, all the summation terms are zero. Hence, $\rho(\bar{a}, \bar{d}, w, l)$ becomes:

$$\rho(\bar{a}, \bar{d}, w, l) = \frac{\sigma_a^2}{\sigma_a \sqrt{\sigma_a^2}} = \frac{\sigma_a^2}{\sigma_a^2} = 1.$$

Since the correlation between two identical vectors is 1, the derivation is correct for the first case. For the second case, $l = 1$ and we can write ρ as follows. Since

1. $d_i \sim N(0, \sigma^2)$ which indicates $\sum_{i=1}^{w} d_i = 0$.

2. d_i and a_i are independent.

Then, $\sum_{i=l}^{w} a_i d_i$ can be written as follows.

$$E[\bar{a}\bar{d}] = \frac{1}{w}\sum_{i=1}^{w} a_i d_i \rightarrow \sum_{i=l}^{w} a_i d_i = wE[\bar{a}\bar{d}]$$

$$E[\bar{a}]E[\bar{d}] = \left(\frac{1}{w}\sum_{i=1}^{w} a_i\right)\left(\frac{1}{w}\sum_{i=1}^{w} d_i\right) \rightarrow \sum_{i=1}^{w} a_i \sum_{i=1}^{w} d_i = w^2 E[\bar{a}]E[\bar{d}]$$

Hence,

$$\sum_{i=1}^{w} a_i d_i = \frac{1}{w}\sum_{i=1}^{w} a_i \sum_{i=1}^{w} d_i \tag{A.2}$$

We plug in Eq.(A.2) to Eq.(A.1).

$$\begin{aligned}
\rho(\bar{a}, \bar{d}, w, l) &= \frac{\sigma_a^2 + \frac{1}{w}\sum_{i=1}^{w} a_i \sum_{i=1}^{w} d_i}{\sigma_a \sqrt{\sigma_a^2 + 2\sum_{i=1}^{w} a_i \sum_{i=1}^{w} d_i + \sum_{i=1}^{w} d_i^2}} \\
&= \frac{\sigma_a^2}{\sigma_a \sqrt{\sigma_a^2 + \sum_{i=1}^{w} d_i^2}} \\
&= \frac{\sigma_a^2}{\sigma_a \sqrt{\sigma_a^2 + \sigma^2}},
\end{aligned}$$

since a_i and d_i are independent and $d_i \sim N(0, \sigma^2)$. In order to check the correctness of this results, we use another definition of correlation coefficient which is:

$$\rho = \frac{COV(\bar{a}, \bar{a'})}{\sigma_a \sigma'_a} = \frac{E[(\bar{a} - \mu_a)(\bar{a'} - \mu'_a)]}{\sigma_a \sigma'_a}. \tag{A.3}$$

Since $E[a'] = E[a]$ and $\sigma'^2_a = \sigma^2_a + \sigma^2$, the numerator of Eq.(A.3) can be written as follows.

$$\begin{aligned}
E[(\bar{a} - \mu_a)(\bar{a'} - \mu_{a'})] &= E[\bar{a}\bar{a'} - \mu_a\bar{a'} - \mu_a\bar{a} + \mu_a^2] \\
&= E[\bar{a}\bar{a'}] - \mu_a E[\bar{a'}] - \mu_a E[\bar{a}] + \mu_a^2 \\
&= E[\bar{a}\bar{a'}] - \mu_a^2 \\
&= E[\bar{a}(\bar{a} + \bar{d})] - \mu_a^2 \\
&= E[\bar{a}^2] - \mu_a^2 + E[\bar{a}]E[\bar{d}] \\
&= \sigma_a^2.
\end{aligned}$$

Then, using Eq.(A.3), the correlation coefficient becomes:

$$\rho = \frac{\sigma_a^2}{\sigma_a \sqrt{\sigma_a^2 + \sigma^2}},$$

which exactly matches with our theoretical derivation.

We compute theoretically the bounds of standard deviation of the disturbance that always bypasses the MIDS and is detected by the MIDS. For this analysis, we make the following assumptions.

1. The correlation coefficient linearly decreases over disturbed samples.

2. μ_ρ and σ_ρ are not updated during attack.

The correlation coefficient decreases as more number of disturbed samples are in the window (i.e., increasing k), and it converges to $\frac{\sigma_a}{\sqrt{\sigma_a^2 + \sigma^2}}$ when all the correlation coefficient samples in the window are disturbed. Since the correlation coefficient is assumed to decrease linearly, the decrement of the correlation coefficient per sample for the upper bound, denoted as Δ_{bypass}, can be written as follows.

$$\Delta_{bypass} = \frac{\rho_0 - \frac{\sigma_a}{\sqrt{\sigma_a^2 + \sigma^2}}}{w}. \tag{A.4}$$

Then, the normalized correlation coefficient becomes as follows.

$$\begin{aligned}\rho'(k-w+1) &= \frac{\rho_0 - (k-n_{normal})\Delta_{bypass} - \mu_\rho}{\sigma_\rho} \\ &= \frac{\rho_0 - \mu_\rho}{\sigma_\rho} - \frac{(k-n_{normal})\Delta_{bypass}}{\sigma_\rho} \\ &= \rho'_0 - \frac{(k-n_{normal})\Delta_{bypass}}{\sigma_\rho} \\ &= -\frac{(k-n_{normal})\Delta_{bypass}}{\sigma_\rho},\end{aligned} \tag{A.5}$$

since $\rho_0 \sim \mu_\rho$. The lower and upper control limits in the CUSUM detector, denoted as L^- and L^+, are updated as $L^- = \max[0, L^- - \rho' - \kappa]$ and $L^+ = \max[0, L^+ + \rho' - \kappa]$. L^+ captures the abnormal increment of the correlation coefficient while the abnormal decrement is tracked by L^-. Since the correlation coefficient under attack is smaller than that without attack, L^- increases under attack. Both control limits remain zero if $\frac{(k-n_{normal})\Delta_{bypass}}{\sigma_\rho} < \kappa$ for all $k \leq n_{normal} + w$. That is, all elements in the window are disturbed. We set $k = n_{normal} + w$ because we find the minimum standard deviation of the disturbance, denoted as σ_{bypass}. σ_{bypass} can be derived as follows using Eq.(A.4).

$$\begin{aligned}\frac{\rho_0 - \frac{\sigma_a}{\sqrt{\sigma_a^2 + {\sigma_{bypass}}^2}}}{w} &= \frac{\kappa\sigma_\rho}{w} \\ \rho_0 - \frac{\sigma_a}{\sqrt{\sigma_a^2 + {\sigma_{bypass}}^2}} &= \kappa\sigma_\rho \\ \frac{\sigma_a}{\sqrt{\sigma_a^2 + {\sigma_{bypass}}^2}} &= \rho_0 - \kappa\sigma_\rho \\ \frac{\sigma_a^2}{\sigma_a^2 + {\sigma_{bypass}}^2} &= (\rho_0 - \kappa\sigma_\rho)^2 \\ \frac{\sigma_a^2 + {\sigma_{bypass}}^2}{\sigma_a^2} &= \frac{1}{(\rho_0 - \kappa\sigma_\rho)^2} \\ 1 + \frac{{\sigma_{bypass}}^2}{\sigma_a^2} &= \frac{1}{(\rho_0 - \kappa\sigma_\rho)^2}\end{aligned}$$

$$\frac{{\sigma_{bypass}}^2}{\sigma_a^2} = \frac{1}{(\rho_0 - \kappa\sigma_\rho)^2} - 1$$
$$\frac{\sigma_{bypass}}{\sigma_a} = \sqrt{\frac{1}{(\rho_0 - \kappa\sigma_\rho)^2} - 1}$$
$$\sigma_{bypass} = \sigma_a \sqrt{\frac{1}{(\rho_0 - \kappa\sigma_\rho)^2} - 1}.$$

Any disturbance whose standard deviation less than σ_{bypass} is guaranteed to be not detected by MIDS (i.e., both control limits are always zero.).

We find the minimum standard deviation of disturbance, denoted as σ_{detect}, that guarantees 100 percent detection probability. Let ρ_{attack} denote the first correlation coefficient right after the attack. Then, let us find the condition of ρ_{attack} such that the attack is detected at the first attack sample. Since $\rho_{attack} < \rho_0$, L^- will increase. L^- right after the attack is updated as follows.

$$L^- \leftarrow \max\left[0, L^- - \frac{\rho_{attack} - \mu_\rho}{\sigma_\rho} - \kappa\right]. \tag{A.6}$$

Then, the condition of ρ_{attack} can be derived as follows.

$$\begin{aligned}
0 - \frac{\rho_{attack} - \mu_\rho}{\sigma_\rho} - \kappa &\geq \Gamma \\
-\frac{\rho_{attack} - \mu_\rho}{\sigma_\rho} - \kappa &\geq \Gamma \\
-\frac{\rho_{attack} - \mu_\rho}{\sigma_\rho} &\geq \Gamma + \kappa \\
-\left(\rho_{attack} - \mu_\rho\right) &\geq \sigma_\rho\left(\Gamma + \kappa\right) \\
\rho_{attack} - \mu_\rho &\leq -\sigma_\rho\left(\Gamma + \kappa\right) \\
\rho_{attack} &\leq \mu_\rho - \sigma_\rho\left(\Gamma + \kappa\right).
\end{aligned} \tag{A.7}$$

Then, let us define the decrement in the correlation coefficient from ρ_0 to ρ_{attack} as Δ_{detect} which is $\sigma_\rho(\Gamma + \kappa)$. With this decrement of Δ_{detect}, let us assume that the correlation coefficient becomes zero at the j-th attack sample. Then, j can be derived as follows.

$$j = \left\lceil \frac{\rho_0}{\rho_0 - \rho_{attack}} \right\rceil, \tag{A.8}$$

where $\lceil\ \rceil$ indicates the ceiling operation.

Then, we can find σ_{detect} as follows.

$$
\begin{aligned}
\frac{\rho_0 - \frac{\sigma_a}{\sqrt{\sigma_a^2 + {\sigma_{detect}}^2}}}{j} &= \sigma_\rho(\Gamma + \kappa) \\
\rho_0 - \frac{\sigma_a}{\sqrt{\sigma_a^2 + {\sigma_{detect}}^2}} &= j\sigma_\rho(\Gamma + \kappa) \\
\frac{\sigma_a}{\sqrt{\sigma_a^2 + {\sigma_{detect}}^2}} &= \rho_0 - j\sigma_\rho(\Gamma + \kappa) \\
\frac{\sigma_a^2}{\sigma_a^2 + {\sigma_{detect}}^2} &= (\rho_0 - j\sigma_\rho(\Gamma + \kappa))^2 \\
\frac{\sigma_a^2 + {\sigma_{detect}}^2}{\sigma_a^2} &= \frac{1}{(\rho_0 - j\sigma_\rho(\Gamma + \kappa))^2} \\
1 + \frac{{\sigma_{detect}}^2}{\sigma_a^2} &= \frac{1}{(\rho_0 - j\sigma_\rho(\Gamma + \kappa))^2} \\
\frac{{\sigma_{detect}}^2}{\sigma_a^2} &= \frac{1}{(\rho_0 - j\sigma_\rho(\Gamma + \kappa))^2} - 1 \\
\frac{\sigma_{detect}}{\sigma_a} &= \sqrt{\frac{1}{(\rho_0 - j\sigma_\rho(\Gamma + \kappa))^2} - 1} \\
\sigma_{detect} &= \sigma_a \sqrt{\frac{1}{(\rho_0 - j\sigma_\rho(\Gamma + \kappa))^2} - 1}.
\end{aligned}
\tag{A.9}
$$

Appendix B

PUBLICATIONS

1. **S. Sagong**, X. Ying, A. Clark, L. Bushnell, and R. Poovendran, "Cloaking the Clock: Emulating Clock Skew in Controller Area Networks," in Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS) 2018.

2. X. Ying, **S. Sagong**, A. Clark, L. Bushnell, and R. Poovendran, "Shape of the Cloak: Formal Analysis of Clock Skew-Based Intrusion Detection System in Controller Area Networks," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 9, pp. 2300-2314, 2019.

3. **S. Sagong**, X. Ying, R. Poovendran and L. Bushnell, "Exploring Attack Surfaces of Voltage-Based Intrusion Detection Systems in Controller Area Networks," in Proceedings of the 16th Embedded Security in Cars (ESCAR) Europe 2018.

4. **S. Sagong**, R. Poovendran, and L. Bushnell, "Mitigating Vulnerabilities of Voltage-based Intrusion Detection Systems in Controller Area Networks," in ArXiv (eprint no. 1907.10783), 2019.

5. **S. Sagong**, R. Poovendran, and L. Bushnell, "Inter-Message Correlation for Intrusion Detection in Controller Area Networks," in Proceedings of the 17th Embedded Security in Cars (ESCAR) Europe 2019.

VITA

Sang Uk Sagong was born in Korea, 1986. He received his M.S. and B.S. in Electrical Engineering from Yonsei University, Korea, in 2011 and 2009, respectively. His research interests include the standard of the in-vehicle communication protocol and security of the automobile. He explores the vulnerabilities of the CAN protocol by demonstrating cyber attacks on the CAN testbed and real vehicles. He also develops hardware and software-based defense mechanisms for the CAN protocol.