Doctoral Dissertation

Offensive and Defensive Strategies on In-Vehicle Networks: Toward Disabling DoS Vulnerability

Shuji Ohira
Program of Information Science and Engineering
Graduate School of Science and Technology
Nara Institute of Science and Technology

Supervisor: Prof. Kazutoshi Fujikawa
Internet Architecture and Systems Lab. (Division of Information Science)

Submitted on March 1, 2023

A Doctoral Dissertation
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of Engineering

Shuji Ohira

Thesis Committee:

Supervisor	Kazutoshi Fujikawa
	(Professor, Division of Information Science)
	Yuichi Hayashi
	(Professor, Division of Information Science)
	Youki Kadobayashi
	(Professor, Division of Information Science)
	Ismail Arai
	(Associate Professor, Division of Information Science)

Offensive and Defensive Strategies on In-Vehicle Networks: Toward Disabling DoS Vulnerability*

Shuji Ohira

Abstract

Due to the increase in the number of automobiles that connect to the Internet, cyberattack on Controller Area Network (CAN) is becoming a severe problem. CAN is one of the in-vehicle network protocols for communicating among Electronic Control Units (ECUs) and it is a de-facto standard of in-vehicle networks. Some security researchers point out several vulnerabilities in CAN such as Denial-of-Service (DoS) attacks and no identifiability of sender. To deal with these vulnerabilities, Intrusion Detection Systems (IDSs) and authentication mechanisms on CAN-bus have been proposed. However, the IDSs focus on detecting spoofing, replaying, and DoS attacks, so that IDSs do not provide protection against these attacks. In addition, a state-of-the-art IDS for DoS attack detection is probably only effective against DoS attacks under naive conditions such as some high-priority messages. It means that an attacker probably evades the IDS by manipulating feature IDS used. On the other hand, the authentication mechanisms focus on protecting against spoofing and replaying attacks. For these reasons, existing IDSs and authentication mechanisms cannot disable DoS attacks on CAN. Thus, to permanently disable DoS attacks on CAN, this dissertation studies the threat of DoS attacks from two different points: *offense* and *defense*.

Our main contributions consist of unveiling a new evasion attack and proposing three defensive strategies (*detection*, *identification*, and *protection*) for disabling DoS attacks including our evasion attacks. First, we derive a new evasion

*Doctoral Dissertation, Graduate School of Science and Technology, Nara Institute of Science and Technology, March 1, 2023.

attack called *entropy-manipulation attack* which is not detectable with a state-of-the-art IDS based on entropy of CAN. To address the evasion attack against the IDS, our similarity-based IDS detects DoS attacks including our evasion attacks using sliding window optimized similarity. Next, we propose a sender identification method based on physical-layer characteristics, called Physical-Layer Identification (PLI). To apply a security patch to the compromised ECU attacking the CAN, PLI accurately identifies the compromised ECU. Our IDS and PLI can detect DoS attacks and identify the compromised ECU that sends DoS attacks however IDS and PLI do not provide protection. Therefore, finally, we introduce IVNProtect which provides isolable and deployable CAN-bus kernel-level protection. Three defensive strategies are evaluated in a CAN-bus prototype and a real-vehicle. And the experimental results show that defensive strategies can successfully handle the attacks in the environments.

Keywords:

Automotive Security, Controller Area Network, Denial-of-Service Attack, Entropy, Sender Identification, Machine Learning

Contents

List of Figures

List of Tables

1. Introduction

Vehicles are currently undergoing a paradigm shift to connect to the Internet. Due to the paradigm shift, vehicles can perform autonomous driving, ride-sharing services, and Over-The-Air (OTA) updating. Unfortunately, the paradigm shift also brings to break down the cybersecurity of vehicles by hacking from remote hackers. Therefore, countermeasures against cybersecurity for vehicles are imperative to secure the current automotive society. This dissertation researches effective and defensive strategies to detect attacks, identify the causes of attacks, and protect vehicles from attacks.

1.1 Security Threats on Automotive IoT

Various vehicles such as automobiles and route buses are connected to the Internet (Fig. 1), and new concepts of services such as carpooling and ride-sharing were born. Due to these influences, Mobility as a Service (MaaS) attracts attention. MaaS enables users to perform useful operations such as route search, reservation, and payment at once on smartphones. Moreover, it is expected to solve traffic congestion and environmental problems in urban areas and to provide transportation for mobility-impaired people in rural areas. Also, we can collect transportation big-data (e.g., vehicle speed, engine speed, and location) from vehicles connected to the Internet. Utilizing transportation big-data collected from vehicles, we can obtain curated data such as accurate bus arrival time and ridership prediction. In this dissertation, vehicles connected to the Internet are defined as *Automotive IoT*.

While the Automotive IoT is increasing due to the aforementioned convenience, there are security threats on Controller Area Network (CAN) [28], which is a network inside vehicles. CAN is an in-vehicle network protocol used for communication between Electronic Control Units (ECUs) and has become a defacto standard of in-vehicle networks. Miller and Valasek successfully controlled a variety of automotive functions and, through In-Vehicle Infotainment system (IVI), exploited the vulnerabilities of CAN [62]. As a result, 1.4 million automobiles were recalled because of this vulnerability. In 2016, Nie *et al.* exploited the respective vulnerabilities of in-vehicle system browsers and CAN to demon-

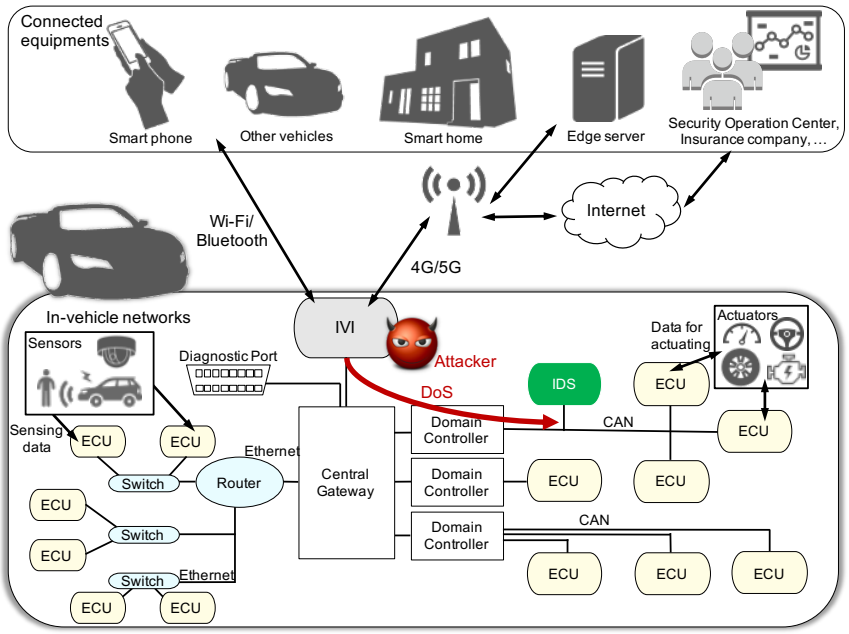


Figure 1: Modern in-vehicle network architecture and our research target attacker.

strate that various functions of the vehicle can be controlled [71]. These attacks use CAN vulnerabilities. CAN-bus is simple and has several vulnerabilities (e.g., Denial-of-Service (DoS) attacks, no identifiability of sender). Thus, once an ECU with entry points to external networks (e.g. IVI) is exploited, an attacker can inject malicious CAN messages from the exploited ECU.

On the other hand, vehicles can also use Vehicular Ad-hoc NETworks (VANETs) (e.g., Vehicle-to-Vehicle and Vehicle-to-Infrastructure) to support drivers and autonomous functions with the data of neighbor situations through Telematics Control Unit (TCU) [20]. VANETs communications enable vehicles to enhance safety with crash avoidance and remote diagnosis. However, if the interface of VANETs has a vulnerability, an attacker can intrude in-vehicle networks through TCU as with the IVI.

1.2 State-of-the-Art Defenses on CAN

To address the attacker/threat described in Sec. 1.1, many researchers have proposed two defensive strategies: message authentication and intrusion detection. Here, we explain these defenses and point out their problems.

1.2.1 Message Authentication

Message authentication has been proposed to prevent spoofing and replaying attacks on CAN [81, 92, 101]. Message authentication uses a hash algorithm to guarantee the authenticity of messages. In detail, a receiver ECU can verify the authenticity of a message by using a hash algorithm with a pre-shared key which is shared with a legitimate sender ECU. Message authentication can disable traditional hacking, in which an attacker sends a spoofed message to a specific ECU. However, even in authenticated CAN-bus, message authentication cannot disable DoS attacks that send many messages to the in-vehicle network, because DoS attacks can delay the authenticated benign messages. The delay caused by DoS attacks is a severe threat to CAN because even only exceeding 30 % bus utilization causes unacceptable delays for some low-priority messages [13].

Moreover, message authentication requires additional hardware and the extension of source codes to authenticate CAN messages. Besides, since the message authentication must manage the key lifecycle based on Public-Key Infrastructure (PKI), the key-management increases the complexity of the automotive system.

1.2.2 Intrusion Detection

Intrusion Detection Systems (IDSs) have an advantage in terms of applicability and implementation cost, unlike message authentication. One such case is IDSs based on characteristics of CAN messages (e.g., frequency [88], entropy [60, 98], ID sequence [59], physical-layer fingerprint [44, 46, 78]). The IDSs can be easily adapted for the CAN bus of modern automobiles and have high detection accuracy against spoofing, replaying, and DoS attacks. However, the IDSs provide no protection for spoofing, replaying, and DoS attacks. Therefore, existing defenses (i.e., message authentication and intrusion detection) cannot prevent DoS attacks. It is necessary to build a protection mechanism that has features from DoS attack

detection to defense.

In addition, these IDSs may have higher false positives for undiscovered evasion attacks which are launched by an attacker to manipulate the result of a model with crafted input data [22]. For instance, physical-layer fingerprint-based IDS cannot detect a spoofing attack which mimics the physical-layer characteristics [7]. Therefore, it is necessary to investigate the feasibility of undiscovered evasion attacks against state-of-the-art IDS.

1.3 Problems

As countermeasures against attacks on CAN, many researchers and industry developers have studied message authentications and IDSs. However, existing message authentications and IDSs cannot disable DoS attacks on CAN. In addition, if these countermeasures have a vulnerability such as evasion attacks, an attacker can perform attacks on numerous vehicles at a large scale. Thus, at first, it is essential to inspect whether a state-of-the-art DoS IDS has a vulnerability such as evasion attacks. Next, we derive defensive strategies to solve the failure of existing countermeasures.

1.3.1 An Undiscovered DoS Vulnerability of Entropy-Based IDS

After compromising an ECU, an attacker can inject the large number of messages to the CAN bus, which is called DoS attacks. The attacker also exploits an arbitration scheme which prioritizes the lower arbitration ID than the higher one. In details, the attacker floods the targeted CAN bus with the highest priority ID (e.g. `0x000`). This attacks causes unexpected alerts of vehicles, violations of the constraints for the arrival time of CAN messages and so on. Therefore, it is required to detect and prevent the DoS attacks. As the state-of-the-art IDS for DoS attacks, entropy-based IDSs have been proposed. The entropy-based IDSs have fast detection capability, adaptation for various CAN baud rate, and low hardware requirements. However, the question of whether the entropy-based IDSs can detect DoS attacks when an attacker manipulates entropy has not been clarified yet. To address this question, we find a way bypassing the entropy-based IDSs from the attacker's perspective.

1.3.2 Drawbacks of Intrusion Detection, Physical-Layer Identification, and CAN-Bus Protection

We unveiled an evasion attack against the entropy-based IDS on CAN. It means that the entropy, which has been claimed to be an effective characteristic for the detection of DoS attacks, actually allows DoS attacks on CAN. To end the arms race between evasion attacks and existing countermeasures, we propose three defense strategies to prevent DoS attacks including our evasion attack on CAN.

Evasion Attacks on State-of-the-Art DoS IDS

To detect DoS attacks on CAN, the entropy-based IDS has been proposed. We discover that the entropy-based IDS has a vulnerability against a DoS attack in our research. To solve this vulnerability accompanied by inheriting the good characteristics of entropy-based IDS, we derive a new characteristic of similarity for detection of DoS attacks.

Misclassification of Delay-Time Based Physical-Layer Identification

An attacker tries to evade IDSs with adversarial attacks which manipulate some features such as entropy, payload, and so on. Hence, identifying the attacker using immutable physical-layer characteristics is an effective countermeasure because a remote attacker cannot manipulate physical-layer characteristics. As one of these countermeasures, Physical-Layer Identification (PLI) has been studied. The idea is to use physical-layer characteristics (e.g., voltage, delay-time) of analog signal of one frame to identify the sender ECUs of the frame. As the one of the PLIs, delay-time based PLI have been proposed which is called Divider. Divider uses the difference of delay of CAN transceiver. The delay is caused by output/input/parasitic capacitance. Divider has the low number of samplings, but a low identification rate. Thus, we improve the identification accuracy of Divider.

Limitation of Deployability for CAN-Bus Protection Dedicated to DoS

To prevent DoS attacks on CAN, some researchers have proposed protective mechanisms. The allowlist-based mitigation method has been proposed as one of the countermeasures. This method can disable DoS attacks with the

highest priority CAN ID:0x000; in other words, it passes some allowlisted messages permanently. Hence, it is ideal for mitigating and isolating a DoS attacker who uses both malicious IDs (e.g. the highest priority CAN ID:0x000) and benign IDs (e.g. a CAN ID which a legitimate ECU sends). Moreover, this method requires additional hardware; therefore, it also has a drawback in deployability. Hence, we derive an isolable and deployable CAN-bus protection to prevent DoS attacks on CAN.

1.4 Dissertation Contributions

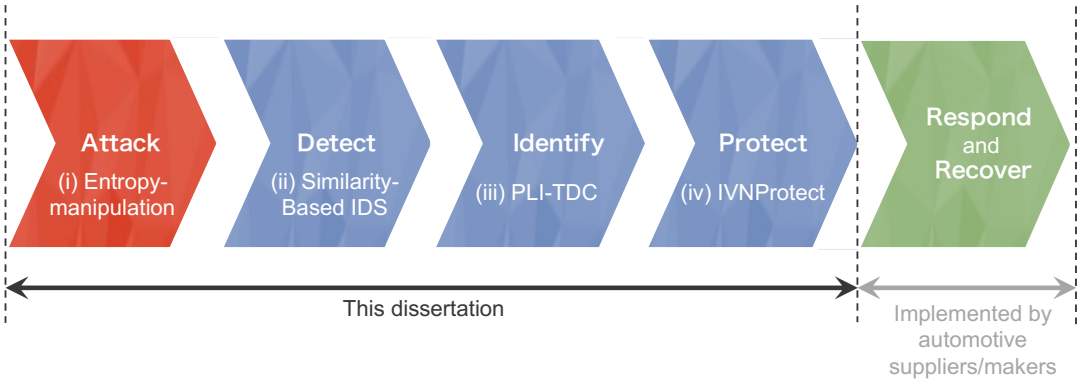
1.4.1 Motivation and Approach

Security researchers confirm that DoS attacking CAN causes unexpected vehicle behavior (e.g., disabling power steering, blocking Advanced Driver-Assistance Systems (ADAS) function). Therefore, it is important to completely eliminate the threat of DoS attacks on CAN. To permanently disable DoS attacks on CAN, we study the threat of DoS attacks from two different perspectives: *offense* and *defense*. Namely, we consider exploiting vulnerabilities for evasion attacks on state-of-the-art DoS IDS from the attacker's perspective. After that, we consider three defensive approaches (*detection*, *identification*, and *protection*) as depicted in Fig. 2, which can solve the problem of state-of-the-art DoS IDS.

1.4.2 Dissertation Goal and Scope

In this dissertation, we aim to disable DoS attacks from an attacker which is based on the actual hacking of vehicles, defined by Chap. 2.4. After achieving the dissertation goal, the security threat of spoofing and replay attacks on CAN is an unsolved problem. However, we can tackle the problem by applying the existing message authentications to CAN. Therefore, we define our goal that we propose defense strategies that can prevent all attacks by combining the existing message authentications. For this reason, we focus on defense strategies to disable all DoS attacks because the existing message authentications can deal with spoofing and replay attacks other than DoS attacks.

To achieve the goal, first, we explore vulnerabilities in state-of-the-art DoS IDS to verify the capabilities of detection. Second, we derive a new detection



To disable DoS attacks on CAN, we envision four stages: (i) derive an undiscovered DoS attack on a state-of-the-art IDS, (ii) detect the undiscovered and previous DoS attacks effectively, (iii) identify the compromised ECU which sends the DoS attacks, (iv) protect the CAN-bus from DoS attacks.

Figure 2: Four dissertation components and defense flow.

approach which detects existing DoS attacks and evasion attacks on the state-of-the-art DoS IDS. Third, we propose a PLI to accurately identify the sender ECU of malicious messages. Finally, we develop a protection mechanism to prevent DoS attacks in the software layer of ECU.

Here, we summarize the contributions of this dissertation.

1.4.3 *entropy-manipulation attack*: Evasion Attack on Entropy-Based IDS

The main contributions of the study of Chap. 3 can be summarized as follows.

1. We found a DoS attack called the *entropy-manipulation attack*, which bypasses the state-of-the-art DoS IDS (entropy-based IDS [98]) by adjusting the entropy of messages. These consist of messages of random CAN ID.

2. We propose an injection technique to evade the detection of entropy-based IDS at boundary between benign messages and DoS messages, which is called *Sliding Window Poisoning Tactic*.

3. We confirmed that an attacker can evade the entropy-based IDS with the entropy-manipulation attack in 6 vehicle environments.

1.4.4 Sliding Window Optimized Similarity Analysis Method

The main contributions of the study of Chap. 4 can be summarized as follows.

1. The proposed method (similarity-based IDS) achieved a detection precision of 100.0% against the all DoS attacks (including entropy-manipulation attack), while the detection precision is 68.3% in the entropy-based IDS.

2. We showed that the detection time is up to 93% (14 µs) shorter than the entropy-based IDS.

1.4.5 PLI-TDC: Physical-Layer Identification with Time-to-Digital Converter for In-Vehicle Networks

The main contributions of the study of Chap. 5 can be summarized as follows:

1. We propose a PLI using Time-to-Digital Converter (TDC) which can fine measure an elapsed time, called PLI-TDC. Our method uses new characteristics in the identification of ECUs in CAN. PLI-TDC does not use continuous characteristics such as voltage, but the delay-time to be observed in each rising edge of the CAN message. Hence, PLI-TDC can identify the ECUs with a lower number of sampling than the voltage-domain based method.

2. PLI-TDC achieved mean accuracy of 99.67 % and 97.04 % on CAN bus prototype and a real-vehicle network, respectively. Additionally, we showed that this approach achieved a 100% true positive rate against two attacker models.

3. We designed PLI-TDC so that it can be robust against features' drift caused by temperature drift. From our experiment, PLI-TDC can achieve a mean accuracy of over 99% even if the temperature is drifted.

4. PLI-TDC solved the problems of detection based on multiple frames, number of probes, and robustness against feature drift.

1.4.6 IVNProtect: Isolable and Traceable Lightweight CAN-Bus Kernel Protection

The main contributions of the study of Chap. 6 can be summarized as follows:

1. We propose an isolable and traceable lightweight CAN-bus protection called IVNProtect. IVNProtect is implemented to a CAN-bus kernel driver on Linux. Hence, IVNProtect can deploy to an ECU just by software updating.

2. We provide a new error state mechanism on IVNProtect for handling security incidents. IVNProtect mitigates DoS attacks and isolates a compromised ECU based on this security error state mechanism.

3. We experimentally confirm the traceability that IVNProtect reports warning messages for legitimate ECUs to distinguish whether the cause of isolation is a fault or a security incident.

4. We show the overhead caused by IVNProtect. As a result, the overhead takes only 9.049 µs, which means that a system with our IVNProtect satisfies in-vehicle real-time demands.

1.5 Outline

The rest of the dissertation is organized as follows. Chap. 2 explains all the relevant backgrounds on CAN and its security issues. In Chaps. 3 and 4, we detail a new DoS attack called *entropy-manipulation attack* and a defensive strategy against it called *similarity-based IDS*, respectively. Next, aiming to end the arms race between evasion attacks like *entropy-manipulation attacks* and defensive strategies like *similarity-based IDS*, we propose physical-layer characteristics-based sender identification and attacker detection called *PLI-TDC* in Chap. 5. In Chap. 6, we propose a CAN-bus kernel-level protection called IVNProtect. Finally, Chap. 7 details the impact of this research on other related fields and elaborates the future research direction.

2. Controller Area Network and Its Vulnerabilities

In this section, we describe the CAN which is a widely utilized in-vehicle network protocol in modern automobiles. We also summarize the vulnerabilities of CAN.

2.1 CAN Primer

CAN is the de facto standard for in-vehicle networks, and generally works on a bus-type network topology as shown in Fig. 3. Normally, both ends of the CAN bus are terminated with 120 Ω resistors to prevent signal reflection. Therefore, the value of the combined resistance of CAN is 60 Ω.

Typical CAN node consists of Micro Controller Unit (MCU), CAN controller, and CAN transceiver. The CAN controller processes various frames according to the CAN. The CAN transceiver converts the logical level (low and high) of the signal and the CAN bus level (dominant and recessive) of the signal between the CAN bus and the CAN controller. ISO 11898 [80] gives the High-Speed CAN bus specification. The specification are given for the maximum baud rate of 1 Mbps and the maximum bus length of 40 m with up to 30 nodes can be connected. A twisted-pair cable is used to ensure robust noise immunity on the CAN bus. The two wires are called CAN-L and CAN-H respectively. As shown in Fig. 4 (a), the voltage of CAN-H/CAN-L are 3.5/1.5 V and 2.5/2.5 V during dominant and recessive (logical 0 and 1), respectively. Also, the CAN data frame has a maximum 8-byte data field whose length is determined by Data Length Code (DLC) in the control field. When dominant and recessive are transmitted simultaneously by multiple nodes, the dominant is preferentially transmitted. Using this feature, an ECU can transmit frames with high priority without interrupting transmission even if multiple nodes simultaneously transmit frames and signals collide. This mechanism is called arbitration, and details are described in Sec. 2.1.2.

Also, in this dissertation, we define "CAN message" as actual data compliant on the data frame (Fig. 4 (b)) on CAN.

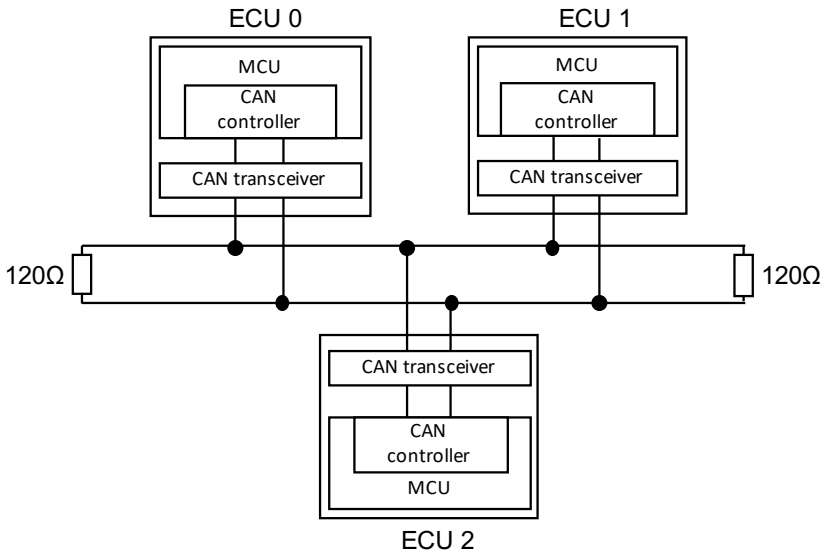
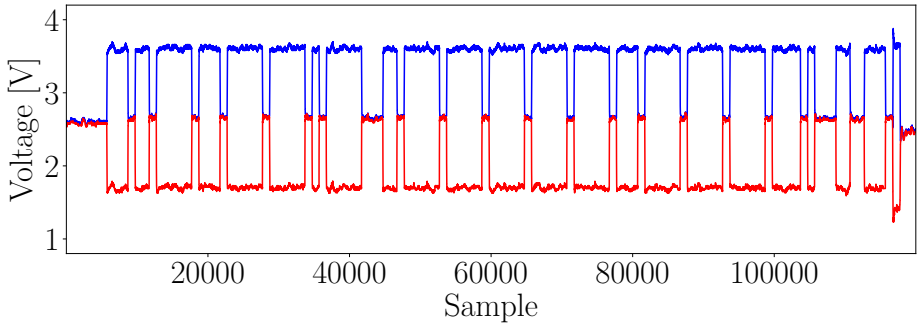
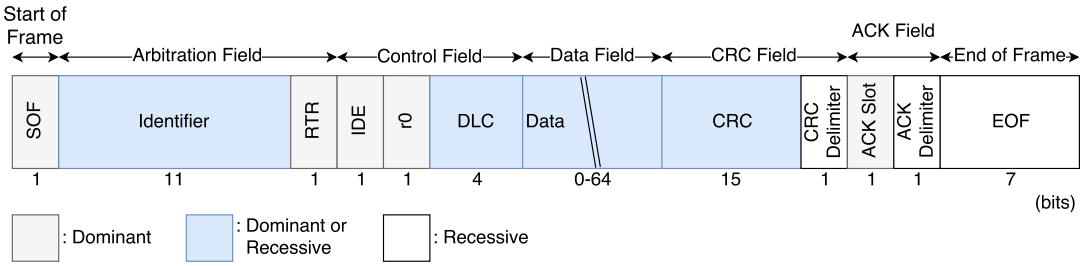


Figure 3: Typical CAN bus.



(a) An example of the CAN message (CAN-H (blue) and CAN-L (red)).



(b) Data frame on CAN.

Figure 4: A CAN message and data frame.

2.1.1 CAN Frame

CAN defines the following four types of frames: data frame, remote frame, error frame, and overload frame. First, the data frame contains data itself, such as sensor data, and is transmitted from the sender ECU to the receiver ECU. Second, the remote frame is used by the receiver ECU to request the transmission of the data frame. Third, the error frame is transmitted to CAN when the transmitted logical value and the CAN differential signal are different. Finally, the overloaded frame is used to add a delay between the previous and current data frames. Although this frame is rarely used because the processing power of the CAN controller and its microcomputers have improved.

The data frame consists of some fields (arbitration field, data field, etc.). In the following, each field is described.

Start Of Frame (SOF)

This field represents the start of the frame using a dominant 1 bit.

Arbitration Field

This field consists of the 11 bit of identifier and the 1 bit of Remote Transmission Request (RTR) which indicates the frame type. The 11 bit of identifier represents the priority of the frame. The lower identifier, the higher the priority of a frame with the identifier. In this dissertation, the identifier is called Arbitration ID (CAN ID). RTR is used to distinguishe whether a frame is data frame or remote frame. When RTR is dominant, it represents a data frame, else a remote frame.

Control Field

This field consists of two reserved bits called IDE and r0 and the DLC.

Data Field

This field represents that the data is transmitted. This field is a variable length within 0–8 bytes.

Cyclic Redundancy Check (CRC) Field

This field is used to check for frame transmission errors. It consists of a 15 bit CRC and a 1 bit CRC delimiter.

acknowledgment (ACK) Field

This field represents acknowledgment sent by receiver ECUs. If receiver ECUs other than an ECU transmitting a CAN message can normally receive up to the CRC field, a 1-bit dominant transmission is performed in the ACK slot as a signal.

End Of Frame (EOF)

The field indicates the end of a frame consisting of 7 bits of recessive.

2.1.2 Arbitration

Since CAN uses the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) method, an ECU confirms that the CAN bus is idle before the ECU sends a data frame, and then the other ECUs can receive the frame. If two or more ECUs start transmission at the same timing, this collision of transmission requests is resolved by bit-by-bit arbitration. This arbitration is performed using the value of the Arbitration ID. Therefore, while sending the Arbitration ID, the ECU sending the CAN message compares whether the transmitted bit and the bit represented by the bus are the same. If the bus expresses a dominant even though it transmitted a recessive, the ECU sending the recessive loses the right to send and must stop sending the CAN message.

Also, if the data frame and the remote frame have the same Arbitration ID and their transmission of them starts at the same time, the data frame is preferentially transmitted because the RTR of the data frame is dominant.

2.1.3 Error Handling

CAN has a fault error state mechanism for high fault tolerance. The error state mechanism defines the following three states as the state of an ECU: error active state, error passive state, and bus-off state as shown in Fig. 5. An ECU starts at the error active state. When the ECU detects several errors (e.g. CRC error), it transits to the error passive state which means that the ECU cannot receive the CAN messages on the bus. Finally, suppose the ECU still detects several errors in the error passive state. In that case, the state of the ECU transits to the bus-off state, where the ECU is logically isolated from the bus. Namely, the

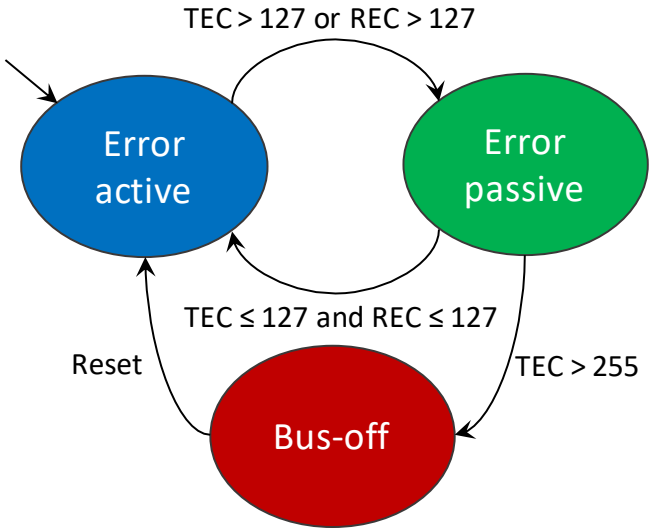


Figure 5: State diagram of CAN error state machanism.

ECU in bus-off state cannot send and receive CAN messages via the bus. Once the ECU enters the bus-off state, the ECU conducts using reboot or a manual procedure by diagnosis to transit to the error active state.

Also, the errors (e.g. CRC error) on CAN increase two types of counters in an ECU: Transmit Error Counter (TEC) and Receive Error Counter (REC). TEC in a sender ECU is counted if the sender ECU confirms the dominant (or recessive) on the bus despite sending the recessive (or dominant) in other than Arbitration and ACK fields. REC in a receiver ECU is counted if received CRC data is different from CRC data calculated by a received CAN message or if a received CAN message has the incorrect data frame. The aforementioned error state mechanisms are managed based on TEC and REC.

2.2 Vulnerablilities of CAN

According to Liu *et al.* [57], CAN vulnerabilities were classified into 4 categories, and attack methods against the vulnerabilities were classified into 5 categories. Fig. 6 shows the classification arranged by Liu *et al.*

The inherent vulnerabilities of CAN are broadcast communication, no encryption, no authentication, and Arbitration ID-based priority schemes. The at-

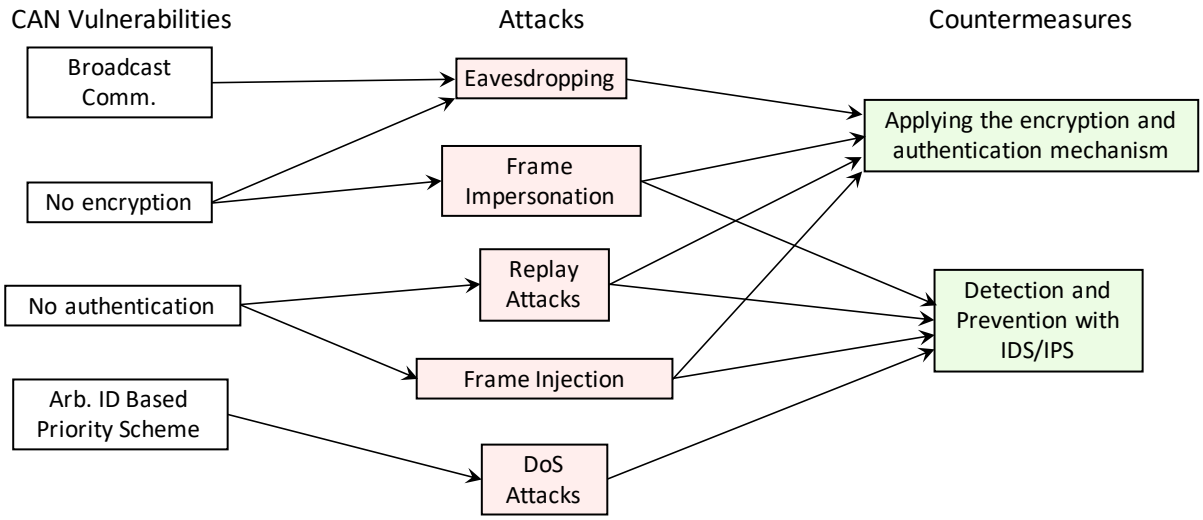


Figure 6: CAN vulnerabilities and the countermeasures.

tack methods caused by these are described below. First, eavesdropping exploits broadcast communication and no encryption. Anyone who can access CAN can eavesdrop on CAN messages because of broadcast communication and no encryption. Second, in a frame impersonation, by using the eavesdropped CAN messages, an attacker can impersonate the CAN message such as operating the meter by impersonating the CAN message related to the vehicle speed. Third, an attacker performs a replay attack, which is an attack that reproduces the CAN messages transmitted by ECUs, and a frame injection that sends an arbitrary CAN message, because CAN has no authentication. Finally, the Arbitration ID-based priority scheme states that are vulnerable to DoS attacks, which overwhelm the bus by sending large numbers of CAN messages that exceed the priority of CAN messages in transmission [13]. Also, as shown in Fig. 4 (b), because the CAN data format does not have a field to indicate the source information, the receiver ECU cannot verify whether the CAN message was sent by the legitimate sender ECU.

Encryption and authentication would be a countermeasure against the above attack methods. However, since the maximum data field of CAN is 8 bytes, authentication is not easily applicable. Furthermore, it is difficult to implement the management of keys needed for encryption/authentication as soon as possible

because it is necessary for automobile manufacturers to appropriately decide and implement key management rules. On the other hand, detection/protection by IDS/IPS can be executed only by connecting IDS/IPS to the CAN bus, Therefore, detection/protection by IDS/IPS has advantages in terms of effectiveness and ease of implementation in current vehicles.

2.3 Attack Surfaces on Automotive IoT

This section describes the intrusion route of attacks from external networks. We focus on the IVI connected to the Internet. As shown in Fig. 7, intrusions from the outside can be roughly divided into three types: long-range, short-range, and internal [15]. Note that a Linux-based IVI such as Automotive Grade Linux (AGL) [24] is assumed.

First, as intrusion routes from the outside (long range), there are the High Speed Synchronous Serial Interface (HSI) driver that handles mobile phone communication and the WPA supplicant process in the user space that handles Wi-Fi communication. If the old versions of these are installed on the IVI, an attacker could break into the IVI. Also, if the IVI is set to automatically connect to an access point with an SSID, an attacker could set up a malicious access point with the same SSID and the attacker could attempt to break into various ports on the IVI.

Second, an attacker from the outside (short range) possibly invades by exploiting the vulnerability of Bluez, a daemon that performs Bluetooth communication.

Third, as a threat other than the external network, an attacker tries to let a legitimate user connects a USB device to IVI to install malware through social hacking. This attack can be exploited if the USB software stack is flawed or if the IVI software is updated via USB. After exploiting the IVI system, the attacker can attempt malware installation such as ransomware targeting vehicles [6].

Also, as a direct entry route to the CAN bus, the ECU diagnostic port On-Board Diagnostics-II (OBD-II) port[1] is used to intrude the CAN bus through direct access [8]. In this dissertation, we conducted an evaluation experiment of attack detection by invading from the OBD-II port.

[1]A diagnosis port for vehicles. Generally, it is installed around the driver's seat.

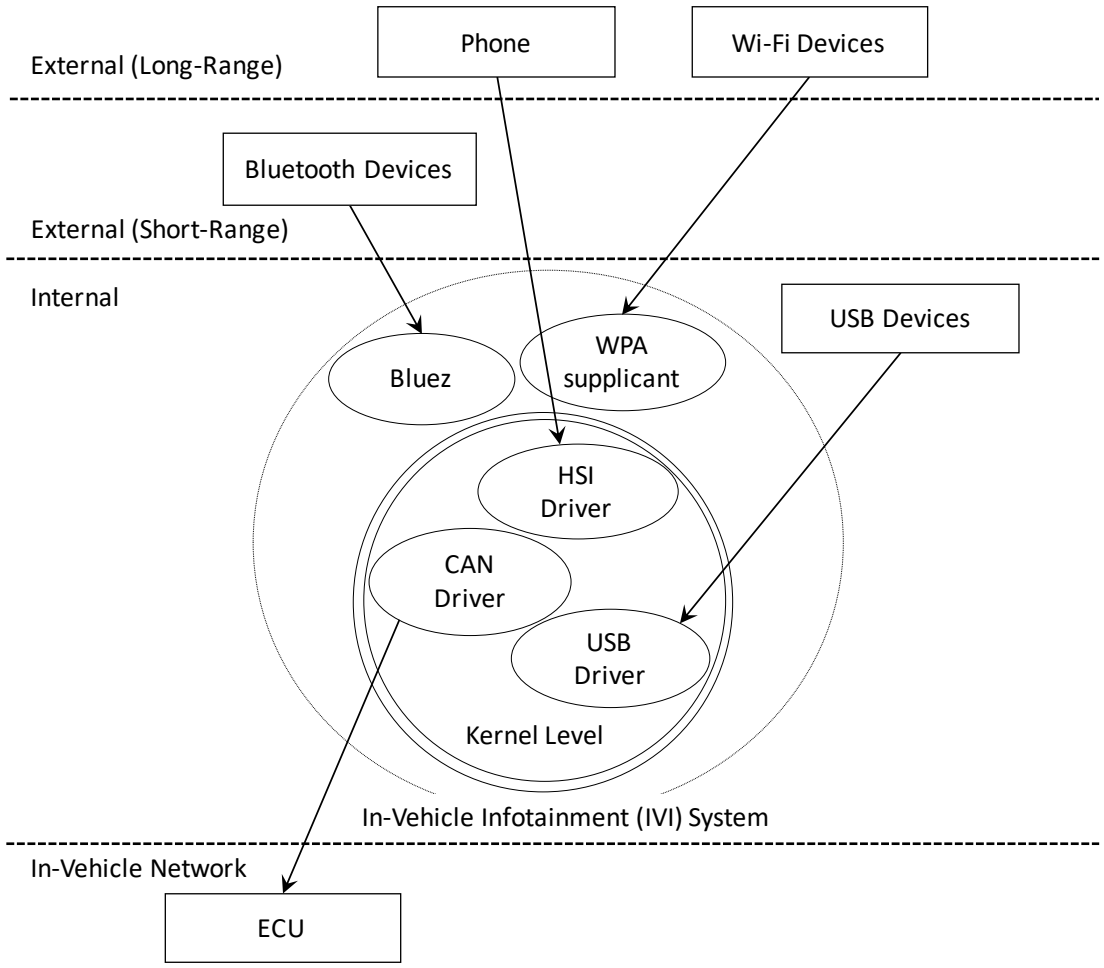


Figure 7: Attack surfaces to the CAN (Linux based IVI)

From the above, the in-vehicle network is directly or indirectly connected to various interfaces, so that attackers can use diverse intrusion routes. Therefore, multi-stage and multi-layered defense measures are important for security countermeasures for securing automotive IoT.

2.4 Our Remote Adversary Model

In this section, we define an adversary model in in-vehicle networks. The remote adversary model is based on the hacking of Jeep Cherokee [62] and the exploitation via Wi-Fi/Bluetooth interface against Display Control Unit (DCU) installed in some Lexus series [14]. In actual hacking of Jeep Cherokee, Miller and Valasek exploited an ECU's update mechanism to inject their code. After getting control of an ECU, it is supposed that the attacker reconnoiters the CAN bus with exploration/reconnaissance tools (e.g., CaringCaribou[2], CANvas Network Mapper [50]). Thus, we suppose that the adversary model can manipulate software in a single compromised ECU. In other words, our model cannot manipulate any direct physical characteristics in the CAN bus. Even though our model is only allowed to manipulate the software layer, the model is a serious threat. For instance, our model with only the software layer can simultaneously conduct the large-scale attacks for thousands of vehicles at the same time by using a single vulnerability [49, 51]. Therefore, we suppose that the adversary model with a single compromised ECU has all software manipulation capabilities.

[2]A friendly car security exploration tool, https://github.com/CaringCaribou/caringcaribou

3. *entropy-manipulation attack*: Evasion Attack on Entropy-Based IDS

3.1 Introduction

Encryption and authentication approaches toward secure in-vehicle networks have been proposed to prevent spoofing, sniffing, and replay attacks [52, 66, 82, 95, 92, 101, 81]. These proposals could disable traditional hacking, in which an adversary sends a spoofed message to a specific ECU. However, even in CAN bus applied to encryption and authentication, these proposals cannot disable a DoS attack that sends many messages to the in-vehicle network, because DoS attacks cause a delay to the encrypted benign messages. To disable DoS attacks, a security solution different from encryption and authentication is necessary. In order to prevent a DoS attack of one type in which an adversary sends messages to flood the buffer of receiving ECU, Intrusion Prevention System (IPS) defenses have been proposed [39, 99, 96]. However, these methods do not affect other DoS attacks, in which an adversary sends many messages of the highest CAN ID priority. Therefore, it is necessary to have an IPS that can prevent all DoS attacks. To prevent all DoS attacks, firstly, we must consider a method to detect all DoS attacks.

Time-intervals IDS [88] has been proposed to detect spoofing attacks and DoS attacks on CAN. This IDS detects DoS attacks with the cutoff of the time interval to 0.2 milliseconds for detecting DoS messages. However, in the case of over 0.2 milliseconds of the time interval of the DoS attack's messages, the IDS cannot detect DoS attacks. In addition, in different baud rates such as CAN and CAN with Flexible Data Rate (CAN-FD) [29], the IDS cannot be adapted to the CAN buses because the time intervals of DoS attacks are different. Furthermore, some methods [88, 31, 98] are probably only effective against the DoS attacks under naive environments, such as some higher priority messages [53].

A machine learning approach for IDS has been proposed [54]. The approach can widely detect attacks such as spoofing, replaying, and DoS attacks with high accuracy. The approach (especially deep learning) is too expensive to implement the training function in the vehicle, although the cost of inferring is reasonable. Also, a secure OTA update [40] has been proposed in modern automotive. There-

fore, the cost of training the additional communication of the OTA update should be reasonable.

Also, an entropy-based IDS [98] using the entropy of a fixed number of messages, called a sliding window, has been proposed against the DoS attacks and the replay attacks. This method has a good advantage in terms of effectiveness and the small computational overhead [100]. However, the entropy might be manipulated by adversaries to avoid the IDS. To validate the feasibility of this attack, we present an evasion DoS attacks called *entropy-manipulation attacks* against the entropy-based IDS. Additionally, we carried out experiments in which the attacker executed entropy-manipulation attacks on a six-vehicle environment. As the result, we confirmed that the entropy-manipulation attacks can evade the entropy-based IDS to mimic the entropy with higher IDs than actual IDs.

The main contributions of this study can be summarized as follows.

1. We found a DoS attack called the *entropy-manipulation attack*, which bypasses the conventional method (entropy-based IDS [98]) by adjusting the entropy of messages. These consist of messages of random CAN IDs.

2. We proposed a sliding window poisoning tactic to inject the DoS messages to the sliding window without varying the entropy at the boundary between the benign and DoS messages.

3. We confirmed that an attacker can evade the entropy-based IDS with the entropy-manipulation attack in a six-vehicle environment.

3.2 Related Work

A lot of works have been done on IDSs on CAN. IDSs on CAN using deep learning have been proposed [43, 86, 18, 19, 17, 36, 48, 70]. The approach is too expensive to implement the training function in the vehicle although the cost of inferring is reasonable. Also, a secure OTA update [40] has been proposed in modern automotive. Therefore, the cost of training the additional communication of the OTA update should be reasonable.

Time-intervals IDS [88] has been proposed to detect spoofing attacks and DoS attacks on CAN. This IDS detects DoS attacks with the cutoff of the time interval

to 0.2 milliseconds for detecting DoS messages. However, in the case of over 0.2 milliseconds of the time interval of the DoS attack's messages, the IDS cannot detect DoS attacks. In addition, in different baud rates such as CAN and CAN-FD, the IDS cannot be adapted to the CAN buses because the time intervals of DoS attacks are different. Furthermore, some methods [88, 31, 98] are probably only effective against the DoS attacks under naive environments such as some highest priority messages [53].

Detection methods based on electrical fingerprint information have been proposed [10, 44]. However, in order to perform electrical fingerprint information-based anomaly detection, some additional hardware such as the A/D converter is necessary. Furthermore, if an original ECU is compromised, the IDS cannot detect a malicious message using CAN ID assigned in compromised ECU itself.

There are various other related studies, but these studies are not superior to the entropy-based IDS in terms of effectiveness to DoS attacks and the small computational overhead.

3.3 Entropy-Based IDS

3.3.1 Fixed Time Based Method

IDSs based on entropy in the fixed time have been proposed [69, 60].

Muter and Asaj proposed the first anomaly detection method based on information-entropy [69]. The method uses the relative distance of entropy of each IDs between two datasets to detect anomalies. This advantage of the method is to detect a slight increase or decrease in the specific CAN ID. However, it also causes false alarms because the frequency of IDs in the fixed time changes by the clock-skew in CAN.

Next, Marchetti et al. proposed the entropy of the fixed time based anomaly detection method [60]. They designed the anomaly detection method through experiments in real CAN messages. However, the method has disadvantages such as no adaptation to different baud rate, misdetection caused by aperiodic CAN messages, and no capability of real-time detection.

3.3.2 Sliding Window Based Method

Wu et al. pointed out that these IDSs [69, 60] cannot be applied to different transmission rates and aperiodic messages because they use fixed-time messages for calculating entropies. Therefore, they proposed a novel entropy-based IDS [98], showing that the entropy-based IDS can fastly detect DoS attacks by using a sliding window (a fixed number of messages) for calculating the entropy. The definition of entropy in the method is as follows, where $I = \{id_1, id_2, id_3, ..., id_n\}$ is a set of different CAN IDs appearing within sliding windows W. Equation (1) is expressed as the entropy of CAN IDs in sliding windows W.

$$H(I) = -\sum_{id_i \in I} P(id_i) \log(P(id_i)) \tag{1}$$

Next, we explain the $P(id_i)$ in Equation (1). Since the method determines the network state by monitoring CAN messages per window W, the total number of messages in the arbitrary network state is the same to window W. Thus, the total number of messages N_{total} in the sliding windows W can be obtained by Equation (2):

$$N_{total} = \sum_{i=1}^{n} Count_{id_i} \tag{2}$$

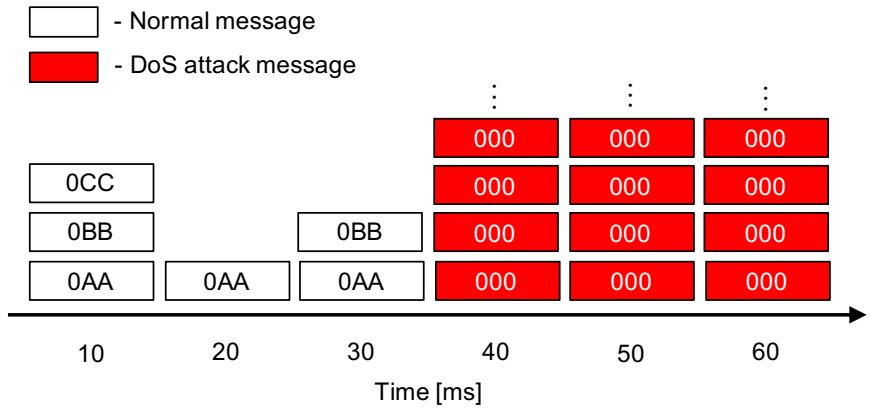
The $Count_{id_i}$ is the number of id_i appearing in W. Then the probability of id_i appearing in W can be represented as

$$P(id_i) = \frac{Count_{id_i}}{N_{total}} \tag{3}$$

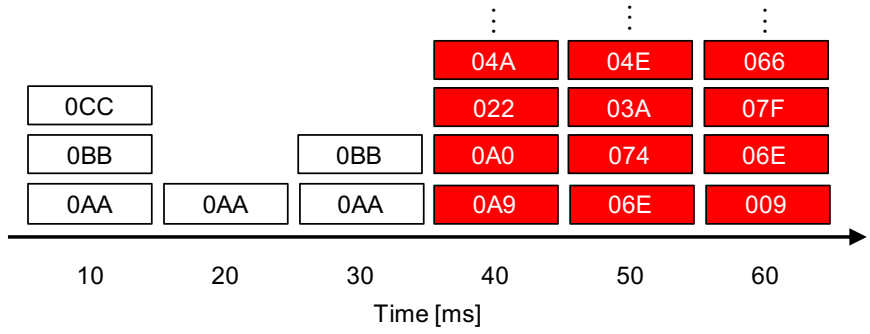
The definition of Equation (1), (2), and (3) is based on the entropy-based IDS [98]. The problem with the entropy-based IDS [98] is that it has a much higher FP rate against the entropy-manipulation attack.

3.4 Attacker Model

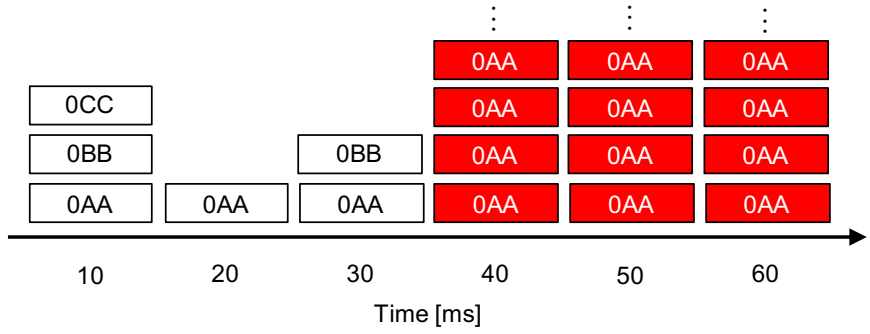
There are three types of DoS attacks on CAN according to the report [39] of Humayed et al. as follows. (Also, shown in Fig. 8.)



(a) Traditional DoS attack



(b) Randomized DoS attack



(c) Targeted DoS attack

Figure 8: The classification of DoS attacks on CAN.

3.4.1 Traditional DoS Attack

An adversary can easily interfere with a CAN bus by using bitwise arbitration. Since the lower CAN ID has a higher priority, the adversary would use CAN ID `0x000` for the DoS attack. As a result, the adversary can induce unexpected behavior of the vehicle. Though it is not difficult to detect the Traditional DoS attack, IDSs must detect it as soon as possible.

3.4.2 Randomized DoS Attack

A randomized DoS attack is the most appropriate attack for broadcasting incorrect values without investigating an in-vehicle network. Messages with a random CAN ID from 0 to 2047 can be transmitted within one second, even on a low-speed network. The difficulty of the detection of the randomized DoS attack is the same with Traditional DoS attacks and should be detected as soon as possible too. An entropy-based IDS can detect randomized DoS attacks of both low and high entropy. However, the entropy-based IDS cannot detect an entropy-manipulation attack in which an adversary adjusts the entropy of a DoS attack to a benign entropy. For example, in case of the sliding window $W = 30$, entropies of two sliding windows are the same value if a sliding window includes 10 messages of normal CAN IDs `0x0AA`, `0x0BB`, and `0x0CC`, respectively, and a sliding window includes 10 messages of the DoS message's IDs `0x000`, `0x001`, and `0x002`, respectively. If an attacker exploits the entropy-manipulation attack, the attacker can bypass the entropy-based IDS. The conditions that an attacker must satisfy are as follows.

1. Knowing the hyperparameters of entropy-based IDS such as window size and average entropy.

2. Existence of Higher priority CAN IDs than actual CAN IDs.

Here, we describe whether an attacker satisfies condition 1. Firstly, after an attacker intruded an ECU from some external network, the attacker sniffs the messages of the CAN bus. And then, the attacker can calculate the hyperparameters of window size and the average entropy using the same algorithm to the entropy-based IDS. Finally, the entropy-manipulation attack is carried out using

the window size and average entropy. IDS should assume adversaries know the algorithm inside it [23].

To confirm the satisfaction of condition 2, we surveyed whether there are these IDs with collecting CAN data of 6 car models in Sec. 3.5.2. Therefore, we explain the detail of condition 2 in the section.

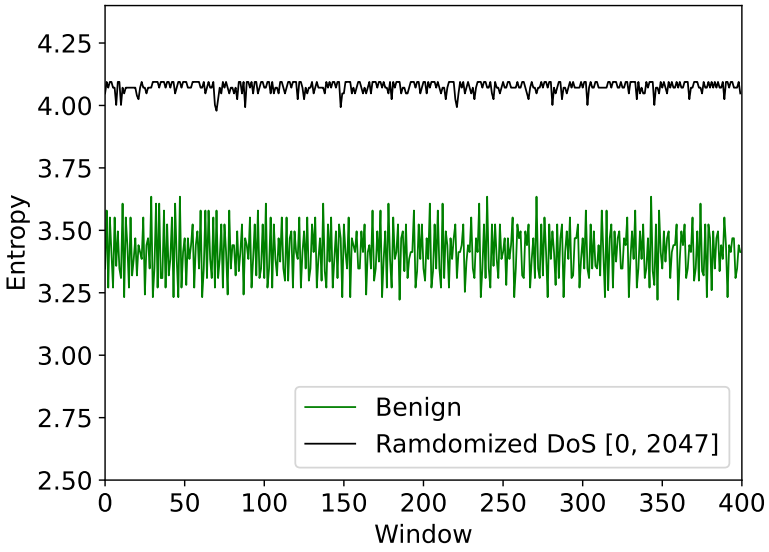
3.4.3 Targeted DoS Attack

A Targeted DoS attack influences buses and ECUs. In this research, we assume an attack on one ECU and define it as a DoS attack using one ID flowing on the bus. This DoS attack could have life-threating consequences for the driver and passengers. However, entropy-based IDS can be used to achieve to detect this DoS attack.

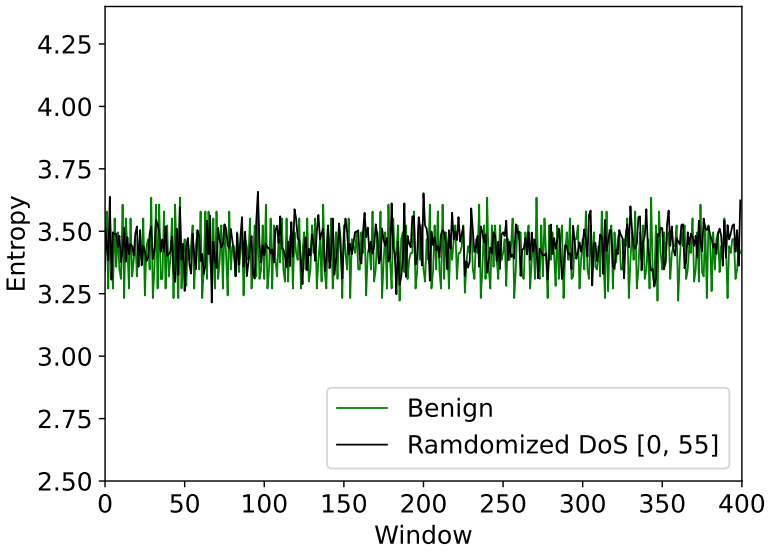
3.5 Feasibility of *entropy-manipulation attack*

3.5.1 Root Cause of Evasion Attack

We show the example of the entropy-manipulation attack. Fig. 9 illustrates our test car's temporal change of entropy against two randomized DoS attacks. Fig. 9 (a) and (b) are randomly generated by randomized DoS attacks with CAN IDs of a range of `[0, 0d2047]` and `[0, 0d55]` respectively. From Fig. 9 (a), we confirm that the entropy-based IDS can distinguish the normal CAN messages and randomized DoS attacks of the extremes of entropy. Fig. 9 (b) shows that the entropies of normal messages and randomized DoS attacks are the same value. This attack is an entropy-manipulation attack. Also, Fig. 10 shows the entropy of randomized DoS attacks under the various ranges of CAN IDs. It shows that an adversary can inject a randomized DoS attack using arbitrary entropy. If the entropy of a randomized DoS attack and the entropy of normal traffic are the same value, like Fig. 9 (b), an adversary can bypass the entropy-based IDS. This problem is caused by that the entropy defined by [98] is based only on the randomness of the CAN ID. In other words, an adversary can configure sliding windows with the same randomness with completely different CAN IDs which are higher priority than normal CAN messages. Therefore, to detect the entropy-manipulation attacks and the other DoS attacks, we consider an approach that



(a) Randomized DoS attacks with the range of `[0, 0d2047]`



(b) Randomized DoS attacks with the range of `[0, 0d55]`

Figure 9: Vulnerability of the entropy-based IDS.

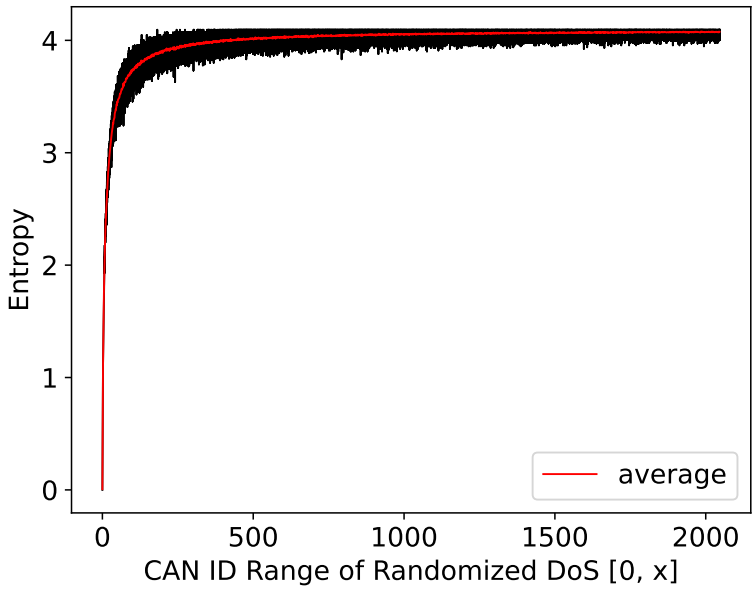


Figure 10: Entropy of randomized DoS attack under different CAN ID range.

can detect the entropy-manipulation attacks based on whether a CAN IDs' set in a sliding window is a normal range.

3.5.2 Higher Priority IDs than Actual IDs

To realize the entropy-manipulation attack, it is necessary to use the higher priority IDs than the actual IDs. We survey whether there are these IDs with collecting CAN data of 6 car models. Table 1 shows the top 10 priority IDs of each real-vehicles. The highest priority IDs in the vehicle A, B, and D are `0x101`, `0x215`, and `0x114`, respectively. Hence, an attacker can exploit at least the random IDs of `[0, 0x101]` to perform the entropy-manipulation attack on these vehicles. On the other hand, the highest priority IDs in the vehicle of HCR Lab, C, and E are `0x018`, `0x002`, and `0x020`, respectively. In the case of these vehicles, an attacker might not be able to do the entropy-manipulation attack because the high priority IDs are assigned as benign IDs. Thus, to mitigate this problem, the attacker can change the attack targets to IDs of 2nd place and below.

Table 2 shows the entropy of each vehicles' data and the entropy of DoS

Table 1: Top high priority IDs in real-vehicles.

	HCR Lab [55]	Vehicle A	B	C	D	E
1st	0x018	0x101	0x215	0x002	0x114	0x020
2nd	0x034	0x151	0x216	0x0D0	0x116	0x024
3rd	0x043	0x152	0x280	0x0D1	0x119	0x025
4th	0x050	0x154	0x2DE	0x0D2	0x120	0x0AA
5th	0x080	0x1D0	0x351	0x0D4	0x122	0x0B4
6th	0x0A0	0x1D1	0x355	0x140	0x124	0x127
7th	0x110	0x200	0x358	0x141	0x130	0x1C4
8th	0x120	0x208	0x35D	0x144	0x131	0x224
9th	0x153	0x210	0x615	0x148	0x13F	0x230
10th	0x164	0x212	0x5C5	0x149	0x180	0x245

attack using the random IDs of the range up to the 1st and 2nd priority of each vehicle. In the vehicle A, if the entropy of the DoS attack using IDs up to the 1st priority is greater than the entropy of the vehicle, the attacker can perform the entropy-manipulation attack interrupting all messages of benign IDs. Therefore, the attacker can perform the entropy-manipulation attack. Also, in vehicle HCR Lab and C, the entropy of the vehicle is greater than the entropy of the DoS attack using IDs up to the 1st priority. However, the entropy of the vehicle is not greater than the entropy of the DoS attack using IDs up to the 2nd priority. In other words, the attacker can carry out the entropy-manipulation attack interrupting the messages of benign IDs except for the highest benign ID.

To sum up this section, we conclude that in 4 out of 6 vehicles the attacker can perform DoS attacks using the higher priority IDs than the actual IDs, while in the remaining 2 vehicles the attacker can execute DoS attacks using the higher priority IDs than the part of actual IDs.

3.5.3 Sliding Window Poisoning Tactics

Here, we describe how an attacker poisons the sliding window without changing the entropy. We envisage that the entropy increases in the boundary between

Table 2: Comparison of average entropies of real-vehicle's CAN data and Randomized DoS attacks using available ranges.

	Average Entropy
HCR Lab	3.05408
[0, 0x018]	2.99363
[0, 0x034]	3.46754
A	3.28269
[0, 0x101]	3.94616
[0, 0x151]	3.97857
B	2.09613
[0, 0x215]	4.01927
[0, 0x216]	4.01904
C	3.25292
[0, 0x002]	1.08202
[0, 0x0D0]	3.91271
D	3.41602
[0, 0x114]	3.95546
[0, 0x116]	3.95483
E	2.88480
[0, 0x020]	3.18932
[0, 0x024]	3.26292

benign and DoS messages. For example, if the sliding window contains half of the benign and half of the DoS messages, the randomness of IDs increases. As a result, the entropy increases in the boundary between benign and DoS messages. Therefore, we propose a tactic that an attacker can poison the sliding window without changing the entropy.

First, we confirm whether the entropy increases in the boundary between benign and DoS messages. Fig. 11 shows that the entropy and CAN ID range of Randomized DoS during sliding window poisoning tactic with static range `[0, 0d55]`. From Fig. 11 (a), we can confirm that the entropy exceeds the benign area around 30 of the number of injected messages. This exceedance is caused by high random degree of CAN ID range of randomized DoS around 30 of the number of injected messages. Thus, next, we try the sliding window poisoning tactic with liner growth. As with Fig. 11, Fig. 12 show that the entropy and CAN ID range of randomized DoS. From Fig. 12 (a), we confirm that the increase of entropy is slightly mitigated. However, the entropy still exceeds the benign area. Finally, as shown in Fig. 13, we grow the CAN ID range with non-linear. Fig. 13 (a) shows that the entropy is not exceeded the benign area because the CAN ID range of randomized DoS is carefully increased.

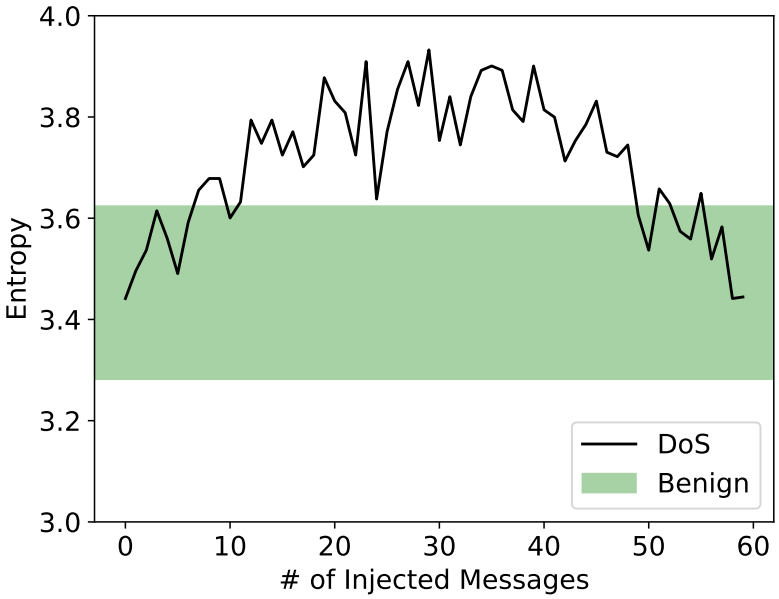
From this comparison of poisoning tactics, we conclude that the sliding window poisoning tactic with non-linear growth can poison the sliding window without changing the entropy.

3.6 Evaluation

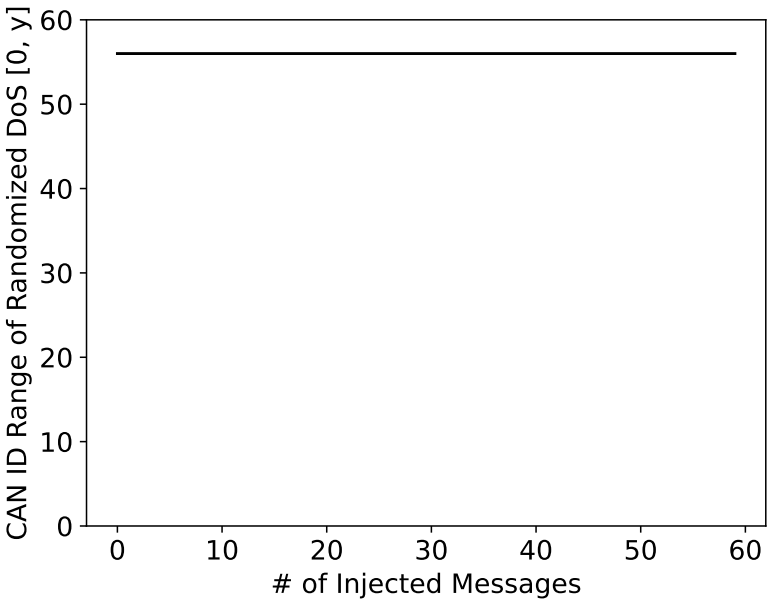
3.6.1 Entropy of Real-Vehicles

In this study, we evaluate the detection capabilities of entropy-based IDS against entropy-manipulation attacks. We use a data set of the real CAN messages provided in [55] and a data set of the five vehicles (A, B, C, D, E) which we logged during driving and stopping. We evaluate the entropy-based IDS using the DoS attack messages (1000 messages) added to the data sets without DoS attack messages.

Table 3 shows the dataset description. We describe the average value H_{ave} and the standard deviation H_{dev} of the entropy at the sliding window $W = 60$,

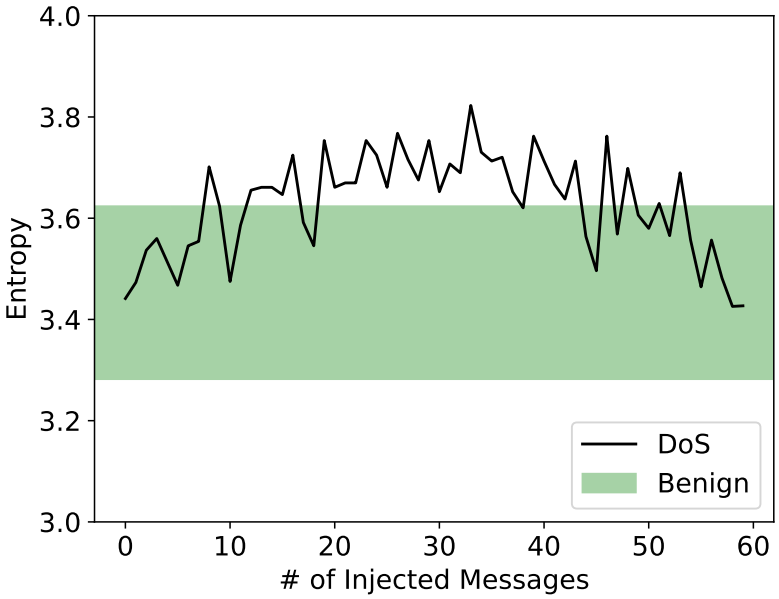


(a) Entropy during the sliding window poisoning tactic.

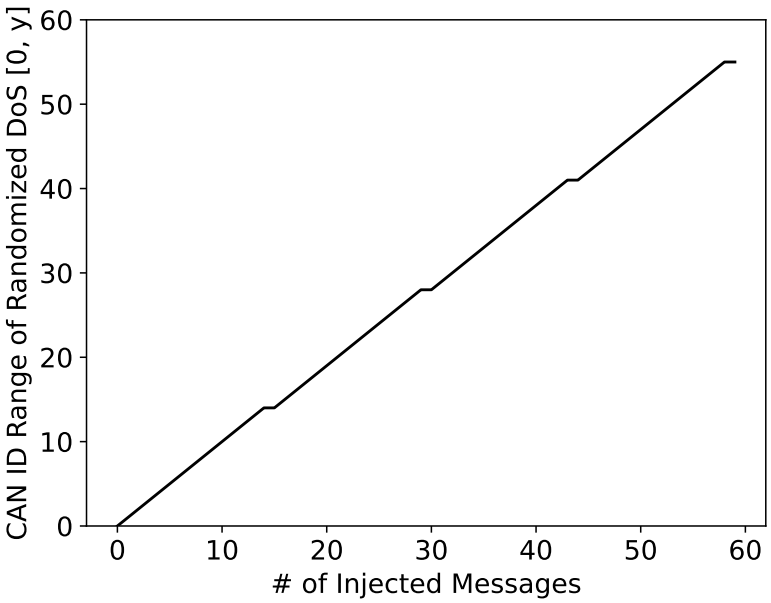


(b) CAN ID range of randomized DoS during the sliding window poisoning tactic.

Figure 11: Sliding window poisoning tactic (static range `[0, 0d55]`).

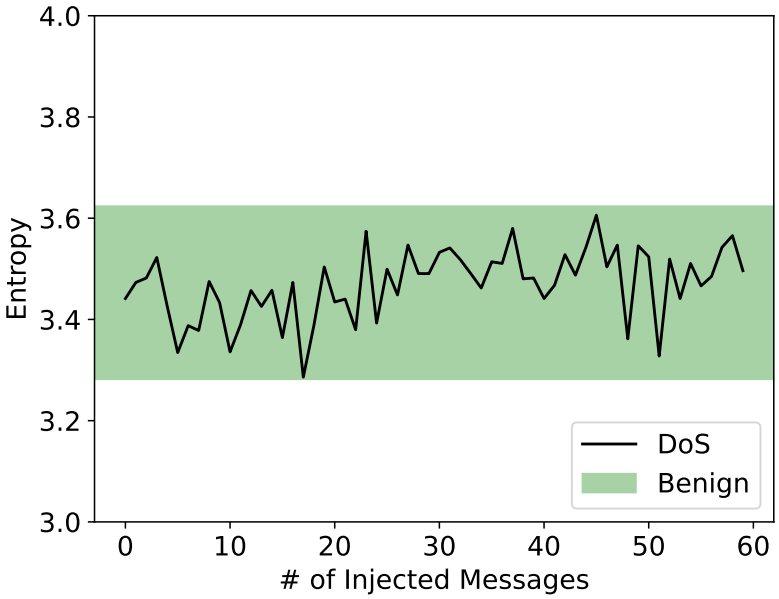


(a) Entropy during the sliding window poisoning tactic.

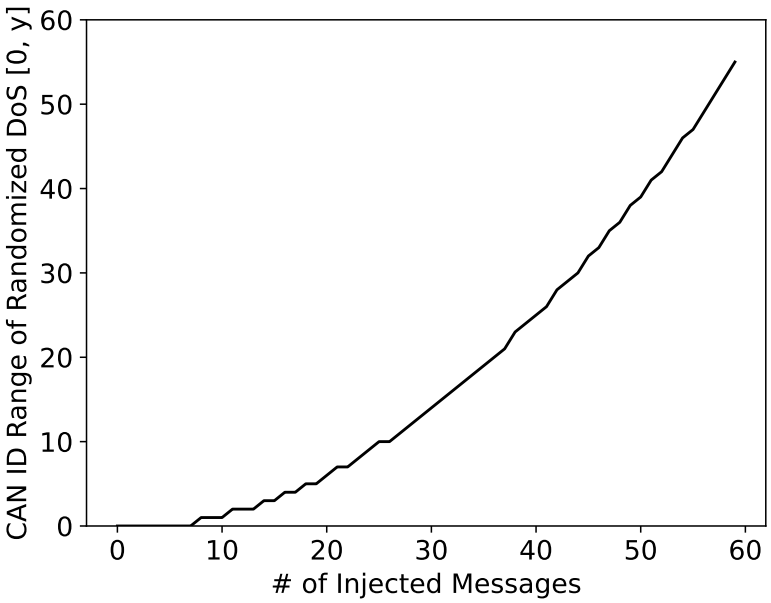


(b) CAN ID range of randomized DoS during the sliding window poisoning tactic.

Figure 12: Sliding window poisoning tactic (linear growth).



(a) Entropy during the sliding window poisoning tactic.



(b) CAN ID range of randomized DoS during the sliding window poisoning tactic.

Figure 13: Sliding window poisoning tactic (non-linear growth).

Table 3: Data set description.

Name	Normal messages	H_{ave}	H_{dev}
HCR Lab	1000	3.05408	0.08345
A	1000	3.28269	0.06991
B	1000	2.09613	0.04416
C	1000	3.25292	0.07083
D	1000	3.41602	0.11795
E	1000	2.88480	0.12877

which was regarded as the optimal parameter in the entropy-based IDS [98]. The entropies depend on the vehicle. Therefore, when these normal entropies and the entropies of randomized DoS attacks are the same value, the detection accuracy of the entropy-based IDS decreases.

We evaluate the proposed attack method with three metrics. The first is the recall of entropy-based IDS under entropy-manipulation attacks. In this evaluation, we confirm whether entropy-manipulation attacks can evade the entropy-based IDS in 6 vehicles. The second is the precision of manipulation-aware entropy-based IDS. We test the scenario that the entropy-based IDS learns entropy-manipulation attacks as DoS attacks. The third is the evasive performance of sliding window poisoning tactics of entropy-manipulation attacks. We validate the three poisoning tactics using real CAN traffic.

3.6.2 Recall of Entropy-Based IDS under *entropy-manipulation attacks*

In this section, we evaluate the recall of entropy-based IDS during entropy-manipulation attacks. In this evaluation, we validate the scenario that the entropy-based IDS learns DoS attacks with `0x000` messages. In other words, the entropy-based IDS cannot learn entropy-manipulation attacks in order that we test the manipulation-unaware entropy-based IDS.

Fig. 14 shows that the recall of entropy-based IDS against entropy-manipulation attacks in each vehicle. From Fig. 14, we confirm that there is attacks with the recall of 0.0 % in each vehicle. This implies that entropy-manipulation attacks

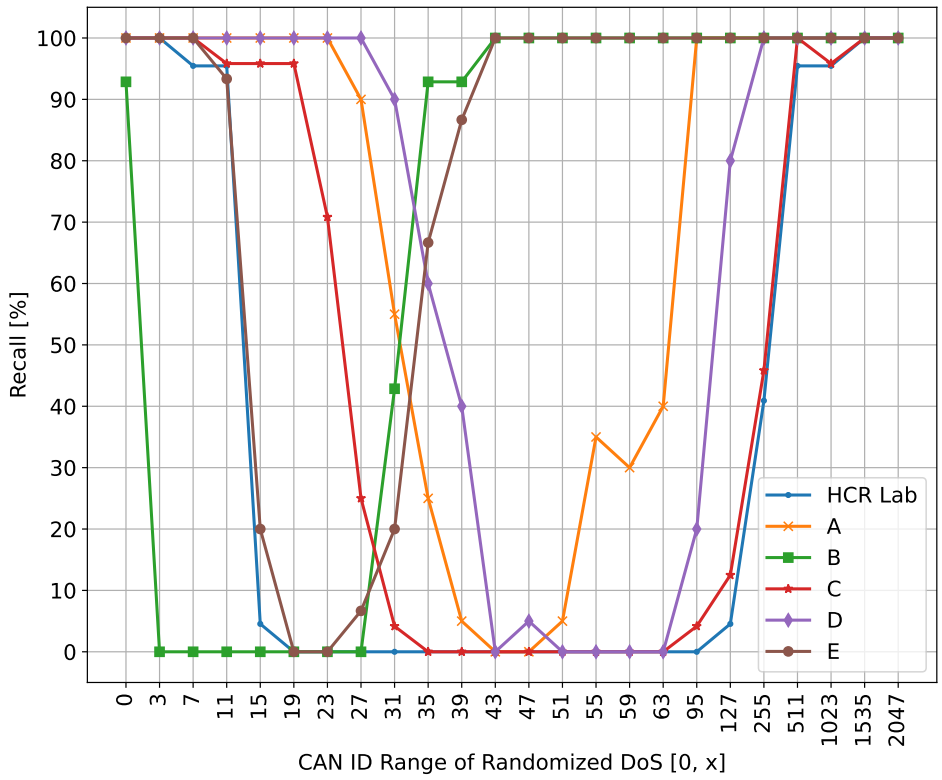


Figure 14: Recall of entropy-based IDS against entropy-manipulation attacks.

could completely evade the entropy-based IDS. We also evaluated the precision, but most of the precision was 0.0 % because IDS rarely classifies as an attack label. Therefore, we conclude that entropy-manipulation attacks can evade the entropy-based IDS in all vehicles.

3.6.3 Precision of Manipulation-Aware Entropy-Based IDS

In this section, we validate the precision of manipulation-aware entropy-based IDS. In other words, the entropy-based IDS can learn entropy-manipulation attacks in off-line learning phase. Fig. 15 shows that the precision of manipulation-aware entropy-based IDS against entropy-manipulation attacks. We confirm that the entropy-based IDS has a range in which the precision decreases. We show

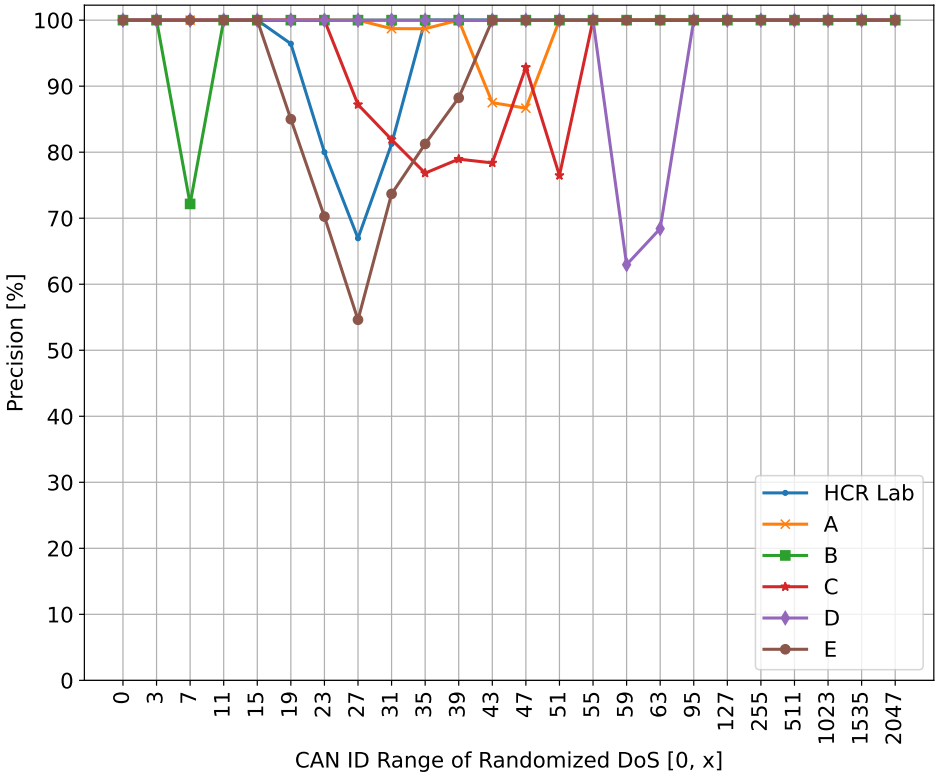


Figure 15: Precision of manipulation-aware entropy-based IDS against entropy-manipulation attacks.

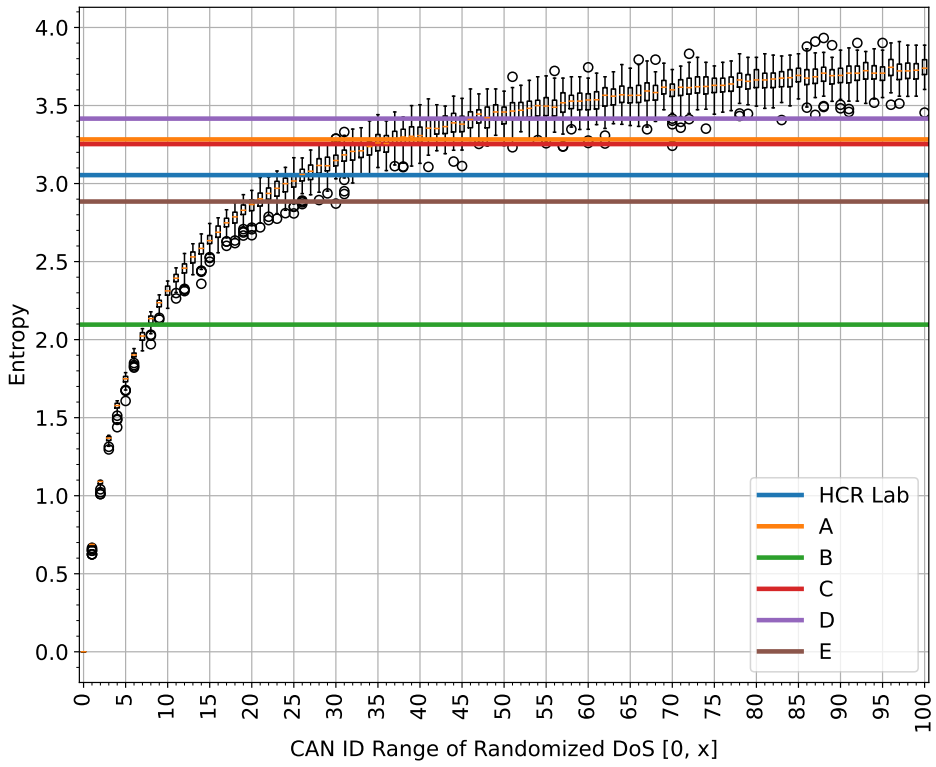


Figure 16: Entropy of entropy-manipulation attack under different CAN ID range `[0, 0]-[0, 0d100]`.

the entropy in the entropy-manipulation attack under different CAN ID ranges in Fig. 16, which shows a correspondence between the entropy and the CAN ID range of entropy-manipulation attacks. Fig. 16 plots the 100 entropies of entropy-manipulation attacks in each range with the boxplots. Furthermore, we depict the 6 lines which express the average entropies of each vehicle in Fig. 16. Focusing on the vehicle D in Fig. 15, the precision of the entropy-based IDS has decreased in the range `[0, 0d59]`. Next, when focusing on the x-axis `[0, 0d59]` in Fig. 16, the average entropy of vehicle D is within the entropy of the entropy-manipulation attack in the range `[0, 0d59]`. Because the entropy of the entropy-manipulation attack and the average benign entropy are the same value, the entropy-based IDS cannot detect the entropy-manipulation attack with high

Table 4: Evasive performance during the each poisoning tactic.

	TPR[%]	FPR[%]	FNR[%]	TNR[%]	Precision[%]	Recall[%]
static	66.7	53.3	33.3	46.7	55.6	66.7
linear	26.7	33.3	73.3	66.7	44.4	26.7
non-linear	0.0	0.0	100.0	100.0	-	0.0

precision. The same applies to other vehicles. Hence, we confirmed that the precision decreases when the average entropy used in detecting DoS attacks is within the range of the entropy of the entropy-manipulation attack.

3.6.4 Evasive Performance during Sliding Window Poisoning Tactics

As described in Sec. 3.5.3, we propose the three poisoning tactics. In this section, we evaluate these poisoning tactics. Table 4 shows the evasive performance during the each poisoning tactic.

As shown by Table 4, the poisoning tactic with randomized DoS of static range can be detected by IDS with a precision of 55.6 %. As with the static range, the poisoning tactic with linear growth range can be detected with a precision of 44.4 %. However, the poisoning tactic with non-linear growth range can evade the IDS because of a precision of 0.0 %. Therefore, we conclude that the sliding window poisoning tactic with non-linear growth can poison the sliding window without changing the entropy.

3.7 Discussion

3.7.1 Feasibility of Evasion Attack on Entropy-Based IDS

To realize entropy-manipulation attacks, some higher priority IDs than actual IDs are required. In Sec. 3.5.2, we surveyed whether there are high priority IDs enough to manipulate the entropy. As the result, we have confirmed that there are higher priority IDs than actual IDs in the six-vehicles.

Besides, we proposed sliding window poisoning tactics in order to inject the DoS messages to benign windows. The proposed poisoning tactics can inject the

DoS messages to the sliding window without changing the entropy.

From the existence of higher priority IDs than actual IDs and the effectiveness of sliding window poisoning tactics, we conclude that an attacker can perform entropy-manipulation attacks in real-vehicle's environments.

3.7.2 Detection of *entropy-manipulation attacks*

The proposed evasion attack can be detected by monitoring the time-intervals of each CAN ID, however a remote attacker can mimic the time-interval of CAN ID as with the entropy. Moreover, the time-interval based IDS has a problem that the IDS cannot apply to aperiodic messages. In order to overcome this problem of time-interval based IDS, the entropy-based IDS also has been proposed. Thus, even if the time-interval based IDS can detect entropy-manipulation attacks, the problem of aperiodic messages reoccurs.

On the other hand, a sender identification method based on ECU-specific characteristics such as voltages (VIDS) has been proposed [44, 46, 84]. The VIDS can detect entropy-manipulation attacks because the attacks cause a lot of messages of the same sender. However, in case that the highest ID is assigned to a compromised ECU, a DoS attack that interrupts the transmission of other messages is feasible. This attack cannot be detected by VIDS because a legitimate ECU performs a DoS attack with a legitimate ID.

To detect entropy-manipulation attacks and other DoS attacks, it is possible to implement an IDS based on similarity rather than the entropy of sliding windows. Since the similarity-based IDS is extended to inheriting the characteristics of the applicability to different baud rate and aperiodic messages of the entropy-based IDS, the problem of time-interval based IDS cannot occur. Therefore, we will propose the similarity-based IDS in Chap. 4.

3.8 Conclusion

To develop the security mechanism for automotive networks, entropy-based IDS has been proposed. This IDS can fastly detect DoS attacks and can be implemented at a low cost. However, we found a vulnerability against the entropy-based IDS, which is called *entropy-manipulation attack*. The proposed attack

can completely evade the manipulation-unaware entropy-based IDS. In addition, even in the case that the IDS is aware of the manipulation, the entropy-based IDS only can detect the proposed attack with a precision of 54.62%. We also released the demonstration of entropy-manipulation attacks at `https://youtu.be/pbRVXO4RnlM`.

4. Sliding Window Optimized Similarity Analysis Method against *entropy-manipulation attack*

4.1 Introduction

In Chap. 3, we discovered the entropy-manipulation attacks which can evade the entropy-based IDS to mimic the entropy with higher IDs than actual IDs. To solve this problem, we aim to detect the entropy-manipulation attack and the other DoS attacks with the proposed method. The entropy-based IDS focuses only on degree of disorder of CAN IDs in a sliding window, but the IDS does not focus on the individual values of CAN IDs in a sliding window. In other words, even though the CAN IDs in a sliding window are completely different and have the same degree of disorder, the entropy is the same value. Therefore, the entropy-manipulation attack can bypass the entropy-based IDS. Hence, in the proposed method, in order to detect anomalies based on degree of disorder of CAN IDs and the individual values of CAN IDs in a sliding window, we use similarity of two sliding windows. One of the two sliding windows is composed of CAN IDs ($WIDs$; Window IDs) which are actually received, and the other is composed of normal CAN IDs ($CIDs$; Criterion IDs) which serves as a criterion to calculate the similarity. Our proposed method calculates the similarity in $WIDs$ and $CIDs$ using the Simpson coefficient, which expresses the similarity between the sets. Also, $CIDs$ is generated as optimized parameter using the Simulated Annearing (SA) algorithm in the proposed method. In addition, in order to optimize the anomaly detection precision and execution time, we use the SA algorithm, which obtains a good local solution using the above algorithm of entropy-based IDS.

The main contributions of this study can be summarized as follows.

1. The proposed method (similarity-based IDS) achieved a detection precision of 100.0% against the above type of DoS attacks, while the detection precision is 68.3% in the conventional method.

2. We showed that the execution time is up to 93% (14μs) shorter than the

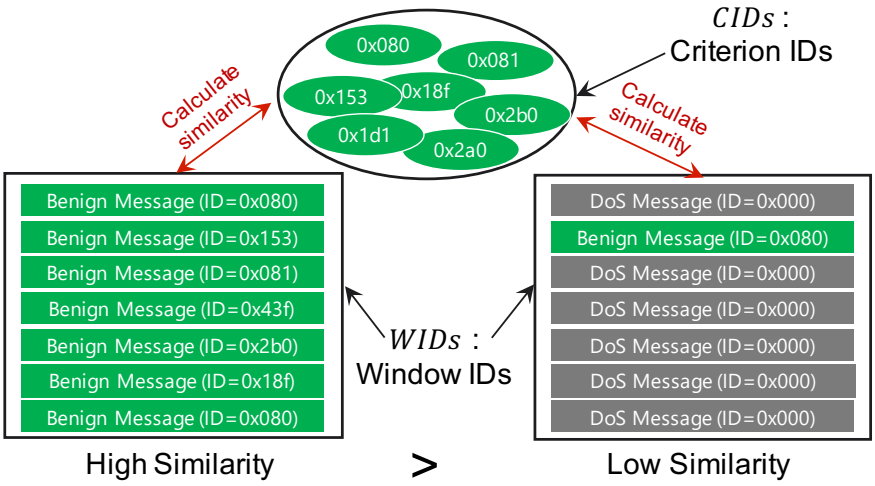


Figure 17: An example of $WIDs$ and $CIDs$.

conventional method.

4.2 Similarity-Based IDS against Various DoS Attacks

As aforementioned in Sec. 4.1, in order to detect anomalies based on the degree of randomness of CAN IDs and the individual values of CAN IDs in a sliding window, we propose an IDS based on the similarity of the sliding windows rather than the entropy of the sliding windows. Our similarity-based IDS calculates the similarity in $WIDs$ and $CIDs$ using the Simpson coefficient, which expresses the similarity between the sets. Fig. 17 shows an example of $WIDs$ and $CIDs$. As shown in Fig. 17, our similarity-based IDS detects DoS attacks based on the similarity between $WIDs$ and $CIDs$. In addition, in order to optimize the anomaly detection precision and detection time, we use the SA algorithm. The SA algorithm is used to obtain a good local solution, so that the entropy-based IDS [98] indicated the effectiveness of the SA algorithm.

Detection time is an important evaluation metric because fast detection can conduct intrusion preventions (e.g. ID-Hopping Mechanism [39, 99, 41], Reactive Defense Mechanism [34], Firewall [38, 56], Client-Side Enforcement [77]) rapidly. Therefore, we consider decreasing the detection time.

4.2.1 Definition of Similarity

Our similarity-based IDS calculates the similarity in $WIDs$ and $CIDs$ using the Simpson coefficient (often called the Overlap coefficient), which expresses the similarity between the sets. If the two sets have an intersection, the Simpson coefficient of the two sets is higher than the Jaccard coefficient and the Dice coefficient. Also, since some non-cyclic messages are sent in CAN, similarity decreases even in normal messages. Therefore, if a part of the two sets is not shared such as non-cyclic messages, the Simpson coefficient that the decreasing of similarity is most low is the most suitable similarity in CAN.

The definition of similarity in CAN is as follows. Equation (4) is used to calculate the similarity between $CIDs$ and $WIDs$ in CAN, where $CIDs$ are a set of bases to calculate the similarity, and $WIDs$ are a set of CAN IDs in a sliding window, W, of a fixed number of messages.

$$\text{overlap}(CIDs, WIDs) = \frac{|CIDs \cap WIDs|}{\min(|CIDs|, |WIDs|)} \tag{4}$$

If we use normal CAN messages to calculate the similarity, $CIDs$ and $WIDs$ probably possess several elements of the same CAN ID. Due to this fact, $CIDs$ and $WIDs$ are multisets. Moreover, Equation (4) can be transformed into Equation specified in (5) because $|CIDs|$ and $|WIDs|$ are always same as $|W|$.

$$\text{overlap}(CIDs, WIDs) = \frac{|CIDs \cap WIDs|}{|W|} \tag{5}$$

We use Equation (5) to calculate the similarity in the proposed similarity-based IDS. Also, note that $|W|$ is a constant value in the On-line detection phase, if the denominator of Equation (5) is incremented whenever a CAN message is received, the Overlap coefficient can be calculated in $\mathcal{O}(1)$. It is a smaller value than $\mathcal{O}(\log(|N|))$ of the entropy calculation of the entropy-based IDS, where N is the number of unique CAN ID included in a sliding window.

Next, we measure similarity under different sliding windows as a preliminary experiment, in which we used preliminary CAN messages that are composed of 1000 messages of both normal and DoS attacks. These preliminary CAN messages

of the first half are normal ones, and the rest is the DoS attack messages of CAN ID `0x000`. We use CAN IDs optimized by the Off-line learning phase as $CIDs$ to calculate similarity. From the experiments, the Off-line learning phase optimally selects a sliding window size in the range of `[0d5, 0d50]`.

4.2.2 Framework of Similarity-Based IDS

The similarity-based IDS has two phases, an Off-line learning phase and an On-line detection phase (Fig. 18).

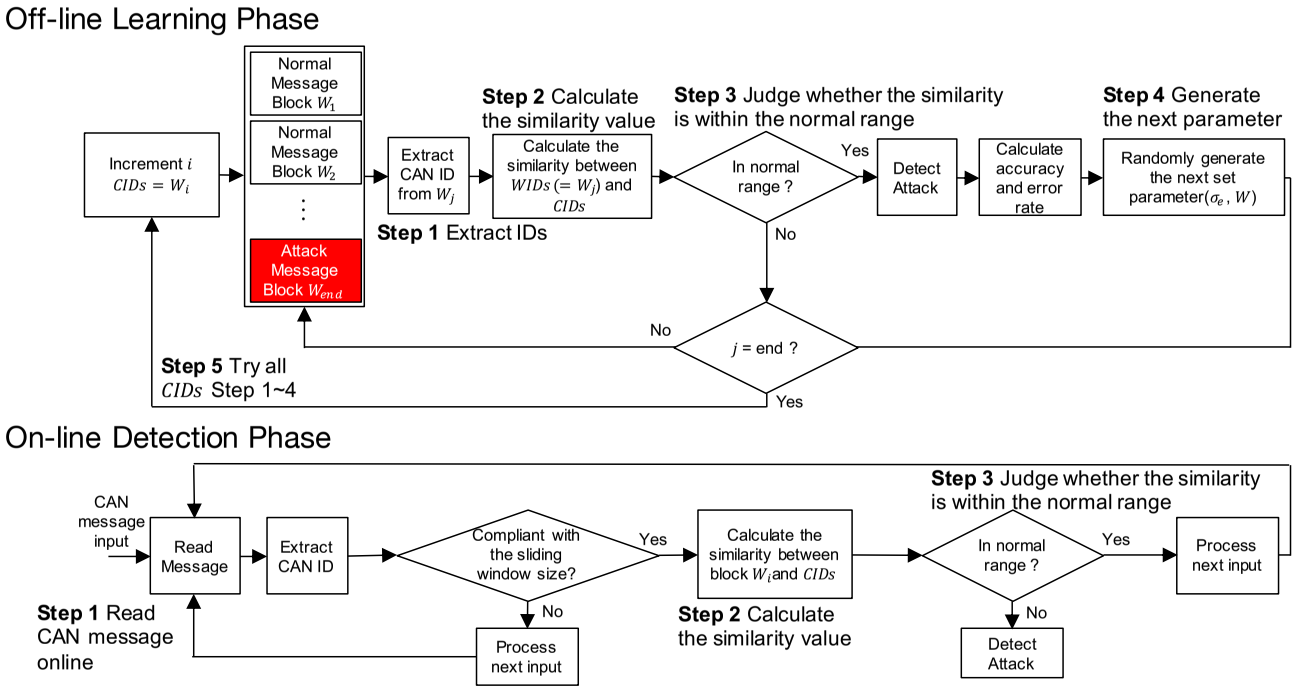


Figure 18: Flow of the similarity-based IDS.

In the first phase, the SA algorithm is used to collect optimal parameters; in the later phase, we detect anomalies by using the optimal parameters collected in the first phase.

1. Off-line learning phase
   The Off-line learning phase mainly includes the following steps.

   - **Step 1:** Extract the CAN IDs from learning traffic.
   - **Step 2:** Calculate the similarity between $WIDs$ and $CIDs$. $WIDs$ are a set of CAN IDs of a certain window. $CIDs$ are a set of normal CAN IDs defined for intrusion detection. Also, $CIDs$ serves as a criterion to calculate the similarity.
   - **Step 3:** Judge whether the similarity (details are shown in Section 4.2.1) calculated in Step 2 falls within the normal range randomly generated by the SA algorithm.
   - **Step 4:** Select the new deviation of the normal range and the sliding window for the next loop.
   - **Step 5:** Try all the $CIDs$ in the CAN messages for learning after that determine the $CIDs$ to calculate the best score. Steps 1-4 are designed based on the SA algorithm. Step 5 is designed to try all the $CIDs$ through steps 1-4 for high accuracy.

2. On-line detection phase
   The On-line detection phase mainly includes the following steps.

   - **Step 1:** Read a CAN message on-line and collect messages until the number of messages is the same as the size of the sliding window.
   - **Step 2:** Calculate the similarity between $WIDs$ and $CIDs$.
   - **Step 3:** Judge whether the similarity calculated in Step 2 falls within the normal range generated in the Off-line learning phase.

4.2.3 On-Line Detection Phase

The proposed algorithm used for the On-line detection phase is depicted in Algorithm 1. In the algorithm, I is a set of CAN IDs from one sliding window

Algorithm 1 Similarity-Based Intrusion Detection Algorithm (On-line detection phase)

Input: $I \Leftarrow \{message_1, message_2, ..., message_W\}, k, \sigma_s, W, CIDs$

1: $u_s \Leftarrow 0.8$
2: **while** True **do**
3: $WIDs \Leftarrow$ extract_CANID(I)
4: Calculate similarity S according to overlap($WIDs, CIDs$) based on Equation (5)
5: **if** S not in normal range $[u_s - k\sigma_s, u_s + k\sigma_s]$ **then**
6: Detect DoS attacks
7: **end if**
8: **end while**

W. The remaining parameters are optimized in the Off-line learning phase. During the On-line intrusion detection phase, the in-vehicle network is monitored in real-time in units per sliding window, W.

The details of the On-line detection phase are as follows:

1. In line 1, we define the average of similarity $u_s = 0.8$. This average has been measured by the result of the average similarity of six car models.

2. In lines 2-4, we calculate the similarity value in sliding window W.

3. In lines 5-7, we judge whether the similarity value is within the normal range.

As described in Section 4.2.1, the time complexity in calculating similarity is $\mathcal{O}(1)$. Thus, the time complexity of Algorithm 1 is $\mathcal{O}(|W|)$.

4.2.4 Off-Line Detection Phase

As a parameter for evaluating our similarity-based IDS, we used precision TP/(TP+FP) (True Positive: TP, False Positive: FP). We selected this precision because it gives the main indicators in the IDS. The detection rate of attack messages R_A (TP rate) is calculated according to equation (6).

$$R_A(\%) = \frac{D_A}{T_A} \times 100 \tag{6}$$

where T_A is the total number of DoS attack blocks, D_A is the detected number of DoS attack blocks. Moreover, the detection error rate of attacks R_N (FP rate) is calculated according to equation (7).

$$R_N(\%) = \frac{D_N}{T_N} \times 100 \tag{7}$$

where T_N is the total number of normal message blocks, D_N is the number of normal message blocks detected incorrectly as attacks by the IDS. Also, if the number of normal messages is greater than the number of DoS messages, the block is labeled as normal.

The proposed algorithm used for the Off-line learning phase is depicted in Algorithm 2 and 3.

Algorithm 2 is the algorithm added to calculate precision to Algorithm 1 for the Off-line learning phase. The $Test\_Data$ of input parameters in Algorithm 2 represents the CAN message chronologically sequenced, including the DoS attack blocks. We employ the SA algorithm to optimize parameters in the Algorithm 3. The energy function used in the SA algorithm is as follows.

$$\mathrm{E}() = C_1 \times R_A(\%) - C_2 \times R_N(\%) - C_3 \times W \tag{8}$$

where E() represents the efficiency of the TP rate, the FP rate, and the detection time. E() is based on Equations (6), (7), and sliding window W. Three weighted parameters C_1, C_2, C_3 are used to adjust the weights to the characteristics of IDS. To get high precision and fast detection time, we set $C_1 = 1.0, C_2 = 0.5, C_3 = 2.0$, which are the same values as in the entropy-based IDS [98] and the sliding window W is in the range of `[5, 0d50]`.

The $Learning\_Data\_with\_DoS\_attack$ of input parameters in Algorithm 3 represents the CAN messages of time sequence, including the DoS attack blocks. The I is a set of one sliding window W. Algorithm 3 optimizes σ_s, W, and $CIDs$ to achieve high precision and fast detection. $(\sigma_s, W)\_set$ is randomly

Algorithm 2 Modified Similarity-Based Intrusion Detection Algorithm for Offline Learning Phase

Input: $Test\_Data, I \Leftarrow \{message_1, message_2, ..., message_W\}, k, \sigma_s, W, CIDs$
Output: Precision $\frac{R_A}{R_A+R_N}$

$u_s \Leftarrow 0.8$
while I in $Test\_Data$ **do**
  $WIDs \Leftarrow$ extract_CANID(I)
  Calculate similarity S according to overlap($WIDs, CIDs$) based on Equation (5)
  if S not in normal range $[u_s - k\sigma_s, u_s + k\sigma_s]$ **then**
    if The window include number of malicious messages greater than number of normal messages **then**
      $D_A += 1$
    else
      $D_N += 1$
    end if
  end if
end while
Calculate TP rate R_A and FP rate R_N, based on Equation (6) and and Equation (7)
return Precision $\frac{R_A}{R_A+R_N}$

Algorithm 3 Sliding Windows Optimization Algorithm (Off-line learning phase)

Input: $Learning\_Data\_with\_DoS\_attack$,
$I \Leftarrow \{message_1, message_2, ..., message_W\}$

Output: $(\sigma_s, W)\_set_{max}, CIDs_{max}$

function neighbor(σ_s, W)
  return random$(\sigma - 0.5, \sigma + 0.5)$, random$(W - 10, W + 10)$
end function
function probability$(e1, e2, T)$
  return $\exp\left(-(\frac{e2-e1}{T})\right)$
end function
while I in $Learning\_Data\_with\_DoS\_attack$ **do**
  $CIDs \Leftarrow$ extract_CANID(I), $k \Leftarrow 0.8$, $T \leftarrow 10000$, $cool \leftarrow 0.99$
  $\sigma_s\_best \Leftarrow \sigma_{e0}$, $W\_best \Leftarrow W_0$, $e_{best} \Leftarrow \mathrm{E}((\sigma_s, W)\_set_0, CIDs)$
  while $T > 0.0001$ and $e_{best} > e$ **do**
    $(\sigma_s, W)\_set_{next} \Leftarrow$ neighbor$((\sigma_s, W)_set)$
    $e_{next} \Leftarrow \mathrm{E}((\sigma_s, W)\_set_{next}, CIDs)$ Calculated by Algorithm_2 $(Learning\_Data\_with\_DoS\_attack, k, (\sigma_s, W)\_set_{next}, CIDs)$
    $p =$ probability(e, e_{next}, T)
    if random() $< p$ **then**
      $(\sigma_s, W)\_set \Leftarrow (\sigma_s, W)\_set_{next}; e \Leftarrow e_{next}$
      if $e > e_{best}$ **then**
        $(\sigma_s, W)\_set_{best} \Leftarrow (\sigma_s, W)\_set; e_{best} \Leftarrow e$
      end if
    end if
    $T \Leftarrow T \times cool$
  end while
  $precision_{best} \Leftarrow$ Algorithm_2 $(Learning\_Data\_with\_DoS\_attack, k, (\sigma_s, W)\_set_{best}, CIDs)$
  if $precision_{max} < precision_{best}$ **then**
    $precision_{max} \Leftarrow precision_{best}$, $(\sigma_s, W)\_set_{max} \Leftarrow (\sigma_s, W)\_set_{best}$,
    $CIDs_{max} \Leftarrow CIDs$
  end if
end while
return $(\sigma_s, W)\_set_{max}, CIDs_{max}$

generated, where σ_s is the deviation, k is the sensitivity deviation, and u_s is the average similarity value. The sensitivity deviation k is 0.8 the same as the average similarity value. The purpose of Algorithm 3 is to obtain the parameter settings that can effectively maximize E(). The details of the Off-line learning phase are as follows:

1. In lines 1-6, functions neighbor() and probability() are defined and later used in lines 17 and 19 respectively.
2. In lines 8-27, we execute the SA algorithm to optimize σ_s, W.
3. In lines 28-33, we calculate $precision_{best}$ using $(\sigma_s, W)\_set_{best}$, and then compare $precision_{best}$ with $precision_{max}$, thereby getting the parameters $(\sigma_s, W)\_set_{best}$ with the highest precision among all $CIDs$.

Since the time complexity of Algorithm 1 is $\mathcal{O}(|W|)$, the time complexity of Algorithm 3 is $\mathcal{O}(|S| \times |T| \times |W|)$, where $|T|$ is the temperature in the SA algorithm, $|S|$ is the number of normal blocks in $Learning\_Data\_with\_DoS\_attack$.

4.3 Evaluation

4.3.1 Precision against Various DoS Attacks

In this section, we evaluate the precision of the similarity-based IDS against each of the DoS attacks. In actual automobiles, the FP rate in the IDS should be low. In other words, the similarity-based IDS is expected to have a high TP rate and a low FP rate. Therefore, we selected a TP rate, an FP rate, and the precision (TP/(TP+FP)) as the evaluation indicators of the similarity-based IDS. Table 5 shows the evaluation indicators of the entropy-based IDS and the similarity-based IDS against Traditional, Randomized, and Targeted DoS attacks.

Table 5: Comparison of the precision at each DoS attack.

	Entropy-based IDS [98]			Similarity-based IDS		
	R_A[%]	R_N[%]	Precision[%]	R_A[%]	R_N[%]	Precision[%]
Traditional DoS	100.0	0.0	100.0	100.0	0.0	100.0
Randomized DoS `[0, 0d31]`	94.6	44.0	68.3	100.0	0.0	100.0
Targeted DoS	100.0	0.0	100.0	100.0	0.0	100.0

Also, we evaluate the entropy-based IDS and the similarity-based IDS using only the HCR Lab data set[3]. Table 5 only shows the precision against the Randomized DoS attack with the range of `[0, 0d31]` in both methods, because this range is suitable as the example of entropy-manipulation attacks.

Table 5 shows that the entropy-based IDS can detect DoS attacks in which an adversary uses a single CAN ID at the high TP rate and high precision. However, the precision of the entropy-based IDS against Randomized DoS attacks is 68.3%. Also, the average and standard deviation of entropies of Randomized DoS attacks are $H_{ave} = 3.08535$, $H_{dev} = 0.07863$, and these are almost the same as the average and standard deviation of the HCR Lab's data set in Table 3. Hence, we confirm that the entropy-based IDS cannot correctly classify a Randomized DoS attack.

On the other hand, Table 5 shows that the similarity-based IDS can detect all DoS attacks with a high TP rate and precision. Therefore, it is confirmed that the similarity-based IDS has superior precision against Randomized DoS attacks, as compared with the entropy-based IDS, and has the same precision as other methods for the other types of DoS attacks. Also, this evaluation made it clear that the similarity-based IDS with the below parameters can detect all DoS attacks in the HCR Lab's data set.

$$
\begin{aligned}
W =& 25, \\
\sigma_s =& 0.52499, \\
CIDs =& \{\texttt{0x80,0x80,0x81,0x81,0x153,0x164,0x165,} \\
& \texttt{0x165,0x18f,0x18f,0x220,0x260,0x2a0,0x2b0,} \\
& \texttt{0x316,0x316,0x329,0x370,0x382,0x43f,0x440,} \\
& \texttt{0x4b0,0x4b1,0x545,0x5a2}\}
\end{aligned}
$$

4.3.2 Precision against Various CAN ID Ranges of Randomized DoS Attack

In this section, we compare the similarity-based IDS with the entropy-based IDS when these IDSs are used under an entropy-manipulation attack. We also compare the two, in Fig. 19, for their precision against entropy-manipulation attacks.

[3]We must hide specific CAN IDs of real vehicle data except the data set of HCR Lab because they are not in public from the companies.

Table 6: The experimental environment.

CPU	Broadcom BCM 2837 1.2GHz 64bit quad-core armv7l
RAM	1GB
OS	Debian 8.0
CAN Interface	PiCAN 2

Fig. 19 shows the precision against entropy-manipulation attacks of various ranges, while Table 5 shows the precision against only the entropy-manipulation attack with the range of `[0, 0d31]` in both methods. We confirm that the entropy-based IDS has a range in which the precision decreases, whereas the similarity-based IDS can detect all ranges.

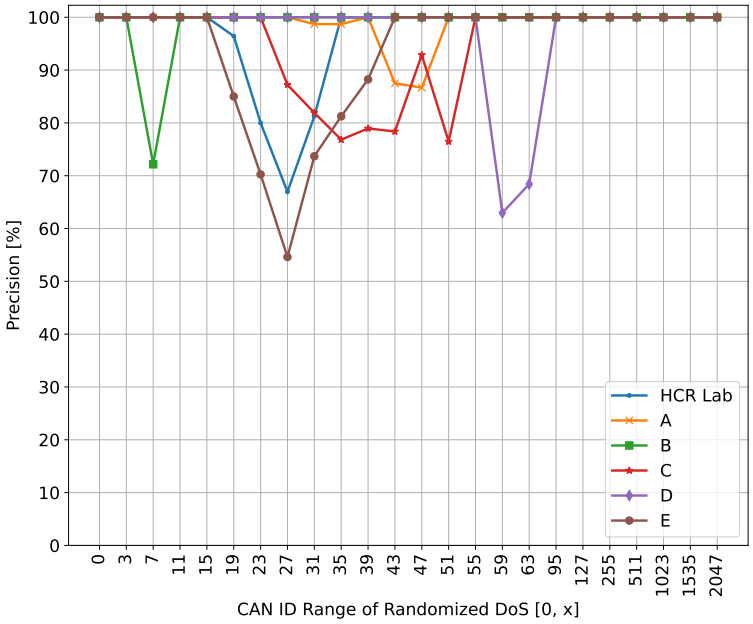
4.3.3 Detection Time

In this section, we compare the detection time of the similarity-based IDS and of the entropy-based IDS in the On-line detection phase. Also, since the Off-line learning phase has nothing to do with real-time detection, we evaluate only the On-line detection phase.

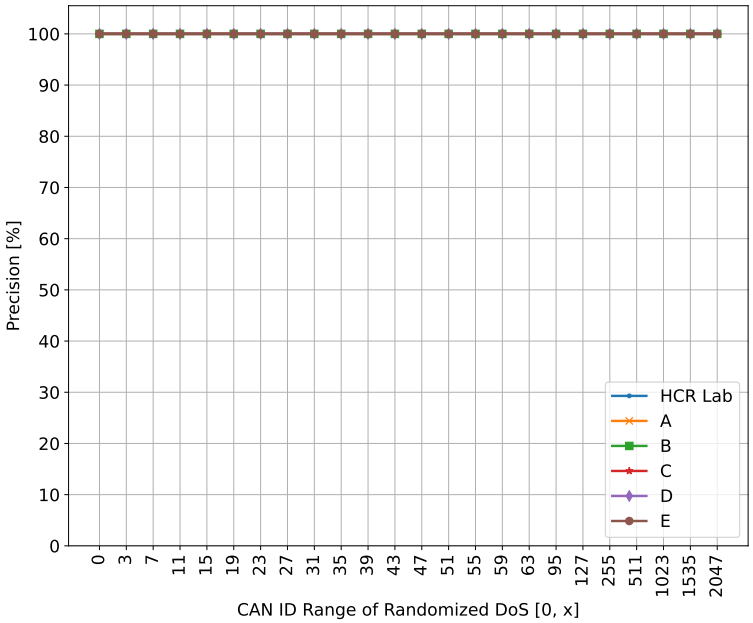
First, we describe the experimental environment (see Table 6). We assumed Raspberry Pi, a low-spec evaluation board, to implement our similarity-based IDS on resource-restricted on-board computers. We also implemented the entropy-based IDS [98] for the comparison between entropy- and similarity-based IDS. Thus, the conventional one was conducted in the same environments of similarity-based IDS for this evaluation.

Next, we define the evaluation time in Fig. 20. The evaluation time T_1 shows the time after the start of the DoS attack until the anomaly is detected. In other words, T_1 is the time that increases in proportion to the sliding windows. The evaluation time T_2 shows the time from receiving W messages as a sliding window until the time of detecting the anomaly. In other words, T_2 is an indicator to compare the calculation times of the entropy and the similarity.

We show the actual requirement in the detection time of similarity-based IDS. In some cars, it is impossible to send messages more often than 10 ms apart due



(a) Entropy-based IDS



(b) Similarity-based IDS

Figure 19: Comparison of precision against entropy-manipulation attack.

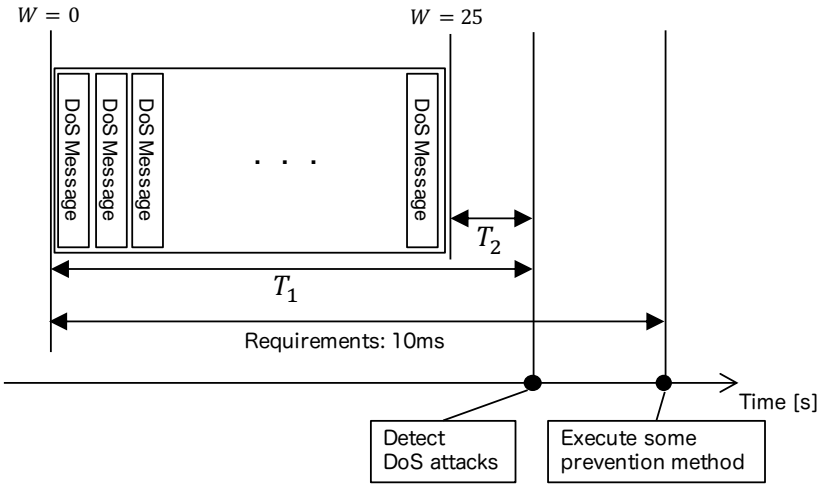
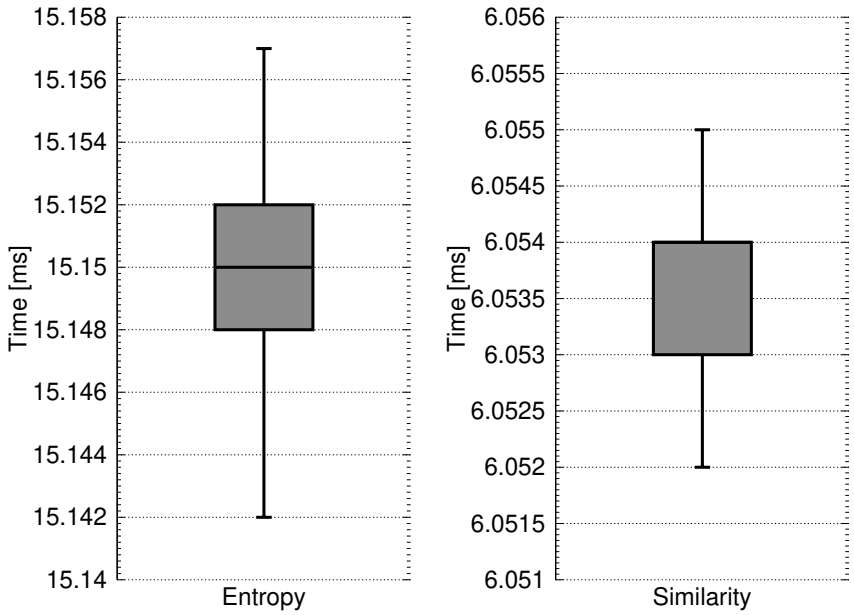


Figure 20: Definition and requirements for detection time in CAN.

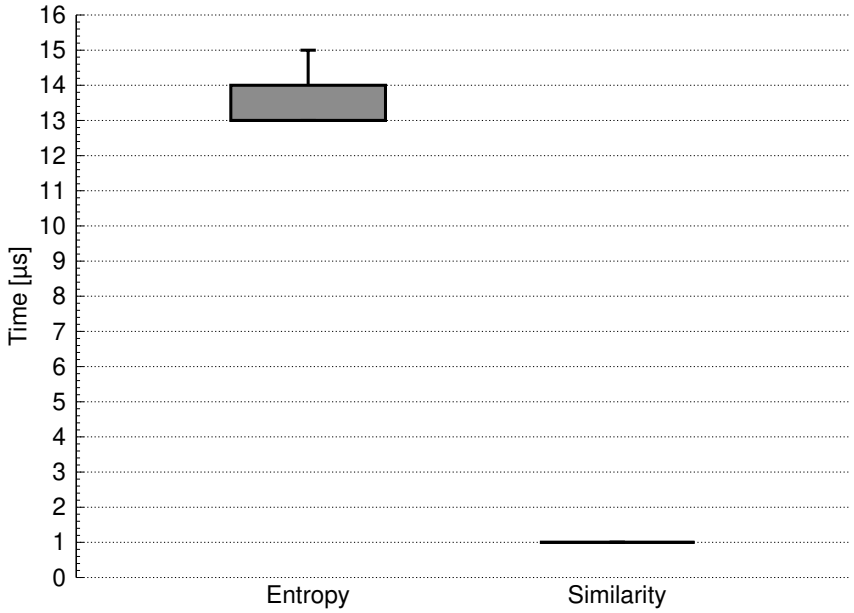
to the load requirements placed by vehicle manufacturers [16]. Therefore, if we can prevent the DoS attacks until 10 ms after starting attacks, the messages can be sent with the correct intervals. Hence, we define a requirement as which the T_1 must be shorter than 10 ms in the similarity-based IDS.

Fig. 21 shows the box-and-whisker diagrams as the results of T_1 and T_2 measured 1000 times each. Also, the median of T_1 is 6.054 ms in the similarity-based IDS. In the figure, the detection times T_1 of the similarity-based IDS are lower by one order to the detection times of the entropy-based IDS. As shown in Fig. 21 (a), the T_1 ranges of the similarity-based IDS and of the entropy-based IDS are 6.052-6.055 ms and 15.142-15.157 ms, respectively. This result is caused by the difference that is an optimized sliding window $W = 25$ in the similarity-based IDS, whereas an optimized sliding window $W = 60$ in the entropy-based IDS.

As shown in Fig. 21 (b), the T_2 ranges of the similarity-based IDS and of the entropy-based IDS are 13-15 µs and 1 µs, respectively. The median of T_2 is 14 µs in the entropy-based IDS. We confirmed that the similarity-based IDS could detect an attack up to 93.33% (14 µs) faster than the entropy-based IDS.



(a) T_1



(b) T_2

Figure 21: Comparison of detection time.

4.4 Discussion

In Section 4.3, the similarity-based IDS can detect all DoS attacks in 100.0% precision and with a faster time than the conventional entropy-based IDS by up to 93.33% (14 µs). We discuss these results in this section.

4.4.1 Precision

We found this DoS attack called the entropy-manipulation attack, which bypasses the conventional entropy-based IDS by adjusting the entropy of messages. The proposed similarity-based IDS can detect the entropy-manipulation attack because it can distinguish between the CAN IDs of the entropy-manipulation attack and the normal CAN IDs. As an experimental result, the similarity-based IDS achieved a detection precision of 100.0% against the entropy-manipulation attacks, while the detection precision is 68.3% in the entropy-based IDS. Since an adversary use CAN IDs with higher priority than normal messages in the entropy-manipulation attacks, the similarity decreases between normal messages and the DoS attack. Therefore, as shown in Fig. 19 (b), the similarity-based IDS can detect entropy-manipulation attacks with 100.0% precision.

However, the similarity-based IDS probably not be able to detect a Replay DoS attack which is a combination of replay attacks and DoS attacks, because an adversary can inject the same CAN IDs with normal CAN messages. Since the entropy-based IDS is effective against a Replay attack, a hybrid implementation of the similarity-based IDS and the entropy-based IDS would be effective against Replay DoS attacks.

4.4.2 Detection Time

Due to load requirements placed by the vehicle manufacturers [16], we defined the requirement as which the T_1 must be shorter than 10 ms in the similarity-based IDS. As the result of Section 4.3.3, the T_1 ranges of the similarity-based IDS and of the entropy-based IDS are 6.052-6.055 ms and 15.142-15.157 ms, respectively. The T_1 of the similarity-based IDS meets the 10 ms of the requirement, whereas the entropy-based IDS's T_1 does not meet the requirement. Thus, we confirmed that the similarity-based IDS is superior to the entropy-based IDS in

actual requirements.

As shown in Fig. 21 (b), the T_2 ranges of the similarity-based IDS and of the entropy-based IDS are 13-15 µs and 1 µs, respectively. Note that the time complexity of the entropy-based IDS is $\mathcal{O}(|W| \times \log(|N|))$, and the computational complexity of the similarity-based IDS is $\mathcal{O}(|W|)$ in the On-line detection phase. Hence, we showed that the detection time is up to 93% (14 µs) shorter than with the entropy-based IDS in Section 4.3.3.

Incidentally, in ID-Hopping Mechanism [39, 99, 96] which avoids Targeted DoS attacks, the average overhead required for AES encryption to generate a one-time ID and for newly setting the CAN ID register are 20.23 µs and 0.2 µs respectively. Therefore, the impact of achieving rapid IDS 14 µs faster than the entropy-based IDS is worth to cancel the overhead for utilizing the conventional IDS and the ID-Hopping Mechanism together.

4.4.3 Comparisons

Some IDSs proposed so far has a good advantage in term of effectiveness to DoS attacks and the small computational overhead. In the following, these IDSs and the similarity-based IDS are compared. Table 7 shows a comparison of the related works.

Table 7: Comparison of the related works.

	Rule-based IDS	Time-interval IDS [88]	Entropy-based IDS [98]	Similarity-based IDS
Traditional DoS	100%	100%	100%	100%
Randomized DoS	100%	100%	68.3%	100%
Targeted DoS	0%	0%	100%	100%
Different Bandwidth	Applicable	Not Applicable	Applicable	Applicable
Threshold	-	Not Optimized	Optimized	Optimized
Time Complexity	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(\|W\| \times \log(N))$	$\mathcal{O}(\|W\|)$

Incidentally, to compare the similarity-based IDS and various methods, we newly define one of the IDS called Rule-based IDS which detects an attacker based on a white-list or black-list of CAN ID.

First, we compare each IDSs based on three types of DoS attacks. Rule-based IDS can detect Traditional and Randomized DoS attacks using a white-list or black-list because Traditional DoS attacks consisted of messages of CAN ID `0x000`. However, Targeted DoS attacks are bypassed because rule-based IDS judges attack with the white-list. Time-interval based IDS can detect Traditional DoS attacks because time-interval based IDS detects anomaly interval of messages of CAN ID `0x000`. When Randomized DoS attacks have messages of same CAN IDs assigned to the CAN, the time-interval of the CAN IDs is shorter than the regular time-interval. Hence, the time-intervals IDS detects Randomized DoS attacks. However, the time-interval IDS cannot detect Targeted DoS attacks using a non-cyclic CAN ID. The time-interval IDS monitors the CAN IDs which ECUs send periodically. In other words, the time-interval IDS does not monitor non-cyclic CAN IDs. As we confirmed in Section 4.3, the entropy-based IDS can detect Traditional and Targeted DoS attacks. However, the entropy-based IDS has a problem that the FP rate is high against Randomized DoS attacks. While our similarity-based IDS using the similarity of sliding windows has a high precision against both Randomized DoS attacks and the other DoS attacks. As we mentioned in Section 3.3.2, the entropy-based IDS bypasses entropy-manipulation attacks which is a kind of Randomized DoS attack, whereas the similarity-based IDS can detect all DoS attacks. From the comparison above, we confirmed that the similarity-based IDS could only detect DoS attacks of all types.

Secondly, we describe the applicability in different bandwidth of CAN. The rule-based IDS can be used in different bandwidth of CAN because this IDS does not use intervals of messages to detect attacks. Similar to the rule-based IDS, the entropy-based IDS and the similarity-based IDS are applicable because the entropy and the similarity are calculated based on CAN IDs of a fixed number of messages. On the other hand, since the time-interval of messages varies in each bandwidth, time-interval IDS cannot be used in different bandwidth of CAN. Thus, we confirmed that the rule-, entropy-, and similarity-based IDSs have advantages in terms of different bandwidth.

Thirdly, we mention whether each IDSs optimize a threshold to judge DoS attacks. Also, the rule-based IDS detects the attacks based on a white-list or black-list of CAN ID, so that this IDS does not have a threshold to judge attacks. The time-interval IDS has a threshold to detect DoS attacks with high accuracy. An expert must manually select this threshold before the IDS is implemented on the actual CAN bus. In addition, the threshold is experimental rather than theoretical; it is possible that there is an optimized threshold to detect DoS attacks. Both the entropy- and the similarity-based IDSs automatically optimize a threshold to judge attacks using SA. Therefore, there is an advantage to determine a threshold in the rule-, entropy-, and similarity-based IDSs.

Finally, we discuss the time complexity. The rule-based IDS uses a white-list or a black-list to detect intrusions so that the time complexity of this IDS is $\mathcal{O}(1)$. The time-interval IDS calculates the time-interval when received some messages. Since this IDS only needs calculating the time-interval and comparing whether the time-interval is normal to detect attacks, the time complexity is $\mathcal{O}(1)$. Next, the time complexity of the entropy-based IDS's On-line detection phase is $\mathcal{O}(|W| \times \log(N))$. As mentioned in Section 4.2.3, the time complexity of the similarity-based IDS is $\mathcal{O}(|W|)$. Thus, the rule-based IDS and Time-interval IDS have advantages in terms of the time complexity. On the other hand, we evaluated the actual detection time of the similarity-based IDS. As a result, we confirmed that our method satisfies the requirement from the vehicle manufacturers [16]. Therefore, we conclude that the similarity-based IDS can be operated in the actual environment.

From these comparisons among related works, we confirm that the similarity-based IDS can detect the DoS attacks of all types. In addition, it was confirmed that the similarity-based IDS has advantages in terms of applicability in CAN of different bandwidth, determining the threshold, and the detection time.

4.5 Conclusion

The growing number of vehicles connected to the internet causes a security risk of cyberattacks such as DoS attacks on modern automobiles. It requires a security solution that can prevent DoS attacks. To prevent all DoS attacks, firstly we must consider a method to detect all DoS attacks. In this research, we proposed

an optimized DoS attack detection method based on the similarity of sliding windows that is capable of detecting every type of DoS attack. In addition, we have solved the entropy-based IDS' problem of a higher FP rate occurring when the entropy-manipulation attack is executed. Furthermore, our similarity-based IDS has lower computational complexity than the entropy-based IDS. We confirmed that the similarity-based IDS detected a DoS attack in 100% of the cases in our experiment, and we showed that the detection time is up to 93.33% (14 µs) shorter than with the entropy-based IDS. We release the source code [73] hoping to promote research on countermeasures against DoS attacks.

5. PLI-TDC: Physical-Layer Identification with Time-to-Digital Converter for In-Vehicle Networks

5.1 Introduction

In Chap. 4, we proposed similarity-based IDS which can fastly detect DoS attacks with lightweight computing resources. Also, we confirmed that similarity-based IDS can detect undiscovered entropy-manipulation attacks and the other DoS attacks with 100 % accuracy. However, similarity-based IDS cannot identify the sender of DoS messages because it just detects DoS attacks. It is ideal that IDS can identify the ECU attacking the bus in order to patch the compromised ECU. Therefore, in this chapter, we study the sender identification based on the physical-layer characteristic of the ECU.

An IDS based on physical-level characteristics such as delay-time which is a gap time between ideal transition time and actual transition time has been proposed, which is called Divider [79]. Divider identifies sender ECUs based on delay-time measured by a Divider's internal clock. Thus, if different ECU's delay-time have similar variations, this approach may not correctly classify legitimate ECUs because the time-resolution of the internal clock is coarse. Therefore, we should focus on enhancing the accuracy of sender identification. In addition, Divider cannot adapt a drift of delay-time caused by the temperature drift. In this research, we propose super fine delay-time based PLI with TDC. The TDC in our method digitizes the delay-time per 154 ps. The proposed method realizes temperature-robustness by learning the temperature as one of the features. Also, our proposed method can identify the ECUs with higher accuracy than Divider and at the same sampling count as Divider.

The main contributions of this study can be summarized as follows:

- We propose a PLI using TDC called PLI-TDC. Our method uses new characteristics in the identification of ECUs in CAN. PLI-TDC does not use continuous characteristics such as voltage, but the delay-time to be observed in each rising edge of the CAN message. Hence, PLI-TDC can identify the ECUs with a lower number of sampling than the voltage-domain based

method.

- PLI-TDC achieved mean accuracy of 99.67 % and 97.04 % on CAN bus prototype and a real-vehicle network, respectively. Additionally, we showed that this approach achieved a 100% true positive rate against two attacker models.
- We designed PLI-TDC so that it can be robust against features' drift caused by temperature drift. From our experiment, PLI-TDC can achieve a mean accuracy of over 99% even if the temperature is drifted.
- PLI-TDC solved the problems of detection based on multiple frames, number of probes, and robustness against feature drift.

5.2 Related Work

Some authentication mechanisms [37, 72, 33, 65, 42] can ensure the authenticity of ECU sending CAN messages. However, these mechanisms required additional hardware and revising some source codes to encrypt CAN messages. Besides, since the authentication mechanisms must manage the key lifecycle based on PKI, it increases the complexity of the automotive system.

PLI can be applied to CAN protocol without these drawbacks. PLI translates some inconsistencies (e.g., hardware and manufacturing) caused by a minute and unique variations to reliable features that can identify ECUs sending CAN messages [25, 27, 26]. Because we do not need to take revising source code and key management into consideration, PLI only requires an additional node to run acquiring fingerprints and classification algorithms.

The timeline of related PLI researches are summarized in the timeline of Fig. 22.

5.2.1 Voltage Domain Characteristics Based PLI

Murvay et al. proposed a method for sender identification based on physical voltage features in CAN [67]. Moreover, Choi et al. improved Murvay's method in [11]. They embed a fixed bit string into the extended identifier field of the CAN frame and sample the signal and identify ECUs by using 17 different features.

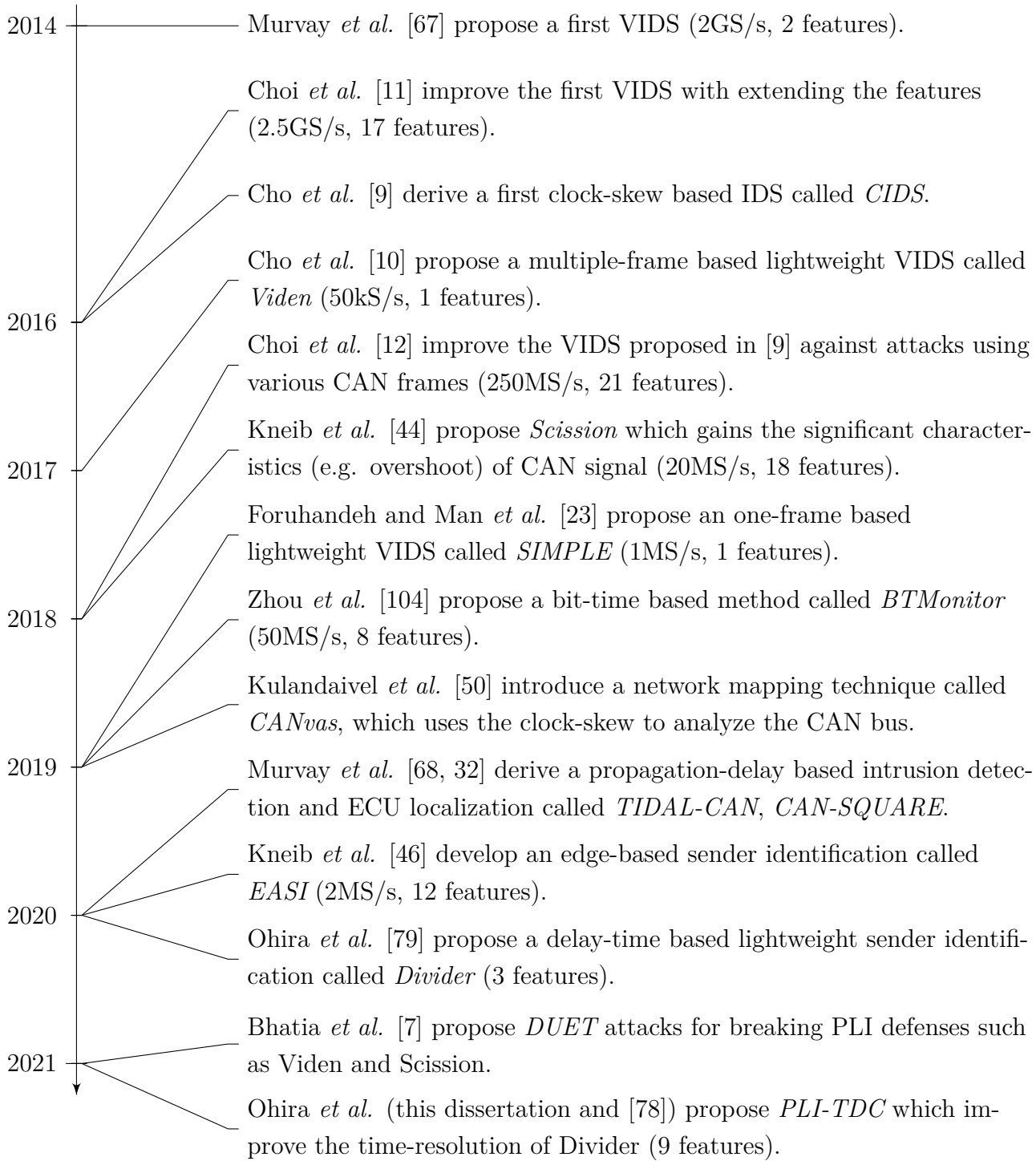


Figure 22: Timeline of PLI researches.

Hence, these methods cannot be implemented on the normal CAN because they require the extended frame format in CAN.

Cho et al. proposed a system for identifying an attacker by using voltage difference among ECUs called Viden [10]. They implemented the system on MCU of a lower sampling rate (50 kS/s) than CAN bus bit rate. Therefore, Viden requires two or three messages to output a voltage instance and updates the profiles. Thus the receiver cannot help rejecting the first forged message.

Besides, since Viden relies on multiple messages to make detection and identification, Viden has vulnerability against the Hill-climbing-style attack [23], in which an attacker sends gradually malicious messages without being either detected or identified. To be robust against the Hill-climbing-style attack, IDS has to detect the attacks using features acquired in one message [11, 44, 45, 23].

Scission [44] solved the problem in the sender identification method proposed by Choi et al. [11] which the method could not get significant characteristics such as the overshoot. Scission achieves 99.85 % of identification accuracy which is higher than that of Viden and the method of Choi et al. However, since Scission uses Fourier Transform to calculate the features of the frequency domain, the time complexity of Scission is $\Omega(n \log n)$ which is higher than the time complexity of SIMPLE [23]. Because SIMPLE only uses means of voltage as a feature of ECUs, the time complexity is $\Theta(n)$. Since these sender identification methods use a result of sampling continuous function, the accuracy of identification depends on the sampling rate. In general, as the sampling rate increases, the accuracy of identification is improved. But the amount of data used for the identification increase too. Hence, IDS which is limited in computing resources on the in-vehicle system needs to be able to identify ECUs with few sampling. Therefore, we focus on the sender identification method using other characteristics with few sampling.

5.2.2 Time Domain Characteristics Based PLI

Table 8 shows a comparison among time-domain PLIs.

Bit-time based PLI called BTMonitor [104] has been proposed. This method achieved a mean accuracy of over 99% based on features extracted from CAN messages ranging from 5 to 50. Similar to Viden, it cannot help in rejecting the first forged message. In addition, a mean accuracy is 90.04% if BTMonitor

Table 8: Comparison among time-domain based physical-layer identifications for CAN.

	BTMonitor [104]	TIDAL-CAN [68]	Divider [79]	PLI-TDC
Accuracy [%]	99.59	100.0	87.20	99.67
Source	Bit -time	Propagation -delay	Transition -time	Transition -time
One frame	no	yes	yes	yes
# of probe	1	2	1	1
Concept drift robustness	yes	no	no	yes

uses the features from one CAN message. Hence, a PLI should achieve a mean accuracy of over 99 % with only one CAN message.

The PLI based on the propagation-delay of wire is caused by wire speed have been proposed [94, 68]. These method require at least two probe points for each CAN. The modern in-vehicle network often divided into multiple parts. Therefore, $2n$ probe points are required for n CANs in this method. It ruins the simplicity of CAN bus due to the requirement of increasing the probe points. In terms of applicability to real-vehicles, the identification method should have one probe point for each CAN. Further, this methods may not be robust against the drift of features caused by temperature change, and so on.

Divider [79] has some advantages in terms of the number of frames used in the detection and the number of probes compared to the other time-domain methods. Divider uses delay-time which is a gap from ideal transition-time to actual transition-time in typical CAN transceiver. If there are some ECUs with similar gaps, Divider cannot correctly classify the sender ECU. Also, Divider can distinguish the ECUs with a mean accuracy of 87.20 % in the case of time-resolution 20 ns. Therefore, Divider has a disadvantage in classification accuracy compared to BTMonitor and TIDAL-CAN. It is required that time-resolution in Divider is

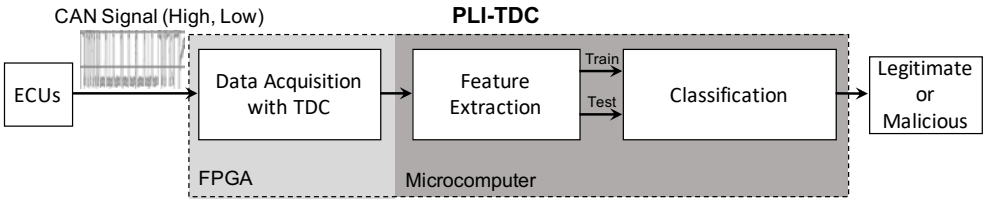


Figure 23: Proposed physical-layer identification.

improved to overcome the problem of classification accuracy. In addition, similar to TIDAL-CAN, Divider is not robust against the drift of features.

In the next section, we propose a fine time-resolution TDC based PLI which has higher accuracy than Divider and overcomes the problems such as needing multi frames and intolerance of temperature change. To solve these problems, we improve Divider by data acquisition using fine time-resolution TDC and adding temperature information as one feature.

5.3 Super Fine Delay-Time Based Physical-Layer Identification

As with general PLI [93], PLI-TDC consists of three phases, data acquisition with TDC, feature extraction and classification, as shown in Fig. 23. In the data acquisition phase, the delay-time is acquired as a digital value using TDC. In the feature extraction phase, the delay-time obtained from TDC is converted into statistics such as average, variance, and so on. Finally, it classifies the source ECU of the CAN message to judge whether the ECU is legitimate or malicious. In the following sections, we describe each phase in order.

5.3.1 Data Acquisition with TDC

The data acquired by PLI-TDC is the delay-time in the rise- and fall- times of the CAN signal, and its definition and observation method are described below.

Definition of delay-time

The delay-time used in PLI-TDC is the same as Divider's delay-time [79] which is the gap time between actual transition-time and ideal transition

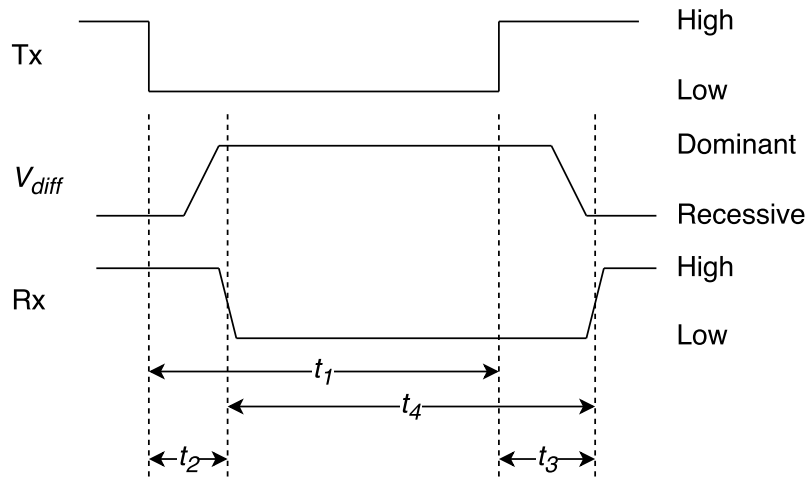


Figure 24: Delay model in CAN.

time of the signal in CAN transceiver. The Divider's delay-time is caused by the load capacitance of the transistor in CAN transceiver. And the factors of load capacitance include three types of output capacitance at the gate of the transistor, input capacitance of the gate and wiring capacitance. Here, the cause of the delay-time and the equation for calculating the delay-time are described.

A delay model in CAN is showed in Fig. 24. The time variables have the following relation. From the definition, we obtain the following equation.

$$t_3 - t_2 = t_4 - t_1 \tag{9}$$

Here, the time actually measured at the IDS is only t_4, and t_1, t_2, t_3 are unknown. Therefore, we use an approximation. From [79], since t_1 is regarded as t_{bit} which is ideal bit time, we obtain the following equation.

$$t_3 - t_2 \approx t_4 - t_{\mathrm{bit}} \tag{10}$$

Also, t_{bit} is 2000 ns in CAN (500 kbps). Hence, PLI-TDC acquire delay-times by observing the $t_4 - t_{\mathrm{bit}}$.

TDC based measurement of delay-time

The TDC is a time digitizer used for physics experiments and time-of-flight

(ToF) technique [89]. An IC containing TDC is sold as a product for about $23.80 [3]. However, the IC does not satisfy a requirement for PLI-TDC, because it is necessary to directly control TDC to match the arbitration ID and measured time in PLI-TDC. On the other hand, the methods of implementing TDC at low cost using Field-Programmable Gate Array (FPGA) have been studied [89, 97]. Also, an arbitration ID and delay-time can be easily matched by implementing TDC in FPGA. Therefore, TDC on FPGA is used to observe the delay-time in CAN with high time-resolution in PLI-TDC.

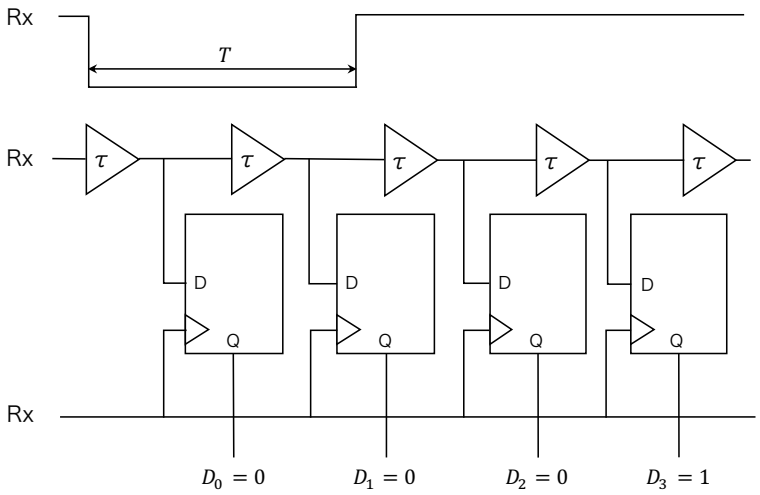
First, we describe the composition of CMOS based TDC (CMOS TDC) which is one of the typical TDC [4]. Fig. 25 (a) shows CMOS TDC circuit. The CMOS TDC is implemented by D-type flip-flops and delay-cells causing a little delay τ ps. Here, we explain the operation of CMOS TDC. We define T is the measured time. Further, we suppose that there is a signal Rx (the top of Fig. 25 (a)) whose logical value is 0 during the time T. This signal Rx is inputted to the two inputs of CMOS TDC simultaneously. After inputting, as shown in Fig. 25 (b), the input signal delayed by the delay-cells by τ and propagates to the entire CMOS TDC. If the signal Rx rises, the output of the D-type flip-flop to the output Q and $D_0D_1D_2D_3$ are determined. Thus, the output of CMOS TDC $D_0D_1D_2D_3 = (0, 0, 0, 1)$ is obtained. Here, assuming the delay $\tau = 100\,\mathrm{ps}$, the signal is propagated from the output $D_0D_1D_2D_3 = (0, 0, 0, 1)$ to the third delay cell. Therefore, $T = 3 \times 100\,\mathrm{ps} = 300\,\mathrm{ps}$. Generally, TDC achieves high time-resolution by the above operation.

Next, we describe the implementation method of FPGA based TDC. Song et al. [89] implemented a delay-line using a multi-bit adder in FPGA. Fig. 25 (c) shows the implementation of delay-line using a multi-bit adder in Tapped-Delay TDC. The boolean equation of each adder is as follows.

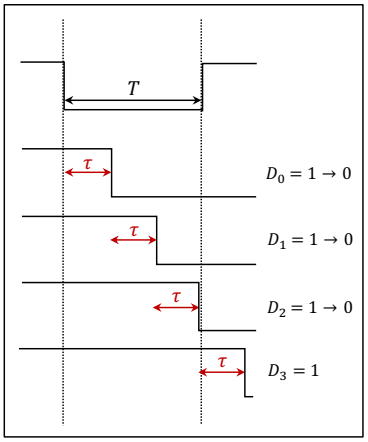
$$S = A \oplus B \oplus C_{\mathrm{i}} \tag{11}$$

$$C_{\mathrm{o}} = AB + (A + B)C_{\mathrm{i}} \tag{12}$$

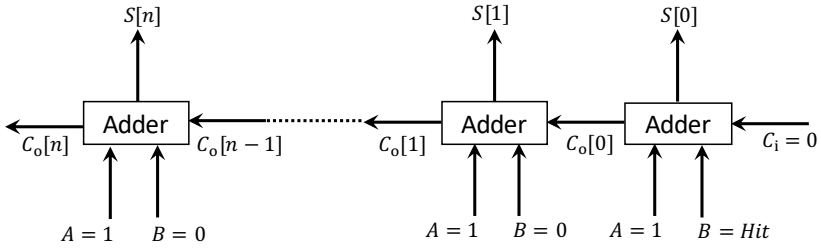
where A and B are the input of the adder, and C_{i} (carry-in bit) is the input carried from the previous adder, C_{o} (carry-out bit) is the output carrying to



(a) Circuit of CMOS TDC.



(b) Timig-chart of CMOS TDC.



(c) Delay-line in FPGA-based TDC.

Figure 25: Implementation method and timig-chart of TDC.

the next adder, and S is the result of the addition. Therefore, the delay-line propagate $C_{\mathrm{o}}[0] = 1$ when a signal Hit becomes $Hit = 1$. And then, The output case S of each adder becomes 0.

We describe that the time-resolution performance of the implemented FPGA based TDC. When a 20 ns pulse was input to the TDC, the signal was transmitted to the 92 delay element. Similarly, when a 40 ns pulse was measured, the signal was transmitted to 183 delay elements. From this results, a delay with one delay-cell is $\frac{40-20}{183-92}$ns = 219 ps. Then, we calculated the root mean square error (RMSE) between the actual value and the true value 20 ns. And we measured the 20 ns pulse in 50000 times. As a result, it was obtained that the RMSE was 154.011 ps. Therefore, the implemented TDC has a time resolution of 154 ps.

The delay-time experimentally observed by TDC is shown in Fig. 26. Six arbitration IDs are plotted from two ECUs. The arbitration IDs of ECU **a** is plotted around 50 ns, and the arbitration IDs of ECU **b** is plotted around 110 ns. Therefore, we confirm that sender identification is possible regardless of the arbitration ID of the CAN message sent from the two ECUs.

Measurement period

As we showed in Fig. 4, length of the data frame on CAN is variable and it is set in the DLC field. Therefore, even if the length of the CAN frame is the shortest (DLC=0), it is necessary to reliably be able to measure the section transmitted by the target node. Then, considering CAN frame such as DLC=0, 35 bits of signal of SOF (1 bit), the arbitration field (12 bits), the control field (6 bits) and CRC filed (16 bits) are transmitted by the ACK field. Here, if we include the CRC delimiter to the measurement period, the rising edge of the ACK slot may be measured. We subtract 1 bit from the 35 bits. Hence, we set the measurement period from SOF to time that passing 34 bits time (68 µs). Since the length of 1 frame is not shorter than the CAN frame in case of DLC=0, this allows us to reliably measure only the signal of the target node. Also, during the measurement of delay-time, the time capture is performed every rising edge of the Rx pin.

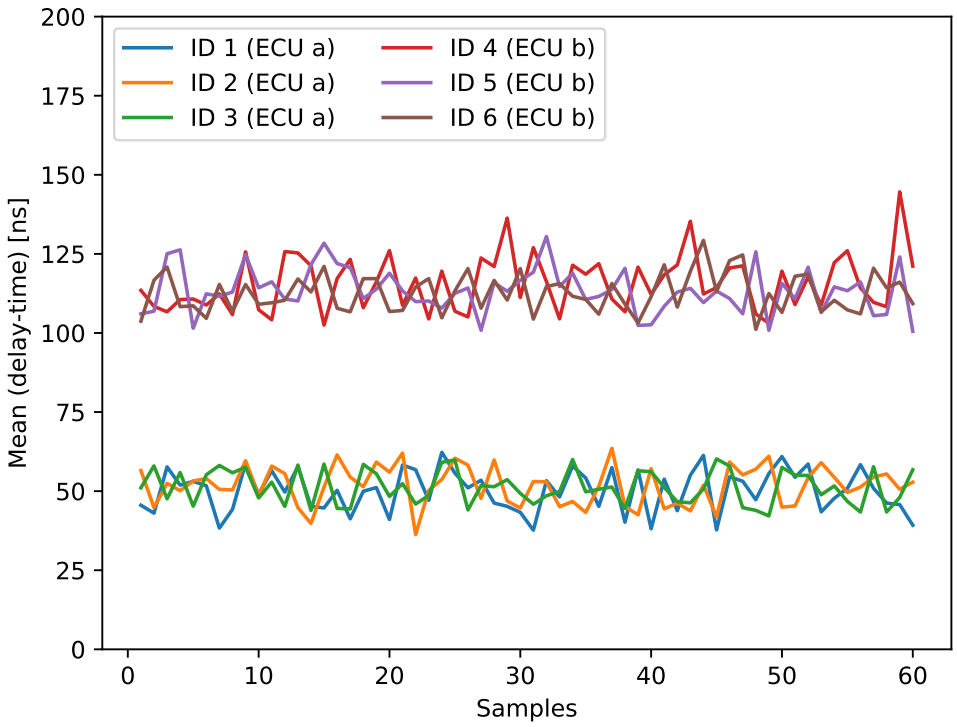


Figure 26: An example of two ECUs' delay-times observed by Time-to-Digital Converter.

We describe how to obtain delay-time, t_{delay} from the measured counter value. As the unit of timer counter value is 0.154 ns, The elapsed time from SOF, t_{elapsed} (ns) can be calculated as:

$$t_{\mathrm{elapsed}} = (\text{capture counter value} - \text{SOF counter value}) \times 0.154 \tag{13}$$

The value of elapsed bits from the SOF at each rising edge can be calculated as follows:

$$\lfloor \frac{t_{\mathrm{elapsed}} + 500}{2000} \rfloor \tag{14}$$

where, 500 is added in the numerator to round t_{elapsed} by 1000 ns, 2000 is the value of t_{bit} in ns. Also, 500 is offset to obtain the correct elapsed bits. And the ideal value of elapsed bits can be obtained with floor function.

Therefore, the ideal elapsed time from SOF, t_{ideal} (ns) can be calculated as follows:

$$t_{\mathrm{ideal}} = \lfloor \frac{t_{\mathrm{elapsed}} + 500}{2000} \rfloor \times 2000 \tag{15}$$

t_{delay} (ns) we want to calculate is:

$$t_{\mathrm{delay}} = t_{\mathrm{elapsed}} - t_{\mathrm{ideal}} = t_{\mathrm{elapsed}} - \lfloor \frac{t_{\mathrm{elapsed}} + 500}{2000} \rfloor \times 2000 \tag{16}$$

Here, we describe the detail of the relation between Equation (10) and (16). Fig. 27 shows an example of the relation in the SOF bit of CAN message. In Fig. 27, the actual signal is observed by IDS and the other ECUs with an unavoidable delay, t_{delay}, which is a gap from 2000 ns of the ideal signal timing.

We confirm that t_{delay} equals $t_4 - t_{\mathrm{bit}}$. First, t_{elapsed} equals t_4 because these are times of the dominant signal observed by IDS and the other ECUs. Next, we explain the relation between t_{bit} and t_{ideal}. t_{bit} is an ideal elapsed bit time, and it is 2000 ns in SOF field of CAN message. In addition, t_{ideal} indicates the ideal rise time estimated from t_{elapsed}. For example, in case of $t_{\mathrm{elapsed}} = 2050$ ns, t_{ideal} is 2000 ns according to Equation (15). In other words, if t_{ideal} can be estimated correctly, then t_{bit} equals t_{ideal}. Therefore, $t_4 - t_{\mathrm{bit}}$ and t_{delay} are equal.

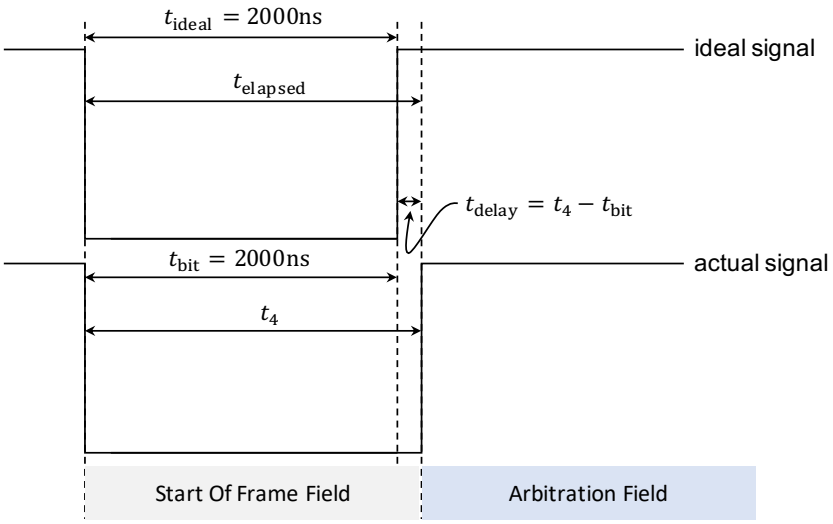


Figure 27: Relation between Equation (10) and (16).

5.3.2 Feature Extraction

Similar to Scission [44], PLI-TDC selects efficient features from the statistics in Table 9. In order to determine the efficient features, the features are ranked using Relief-F [47], which is an algorithm that calculates the weight of the features.

Table 10 shows the result of Relief-F in data obtained from a CAN bus prototype and a real-vehicle's bus. In order to reduce the complexity of the model and the time required to calculate the features, PLI-TDC uses only the features which weight of Relief-F is 0.01 or more in both the CAN bus prototype and the real-vehicle. As a result, eight statistics except energy and variance are selected. In the following, the eight statistics are defined as features.

5.3.3 Classification

Sender identification results in a classification problem. In PLI-TDC, the mean accuracy of various learning algorithms is evaluated. And an algorithm with the highest mean accuracy is used in the classification phase of PLI-TDC.

Here, we describe a comparison of the various learning algorithm. We compare typical learning algorithms composed of function values, distances, trees, and so on. The comparison is summarized in Table 11. The abbreviations express Logis-

Table 9: A list of statistical features considered in the selection. x is the delay-time in one CAN message, N is the number of measured delay-time in one CAN message.

Feature	**Description**
Mean	$\mu = \frac{1}{N}\sum_{i=1}^{N} x(i)$
Standard Deviation (Stdev)	$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x(i)-\mu)^2}$
Variance	$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x(i)-\mu)^2$
Skewness	$skew = \frac{1}{N}\sum_{i=1}^{N}(\frac{x(i)-\mu}{\sigma})^3$
Kurtosis	$kurt = \frac{1}{N}\sum_{i=1}^{N}(\frac{x(i)-\mu}{\sigma})^4$
Root Mean Square (RMS)	$rms = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x(i)^2}$
Max	$max = \max(x(i))_{i=1\ldots N}$
Min	$min = \min(x(i))_{i=1\ldots N}$
Energy	$en = \frac{1}{N}\sum_{i=1}^{N} x(i)^2$

Table 10: Ranking of the features calculated by Relief-F [47].

Rank	CAN bus prototype	Weight	real-vehicle	Weight
1	Mean	0.11025	Stdev (fine time)	0.09311
2	Min	0.08773	Mean	0.05028
3	RMS	0.05644	RMS	0.04833
4	Max	0.04696	Min	0.04613
5	Kurtosis	0.03398	Kurtosis	0.04090
6	Stdev (fine time)	0.02949	Skewness	0.03694
7	Skewness	0.02307	Max	0.02468
8	Stdev	0.01282	Stdev	0.01746
9	Energy	0.00878	Energy	0.01639
10	Variance	0.00104	Variance	0.00723

Table 11: Comparison of machine learning algorithms for PLI-TDC

	Classification Speed	Training Speed	Model Adjustment	Overall Complexity
LR	✓	×	✓	✓
Naive Bayes	✓	✓	✓	✓
MLP	×	×	×	×
KNN	×	✓	✓	×
Decision Tree	✓	✓	×	✓
Random Forest	✓	✓	×	✓
SVM (RBF)	×	×	✓	×

tic Regression (LR), Multi-Layer Perceptron (MLP), K-Nearest-Neighbor (KNN), Support Vector Machine (SVM), Radial Basis Function (RBF), respectively. In PLI-TDC, classification speed is essential to identify all messages. LR, Naive Bayes, Decision Tree, and Random Forest meet this requirement. In addition, in in-vehicle networks, some features' drift may be caused by material wear and temperature fluctuations. Therefore, the model should be tolerant of the drift. LR and Naive Bayes may do, but Decision Tree and Random Forest may not. On the other hand, Decision Tree and Random Forest have the advantage of fast training speed in case that the number of trees is few. Therefore, the Decision Tree and Random Forest can adapt the drift by deploying the new model per fixed times. Therefore, PLI-TDC uses an algorithm with the highest accuracy among these four faster algorithms.

5.3.4 Enhancing the Concept Drift Robustness

In some voltage-based sender identification methods [10, 9, 23, 104], it has been confirmed that the sender's features such as voltage are changed by the drift of temperature. Therefore, we must investigate whether or not the change of delay-time is caused by drift of temperature. We conducted an experimental investigation to determine whether the delay-time has a drift of temperature in Sec. 5.4.4. As a result, some ECUs had a delay-time that increases monotonically and some ECUs did not show a change in the delay-time. Therefore, PLI-TDC must

be robust against the drift of temperature as in the voltage-based source identification method. The approaches to avoid drift of temperature in the voltage-based PLI is as follows.

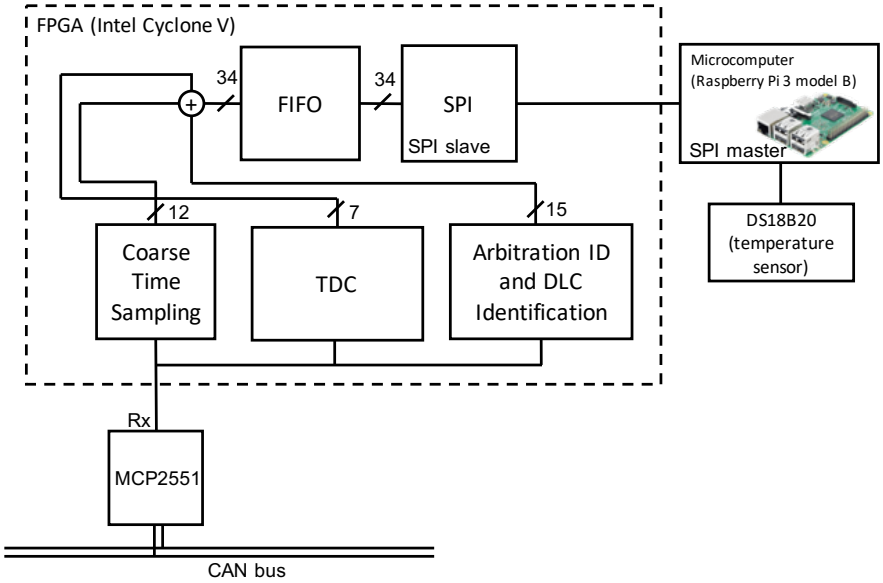
1. Correction of features with liner regression [23]
2. Detection with multi models [104]
3. Tracking to features' drift [10, 9]

The first method employs linear regression. It cannot be applied to PLI-TDC because the delay-time does not increase linearly with the drift of temperature as described in Sec. 5.4.4. The second method uses multiple models. It makes the memory usage in a resource-limited system increase compared to one model. Thus, it is ideal to avoid the drift with one model. The third feature tracking method which is used in CIDS [9] and Viden [10] is vulnerable to Hill-climbing-style attack, because they use several CAN message lastly received for learning [23]. Therefore, in the PLI-TDC, the temperature is added as one of the features.

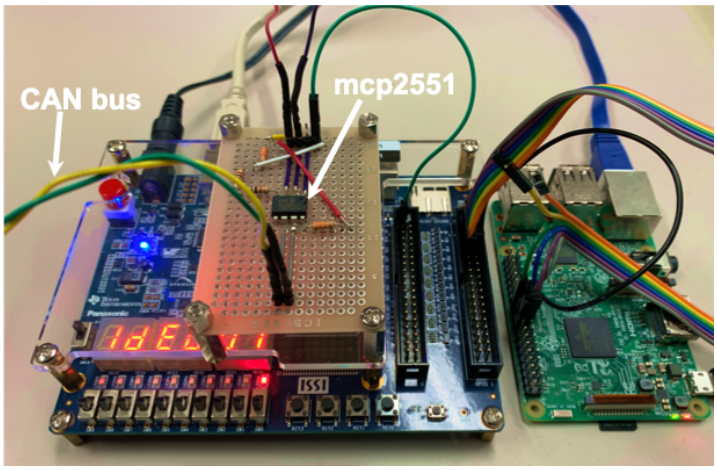
5.3.5 Implementation

In this section, we describe the implementation of PLI-TDC. As mentioned in Sec. 5.3.1, PLI-TDC measures the 34 bits to observe delay-time no matter what length of the data field is received. We show the block diagrams of the implementation of PLI-TDC in Fig. 28 (a). The MCP2551 is a CAN transceiver as the interface between a CAN controller and the physical bus. We also selected an FPGA as a measurement device, because a measurement with software cannot process all messages without missing ones due to the limitation of the ability of microcomputer. We show the prototype of PLI-TDC in Fig. 28 (b). We developed the prototype of the proposed method using FPGA and microcomputer, selected the DE0-CV Cyclone V Board (5CEBA4F23C7) as an FPGA and Raspberry Pi 3 Model B+ as a microcomputer.

Here, we describe the circuits of FPGA in PLI-TDC. The circuits are divided into five operations as follows.



(a) Implementation of PLI-TDC.



(b) Prototype of PLI-TDC.

Figure 28: Implementation and prototype of PLI-TDC.

- Coarse Time Sampling Circuit
  This circuit measures a period of CAN message in the measurement period of 34 bits with counting per 20 ns.

- TDC Circuit
  The TDC circuit measures the period with counting per 154 ps. The coarse time sampling and TDC circuit send counting value to a FIFO queue per Rx rising edge of CAN.

- Arbitration ID and DLC Identification Circuit
  As its name suggests, we observe and store the arbitration ID and DLC of every message. Similar to the coarse sampling circuit, the arbitration ID and DLC sampling circuit sends arbitration ID to FIFO queue per Rx rising edge too.

- FIFO Circuit
  In this circuit, we stack the measured data of 34 bits constructed of an arbitration ID of 11 bits, a DLC of 4 bits, and the counter value of coarse / fine time of 19 bits.

- SPI Slave Circuit
  We implement the SPI slave module to send measurement data to the Raspberry Pi.

The operation of the measurement is as follows.

1. Starting the capture of measurement time and arbitration ID, an occurrence at the falling edge of SOF bit.

2. Send the measurement data (arbitration ID and measurement time like as shown in Fig. 29) with every rising edge of Rx to a FIFO queue. We calculate the delay-time with equation (16) and record the delay-time after Raspberry Pi receives the measurement data from the FPGA.

3. After reading 34 bits from SOF, the measurement is ended.

4. When CAN frame is completely received, the coarse time sampling circuit, TDC circuit, and arbitration ID and DLC identification circuit are waiting SOF bit.

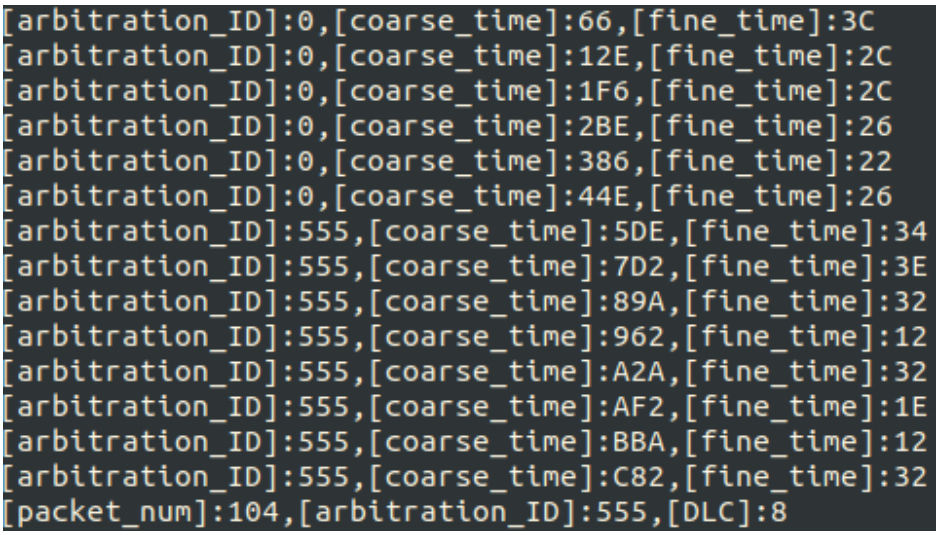


Figure 29: An example of outputted measurement data from PLI-TDC.

Finally, we describe the receiving performance of PLI-TDC. Fig. 30 shows the CAN message loss rate when the CAN bus occupancy of PLI-TDC is changed. Fig. 30 also shows the loss rate when the queue length of the FIFO module in PLI-TDC is 512, 2048, and 8192, respectively. As shown by 30, when the queue length is 512 and 2048, we confirmed that the loss rate increases as the bus occupancy rate increases. On the other hand, In the case that the queue length is 8192, the loss rate is 0 %. Therefore, we experimentally confirmed that PLI-TDC can measure the delay-time without spilling the CAN message even when the bus occupancy rate is 100 %.

5.4 Evaluation

In this section, we describe the evaluation of the results. First, we evaluate the accuracy of the identification of ECUs. Second, we experiment on two attacker models and evaluate the attacker detection performance of PLI-TDC. Third, we confirm the robustness of PLI-TDC under different temperatures. Finally, we measure the detection time of PLI-TDC from the feature extraction phase to the classification phase.

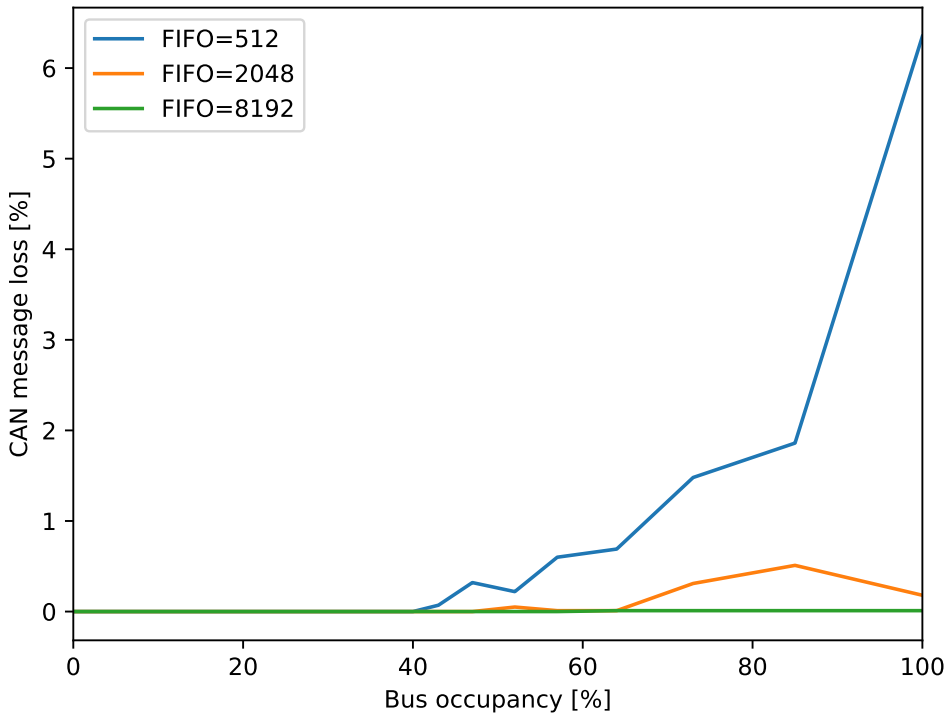
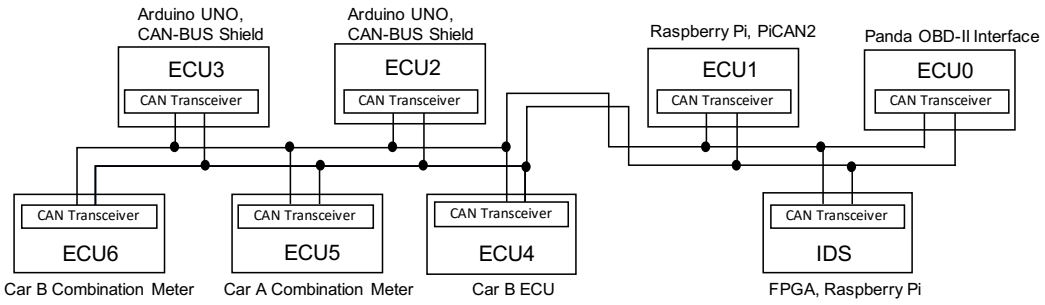
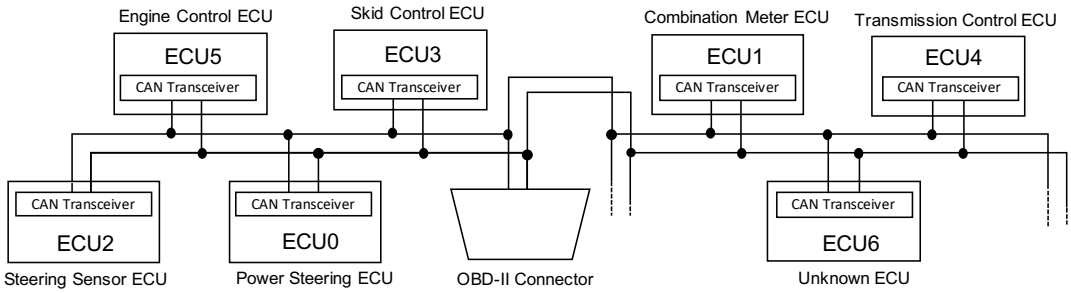


Figure 30: CAN message loss rates of PLI-TDC when changing the CAN bus occupancies.



(a) The CAN bus prototype.



(b) A part of CAN bus in the real-vehicle.

Figure 31: Environments for evaluations.

5.4.1 Environments and Attacker Models

In this section, we describe the environments for evaluating PLI-TDC.

Fig. 31 (a) shows the prototype of the CAN bus topology we implemented in our experiment. We prepare various ECUs to evaluate PLI-TDC. The various ECUs we prepared are described here. As shown in Fig. 31 (a), ECUs 0, 1, 2, and 3 are implemented by microcomputers and dedicated CAN boards. ECU4 is an actual ECU not connected other than CAN, ECUs 5 and 6 are an actual combination meter of each different car model. We cannot control sending CAN messages of ECUs 4, 5, and 6 but these ECUs automatically send some messages periodically, so that PLI-TDC uses the messages to fingerprint ECU.

Fig. 31 (b) shows a part of CAN in real-vehicle which is used to evaluate PLI-TDC. The real-vehicle has multiple CAN buses. One of these CAN buses has a realistic environment in which each ECU has a yaw-rate sensor or an acceleration sensor sends the information to the meter ECU. This CAN bus also has an OBD-

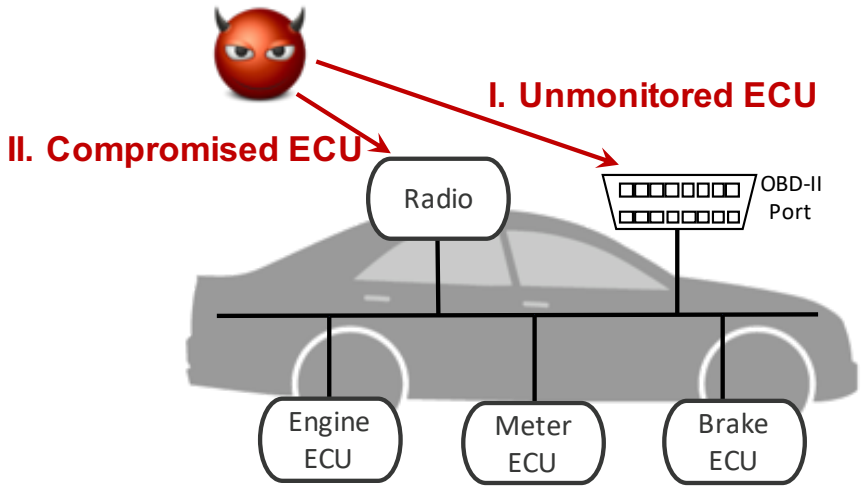


Figure 32: Attacker models defined by Sec. 2.4.

II port. In the real-vehicle experiment, we have collected the datasets during driving and stopping.

We define two types of attacker models (Fig. 32) as follows. The first model is based on the hacking of Jeep Cherokee [62]. In actual hacking of Jeep Cherokee, Miller and Valasek exploited a passive or unmonitored ECU's update mechanism to inject their code. Thus, we suppose the type of attacker called unmonitored ECU. By the way, since an ECU has some connectivity interfaces such as Wi-Fi or Bluetooth, some attackers may exploit the attack surfaces such as Wi-Fi or Bluetooth [14]. Therefore, we suppose the attacker model called compromised ECU which is an ECU exploited by the attacker through attack surfaces.

5.4.2 Idetification of ECUs

First, we evaluate the mean accuracy of the four algorithms described in Sec. 5.3.3. We show a result of the evaluation of each algorithm in Table 12. Also, we used Random Forest of the number of trees is 50. As the result, we confirmed that Random Forest classifier is the highest accuracy in both CAN bus prototype and real-vehicle. Therefore, we decide that PLI-TDC uses a Random Forest classifier.

Next, we evaluate the mean accuracy with the Random Forest classifier of each ECU in the CAN bus prototype. We have captured 9000 messages from each ECU, dividing the messages into 80 % and 20 % for learning and testing

Table 12: Mean accuracy of each algorithm.

	Mean accuracy [%]	
Algorithms	CAN bus prototype	real-vehicle
LR	92.86	74.83
Naive Bayes	88.05	75.94
Decision Tree	99.48	95.49
Random Forest	**99.67**	**97.04**

Table 13: Confusion matrix for the identification of ECUs of the CAN bus prototype.

		Predicted label						
		ECU0	ECU1	ECU2	ECU3	ECU4	ECU5	ECU6
Actual label	ECU0	**99.04**	0.06	0.00	0.00	0.00	0.89	0.00
	ECU1	0.06	**99.78**	0.06	0.00	0.00	0.11	0.00
	ECU2	0.00	0.00	**99.94**	0.06	0.00	0.00	0.00
	ECU3	0.00	0.00	0.06	**99.94**	0.00	0.00	0.00
	ECU4	0.00	0.00	0.00	0.00	**100.00**	0.00	0.00
	ECU5	1.29	0.00	0.11	0.00	0.00	**98.60**	0.00
	ECU6	0.00	0.00	0.00	0.00	0.06	0.00	**99.95**

respectively, calculating the eight features from the messages, putting the features into machine learning algorithms. We evaluate the proposed method using K-fold cross-validation in K=5. As a result, the mean accuracy is 99.67 %. A confusion matrix in K-fold cross-validation is shown in Table 13. PLI-TDC can identify correctly with up to 100.00 %. While a minimal identification rate is 98.60 % in CAN bus prototype.

We have also evaluated ECU identification accuracy in real vehicle's CAN bus. We have run around the our university (1.1 km) with 10 km/h to 30 km/h. We have captured 400000 messages from the ECUs. Half of the 400000 messages were observed in stopping and the rest were observed in running. We used the 6010 of the 400000 messages. The number of messages of ECUs 0, 1, 2, 3, 4, and 5 is 1000 messages, respectively. But, the number of ECU6's messages is

Table 14: Confusion matrix for the identification of ECUs of the real-vehicle.

		Predicted label						
		ECU0	ECU1	ECU2	ECU3	ECU4	ECU5	ECU6
Actual label	ECU0	**99.50**	0.00	0.00	0.50	0.00	0.00	0.00
	ECU1	0.00	**97.97**	0.00	0.00	2.03	0.00	0.00
	ECU2	0.00	0.00	**98.50**	0.00	0.00	0.15	0.00
	ECU3	0.00	0.00	0.00	**95.31**	0.00	4.69	0.00
	ECU4	1.99	1.99	0.00	0.00	**96.02**	0.00	0.00
	ECU5	0.47	0.00	1.41	7.55	0.00	**90.09**	0.00
	ECU6	0.00	0.00	0.00	0.00	0.00	0.00	**100.00**

only 10 messages, because it is non-periodic messages. As with the CAN bus prototype, we divided the CAN messages of the delay-time into learning data and testing data. Hence, we evaluate PLI-TDC using K-fold cross-validation in K=5. From the K-fold cross-validation, PLI-TDC performed well with an average accuracy of 97.04 %. A confusion matrix is shown in Table 14. We confirmed that PLI-TDC can identify each ECU correctly with up to 100.00 % while a minimal identification rate is 90.09 %.

5.4.3 Attacker Detection

In this section, we evaluate the intrusion detection capability of the learned model. To reproduce unmonitored ECU, we attach a new ECU which is ELM327 to the CAN bus prototype. Additionally, we sent an arbitration ID: x assigned in ECU3 from unmonitored ECU spoofed to ECU3. Spoofing attacks were performed for three minutes from unmonitored ECU, and the data during attacks of ECUs were classified by the learned model. The results are shown in Table 15. "Predicted: Attack" is when PLI-TDC classifies messages of ID: x as other than ECU3, "Predicted: Normal" is when PLI-TDC classifies messages of ID: x as ECU3. We confirm the true positive rate against compromised ECU is 100.00 % and the true negative rate is 99.32 %.

We evaluate the ability of intrusion detection against compromised ECU. We attached the Arduino UNO (the ECU2 in the prototype of CAN bus) as a com-

Table 15: Confusion matrix against sending ID: x from compromised ECU spoofed to ECU3.

	Predicted: Attack	Predicted: Normal
Actual: Attack	1.0000	0.0000
Actual: Normal	0.0068	0.9932

Table 16: Confusion matrix against sending ID: y from compromised ECU spoofed to ECU3.

	Predicted: Attack	Predicted: Normal
Actual: Attack	1.0000	0.0000
Actual: Normal	0.0570	0.9430

promised ECU in the CAN of the real-vehicle. We assume the spoofing attacks of speed information from the compromised ECU. Therefore, the compromised ECU sends ID: y assigned as arbitration ID of speed in the real-vehicle. Spoofing attacks were performed for three minutes from compromised ECU, and the data during sending messages of ECU3 (legitimate ECU of ID: y) and compromised ECU were classified by the learned model. The results are shown in Table. 16. We confirm the true positive rate against compromised ECU is 100.00 % and the true negative rate is 94.30 %.

5.4.4 Identification of ECUs under Temperature Concept Drift

Here, we evaluate PLI-TDC under different temperatures. We used cardboard to cover the CAN bus prototype and increase the ambient temperature of the CAN bus prototype using a heat-gun, as shown in Fig. 33. And we corrected data during increasing temperature. We have received 100000 messages from seven ECUs in the CAN bus prototype. We use the messages to calculate R-squared (R^2) in case that X is temperature and Y is delay-time. The average R^2 and mean square error (MSE) are given in Table 17. It shows the R^2 and MSE using the data from 30 °C to 45 °C. The R^2 varies depending on the type of CAN transceiver. Therefore, a learned model must respond to feature drift caused by drift of temperature for ECUs 1, 2, and 3.

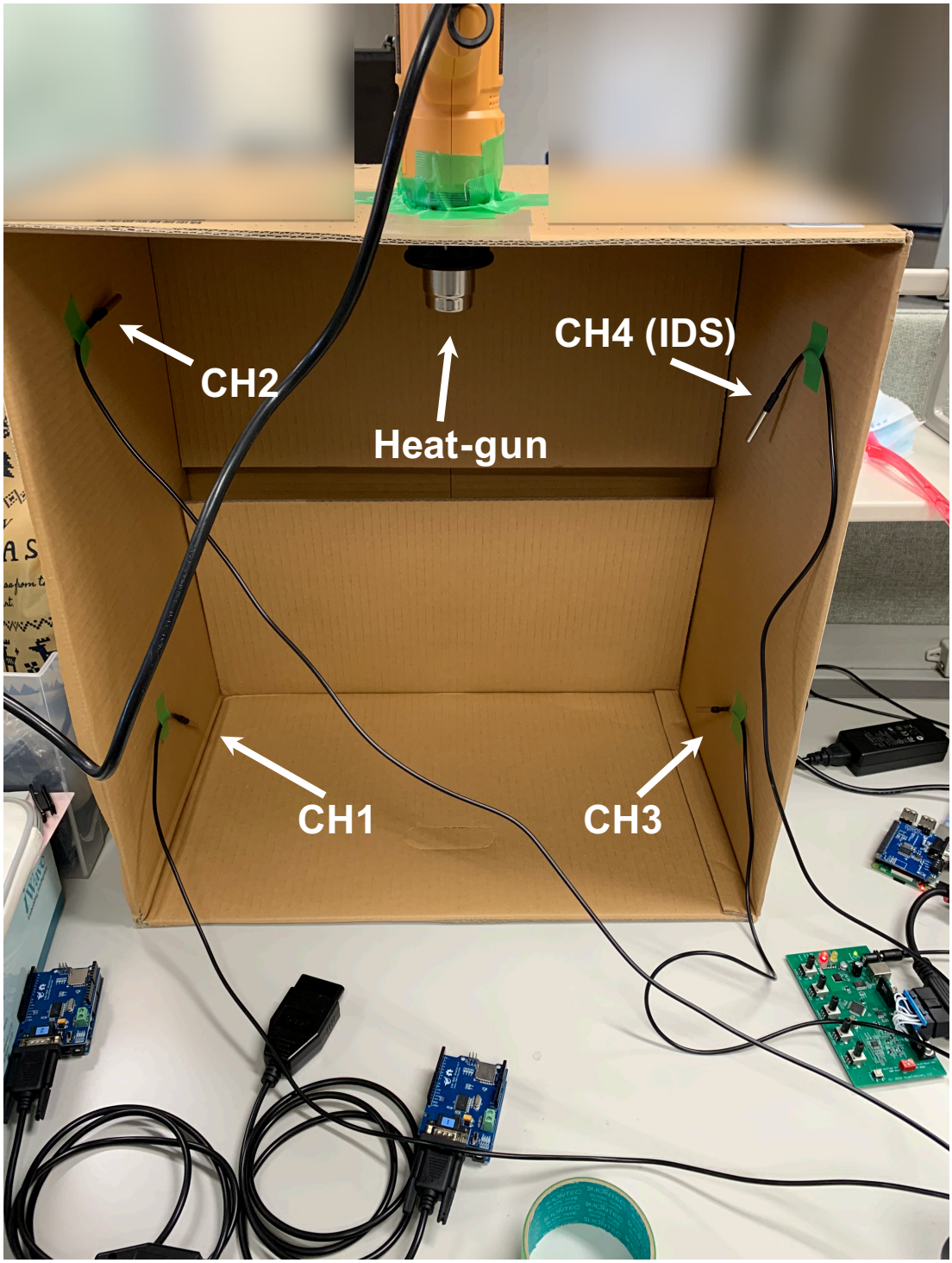


Figure 33: Our testing environment for changing the ambient temperature of the CAN bus prototype.

Table 17: A result of liner regression against temperature drift (CAN bus prototype).

ECU (CAN transceiver)	R^2	MSE
ECU0 (TJA1040)	0.0006	0.9994
ECU1 (MCP2551)	0.8544	0.1456
ECU2 (MCP2551)	0.8242	0.1758
ECU3 (MCP2551)	0.6947	0.3053
ECU4 (TJA1040)	0.0948	0.9052
ECU5 (SE706)	0.0706	0.9294
ECU6 (TJA1042)	0.0102	0.9898

Furthermore, we validated a new robust model for which temperature is added as one of the features. First, we validated the Random Forest model that learned only the eight features' data at each temperature, and the mean accuracies are shown in the Table 18.

Table 18: Mean accuracies under different temperature in CAN bus prototype (eight features).

		Testing data															
		30 °C	31 °C	32 °C	33 °C	34 °C	35 °C	36 °C	37 °C	38 °C	39 °C	40 °C	41 °C	42 °C	43 °C	44 °C	45 °C
Training data	30 °C	**99.8**	97.2	95.9	95.5	96.0	96.5	96.1	94.4	95.7	94.8	96.0	96.2	96.3	90.8	84.4	80.5
	31 °C	96.1	**99.7**	95.7	95.9	95.9	96.5	96.0	94.6	95.7	95.0	96.2	96.2	96.5	91.2	85.2	82.3
	32 °C	96.4	97.3	**99.5**	96.4	96.3	97.1	96.8	95.6	96.4	95.9	96.5	96.4	96.6	91.2	84.2	80.2
	33 °C	95.6	96.6	96.4	**99.7**	96.7	96.9	96.8	95.6	96.2	95.7	96.6	96.6	96.7	91.6	85.5	83.2
	34 °C	95.3	96.4	95.7	96.0	**99.8**	97.2	97.1	95.7	96.4	95.9	96.7	96.9	97.0	92.1	86.0	83.9
	35 °C	95.7	96.7	96.2	96.3	97.3	**99.9**	97.6	97.1	97.4	96.8	97.4	97.4	97.4	93.0	87.5	84.4
	36 °C	94.6	95.7	95.1	95.9	96.9	97.4	**99.9**	97.5	97.6	97.3	97.5	97.4	97.5	93.5	87.8	85.3
	37 °C	93.9	94.4	94.5	94.4	95.9	97.0	97.5	**99.9**	98.2	98.1	98.1	98.0	97.9	94.0	89.0	86.3
	38 °C	91.9	93.9	93.6	93.0	94.9	95.9	96.6	97.8	**99.8**	97.9	98.1	98.1	97.9	94.1	89.4	85.7
	39 °C	89.7	92.1	92.7	91.7	93.9	94.9	95.2	96.5	97.4	**99.9**	98.4	98.4	98.5	95.9	91.5	88.3
	40 °C	89.8	92.4	92.3	91.2	92.5	93.4	94.4	95.8	97.4	98.3	**99.9**	98.7	98.8	96.6	93.1	89.7
	41 °C	87.2	90.3	90.9	89.3	91.5	92.9	93.2	94.3	97.0	97.8	98.6	**100.0**	99.0	96.8	94.0	91.0
	42 °C	86.7	89.0	89.6	88.3	90.6	91.4	92.0	93.6	96.6	97.8	98.5	98.9	**100.0**	98.1	95.9	93.1
	43 °C	85.2	88.4	88.8	87.0	89.6	90.4	89.8	90.6	94.2	96.4	97.1	98.2	98.9	**99.9**	97.3	96.3
	44 °C	83.5	87.3	88.3	86.3	88.9	89.4	89.0	89.8	93.6	95.6	96.7	97.9	98.6	98.5	**99.9**	97.0
	45 °C	83.9	87.4	88.3	86.2	88.9	89.8	89.1	89.8	93.2	95.5	96.2	97.4	98.2	97.9	97.4	**99.8**

We only showed the representative range 30 °C to 45 °C, because the results in range 20 °C to 60 °C is similar to the results in representative range. The rows of Table 18 shows the training data, and the columns show the testing data in each temperature. For instance, from Table 18, the mean accuracy was 80.5 % when a model constructed using the training data of 30 °C classified the testing data of 45 °C. Hence, we confirmed that the mean accuracy decreases as the difference between the training data temperature and the testing data temperature increases.

Table 19: Mean accuracies under different temperature in CAN bus prototype (eight features and temperature).

	Testing data															
	30 °C	31 °C	32 °C	33 °C	34 °C	35 °C	36 °C	37 °C	38 °C	39 °C	40 °C	41 °C	42 °C	43 °C	44 °C	45 °C
Roboust Model	99.8	99.8	99.8	99.8	99.8	99.8	99.8	99.7	99.3	99.3	99.3	99.2	99.2	99.2	99.3	99.3

Next, we show the mean accuracies of the learned Random Forest model in case that temperature is added to one of the feature. As shown in Table 19, the model achieved 99 % of accuracy in all data from 30 °C to 45 °C. Therefore, we conclude that it is possible to construct a robust learned model against feature drift due to drift of temperature by adding temperature as a feature.

5.4.5 Detection Time

In this section, we evaluate the detection time of PLI-TDC from feature extraction phase to classification phase. First, we describe the details of the experimental device. The experiment was conducted on a Raspberry Pi model B+ with 1.2GHz 64-bit quad-core Cortex-A53 CPU, 1GB RAM, and Debian 10. In our experiment, PLI-TDC executed the inference of the Random Forest classifier (# of tree is 50) implemented in C++ per receiving the CAN messages. As a result, the time of the feature extraction phase was 13.590 µs, and the classification phase was 50.217 µs. Therefore, we concluded that the detection time in PLI-TDC is 63.807 µs.

5.5 Discussion

5.5.1 Identification / Detection Accuracy

The mean accuracies of PLI-TDC in CAN bus prototype and real-vehicle were 99.67 % and 97.04 %, respectively. The results of classifying the data observed with the same time-resolution (20 ns) as Divider [79] were 81.43 % and 76.75 %, respectively. Besides, we evaluated the mean accuracy rate of PLI-TDC under different time-resolution in the CAN bus prototype. We can change the time-resolution by regarding multiple delay cells in TDC as one delay-cell. For example, when delay-time per one delay-cell is 0.154 ns, we can obtain the time-resolution 0.616 ns by regarding four delay-cells as one delay-cell. Table 20 shows the mean accuracy of PLI-TDC under different time-resolutions. Also, the eight features selected by Relief-F were used for Random Forest classifier. As shown in Table 20, we confirmed that the mean accuracy is increased by improving the time-resolution. Therefore, we concluded that the mean accuracy of PLI-TDC improves with high time-resolution. Since we confirmed that the mean accuracy is increased by improving the time-resolution, we will try the wave union TDC

Table 20: Mean accuracy under different time-resolution.

time-resolution [ns]	20.000	7.700	3.850	1.540	0.616	0.154
Mean accuracy [%]	87.20	93.02	97.47	98.31	98.46	99.67

Table 21: Comparison among voltage-domain based methods in Accuracy of Identification (A.I.), Sampling Rate (S.R.), Best Number of Samplings per message (B.N.S.), Worst Number of Samplings per message (W.N.S.), Time Complexity in Feature extraction (T.C.F.), Detection Time (D.T.).

	Choi et al. [11]	Scission [44]	SIMPLE [23]	EASI [46]	PLI-TDC
A.I. [%]	96.48	99.85	99.1	99.98	99.67
S.R. [MHz]	2000	20	0.5	$1.4 \sim 10$	-
B.N.S.	136×10^3	1360	34	20	5
W.N.S.	392×10^3	3920	98	20	14
T.C.F.	$\mathcal{O}(n \log n)$	$\mathcal{O}(n \log n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
D.T. [µs]	-	-	1.575	124.9	63.8

[97] which has a higher resolution than our TDC as future work.

In addition, we discuss a comparison of accuracy among some voltage-based methods and PLI-TDC with Table 21. The accuracy of PLI-TDC is lower than Scission and EASI. Particularly, EASI can identify the highest accuracy and few numbers of samplings. However, in EASI, it is required to change the sampling rate, because the required sampling rate depends on the length of the payload. On the other hand, PLI-TDC can reliably measure the delay-time without the additional effort, regardless of the data field length. In addition, changes in the voltages may be caused by Electromagnetic Interference (EMI) sources. Therefore, voltage-based IDS may cause unexpected false alarms. In the contrast, PLI-TDC can mitigate this problem because the delay-time is a time-domain

characteristic not directly influenced by EMI. In addition, manipulation of the delay-time by a remote attacker is expected to be challenging, as the attacker would have to control the timing of the CAN signal with pico-second precision from the software-layer.

Even with a limited number of ECUs in the real-vehicle, the accuracy of PLI-TDC is still 97.04%. The 2.96% inaccuracy is due to two ECUs with almost the same delay-time (i.e., ECU3 and ECU5 in the real vehicle). As result, PLI-TDC may generate many false alarms per day. While PLI-TDC achieved the lowest B.N.S. and W.N.S. close to the theoretical limit (Table 21), it means that PLI-TDC has the potential to realize a highly accurate physical-layer IDS with little additional computational complexity in combination with multiple physical characteristics such as voltage and delay-time. Thus, we will try to combine physical-characteristics to solve the problem of false alarms as future work.

5.5.2 Number of Samplings

Next, we discuss the number of samplings performed by sender identification methods for each CAN message. Since the number of samplings to be processed per one message influences the performance of PLI such as memory usage and detection time, it is required to reduce the number of samplings in PLI.

The number of samplings per one CAN message for Choi's method, Scission, and SIMPLE depends on the length of the data field. Thus, we consider the case when the data field is the shortest (0 byte) and longest (8 byte). If the data field is the shortest (0 byte), the length of CAN message from SOF to CRC delimiter is 34 bit from Fig. 4. Also, when the bit rate of CAN is 500 kbps, the transmission time for 1 bit is 2 µs. Hence, the sampling rate of each method is multiplied by $34 \times 2 \times 10^{-6}$. Thus, for instance, the best number of samplings in Choi's method is $2000\,\mathrm{MHz} \times 34 \times 2 \times 10^{-6} = 136 \times 10^3$. As a result, the best number of samplings per one CAN message in Choi's method, Scission, and SIMPLE is 136×10^3, 1360, and 34, respectively. Similarly, if the data field is the longest (8 byte), the length of the CAN message from SOF to CRC delimiter is 98 bit. Therefore, the worst number of samplings is 392×10^3, 3920, 98 respectively. Also, EASI's number of samplings is 20 for both best and worst. The number of samplings per message in PLI-TDC depends on the number of signal transitions from 0 to 1, not the

length of the data field. Consequently, The minimum and the maximum number of samplings of PLI-TDC are discussed with Arbitration ID `0x000`, which has a small number of bit transitions, and Arbitration ID `0x555`, which has a large number of transitions. As a result, the best number of samplings reached 5. The worst number of samplings reached 14. The results show that PLI-TDC has the least number of samplings at the data acquisition phase; in other words, PLI-TDC has the smallest n at the feature extraction phase. Hence, the feature extraction of PLI-TDC is possible with light processing.

Finally, we discuss computational complexity. The method of Choi et al. and Scission use time and frequency domain features. Therefore, these methods need $\mathcal{O}(n \log n)$ time because these methods perform Fourier Transforms to calculate the frequency domain feature. Also, the feature extraction phases of SIMPLE and EASI takes $\mathcal{O}(n)$. Because PLI-TDC uses statistic features in Table 9, PLI-TDC needs $\mathcal{O}(n)$. Therefore, we confirmed that the computational complexities of SIMPLE, EASI, and PLI-TDC are lower than the computational complexities of other methods.

From these comparisons among related works, we confirmed that PLI-TDC can reduce the amount of data in the data acquisition phase than the other voltage-based methods.

5.5.3 Detection Time

We discuss whether PLI-TDC can identify all messages without spilling messages. In Sec. 5.4.5, the evaluation of detection time was conducted on a Raspberry Pi model B+ which used runs at a 1.2GHz 64-bit quad-core Cortex-A53 CPU. This environment is the same as a specification of the system used for autonomous, in-vehicle infotainment, and gateway function [85]. Thus, the experiment environment of PLI-TDC can be implemented in actual vehicles.

As described in Sec. 5.4.5, PLI-TDC's detection time is 63.807 µs. The detection time is illustrated as shown in Fig. 34. A minimum interval of between two CAN frames is 222 µs in CAN of 500 kbps. Since the detection time is lower than the minimum interval in CAN, we conclude that PLI-TDC can validate all CAN messages without spilling messages.

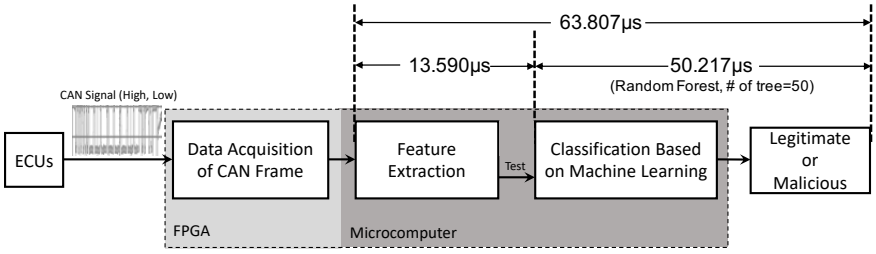


Figure 34: Detection time on PLI-TDC.

5.5.4 Stability and Life-Cycle

In this section, we describe the stability of the delay-time and life-cycle of the learned model in PLI-TDC. PLI-TDC may not adapt long-term gradual feature drift caused by material wear and so on. Thus, we collected the 24 GB of data for 86 days from the real-vehicle. However, we could not find any gradual drift of delay-time in the 86 days. Hence, we confirmed that PLI-TDC has the stability of feature at least 86 days. We also recommend updating a delay-time based model after 86 days from deploying the model. Besides, as the other feature drift, in case of sudden drift caused by some fail, we can update immediately a learned model because Random Forest classifiers can learn data fastly with a few seconds [102].

5.5.5 Comparison with Time Domain Reflectometry

Time Domain Reflectometry (TDR) based PLI [83] has been proposed as an approach to detect the connection of an illegal device based on the variation of the characteristic impedance of the cable in CAN-H, CAN-L, and other CANs. In this method, an impulse signal is applied to a CAN from a pulse generator, and the returned reflection waveform is observed by an oscilloscope. Thus, the method using TDR can be regarded as an active method in which an IDS that detects a rogue device applies a signal to the CAN for measurement. In other words, the application of the measurement signal may interfere with the CAN communication when the vehicle is in operation. Therefore, it is necessary to consider a passive method that can detect an illegal device message by simply observing the CAN message. On the other hand, our PLI-TDC only requires

observing the delay-time characteristics passively. Hence, we conclude that our PLI-TDC can solve the drawbacks of TDR based PLI.

5.5.6 Limitation

We achieved a mean accuracy of 99.67 % and 97.04 % in the CAN bus prototype and the real-vehicle, respectively. However, it may occur that EMI which is interference to devices (e.g. ECU) by electromagnetic waves slightly changes the delay-time in the actual operation of vehicles. In this case of the occurrence of EMI on CAN and ECUs, PLI-TDC can adapt by updating a delay-time based model after detection of the concept drift by EMI. Also, the updating scheme of the machine learning model can be an attack-surface because an attacker tries to inject malicious messages into training data. Fortunately, we can solve this problem using message authentications. Message authentications can support the creation of training data that does not contain injected malicious messages because we can exclude malicious messages by forcing message authentications during enabling the updating scheme.

Furthermore, we confirmed the robustness for the concept drift of temperature. However, we suppose that the ambient temperature of IDS and ECUs simultaneously increases like inside of cardboard (Fig. 33). In actual environments of vehicles, ECUs distribute at various places of vehicles. It means that IDS and ECUs may not increase simultaneously in actual environments. In this case that IDS and ECUs have a difference in ambient temperature, PLI-TDC misclassifies the CAN messages sent by ECU which has a difference in ambient temperature. Therefore, to address this type of concept drift, PLI-TDC may require numerous temperature sensors in actual environments.

5.6 Conclusion

To avoid the security risk on automobiles, PLIs in CAN have been proposed. To meet requirements of high accuracy, the small number of samplings of feature, and robustness against temperature change, we proposed a delay-time based sender identification called PLI-TDC which is higher accuracy than Divider and gets the features with a few sampling. In addition, PLI-TDC realizes temperature-

robustness by learning the temperature as one of the features. We implemented the experimental devices using FPGA and microcomputer to verify our method for identification. As a result, we confirm that PLI-TDC achieved a mean accuracy of 99.67 % in the CAN bus prototype and of 97.04 % in the real-vehicle. We have released our research [74] in the hope to promote research on sender identification.

6. IVNProtect: Isolable and Traceable Lightweight CAN-Bus Kernel-Level Protection

6.1 Introduction

Encryption and authentication [81, 52] for CAN have been proposed to prevent spoofing attacks from a compromised ECU. However, encryption and authentication mechanisms cannot deal with DoS attacks since an attacker can flood encrypted CAN-bus with DoS using a high-priority ID. Therefore, a dedicated countermeasure is required to prevent DoS attacks.

On the other hand, many automotive security researchers have proposed IDSs based on various characteristics (e.g., frequency [88], entropy [98], ID sequence [59], message correlation [61, 76], physical-layer fingerprint [44, 46, 78]). These IDSs achieved the detection of various attacks including DoS attacks with high accuracy. However, these IDSs do not provide any defense mechanism to protect against the DoS attacks. Therefore, it is necessary to build a protection mechanism with features from attack detection to defense.

Some countermeasures to disable DoS attacks on CAN have been researched. The allowlist-based mitigation method [21] has been proposed as one of the countermeasures. This method filters message transmission based on a predefined allowlist on a CAN-specific hardware. It can disable DoS attacks with the highest priority CAN ID:`0x000`; in other words, it passes only allowlisted messages based DoS attacks permanently. Hence, it is ideal for mitigating and isolating a DoS attacker who uses both malicious and benign messages. Moreover, this method requires additional hardware; therefore, it also has a drawback in deployability. On the other hand, a frame-corruption method [90] conducts malicious frame-corruption with error frames by a legitimate ECU with IDS. This method transfers a compromised ECU to bus-off state, which is a disconnected state of an ECU from the CAN-bus (i.e., an ECU in the bus-off state cannot send/receive CAN messages). Originally, the bus-off state is designed to handle a fault in the ECUs, so that the frame-corruption method cannot distinguish whether the bus-off state occurred due to a fault or a security incident. In consequence, if the bus-off state of a compromised ECU occurs due to a security incident, the other

ECUs cannot conduct incident response operations such as switching to manual operation of the vehicle. In other words, if there is a protection method that can distinguish whether the bus-off state occurred through a fault or a security incident, the ECUs on the attacked vehicle can conduct some operations to minimize the security risk. In summary, these countermeasures have problems in terms of the isolability of a compromised ECU, deployability, and the traceability of the root cause for isolation.

To solve these drawbacks, we propose an isolable and traceable In-Vehicle Network bus kernel-level Protection approach called IVNProtect. IVNProtect can be installed on an ECU that has a wireless interface with just a software update because it is implemented in the CAN-bus kernel driver. We also confirm that our IVNProtect can mitigate two types of DoS attacks without distinguishing malicious/benign CAN IDs. After mitigating DoS attacks, IVNProtect isolates a compromised ECU with a security error state mechanism, which handles the security error in IVNProtect. Furthermore, we evaluate the traceability that an ECU with IVNProtect, which can report warning messages to the other ECUs on the bus even while being forced to send DoS attacks by an attacker. In addition, we experimentally show that the ECU installed IVNProtect can send the warning messages with 0 % of transmission loss rate. Moreover, the overhead of IVNProtect is 9.049 µs, so that IVNProtect can be installed on insecure ECUs with minimal side-effects. We release the source code of our IVNProtect, which is available on a GitHub repository [75].

The main contributions of this paper can be summarized as follows.

- We propose an isolable and traceable lightweight CAN-bus protection called IVNProtect. IVNProtect is implemented to a CAN-bus kernel driver on Linux. Hence, our IVNProtect can deploy to an ECU by just software updating.

- We provide a new error state mechanism on IVNProtect for handling security incidents. IVNProtect mitigates DoS attacks and isolates a compromised ECU based on this security error state mechanism.

- We experimentally confirm the traceability that IVNProtect reports warning messages for legitimate ECUs to distinguish whether the cause

of isolation is a fault or security incident.

- We show the overhead caused by IVNProtect. As a result, the overhead takes only 9.049 µs, which means that a system with our IVNProtect satisfies in-vehicle real-time demands.

6.2 Related Work

Various intrusion detection methods (e.g., arrival time [88], entropy [98], and voltage [44]), and authentication mechanisms [81, 52] have been studied in order to secure IVNs. These intrusion detection methods focus on detecting spoofing, replaying, and DoS attacks but do not provide protection against the attacks. Conversely, the authentication mechanisms focus on protecting spoofing and replaying attacks. In other words, the existing intrusion detection methods and authentication mechanisms cannot protect DoS attacks on CAN-bus.

In the rest of this section, we introduce some existing protection methods dedicated to preventing DoS attacks.

6.2.1 Protections Implemented on Bus

Wu *et al.* proposed an ID-hopping moving target defense [39, 96, 99] to protect a DoS attack against individual ECUs (called *targeted DoS attacks*) and reverse engineering CAN messages. They implemented the ID-hopping mechanism to the CAN controller, hardware embedded on an ECU. The ID-hopping is carried out in all ECUs on the CAN-bus, so it is required to install the ID-hopping CAN controller to all ECUs. While, it can protect against targeted DoS attacks, but it cannot protect against DoS attacks with the highest priority CAN ID: `0x000`.

A relays-based reactive defense called CANARY [34] divides the bus by activating/deactivating some relays implemented on the bus. To protect the bus against DoS attacks, CANARY divides the bus by deactivating the relays surrounding the compromised ECU. A CANARY-based bus also has a Bus-Guardian node that detects attacks and manipulates relays. Thus, even in the isolation of the compromised ECU, the others ECUs can trace the root cause of isolation via the Bus-Guardian ECU. However, since CANARY needs additional hardware (relays, resistors, and wires), it has a drawback in deployment cost. Moreover,

the relay action has a negative aspect because it corrupts the benign message sent during activating/deactivating relays.

6.2.2 IDS-side Protection

The frame-corruption approach [90] has been studied. This approach uses error frames to forcibly transfer a targeted compromised ECU's state to the bus-off state. It is possible to send error frames with unmodified CAN controllers so that the approach easily deploys ECUs through software update. However, because the frame-corruption approach uses error frames, it is inherently difficult to distinguish whether the cause of isolation is a fault or a security incident. Furthermore, the error frames generated by this approach contaminate the communication of the bus, which negatively influences the arrival time of some messages and busloads.

6.2.3 ECU-side Protections

Unlike the protections of IDS-side and on-bus, the ECU-side protection has an advantage in unharming for messages on the bus when its protection is triggered. Elend *et al.* developed a secure CAN transceiver [21] that filters message transmission based on a predefined allowlist in the CAN transceiver. In other words, the secure CAN transceiver protects a compromised ECU from sending malicious ID messages. But, an attacker who compromised an ECU with this transceiver can transmit some allowlisted messages permanently. Therefore, it is necessary to deal with the flooding with malicious/benign ID. Moreover, the deployment cost is high since it needs to replace the CAN transceiver hardware.

The most relevant research to our IVNProtect is TEECheck [63], proposed by Mishra *et al.* TEECheck has advantages regarding the complete isolation of a compromised ECU, traceability of the root cause of the bus-off state, and unharming benign messages. This approach isolates a malicious process in a compromised ECU using TrustZone, a Trusted Execution Environment (TEE). Namely, TEECheck does not carry out the disconnection at ECU-unit such as the division of buses using CANARY, because an attacker is isolated at the processing unit. It means that TEECheck ensures traceability to the root cause of isolation since an ECU with TEECheck only becomes the bus-off state caused by fault

incidents. On the other hand, the vulnerabilities/bugs related to the implementation of TrustZone have been reported [58, 35]. Therefore, an attacker possibly exploits the TEE's vulnerabilities for flooding the bus. In addition, TEECheck can only deploy to ARM-based ECUs, so it has the limited deployability.

Table 22 shows a comparison between the aforementioned protections and our proposal. The existing protections have problems in incomplete isolation, traceability on the cause of isolation, deployment cost, and side-effect against some benign messages of legitimate ECUs. Detailed comparisons of ECU-side protections are discussed in section 6.6.1. To solve these problems, we propose an isolable, traceable, and deployable kernel-level protection in the next section.

Table 22: Comparison among CAN-bus protections in Implementation Location (I.L.), Isolability, Traceability for Root-cause of Isolation (T.R.I.), Deployment Cost (D.C.), Harm for Benign Messages of legitimate ECUs (H.B.M.).

	CANARY [34]	ID-Hopping [39, 96, 99]	Frame-Corruption [90]	Elend *et al.* [21]	TEECheck [63]	IVNPROTECT
I.L.	Bus	Bus	IDS	ECU	ECU	ECU
Isolability	Complete	Partial	Complete	Partial	Complete	Complete
T.R.I.	Yes	-	No	-	Yes	Yes
D.C.	High	High	Low	High	Middle	Low
H.B.M.	Harmful	Harmful	Harmful	Unharmful	Unharmful	Unharmful

6.3 Proposed CAN-Bus Kernel-Level Protection

6.3.1 Threat Models

At first, we describe the assumption of the threat model. We assume that the attacker has privileged access. But, the attacker cannot replace kernel modules because we suppose installed kernel modules are signed by a trusted party. Moreover, the attacker also does not disrupt its kernel since the attacker only has the motivation to disrupt CAN-bus communication. In other words, we assume that the attacker does not shutdown attacks on the compromised system. In the following section, we define two DoS attacks as threat models.

Malicious ID DoS

Malicious ID DoS is the most critical threat to CAN-bus communication. This attack uses the highest priority ID (`0x000`) on CAN to fill the networks. It brings unexpected vehicle behavior and stops some functions (e.g., power steering).

Benign ID DoS

Benign ID DoS abuses the highest priority ID assigned to a compromised ECU. In other words, benign ID DoS is a DoS attack using the highest priority legitimate ID used by an actual ECU. It implies that benign ID DoS is less aggressive than the aforementioned malicious ID DoS because it is expected that some benign IDs are higher priority than the ID used by benign ID DoS. However, benign ID DoS is still a critical threat against some benign IDs that are lower priority than the ID used by benign ID DoS.

6.3.2 Problem Statement

We design an isolable, traceable, and deployable kernel-level protection (IVNProtect) that satisfies the following statements.

P1: Prevent DoS attacks (malicious ID DoS and benign ID DoS)

An attacker tries to flood the CAN bus by sending numerous messages. To deal with the attacker, IVNProtect monitors the CAN IDs of transmitted

messages in the compromised ECU. In the case CAN ID of a transmitted message is not a benign ID (i.e., not an allowlisted ID), IVNProtect cancels the message transmission. In contrast, if the CAN ID of a transmitted message is a benign ID, IVNProtect also analyzes the context of the transmitted IDs and then reduces the transmission rate if the context is an anomaly.

P2: Isolate a compromised ECU

To prevent the permanent malicious activity by an attacker in a compromised ECU, our protection isolates such compromised ECUs from the CAN-bus. However, it is important to determine when to isolate a compromised ECU. Specifically, it is required that IVNProtect should isolate the compromised ECU after reporting some warning messages to the other ECUs on the bus because the other ECUs can change some operations to minimize the security risk after receiving the warning messages. For instance, if the attacked vehicle has autonomous functions, the other ECUs can switch to manual operations to prevent exploiting autonomous functions.

P3: Report warning messages during DoS

As described in **P2**, IVNProtect must isolate the compromised ECU after reporting some warning messages. To ensure working this function, IVNProtect is required to satisfy the following requirements:

1. A benign transmission loss rate of 0 % during DoS activities
2. A worst arrival time of benign messages, less than isolation time by IVNProtect
3. A higher priority of warning messages than the others IDs

P4: Deploy IVNProtect with a slight overhead

Due to deploying to resource-constrained ECUs, we must minimize the overhead caused by running IVNProtect. Specifically, we define a requirement that the IVNProtect's overhead must be below 494 µs which is the transmission overhead by TEECheck [63].

6.3.3 Overview of IVNProtect

IVNProtect consists of four stages, allowlist and similarity-based detection modules, security error states, sending function and discarding function, as shown in Fig. 35. In the allowlist and similarity-based detection modules stage, these modules detect malicious ID and benign ID DoS attacks. In the security error states stage, the security error such as DoS attacks is managed by the state in this stage. Finally, IVNProtect determines whether to send the CAN message or discard it based on the state of security error states stage. Also, by installing IVNProtect to all ECU on CAN, it is possible to isolate a compromised ECU no matter which compromised ECU the attacker intrudes in.

Also, we assume that IVNProtect is installed on Linux-based ECUs such as AGL based in-vehicle infotainment system [87], because such ECUs with entry points to external networks have been compromised in the actual hacking [71, 62]. In the following sections, we describe each stage in order.

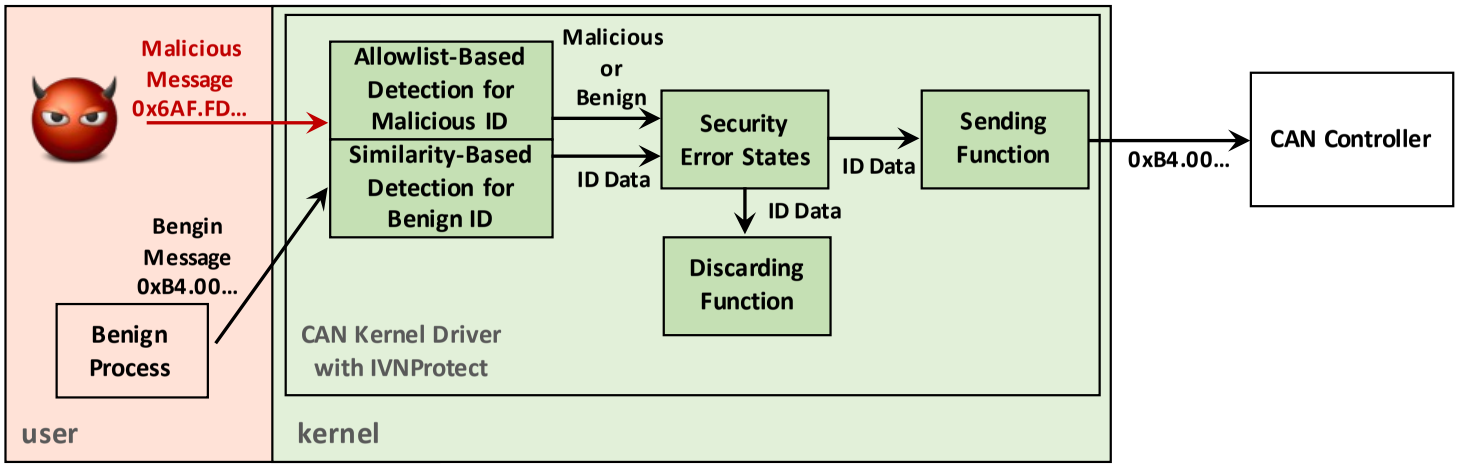


Figure 35: Diagram of the proposed IVNProtect.

6.3.4 Detection Modules

To solve **P1**, we provide two detection modules in this section. Based on the results of these modules, IVNPROTECT determines whether to conduct protection procedures (dropping messages or inserting a delay). The first is the allowlist-based detection module, which is introduced to prevent malicious ID DoS. The second one is the similarity-analysis-based detection module, which can detect benign ID DoS.

Allowlist-Based Detection Module

To disable malicious ID DoS, we add a module that discards incoming messages based on an allowlist of CAN IDs. We assume that this allowlist is pre-defined in the CAN-bus kernel driver before factory shipment based on the CAN IDs that the ECU sends. Therefore, if an attacker who intrudes on the ECU installing IVNPROTECT and tries to modify the allowlist, the attacker has to replace the CAN-bus kernel driver with the attacker's driver. We also suppose that the original CAN-bus kernel driver is signed by a trusted party such as automotive suppliers. It implies that the attacker cannot replace the original CAN-bus kernel driver with the attacker's driver.

Similarity Analysis-Based Detection Module

To prevent benign ID DoS, we provide the similarity-analysis-based detection module in this section. This module uses a similarity-analysis-based detection method [76] to detect DoS attacks quickly by a single ID or randomized IDs. This detection method detects DoS attacks by calculating the similarity (called Simpson coefficient) between Window IDs (*WIDs*) and Criterion IDs (*CIDs*). The *CIDs* are composed of the W number of benign CAN IDs, which are pre-defined before the detection-phase. The *WIDs* include the W number of the latest CAN IDs of recently received CAN messages. *CIDs* and W are pre-defined by an optimization algorithm based on SA before the detection-phase. As one example, we show the pre-defined parameters calculated from our evaluation dataset using the

SA-based algorithm [76] as follows.

$$\begin{aligned} W &= 7, \\ \sigma_s &= 0.3414, \\ CIDs &= \{0\text{x}3\text{b}8, 0\text{x}3\text{b}9, 0\text{x}3\text{ba}, 0\text{x}3\text{ba}, 0\text{x}3\text{bc}, 0\text{x}3\text{bd}, 0\text{x}463\} \end{aligned}$$

where σ_s is the deviation of benign similarity.

This method also requires defining these parameters in the CAN-bus kernel driver as with the allowlist-based detection module. These parameters are calculated and pre-defined before factory shipment for each ECUs. Thus, as with the allowlist-based detection module, these parameters related to similarity-analysis-based detection are protected from the attacker's modification.

6.3.5 Security Error State Mechanism

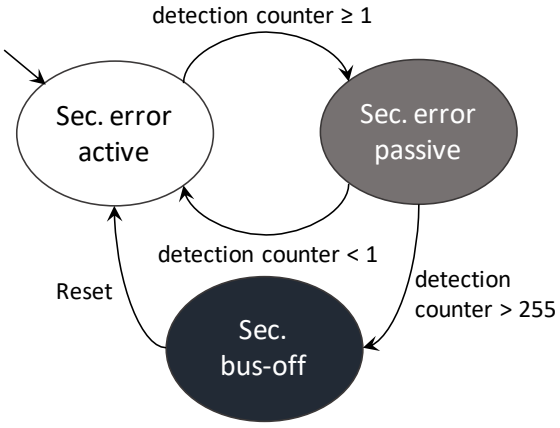
To meet **P2** and **P3**, we employ a security error state mechanism (Fig. 36) for the isolation of a compromised ECU, inspired by the fault error state mechanism specified in CAN. This section explains the functions and the statements of transition in each state.

Security Error Active State

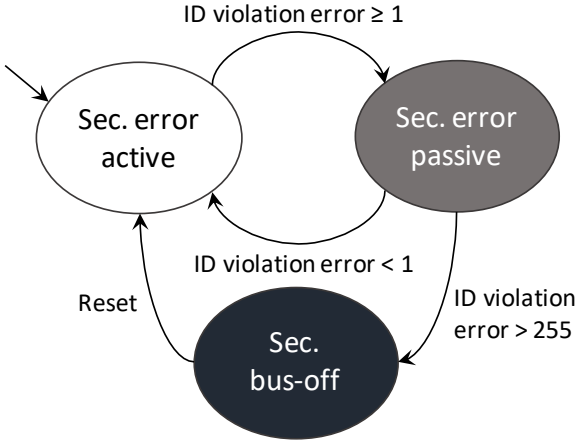
Security error active state implies that there is no security incident so that IVNPROTECT does not execute any functions in this state. As shown in Fig. 36 (a), if a detection module (e.g., allowlist-based detection) finds some malicious activity in this state, the ECU immediately transfers to the following security error passive state.

Security Error Passive State

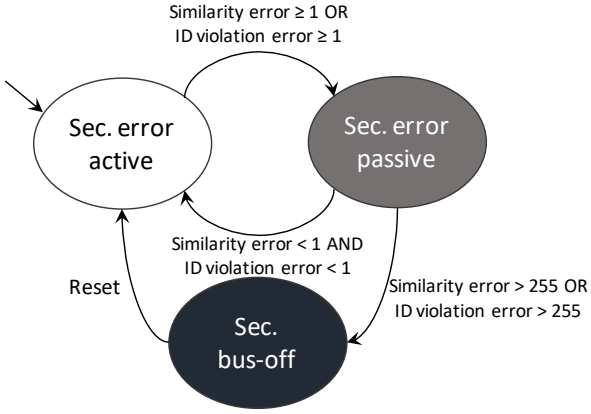
Security error passive state expresses that DoS attack activity has begun to be observed. For example, as shown in Fig. 36 (b), this state is entered when an ID outside the allowlist is sent (i.e., ID violation error). The security error states for both malicious and benign DoS attacks are shown in Fig. 36 (c). As the same as Fig. 36 (b) this state is entered when an ID outside the allowlist is sent (i.e., ID violation error). In addition, in case of



(a) General security error states.



(b) Security error states for malicious ID DoS.



(c) Security error states for malicious/benign ID DoS.

Figure 36: Security (Sec.) error states on IVNProtect.

sending messages with malignant similarity, this state also is entered. We define the case of sending messages with malignant similarity as similarity error.

Moreover, a compromised ECU reports a warning message for the other ECUs to change some operations to minimize the security risk. For instance, if the attacked vehicle has autonomous functions, the not compromised ECUs can change to manual operations to prevent exploiting autonomous functions.

Security Bus-off State

Security bus-off state completely disables the ability to send and receive to isolate a compromised ECU. This state can prevent an attacker from sending benign ID messages permanently or spreading the attacker's infection with some software updating scheme on CAN (e.g., ISO-TP).

This state is entered if either a lot of similarity error or an ID violation error increases. However, it is important to determine when to isolate the compromised ECU. The optimal threshold is defined in Sec. 6.6.2.

6.4 Implementation

In this section, we describe the implementation of IVNProtect. We implemented the procedures related to IVNProtect to the CAN-bus kernel driver [91] in Linux kernel v.5.10.

First, we explain the procedures of detection modules. We add the allowlist- and similarity analysis-based detection to the message transmission function `mcp251x_tx_work_handler()`. Algorithm 4 provides the message transmission function including IVNProtect procedures.

To implement the allowlist-based discarding messages, we used two Linux kernel functions, `dev_kfree_skb()` (line 35), which frees a buffer for storing packet data, and `netif_wake_queue()` (line 36), which wakes up the currently stopped queue and asks the kernel to resume sending messages. Specifically, in case of transmission of malicious ID, our IVNProtect executes `dev_kfree_skb()` to eliminate malicious transmission and then `netif_wake_queue()` to immediately resume next sending for benign messages. By the way, note that the other CAN

Algorithm 4 CAN message transmission algorithm in a kernel driver with IVN-PROTECT (Part 1).

Input: $can\_priv, WIDs, CIDs, W, allowlist$
Output: None

function calculate_similarity($set1, set2, set\_size$)
  return $\frac{|set1 \cap set2|}{set\_size}$
end function

$can\_frame \Leftarrow can\_priv\text{->}packet\_data$;
$similarity \Leftarrow 0$;
$is\_ID\_violation\_error \Leftarrow False$;
$is\_similarity\_error \Leftarrow False$;

// calculate similarity
$can\_frame\text{->}can\_id$ is added to $WIDs$
if $++window\_idx \geq W$ **then**
  $similarity \Leftarrow$ calculate_similarity($WIDs, CIDs, W$);
  $window\_idx \Leftarrow 0$;
  memset($WIDs, 0,$ sizeof($WIDs$)); // initialize $WIDs$
end if

// validate CAN ID and similarity
mutex_lock(&can_priv)
if $can\_frame\text{->}can\_id$ is not in $allowlist$ **then**
  $can\_priv\text{->}can\_device\_sec\_stats\text{->}ID\_violation\_error{+}{+}$;
  $is\_ID\_violation\_error \Leftarrow True$;
  update $can\_priv\text{->}sec\_state$
else if $similarity$ exceeds a benign range defined by σ_s **then**
  $can\_priv\text{->}can\_device\_sec\_stats\text{->}similarity\_error{+}{+}$;
  $is\_similarity\_error \Leftarrow True$;
  update $can\_priv\text{->}sec\_state$
end if

Algorithm 4 CAN message transmission algorithm in a kernel driver with IVNProtect (Part 2).

```
30: // send CAN message
31: if can_priv->sec_state is Security bus-off state then
32:    free the packet data in can_priv with dev_kfree_skb()
33: else if  then
34:    if is_ID_violation_error then
35:       free the packet data in can_priv with dev_kfree_skb()
36:       wake up the CAN device with netif_wake_queue()
37:    else if is_similarity_error then
38:       mdelay(10);
39:    end if
40:    send CAN message of can_priv->packet_data
41: end if
42: mutex_unlock(&can_priv)
```

kernel driver also has the message transmission function like `mcp251x_tx_work_handler()`. Therefore, IVNProtect can be implemented on various CAN peripherals.

Next, we describe the detail of similarity analysis-based detection. To implement the module, we added a new similarity calculation function (line 1-3) and some variants for the calculation. IVNProtect conducts the similarity calculation per fixed messages (line 11-16). Additionally, in case of malignant similarity, IVNProtect executes delay function `mdelay(10)` (line 38), which stops the sending procedure during 10 ms.

Finally, we explain the implementation of security error states. At first, we added two member variants, `enum sec_state` and `struct can_device_sec_stats` into `struct can_priv` which contains CAN common private data such as error state, sending data, etc. The `sec_state` expresses the current security error state of the CAN interface; for example, if `sec_state` is 0, it expresses that the current state is the security error active state. The `can_device_sec_stats` contains the detection counter, such as ID violation error for security error states. Thus, if the detection module discovers some mali-

cious activities, the detection counter in `can_device_sec_stats` increases. Then, IVNPROTECT manages the security error states based on this detection counter in `can_device_sec_stats` in our implementation.

6.5 Evaluation

6.5.1 Environment and Dataset

In this section, we explain our experimental environment and the specification of devices. At first, we set up the experimental CAN-bus in our laboratory. This CAN-bus includes three ECUs, (1) a Raspberry Pi 3 Model B with IVNPROTECT, (2) an actual combination-meter ECU that automatically sends CAN messages, (3) an experimental ECU which is called *ECUsim 2000* which supplies the power to the bus.

To emulate the CAN messages sent by one ECU, we logged 10090 messages from the actual combination-meter ECU. Hereafter, we evaluated various metrics of our IVNPROTECT, such as transmission loss rate and transmission delay when a DoS attack is performed while sending this 10090 message.

6.5.2 Prevention of DoS Attacks

At first, we evaluate the ability of IVNPROTECT to prevent DoS attacks. To evaluate this ability, we made Raspberry Pi 3 Model B both with and without IVNPROTECT, which simultaneously sends 10090 benign messages and DoS attacks. We show the comparison of busloads with and without IVNPROTECT in Fig. 37. As shown in Fig. 37 (a), in the case without IVNPROTECT, the busload reached over 45 % during DoS attacks.

In contrast, as shown in Fig. 37 (b), we confirmed that IVNPROTECT mitigated DoS attacks. Specifically, in the case of sending benign messages and malicious ID DoS simultaneously, there was no increase in busload. It means that the allowlist-based message discarding works against malicious ID DoS effectively. Next, we describe the case of sending benign messages and benign ID DoS. Since the allowlist of IVNPROTECT includes the benign ID used by the benign ID DoS, benign ID DoS passes the allowlist-based message discarding. However, due to the malignant similarity of transmission messages, IVNPRO-

TECT inserts a delay to mitigate flooding. As shown in Fig. 37 (b), we confirm that the busload is reduced to about 12 % by inserting the delay. Considering the busload of only benign messages is about 1 %, IVNPROTECT allows the benign ID DoS to increase the busload by 11 %. In general, the busloads of the real-world CAN-buses are limited to about 20 % in order to avoid unacceptable delays for low-priority messages. Thus, when an attacker executes benign ID DoS, the busload of real CAN-bus increases up to 31 %. From this result, we conclude that an attacker cannot achieve DoS attacks on the bus because the ECUs can send benign messages normally with a busload of about 31 %.

6.5.3 Isolation Time

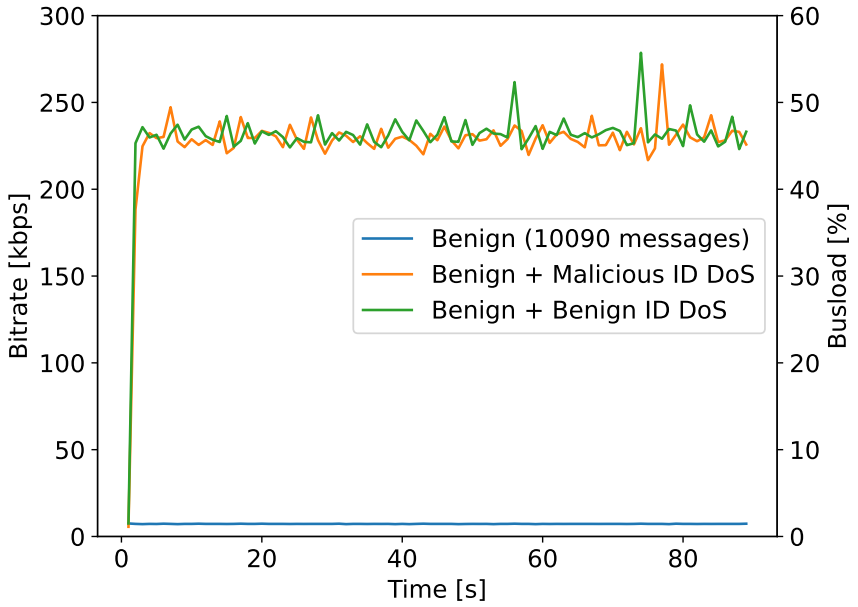
Next, we evaluate the isolation time, which expresses from starts from the DoS attacks to isolation by IVNPROTECT. Like the previous experiment, we had the Raspberry Pi 3 Model B send 10090 benign messages and DoS attacks simultaneously. In Fig. 38, we show the isolation times with different thresholds of the detection counter for transferring the security bus-off state. As shown in Fig. 38 (a), we confirm that the maximum isolation time is 170 ms if the threshold is 255. Additionally, as shown in Fig. 38 (b), we confirm that the maximum isolation time is 4.073 s if the threshold is 255. In other words, we conclude that there is the time (170 ms and 4.073 s) to report some warning messages for malicious and benign ID DoS, respectively.

6.5.4 Benign Transmission Loss Rate under DoS Attacks

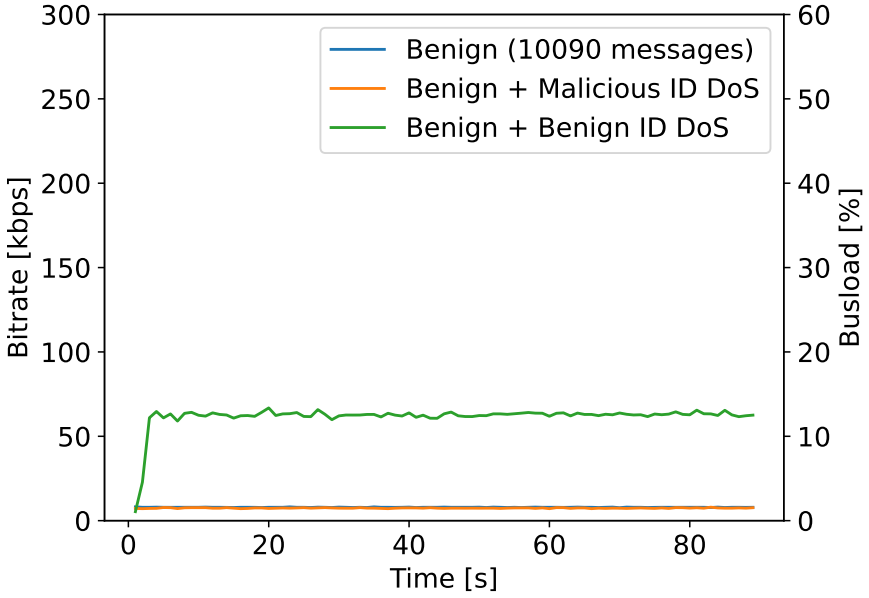
To ensure that IVNPROTECT can report the warning messages, we evaluate the transmission loss rate of benign messages during DoS attacks. Fig. 39 shows the benign transmission loss rate of IVNPROTECT during DoS attacks. The transmission loss rate is 0 % when the transmission queue length is over 350. Thus, we set the transmission queue length to 350 in the following evaluations.

6.5.5 Benign Transmission Delay under DoS Attacks

To verify the number of messages that IVNPROTECT can send during a DoS attack, we evaluate the transmission delay of benign messages during DoS attacks.

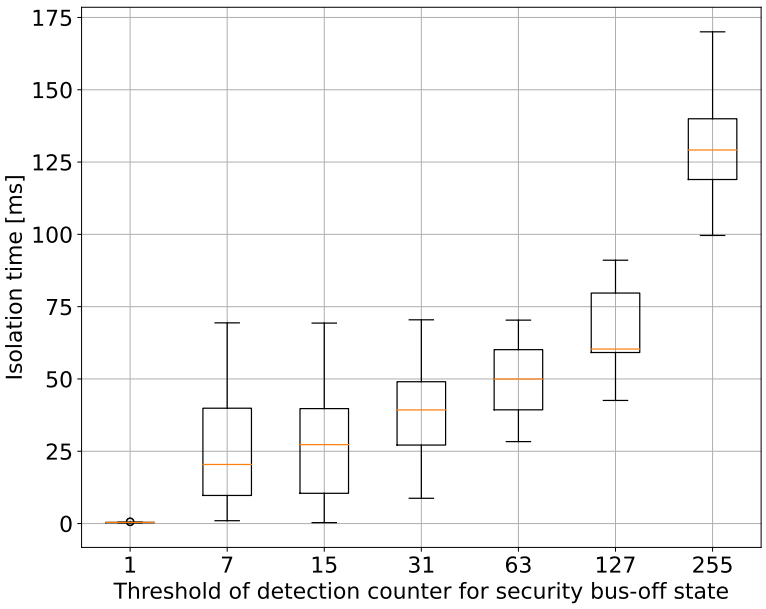


(a) Without IVNPROTECT.

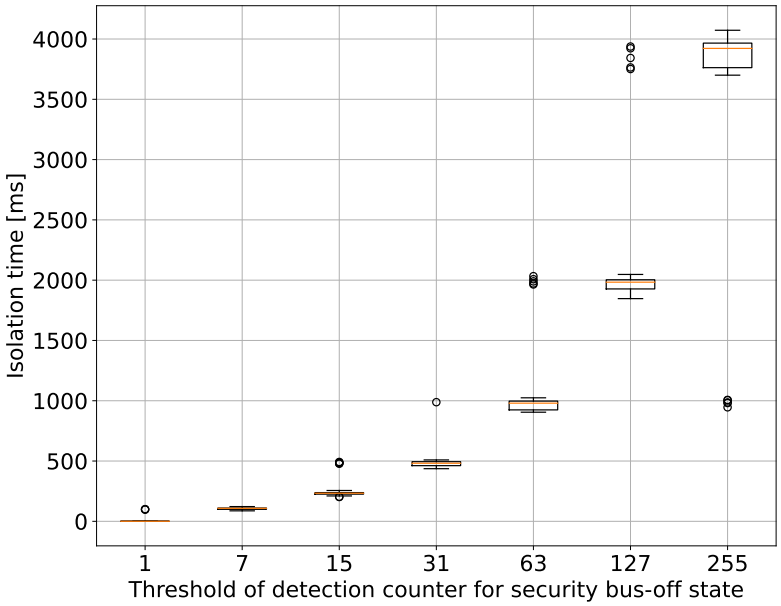


(b) With IVNPROTECT.

Figure 37: Comparison of bitrate and busload between with and without IVNPROTECT.



(a) Malicious ID DoS.



(b) Benign ID DoS.

Figure 38: Isolation time under different thresholds of detection counter for security bus-off state.

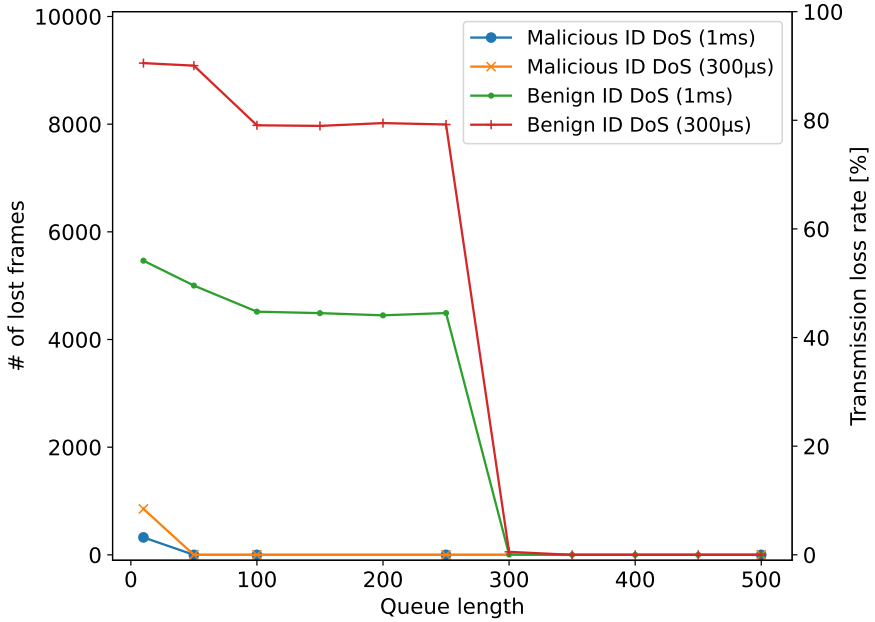


Figure 39: Benign transmission (10090 messages) loss rate under different tx queue size during DoS attacks.

Table 23: Statistics of the arrival time of benign messages.

Traffic	Mean [ms]	Stddev [ms]	Max [ms]
benign	99.977	4.916	111.085
benign+malicious ID DoS (1ms)	99.977	4.709	111.146
benign+malicious ID DoS (300µs)	99.977	5.055	110.976
benign+benign ID DoS (1ms)	100.040	75.665	199.989
benign+benign ID DoS (300µs)	99.989	115.637	333.057

Table 23 shows the arrival time of benign messages during various DoS attacks. The benign traffic in Table 23 expresses the arrival time of benign messages without sending DoS attacks. The benign + malicious ID DoS (1ms) represents the arrival time of benign messages during malicious DoS attacks, which send messages per 1 ms. Similarly, benign + malicious ID DoS (300 µs) represents the arrival time of benign messages during malicious DoS attacks (300 µs), which is the maximum transmission rate.

Comparing benign with benign + malicious ID DoS in Table 23, we confirm that IVNPROTECT does not affect the arrival time of benign messages during malicious ID DoS. In contrast, in benign ID DoS, the maximum delay of benign messages is 199.989 ms and 333.057 ms. In other words, if IVNPROTECT isolates compromised ECU at 333.057 ms after the occurrence of DoS attacks, IVNPROTECT can send at least one message. Based on this evaluation, we determine the optimal threshold for transferring security bus-off state in 6.6.2.

6.5.6 Overhead with IVNProtect

In this section, we evaluate the overhead with IVNPROTECT. To evaluate the overhead, we considered two cases: the Raspberry Pi 3 Model B sends 1000 benign messages with and without IVNPROTECT. Then, we measured the average time between starting to send a benign message and finishing it. As a result, we observed that in the case without IVNPROTECT, the average time was 20.090 193 ms, while in the case with IVNPROTECT, it was 20.099 242 ms. The overhead with IVNPROTECT is 9.049 µs, which is less than the TEECheck's

(494 µs), so we confirm that IVNPROTECT does not affect on benign messages.

6.6 Discussion

6.6.1 Comparison among ECU-side Protections

In this section, we compare IVNPROTECT with previous protections. As described in Sec. 6.2, IVNPROTECT has advantages in terms of unharming messages on the bus when its protection is triggered unlike IDS-side and on-bus protections.

Next, we compare IVNPROTECT with previous ECU-side protections. Table 24 shows a comparison among the ECU-side protections and IVNPROTECT. As described in Sec. 6.2, the secure CAN transceiver proposed by Elend *et al.* [21] can be bypassed using some allowlisted messages so it is effective only against malicious ID DoS attacks. In addition, the secure CAN transceiver uses the leaky-bucket algorithm to manipulate the sending rate of CAN messages. Additionally, it causes the transmission loss rate of benign messages during DoS attacks to increases. Therefore, the secure CAN transceiver cannot report warning messages to others ECUs on the bus while being forced to send DoS attacks by an attacker.

TEECheck [63] proposed by Mishra *et al.* has advantages regarding the complete isolation of a compromised ECU, traceability of the root cause of the bus-off state, and benign messages that are not harmful. However, TEECheck can deploy only to ARM-based ECUs, so it has the limitation of deployability. Moreover, Mishra *et al.* assume that ECUs with TEECheck send only periodic messages as a real-time task model. In other words, TEECheck cannot be deployed to ECUs which carry out aperiodic tasks. In CAN-bus, aperiodic tasks are generally implemented in real-world ECUs, which these tasks are used to implement event-triggered tasks such as airbag control. Therefore, we conclude that TEECheck has a severe limitation of deployment for ECUs with aperiodic tasks.

In contrast, our IVNPROTECT allows the aperiodic tasks to send the messages because the similarity analysis-based detection module does not detect the aperiodic messages as malicious messages [76]. In addition, IVNPROTECT can be installed to an ECU just by software updating without the limitation of hardware such as requiring TrustZone. From the above comparison, IVNPROTECT

Table 24: Comparison among ECU-side protections in Isolability, Traceability for Root-cause of Isolation (T.R.I.), Deployment Cost (D.C.), Harm for Benign Messages of legitimate ECUs (H.B.M.), Adaptation of Aperiodic Messages (A.A.M.), benign Transmission Loss Rate during DoS attacks (T.L.R.).

	Elend *et al.* [21]	TEECheck [63]	IVNProtect
Isolability	Partial	Complete	Complete
T.R.I.	-	Yes	Yes
D.C.	High	Middle	Low
H.B.M.	Unharmful	Unharmful	Unharmful
A.A.M.	Yes	No	Yes
T.L.R.	$\gg 0\%$	0%	0%

has advantages in isolability, deployability, and adaptation to aperiodic tasks.

6.6.2 Warning Response using IVNProtect

For **P3** described in Sec. 6.3.2, IVNProtect must isolate the compromised ECU after reporting some warning messages. To ensure this function work, IVNProtect is required to satisfy the following requirements:

1. A benign transmission loss rate of 0 % during DoS activities
2. A worst arrival time of benign messages less than isolation time by IVNProtect
3. A higher priority of warning messages than the others IDs

As described in Sec. 6.5.4, we confirmed that IVNProtect satisfies the first requirement that the benign transmission loss rate during DoS activities is 0 %. The second requirement depends on the thresholds of the detection counter for transferring the security bus-off state. Specifically, the worst arrival time of benign messages is 0.110976s and 0.333057s for malicious and benign DoS attacks, respectively. Hence, for the second requirement, IVNProtect must isolate the compromised ECU after these times when starting attacks. From the evaluation

of isolation time (Sec. 6.5.3), IVNPROTECT can satisfy the requirement if the thresholds of ID violation error and similarity error are 255 and 31, respectively. Finally, we discuss the third requirement that the ID of warning messages requires a higher priority than the other IDs on the bus. To prevent conflicting the warning messages and the other messages, IVNPROTECT must send the warning messages with the highest priority ID on the bus. Fortunately, we easily solve this problem because such IDs generally exist in the real-world CAN-bus. For example, since the highest ID of our lab's vehicle is `0x020`, we can use the IDs over `0x020` as warning messages.

We conclude that IVNPROTECT can report the warning messages during DoS attacks if the thresholds of ID violation error and similarity error are 255 and 31 and the ID of warning messages is higher than the other messages on the bus.

6.7 Limitation

In this section, we elaborate on the limitation of IVNPROTECT. IVNPROTECT can detect malicious ID DoS and benign ID DoS by allowlist and similarity-analysis. The similarity-analysis module detects the DoS attacks if the similarity of CAN messages exceeds a benign range defined by σ_s. Thus, if an attacker tries to manipulate the similarity of DoS attacks without exceeding the benign range, IVNPROTECT misses the DoS attacks to the CAN-bus. It causes a high busload of the CAN-bus and unexpected vehicle behavior (e.g., disabling power steering, blocking ADAS function). We define this evasion DoS attack against IVNPROTECT as *similarity-manipulation attacks*.

Fortunately, we can mitigate this attack with a well-architected assignment of the CAN IDs in practice. Specifically, we assign the low-priority CAN IDs to an exploitable ECU that has a wireless connection and IVNPROTECT. Moreover, we assign the higher priority CAN ID than the exploitable ECU to the other important ECUs. In consequence, an attacker cannot deny the benign CAN messages using similarity-manipulation attacks because the DoS attack consists of the low-priority CAN IDs. However, to deploy the well-architected assignment of CAN IDs to real-world CAN-buses, automotive manufacturers may have to change the existing assignment of CAN IDs.

6.8 Conclusion

In this paper, we proposed isolable and traceable lightweight CAN-bus kernel-level protection called IVNPROTECT, which has advantages such as deployability and slight overhead compared with previous CAN-bus protections. To allow the other ECUs to trace the root cause of isolation of compromised ECU, IVNPROTECT has a reporting function to send warning messages while an attacker is compromising. Moreover, we confirmed that this reporting function of IVNPROTECT works even during a DoS attack. Finally, since there is not much research on protection for attacking CAN yet, we hope that our kernel-level protection will encourage this research field.

7. Conclusions and Future Research Direction

This dissertation has tackled DoS attacks on CAN from offensive and defensive perspectives. Chap. 3 derives an unveiled evasion attack against state-of-the-art DoS IDS. The last three chapters proposed detection, identification, and protection strategies to permanently disable DoS attacks on CAN. This chapter reviews whether our proposed strategies can prevent DoS attacks on CAN. Additionally, this chapter discusses the open issues toward secure in-vehicle networks.

7.1 Reviewing the Research Contributions

First, we review an unveiled evasion attack and its defensive approach. As mentioned in Sec. 1.4.1, to minimize the security risk of in-vehicle networks, we are motivated to analyze DoS attacks on CAN from offensive and defensive perspectives. In Chap. 3, we unveiled a new evasion attack called *entropy-manipulation attack*. In Sec. 3.6, we confirmed that entropy-manipulation attack can completely evade the state-of-the-art entropy-based IDS. To address the vulnerability of the state-of-the-art IDS, we proposed similarity-based IDS which detects DoS attacks including entropy-manipulation attack with exploiting similarity of CAN messages in Chap. 4. As Sec. 4.3 mentioned, we showed our proposed similarity-based IDS can detect all DoS attacks with 100 % of precision. Moreover, similarity-based IDS reduced the time-complexity of the state-of-the-art IDS's detection phase. It means that similarity-based IDS successfully detects DoS attacks including entropy-manipulation attacks with a more lightweight procedure.

Second, we review physical-layer characteristic-based sender identification. In Chap. 5, we proposed physical-layer identification using time-to-digital converter, called PLI-TDC, which identifies the ECU attacking the bus in order to patch the compromised ECU. In Sec. 5.4.2, we confirmed that our PLI-TDC can identify the ECU in CAN-bus prototype and real-vehicle with 99.67 % and 97.04 % of accuracy, respectively. Hence, we concluded that PLI-TDC satisfies the requirements of identification of compromised ECU which sends DoS attacks.

Third, we review CAN-bus kernel-level protection called IVNPROTECT, which has advantages such as deployability and slight overhead compared with previous

CAN-bus protections. In Sec. 6.5, we confirmed that IVNPROTECT can mitigate the high busload caused by DoS to the benign busload. Also, we verified whether IVNPROTECT can isolate a compromised ECU that conducts malicious activity from the CAN-bus. As the result, we confirmed that IVNPROTECT carried out the isolation of the compromised ECU based on the security error state mechanism which we introduced. In addition, the overhead of IVNPROTECT is 9.049 µs, so that IVNPROTECT can be installed to insecure ECUs with a slight side-effect. Therefore, we concluded that IVNPROTECT prevents and isolates a DoS attacker ECU with slight overhead.

Finally, we summarize the above reviews of the research contributions. First, we disclosed an evasion attack against the state-of-the-art DoS IDS, called *entropy-manipulation attacks*. To address DoS attacks including *entropy-manipulation attacks*, we derived three defensive strategies: similarity-based IDS, PLI-TDC, and IVNPROTECT. From the above reviews, we concluded that the three defensive strategies can provide countermeasures: detection, identification, and protection. Thus, exploiting our three defensive strategies, automotive suppliers/manufacturers can address DoS attacks by just implementing the responsive and recovery function (e.g. reporting for Vehicle Security Operations Center (VSOC) analysts) in the case of DoS attacks.

7.2 Open Issue and Future Research Direction

Although we achieved the disabling DoS attacks on CAN in this dissertation, the studies are still a part of an ultimate goal. The ultimate goal of automotive network security research is to protect and respond to the various IVNs from all types of cyberattacks. This section elaborates the open issues to realize the ultimate goal of automotive network security research.

7.2.1 Protection of Physical-Layer Attacks

Physical-layer data manipulation attack which directly manipulates the voltage of physical bit from compromised ECUs has been proposed [64]. This attack can manipulate the physical bit of CAN to cause `0` $\rightarrow$ `1` by exploiting a physical property of CAN. This attack also requires multiple rogue ECUs to perform the

manipulation of physical voltage so that an attacker cannot perform the physical-layer data manipulation attack on numerous vehicles at a large scale. Therefore, this attack does not pose a critical threat to the critical vehicle-infrastructure. However, it is ideal to protect the vehicles against such targeted attacks.

We proposed the similarity-based IDS and PLI-TDC as state-of-the-art intrusion detection systems in this dissertation. But, these systems cannot identify the physical-layer data manipulation attack because our system exploits message-based characteristics instead of physical bits to detect attacks.

To detect the physical-layer data manipulation attack, monitoring/detecting the bit-wise voltage spikes is one of the practical approaches. However, the voltage spikes of physical bits are also caused by genuine error frames of CAN. Thus, we argue to need to research a method that distinguishes root-cause of voltage spikes and a protection system against the physical-layer data manipulation attack as an open issue.

7.2.2 Sophistication of IDS and PLI's alerts for VSOC analysts

In this dissertation, we confirmed that our similarity-based IDS and PLI-TDC can detect DoS attacks and spoofing attacks with an accuracy of 99 %. However, we have to discuss what to do when the vehicle incorporating the IDS and PLI raise a security alert after deploying our IDS and PLI to real-world CAN-buses. In the case of false alerts, the vehicle incorporating the IDS and PLI may have to stop driving on a shoulder of a road to keep the safety of the vehicle. In consequence, false alerts cause significantly reduce the availability of the vehicle. To mitigate the problem, VSOC can analyze the alerts and then determine whether the vehicle needs to stop driving.

We envisage that the vehicle monitoring system with this VSOC will encounter the same problem as SOC on the Internet. In a previous survey, it was revealed that security analysts in a SOC confirmed the high false alerts of the IDS used, requiring manual validation [1]. To efficiently deal with the high false alerts, it is ideal that the security analysts can distinguish between types of false alerts. Alahmadi *et al.* [1] improved security alerts of IDSs by adding five properties (Reliable, Explainable, Analytical, Contextual, Transferable) required to foster effective and quick validation of alerts. Hence, it is an open issue to identify

suitable properties for improving the security alerts of VSOC to sophisticate IDS's and PLI's alerts.

7.2.3 Robustness of IDS and PLI for Concept Drift Caused by Various Vehicles

In this section, we discuss that robustness of IDS and PLI for concept drift caused by various vehicles. CAN and IVN traffic depend on the vehicle model, generations of the vehicle model and some additional options (e.g., ADAS function). In other words, CAN traffic has the difference even in the same vehicle model. Our similarity-based IDS can adapt to the difference in CAN traffic because it can control a benign range defined by σ_s to permit the difference of CAN traffic. On the other hand, PLI-TDC cannot deal with the difference in CAN traffic because the physical-layer characteristics (e.g., delay-time and voltage) depend on individual vehicles. To address the difference in CAN traffic, PLI-TDC must update its machine learning model to adapt to the difference in physical-layer characteristics. As with the issue of EMI described in Sec. 5.5.6, the updating scheme of the machine learning model can be an attack-surface. It means that an attacker can poison the training data by malicious messages. To deal with the attacker, PLI-TDC requires legitimate training data that do not contain malicious messages. Therefore, it is also a future research direction to be robust in the machine learning model of PLI-TDC against the difference of individual vehicles (same vehicle model).

7.2.4 Improving Accuracy of Physical-Layer Identification

PLI-TDC achieved a mean accuracy of 97.04 - 99.67 %. Also, PLI-TDC detected the messages sent by incorrect sender ECUs with 100 % in the CAN bus prototype and real-vehicle. It means that PLI-TDC can detect the messages sent by an incorrect sender ECU with 100 % and however identify which ECUs are the incorrect sender ECU with a mean accuracy 97.04 - 99.67 %. We assume that PLI-TDC is used to identify a compromised ECU which sends the DoS attacks for responding and recovering from the DoS attacks. Specifically, if PLI-TDC can accurately identify the compromised ECU, VSOC can immediately patch and replace ECU for responding and recovering. To realize the procedure for

responding and recovering, it is ideal to identify the ECUs with a mean accuracy of 100 %. Therefore, it is a future research topic to implement a more accurate PLI than PLI-TDC. One of the ways to implement the accurate PLI is to exploit both voltage-domain and time-domain characteristics. However, it increases the complexity of the machine learning model and requires computational resources. Thus, it is important to study the trade-off between the accuracy of PLI and requiring computational resources which satisfy the constraint of vehicle environments.

7.2.5 Security Analysis and Mechanism for Next-Generation In-Vehicle Networks

Currently, to replace CAN with next-generation in-vehicle networks, various new in-vehicle network protocols have been proposed. Notably, the automotive industry has an interest in *Automotive Ethernet* [2] in terms of high bandwidth, high throughput, and low-cost characteristics. Automotive Ethernet (100Base-T1) is a next-generation in-vehicle network based on BroadR-Reach[4] technology which Broadcom developed. Deploying Automotive Ethernet to real-vehicles, the in-vehicle network enables ECUs to provide service-oriented communication. As one of the service-oriented communication, *Scalable service-Oriented MiddlewarE Over IP* (SOME/IP) [5] was designed to fit small devices like cameras, ECUs, and up to head units or telematics devices. ECUs can use SOME/IP for a new application installed to an ECU by software updating to find some functions in the vehicle. However, since the SOME/IP specification does not consider any security mechanisms, man-in-the-middle attacks have been demonstrated [103].

On the other hand, CAN-XL [30] has been designed to provide 10Mbps as the next step in CAN and CAN-FD evolution. CAN-XL extends the data field length to 2048 bytes from 8 bytes of CAN's data field length. It means that the authentication mechanisms can be easily applied to CAN-XL because there is no limitation to implementing the authentication mechanisms, different from CAN's data field. However, since CAN-XL also inherits the CAN's arbitration scheme, it also requires a DoS attack defense mechanism against CAN-XL. Thus,

[4]BroadR-Reach is a Boradcom point-to-point Ethernet physical-layer (PHY) technology, adopted by the One-Pair Ether-Net Alliance BroadR-Reach PHY (OABR PHY).

IVNPROTECT derived in this dissertation can still be effective in mitigating DoS attacks in CAN-XL.

For securing the next-generation in-vehicle networks, we also have to consider the defensive mechanisms as with CAN. Fortunately, SOME/IP is implemented on the TCP/IP stacks so that we can adapt the network security techniques that do not depend on a specific application layer. Moreover, the automotive security techniques to analyze the characteristics of automotive packets/messages and to meet real-time demands have been studied such as in this dissertation. Therefore, we can apply these techniques to be secure and resilient Automotive Ethernet-based in-vehicle networks. To summarize, in order to realize the protection of the next-generation in-vehicle networks, it is necessary to analyze the networks from both theory and implementation perspectives and to apply the previous network and automotive security techniques.

Acknowledgements

I would like to express my sincere gratitude to all related to my journey to get Ph.D. degree.

First, I thank my advisor, Professor Kazutoshi Fujikawa. He provided appropriate research guidance and research support from various aspects such as gaining experience in international presentations and extensive knowledge. I was always struck and helped by his generosity, and I am sure it made the atmosphere of our laboratory so good. I could never have achieved all my research goals without his support.

I am grateful to Associate Professor Ismail Arai. I was always inspired by his passion for doing research and his leadership in some projects. When our paper was rejected by an international conference and journal, he encouraged me. And, I was able to challenge again an international conference and journal. His great insights also made a big contribution to my research. So, no doubt he was one of the people needed to achieve my research goals.

I appreciate Assistant Professor Masatoshi Kakiuchi, for his various support such as guidance of system administration related to Mandara network and servers. I thank Assistant Professor Arata Endou, for his many comments to improve my research and this dissertation during the laboratory meetings.

I am grateful to Professor Hiroyuki Inoue of Kyoto Sangyo University. He was my advisor during my undergraduate, and I could learn the excitement of cybersecurity research such as automotive security. When I submitted the first and second journals, he helped me by doing meetings and commenting on my journal. He was also one of the people needed to achieve my research goals.

I especially thank all inet-lab members and my friends for their camaraderie. In particular, I am thankful to Dr. Araya Kibrom Desta and Mr. Tomoya Kitagawa. They were labmates in the same research field of automotive security. They also provided many discussions and comments to me on my research. I thank Ms. Ayako Nakano and Ms. Rie Tsujimoto, who are secretaries of the inet-lab, for their warmful support.

Finally, I would like to express my sincere gratitude to my family for understanding my life and providing a great deal of support.

This research in this dissertation was partially supported by Japan Society

for the Promotion of Science (JSPS) Research Fellowships for Young Scientists (DC2) Grant Number: 21J10516.

References

[1] Bushra A Alahmadi, Louise Axon, and Ivan Martinovic. 99% False Positives: A Qualitative Study of SOC Analysts ' Perspectives on Security Alarms. In *Proceedings of the 31st USENIX Security Symposium (USENIX Security 22)*, pages 10–12, 2022.

[2] OPEN Alliance. Automotive Ethernet Specifications. `https://www.opensig.org/about/specifications/`, 12 2021. (Accessed on 7/8/2022).

[3] ams. AS6500 Time-to-Digital Converter. `https://ams.com/ja/as6500`, 2020. (Accessed on 10/25/2020).

[4] Yasuo Arai and Masahiro Ikeno. A Time Digitizer CMOS Gate-Array with a 250 ps Time Resolution. *IEEE Journal of Solid-State Circuits*, 31(2):212–220, 1996.

[5] AUTOSAR. SOME/IP Protocol Specification. `https://www.autosar.org/fileadmin/user_upload/standards/foundation/1-0/AUTOSAR_PRS_SOMEIPProtocol.pdf`, 11 2016. (Accessed on 7/8/2022).

[6] Pranshu Bajpai, Richard Enbody, and Betty HC Cheng. Ransomware Targeting Automobiles. In *Proceedings of the Second ACM Workshop on Automotive and Aerial Vehicle Security (AutoSec)*, pages 23–29, 2020.

[7] Rohit Bhatia, Vireshwar Kumar, Khaled Serag, Z Berkay Celik, Mathias Payer, and Dongyan Xu. Evading Voltage-Based Intrusion Detection on Automotive CAN. In *Proceedings of the 28th Network and Distributed System Security Symposium (NDSS)*, 2021.

[8] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno, et al. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *Proceedings of the 20th USENIX Security Symposium (USENIX Security 11)*, volume 4, pages 447–462. San Francisco, 2011.

[9] Kyong-Tak Cho and Kang G Shin. Fingerprinting Electronic Control Units for Vehicle Intrusion Detection. In *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*, pages 911–927, 2016.

[10] Kyong-Tak Cho and Kang G Shin. Viden: Attacker Identification on In-Vehicle Networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1109–1123. ACM, 2017.

[11] Wonsuk Choi, Hyo Jin Jo, Samuel Woo, Ji Young Chun, Jooyoung Park, and Dong Hoon Lee. Identifying ECUs Using Inimitable Characteristics of Signals in Controller Area Networks. *IEEE Transactions on Vehicular Technology*, 67(6):4757–4770, 2018.

[12] 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.

[13] JA Cook and JS Freudenberg. Controller Area Network (CAN). *EECS*, 461:1–5, 2007.

[14] The MITRE Corporation. CVE-2020-5551. `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5551`, 01 2020. (Accessed on 04/26/2020).

[15] Craig Smith. The Car Hacker's Handbook A Guide for the Penetration Tester. `https://docs.alexomar.com/biblioteca/thecarhackershandbook.pdf`. (Accessed: 2019-12-17).

[16] Robert I Davis, Alan Burns, Reinder J Bril, and Johan J Lukkien. Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised. *Real-Time Systems*, 35(3):239–272, 2007.

[17] Araya Kibrom Desta, Shuji Ohira, Ismail Arai, and Kazutoshi Fujikawa. ID Sequence Analysis for Intrusion Detection in the CAN Bus Using Long Short Term Memory Networks. In *2020 IEEE International Conference*

on Pervasive Computing and Communications Workshops (PerCom Workshops), pages 1–6. IEEE, 2020.

[18] Araya Kibrom Desta, Shuji Ohira, Ismail Arai, and Kazutoshi Fujikawa. Long Short-Term Memory Networks for In-Vehicle Networks Intrusion Detection Using Reverse Engineered Automotive Packets. *Journal of Information Processing*, 28:611–622, 2020.

[19] Araya Kibrom Desta, Shuji Ohira, Ismail Arai, and Kazutoshi Fujikawa. MLIDS: Handling Raw High-Dimensional CAN Bus Data Using Long Short-Term Memory Networks for Intrusion Detection in In-Vehicle Networks. In *2020 30th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–7. IEEE, 2020.

[20] Tamer ElBatt, Siddhartha K Goel, Gavin Holland, Hariharan Krishnan, and Jayendra Parikh. Cooperative collision warning using dedicated short range wireless communications. In *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, pages 1–9, 2006.

[21] Bernd Elend and Tony Adamson. Cyber Security Enhancing CAN Transceivers. In *Proceedings of the 16th International CAN Conference*, 2017.

[22] Alessandro Erba, Riccardo Taormina, Stefano Galelli, Marcello Pogliani, Michele Carminati, Stefano Zanero, and Nils Ole Tippenhauer. Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems. In *Annual Computer Security Applications Conference*, pages 480–495, 2020.

[23] Mahsa Foruhandeh, Yanmao Man, Ryan Gerdes, Ming Li, and Thidapat Chantem. SIMPLE: Single-Frame Based Physical Layer Identification for Intrusion Detection and Prevention on In-Vehicle Networks. In *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC)*, pages 229–244, 2019.

[24] Linux Foundation. Automotive Grade Linux. `https://www.automotivelinux.org/`, 7 2019. (Accessed on 07/08/2019).

[25] Ryan M Gerdes, Thomas E Daniels, Mani Mina, and Steve Russell. Device Identification via Analog Signal Fingerprinting: A Matched Filter Approach. In *Proceedings of the 13th Network and Distributed System Security Symposium (NDSS)*. Citeseer, 2006.

[26] Ryan M Gerdes, Mani Mina, Steve F Russell, and Thomas E Daniels. Physical-Layer Identification of Wired Ethernet Devices. *IEEE Transactions on Information Forensics and Security*, 7(4):1339–1353, 2012.

[27] Ryan Michael Kepke Gerdes. *Physical Layer Identification: Methodology, Security, and Origin of Variation*. PhD thesis, Iowa State University, 2011.

[28] Robert Bosch GmbH. CAN Specification Version 2.0. `http://esd.cs.ucr.edu/webres/can20.pdf`, 1991. (Accessed on 07/08/2019).

[29] Robert Bosch GmbH. CAN with Flexible Data-Rate Specification Version 1.0. `https://can-newsletter.org/assets/files/ttmedia/raw/e5740b7b5781b8960f55efcc2b93edf8.pdf`, 7 2019. (Accessed on 07/08/2019).

[30] Robert Bosch GmbH. CAN XL – THE NEXT STEP IN CAN EVOLUTION. `https://www.bosch-semiconductors.com/media/ip_modules/pdf_2/can_xl_1/canxl_intro_20210225.pdf`, 2 2021. (Accessed on 11/27/2021).

[31] Bogdan Groza and Pal-Stefan Murvay. Efficient Intrusion Detection with Bloom Filtering in Controller Area Networks. *IEEE Transactions on Information Forensics and Security*, 14(4):1037–1051, 2018.

[32] Bogdan Groza, Pal-Stefan Murvay, Lucian Popa, and Camil Jichici. CANSQUARE-Decimeter Level Localization of Electronic Control Units on CAN Buses. In *European Symposium on Research in Computer Security (ESORICS)*, 2021.

[33] Bogdan Groza, Stefan Murvay, Anthony Van Herrewege, and Ingrid Verbauwhede. LiBrA-CAN: A Lightweight Broadcast Authentication Protocol for Controller Area Networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(3):90, 2017.

[34] Bogdan Groza, Lucian Popa, Pal-Stefan Murvay, Yuval Elovici, and Asaf Shabtai. {CANARY}-A Reactive Defense Mechanism for Controller Area Networks based on Active RelaYs. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security 21)*, pages 1–18, 2021.

[35] Joffrey Guilbon. Attacking the ARM's TrustZone. `https://blog.quarkslab.com/attacking-the-arms-trustzone.html`, 2018. (Accessed on 02/17/2022).

[36] Markus Hanselmann, Thilo Strauss, Katharina Dormann, and Holger Ulmer. CANet: An Unsupervised Intrusion Detection System for High Dimensional CAN Bus Data. *IEEE Access*, 8:58194–58205, 2020.

[37] Anthony Van Herrewege, Dave Singelee, and Ingrid Verbauwhede. CANAuth-A Simple, Backward Compatible Broadcast Authentication Protocol for CAN Bus. In *ECRYPT Workshop on Lightweight Cryptography*, volume 2011, pages 1–7, 2011.

[38] Abdulmalik Humayed, Fengjun Li, Jingqiang Lin, and Bo Luo. CANSentry: Securing CAN-Based Cyber-Physical Systems against Denial and Spoofing Attacks. In *European Symposium on Research in Computer Security (ESORICS)*, pages 153–173. Springer, 2020.

[39] Abdulmalik Humayed and Bo Luo. Using ID-Hopping to Defend against Targeted DoS on CAN. In *Proceedings of the 1st International Workshop on Safe Control of Connected and Autonomous Vehicles*, pages 19–26. ACM, 2017.

[40] Muhammad Sabir Idrees, Hendrik Schweppe, Yves Roudier, Marko Wolf, Dirk Scheuermann, and Olaf Henniger. Secure Automotive On-Board Protocols: A Case of Over-The-Air Firmware Updates. In *International Workshop on Communication Technologies for Vehicles*, pages 224–238. Springer, 2011.

[41] Riadul Islam and Rafi Ud Daula Refat. Improving CAN Bus Security by Assigning Dynamic Arbitration IDs. *Journal of Transportation Security*, 13(1):19–31, 2020.

[42] Shalabh Jain and Jorge Guajardo. Physical Layer Group Key Agreement for Automotive Controller Area Networks. In *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, pages 85–105. Springer, 2016.

[43] Min-Joo Kang and Je-Won Kang. Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security. *PloS one*, 11(6):e0155781, 2016.

[44] Marcel Kneib and Christopher Huth. Scission: Signal Characteristic-Based Sender Identification and Intrusion Detection in Automotive Networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 787–800. ACM, 2018.

[45] Marcel Kneib, Oleg Schell, and Christopher Huth. On the Robustness of Signal Characteristic-Based Sender Identification. 2019.

[46] Marcel Kneib, Oleg Schell, and Christopher Huth. EASI: Edge-Based Sender Identification on Resource-Constrained Platforms for Automotive Networks. In *Proceedings of the 27th Network and Distributed System Security Symposium (NDSS)*, pages 1–16, 2020.

[47] Igor Kononenko. Estimating Attributes: Analysis and Extensions of RELIEF. In *European conference on machine learning*, pages 171–182. Springer, 1994.

[48] Vipin Kumar Kukkala, Sooryaa Vignesh Thiruloga, and Sudeep Pasricha. INDRA: Intrusion Detection Using Recurrent Autoencoders in Automotive Embedded Systems. *arXiv preprint arXiv:2007.08795*, 2020.

[49] Sekar Kulandaivel. *Revisiting Remote Attack Kill-Chains on Modern In-Vehicle Networks*. PhD thesis, Carnegie Mellon University, 2021.

[50] Sekar Kulandaivel, Tushar Goyal, Arnav Kumar Agrawal, and Vyas Sekar. CANvas: Fast and Inexpensive Automotive Network Mapping. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security 19)*, pages 389–405, 2019.

[51] Sekar Kulandaivel, Shalabh Jain, Jorge Guajardo, and Vyas Sekar. CANnon: Reliable and Stealthy Remote Shutdown Attacks via Unaltered Automotive Microcontrollers. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 195–210. IEEE, 2021.

[52] Ryo Kurachi, Yutaka Matsubara, Hiroaki Takada, Naoki Adachi, Yukihiro Miyashita, and Satoshi Horihata. CaCAN-Centralized Authentication System in CAN (Controller Area Network). In *14th Int. Conf. on Embedded Security in Cars (ESCAR 2014)*, 2014.

[53] Ryo Kurachi, Hiroaki Takada, Naoki Adachi, Hiroshi Ueda, and Yukihiro Miyashita. DDCAN: Delay-Time Deliverable CAN Network. In *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 36–41. IEEE, 2019.

[54] Takuya Kuwahara, Yukino Baba, Hisashi Kashima, Takeshi Kishikawa, Junichi Tsurumi, Tomoyuki Haga, Yoshihiro Ujiie, Takamitsu Sasaki, and Hideki Matsushima. Supervised and Unsupervised Intrusion Detection Based on CAN Message Frequencies for In-Vehicle Network. *Journal of Information Processing*, 26:306–313, 2018.

[55] Hyunsung Lee, Seong Hoon Jeong, and Huy Kang Kim. OTIDS: A Novel Intrusion Detection System for In-Vehicle Network by Using Remote Frame. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 57–5709. IEEE, 2017.

[56] Teri Lenard and Roland Bolboaca. A Statefull Firewall and Intrusion Detection System Enforced with Secure Logging for Controller Area Network. In *European Interdisciplinary Cybersecurity Conference (EICC)*, pages 39–45, 2021.

[57] Jiajia Liu, Shubin Zhang, Wen Sun, and Yongpeng Shi. In-Vehicle Network Attacks and Countermeasures: Challenges and Future Directions. *IEEE Network*, 31(5):50–58, 2017.

[58] Aravind Machiry, Eric Gustafson, Chad Spensky, Christopher Salls, Nick Stephens, Ruoyu Wang, Antonio Bianchi, Yung Ryn Choe, Christopher

Kruegel, and Giovanni Vigna. BOOMERANG: Exploiting the Semantic Gap in Trusted Execution Environments. In *NDSS*, 2017.

[59] Mirco Marchetti and Dario Stabili. Anomaly Detection of CAN Bus Messages through Analysis of ID Sequences. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1577–1583. IEEE, 2017.

[60] Mirco Marchetti, Dario Stabili, Alessandro Guido, and Michele Colajanni. Evaluation of Anomaly Detection for In-Vehicle Networks through Information-Theoretic Algorithms. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–6. IEEE, 2016.

[61] Michael Müter, André Groll, and Felix C. Freiling. A Structured Approach to Anomaly Detection for In-Vehicle Networks. In *2010 Sixth International Conference on Information Assurance and Security*, pages 92–98, 2010.

[62] Charlie Miller and Chris Valasek. Remote Exploitation of An Unaltered Passenger Vehicle. *Black Hat USA*, 2015:1–91, 2015.

[63] Tanmaya Mishra, Thidapat Chantem, and Ryan Gerdes. TEECheck: Securing Intra-Vehicular Communication Using Trusted Execution. In *Proceedings of the 28th International Conference on Real-Time Networks and Systems (RTNS)*, pages 128–138, 2020.

[64] Abdullah Zubair Mohammed, Yanmao Man, Ryan Gerdes, Ming Li, and Z Berkay Celik. Physical Layer Data Manipulation Attacks on the CAN Bus. In *Workshop on Automotive and Autonomous Vehicle Security (AutoSec) 2022*, pages 1–5, 2022.

[65] Philipp Mundhenk, Andrew Paverd, Artur Mrowca, Sebastian Steinhorst, Martin Lukasiewycz, Suhaib A Fahmy, and Samarjit Chakraborty. Security in Automotive Networks: Lightweight Authentication and Authorization. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 22(2):25, 2017.

[66] Philipp Mundhenk, Sebastian Steinhorst, Martin Lukasiewycz, Suhaib A Fahmy, and Samarjit Chakraborty. Lightweight Authentication for Secure Automotive Networks. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 285–288. IEEE, 2015.

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

[68] Pal-Stefan Murvay and Bogdan Groza. TIDAL-CAN: Differential Timing Based Intrusion Detection and Localization for Controller Area Network. *IEEE Access*, 8:68895–68912, 2020.

[69] Michael Müter and Naim Asaj. Entropy-Based Anomaly Detection for In-Vehicle Networks. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 1110–1115. IEEE, 2011.

[70] Minki Nam, Seungyoung Park, and Duk Soo Kim. Intrusion Detection Method Using Bi-Directional GPT for In-Vehicle Controller Area Networks. *IEEE Access*, 9:124931–124944, 2021.

[71] Sen Nie, Ling Liu, and Yuefeng Du. Free-Fall: Hacking Tesla from Wireless to CAN Bus. *Black Hat USA*, 2017:1–16, 2017.

[72] Stefan Nürnberger and Christian Rossow. –vatiCAN–Vetted, Authenticated CAN Bus. In *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, pages 106–124. Springer, 2016.

[73] Shuji Ohira. Similarity-based IDS. `https://github.com/shuji-oh/similarity_CAN_IDS`, 7 2019. (Accessed on 07/08/2019).

[74] Shuji Ohira. PLI-TDC. `https://github.com/shuji-oh/PLI_TDC_for_CAN`, 2020. (Accessed: 2020-10-25).

[75] Shuji Ohira. IVNProtect. `https://github.com/shuji-oh/ivnprotect`, 2022. (Accessed: 2022-02-21).

[76] Shuji Ohira, Araya Kibrom Desta, Ismail Arai, Hiroyuki Inoue, and Kazutoshi Fujikawa. Normal and Malicious Sliding Windows Similarity Analysis Method for Fast and Accurate IDS against DoS Attacks on In-Vehicle Networks. *IEEE Access*, 8:42422–42435, 2020.

[77] Shuji Ohira, Hiroyuki Inoue, Ismail Arai, and Kazutoshi Fujikawa. DoS Attack Mitigation Method on CAN Bus by Whitelist and Delay Addition in In-Vehicle Infotainment System. *Computer Security Symposium 2018*, 2018(2):1128–1133.

[78] Shuji Ohira, Araya Kibrom Desta, Ismail Arai, and Kazutoshi Fujikawa. PLI-TDC: Super Fine Delay-Time Based Physical-Layer Identification with Time-to-Digital Converter for In-Vehicle Networks. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pages 1–11. ACM, 2021.

[79] Shuji Ohira, Araya Kibrom Desta, Tomoya Kitagawa, Ismail Arai, and Kazutoshi Fujikawa. Divider: Delay-Time Based Sender Identification in Automotive Networks. In *IEEE 44th Annual Computer Software and Applications Conference (COMPSAC)*, pages 1490–1497. IEEE, 2020.

[80] International Organization and Standardization (ISO). ISO 11898: Road Vehicles–Interchange of Digital Information–Controller Area Network (CAN) for High-Speed Communication. `https://www.iso.org/standard/20380.html`, 1993.

[81] Mert D. Pesé, Jay W. Schauer, Junhui Li, and Kang G. Shin. S2-CAN: Sufficiently Secure Controller Area Network. In *Proceedings of the 37th Annual Computer Security Applications Conference (ACSAC)*, ACSAC, page 425–438, New York, NY, USA, 2021. Association for Computing Machinery.

[82] Andreea-Ina Radu and Flavio D Garcia. LeiA: A Lightweight Authentication Protocol for CAN. In *European Symposium on Research in Computer Security (ESORICS)*, pages 283–300. Springer, 2016.

[83] Marcel Rumez, Jürgen Dürrwang, Tim Brecht, Timo Steinshorn, Peter

Neugebauer, Reiner Kriesten, and Eric Sax. CAN Radar: Sensing Physical Devices in CAN Networks based on Time Domain Reflectometry. 2019.

[84] Oleg Schell and Marcel Kneib. VALID: Voltage-Based Lightweight Intrusion Detection for the Controller Area Network. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 225–232. IEEE, 2020.

[85] James Scobie. A Starter's Guide to Arm Processing Power in Automotive - Embedded blog - System - Arm Community. `https://community.arm.com/developer/ip-products/system/b/embedded-blog/posts/a-starters-guide-to-arm-processing-power-in-automotive`, 07 2018. (Accessed on 05/31/2020).

[86] Eunbi Seo, Hyun Min Song, and Huy Kang Kim. GIDS: GAN Based Intrusion Detection System for In-Vehicle Network. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6. IEEE, 2018.

[87] P Sivakumar, RS Sandhya Devi, A Neeraja Lakshmi, B VinothKumar, and B Vinod. Automotive grade linux software architecture for automotive infotainment system. In *2020 International Conference on Inventive Computation Technologies (ICICT)*, pages 391–395. IEEE, 2020.

[88] Hyun Min Song, Ha Rang Kim, and Huy Kang Kim. Intrusion Detection System Based on the Analysis of Time Intervals of CAN Messages for In-Vehicle Network. In *2016 international conference on information networking (ICOIN)*, pages 63–68. IEEE, 2016.

[89] Jian Song, Qi An, and Shubin Liu. A High-Resolution Time-to-Digital Converter Implemented in Field-Programmable-Gate-Arrays. *IEEE Transactions on Nuclear Science*, 53(1):236–241, 2006.

[90] Masaru Takada, Yuki Osada, and Masakatu Morii. Counter Attack against the Bus-off Attack on CAN. In *2019 14th Asia Joint Conference on Information Security (AsiaJCIS)*, pages 96–102. IEEE, 2019.

[91] Linus Torvalds. mcp251x.c - Kernel source tree for Raspberry Pi-provided kernel builds. `https://github.com/raspberrypi/linux/blob/4a1f59200d36993f6b32742c03c154ae275bd89c/drivers/net/can/spi/mcp251x.c`, 2022. (Accessed: 2022-02-21).

[92] Jo Van Bulck, Jan Tobias Mühlberg, and Frank Piessens. VulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks. In *Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC)*, pages 225–237. ACM, 2017.

[93] Cliff Wang, Ryan M Gerdes, Yong Guan, and Sneha Kumar Kasera. *Digital Fingerprinting*. Springer, 2016.

[94] Qian Wang, Yiming Qian, Zhaojun Lu, Yasser Shoukry, and Gang Qu. A Delay Based Plug-in-Monitor for Intrusion Detection in Controller Area Network. In *2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pages 86–91. IEEE, 2018.

[95] Zhuo Wei, Yanjiang Yang, and Tieyan Li. Authenticated CAN Communications Using Standardized Cryptographic Techniques. In *International Conference on Information Security Practice and Experience*, pages 330–343. Springer, 2016.

[96] Samuel Woo, Daesung Moon, Taek-Young Youn, Yousik Lee, and Yongeun Kim. CAN ID Shuffling Technique (CIST): Moving Target Defense Strategy for Protecting In-Vehicle CAN. *IEEE Access*, 7:15521–15536, 2019.

[97] Jinyuan Wu and Zonghan Shi. The 10-ps Wave Union TDC: Improving FPGA TDC Resolution beyond Its Cell Delay. In *2008 IEEE Nuclear Science Symposium Conference Record*, pages 3440–3446. IEEE, 2008.

[98] Wufei Wu, Yizhi Huang, Ryo Kurachi, Gang Zeng, Guoqi Xie, Renfa Li, and Keqin Li. Sliding Window Optimized Information Entropy Analysis Method for Intrusion Detection on In-Vehicle Networks. *IEEE Access*, 6:45233–45245, 2018.

[99] Wufei Wu, Ryo Kurachi, Gang Zeng, Yutaka Matsubara, Hiroaki Takada, Renfa Li, and Keqin Li. IDH-CAN: A Hardware-Based ID Hopping CAN Mechanism With Enhanced Security for Automotive Real-Time Applications. *IEEE Access*, 6:54607–54623, 2018.

[100] Wufei Wu, Renfa Li, Guoqi Xie, Jiyao An, Yang Bai, Jia Zhou, and Keqin Li. A Survey of Intrusion Detection for In-Vehicle Networks. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[101] Yang Xiao, Shanghao Shi, Ning Zhang, Wenjing Lou, and Y Thomas Hou. Session Key Distribution Made Practical for CAN and CAN-FD Message Authentication. In *Proceedings of the 36th Annual Computer Security Applications Conference (ACSAC)*, pages 681–693, 2020.

[102] Mahmut Yazici, Shadi Basurra, and Mohamed Gaber. Edge Machine Learning: Enabling Smart Internet of Things Applications. *Big Data and Cognitive Computing*, 2(3):1–17, 2018.

[103] Daniel Zelle, Timm Lauser, Dustin Kern, and Christoph Krauß. Analyzing and Securing SOME/IP Automotive Services with Formal and Practical Methods. In *The 16th International Conference on Availability, Reliability and Security*, pages 1–20, 2021.

[104] Jia Zhou, Prachi Joshi, Haibo Zeng, and Renfa Li. BTMonitor: Bit-Time-Based Intrusion Detection and Attacker Identification in Controller Area Network. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(6):1–23, 2019.

Achievements

Papers marked with (*) are closely related papers of this doctoral dissertation.

Refereed Journals

*Shuji Ohira, Araya Kibrom Desta, Ismail Arai, Hiroyuki Inoue, Kazutoshi Fujikawa, "Normal and Malicious Sliding Windows Similarity Analysis Method for Fast and Accurate IDS against DoS Attacks on In-Vehicle Networks," *IEEE Access*, Vol.8, pp.42422-42435, Feb. 2020.

大平修慈, 井上博之, 新井イスマイル, 藤川和利, "車載 LAN へ侵入するマルウェアの証拠保全を行うカーネル上のフォレンジック機構," 情報処理学会論文誌, Vol.60, No.3, pp.791-802, Mar. 2019.

Araya Kibrom Desta, **Shuji Ohira**, Ismail Arai, Kazutoshi Fujikawa, "Rec-CNN: In-Vehicle Networks Intrusion Detection Using Convolutional Neural Networks Trained on Recurrence Plots," *Elsevier Vehicular Communications*, Vol.35, pp.1-13, June. 2022.

Araya Kibrom Desta, **Shuji Ohira**, Ismail Arai, Kazutoshi Fujikawa, "Long Short-Term Memory Networks for Intrusion Detection Using Reverse Engineered Automotive Packets," *Journal of Information Processing*, Vol.28, pp.611-622, Sep. 2020.

International Conference

*Shuji Ohira, Araya Kibrom Desta, Ismail Arai, Kazutoshi Fujikawa, "IVNProtect: Isolable and Traceable Lightweight CAN-Bus Kernel-Level Protection for Securing In-Vehicle Communication," *The 9th International Conference on Information Systems Security and Privacy (ICISSP)*, pp.17-28, Feb. 2023.

*Shuji Ohira, Araya Kibrom Desta, Ismail Arai, Kazutoshi Fujikawa, "PLI-TDC: Super Fine Delay-Time Based Physical-Layer Identification with Time-

to-Digital Converter for In-Vehicle Networks," *The 16th ACM ASIA Conference on Computer and Communications Security (ASIACCS)*, pp.176-186, June. 2021.

*__Shuji Ohira__, Araya Kibrom Desta, Tomoya Kitagawa, Ismail Arai, Kazutoshi Fujikawa, "Divider: Delay-Time Based Sender Identification in Automotive Networks," *IEEE 44th Annual Computer Software and Applications Conference (COMPSAC)*, pp.1490-1497, July. 2020.

Araya Kibrom Desta, __Shuji Ohira__, Ismail Arai, Kazutoshi Fujikawa, "U-CAN: A Convolutional Neural Network Based Intrusion Detection for Controller Area Networks," *IEEE 46th Annual Computer Software and Applications Conference (COMPSAC)*, pp.1481-1488, July. 2022.

Araya Kibrom Desta, __Shuji Ohira__, Ismail Arai, Kazutoshi Fujikawa, "MLIDS: Handling Raw High-Dimensional CAN Bus Data using Long Short-Term Memory Networks for Intrusion Detection in In-Vehicle Networks," *International Telecommunication Networks and Applications Conference (ITNAC)*, IEEE, pp.1-7, Nov. 2020.

Araya Kibrom Desta, __Shuji Ohira__, Ismail Arai, Kazutoshi Fujikawa, "ID Sequence Analysis for Intrusion Detection in the CAN bus using Long Short Term Memory Networks," *IEEE International Conference on Pervasive Computing (PerCom), SPT-IoT: 4th Workshop on Security, Privacy and Trust in the Internet of Things*, pp.59-64, Mar. 2020.

Domestic Conference

*__大平修慈__, Araya Kibrom Desta, 新井イスマイル, 藤川和利, "TDC による遅延時間の高時間分解能観測に基づく CAN メッセージの送信元識別手法," 研究報告コンピュータセキュリティ（*CSEC*）, pp.1-8, Dec. 2019.

*__大平修慈__, 新井イスマイル, 井上博之, 藤川和利, "車載インフォテインメン

トシステムにおけるホワイトリストと遅延付加による CAN バス上の DoS 攻撃緩和手法," コンピュータセキュリティシンポジウム *2018* 論文集, pp.1128-1133, Oct. 2018.