

IVNPROTECT: Isolable and Traceable Lightweight CAN-Bus Kernel-Level Protection for Securing in-Vehicle Communication

Shuji Ohira¹^a, Araya Kibrom Desta¹^b, Ismail Arai²^c and Kazutoshi Fujikawa²^d

¹Graduate School of Science and Technology, Nara Institute of Science and Technology, Ikoma 630-0192, Japan

²Information Initiative Center, Nara Institute of Science and Technology, Ikoma 630-0192, Japan

Keywords: Automotive Security, Controller Area Network, Intrusion Prevention System, Operation System Kernel, Loadable Kernel Module.

Abstract: Cyberattacks on In-Vehicle Networks (IVNs) are becoming the most urgent issue. The Controller Area Network (CAN), one of the IVNs, is a standard protocol for automotive networks. Many researchers have tackled the security issues of CAN, such as the vulnerability of Denial-of-Service (DoS) attacks and impersonation attacks. Though existing methods can prevent DoS attacks, they have problems in deployment cost, isolability of a compromised Electronic Control Unit (ECU), and traceability for the root cause of isolation. Thus, we tackle to prevent DoS attacks on CAN. To solve these problems of the existing methods, we propose an isolable and traceable CAN-bus kernel-level protection called IVNPROTECT. IVNPROTECT can be installed on an ECU, which has a wireless interface, just by the software updating 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 identifiers. After mitigating DoS attacks, IVNPROTECT isolates a compromised ECU with a security error state mechanism, which handles security errors in IVNPROTECT. And, we evaluate the traceability that an ECU with IVNPROTECT can report warning messages to the other ECUs on the bus even while being forced to send DoS attacks by an attacker. In addition, the overhead of IVNPROTECT is 9.049 μ s, so that IVNPROTECT can be installed on insecure ECUs with a slight side-effect.

1 INTRODUCTION

Recently, cyberattacks on In-Vehicle Networks (IVNs) are becoming a severe problem (Nie et al., 2017; Miller and Valasek, 2015). These attacks abuse vulnerable Controller Area Network (CAN) (Robert-Bosch-GmbH, 1991), one of the IVNs used to communicate among Electronic Control Units (ECUs), which has been the de-facto standard. Also, ECU is an in-vehicle embedded system to electronically manipulate automotive functions (e.g., engine, gear, steering). Nie *et al.* successfully controlled some automotive functions, exploiting the vulnerabilities in a CAN and a web browser in the in-vehicle system implemented by WebKit of the old version (Nie et al., 2017). And, Miller and Valasek also successfully controlled a variety of automotive functions through an

in-vehicle infotainment system with the QNX real-time operating system (Miller and Valasek, 2015). In addition, automotive manufacturers and suppliers promote the use of open-source software such as Automotive Grade Linux (AGL) (Linux-Foundation, 2019) in in-vehicle infotainment systems to enhance the reusability of source codes. If the open-source software has vulnerabilities, an attacker can launch malware against the vulnerable systems on a large scale. Therefore, cybersecurity countermeasures for automobiles are urgently required.

Encryption and authentication (Pesé et al., 2021; Kurachi et al., 2014) for CAN have been proposed to prevent spoofing attacks from a compromised ECU. However, encryption and authentication mechanisms cannot deal with Denial-of-Service (DoS) attacks since an attacker can flood encrypted CAN-bus with DoS using a high-priority identifier (ID). Therefore, a dedicated countermeasure is required to prevent DoS attacks.

On the other hand, many automotive security researchers have proposed intrusion detection systems

^a <https://orcid.org/0000-0002-6966-0977>

^b <https://orcid.org/0000-0002-4363-8560>

^c <https://orcid.org/0000-0002-5353-2401>

^d <https://orcid.org/0000-0003-1055-5790>

(IDSs) based on various characteristics (e.g., frequency (Song et al., 2016), entropy (Wu et al., 2018a), ID sequence (Marchetti and Stabili, 2017), message correlation (Müter et al., 2010; Ohira et al., 2020), physical-layer fingerprint (Kneib and Huth, 2018; Kneib et al., 2020; Ohira et al., 2021)). These IDSs achieved the detection of various attacks including DoS attacks with high accuracy. However, these IDSs do not provide any prevention mechanism. Therefore, it is necessary to build a prevention mechanism with features from attack detection to prevention.

Some countermeasures to disable DoS attacks on CAN have been researched. The allowlist-based mitigation method (Elend and Adamson, 2017) 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 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 (Takada et al., 2019) 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, the 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 a 0% of transmission loss rate. Moreover, the overhead of IVNPROTECT is $9.049\ \mu\text{s}$, 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 repository¹.

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 a security incident.
- We show the overhead caused by IVNPROTECT. As a result, the overhead takes only $9.049\ \mu\text{s}$, which means that a system with our IVNPROTECT satisfies in-vehicle real-time demands.

2 PRELIMINARIES

2.1 Controller Area Network

CAN is one of the IVNs widely used to interconnect among ECUs and has been a standard protocol. As shown in Fig. 1, a CAN message has a maximum data field of 8 bytes, whose length is determined by the Data Length Code (DLC) in the control field. The data field delivers sensor data and control data such as running speed, engine speed, and gear position. The arbitration field is assigned via a unique ID expressed from $0x000$ to $0x7FF$ for each data. This ID is called CAN ID, and it prioritizes the messages when

¹<https://github.com/shuji-oh/ivnprotect>

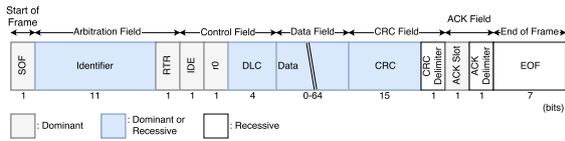


Figure 1: CAN data frame format.

multiple messages are being sent simultaneously from ECUs. Lower CAN IDs have a higher priority. In other words, $0x000$ is the highest priority ID for CAN protocol.

Next, we explain the error handling mechanism of CAN. CAN has a fault error state mechanism for high fault tolerance. The error state mechanism consists of three states (error active state, error passive state, and bus-off state). At first, an ECU starts in the error active state. Then if the ECU detects several errors (i.g. a CRC error), it transits into the error passive state in which the ECU cannot receive the messages on the bus. Finally, suppose the ECU still detects several errors in the error passive state. In that case, the ECU will be at the bus-off state, which expresses the ECU is logically isolated from the bus. Namely, the ECU in bus-off state cannot send the messages to the bus and receive the messages on the bus. To recover from bus-off state, the ECU in bus-off state needs to reset software or reboot.

2.2 Related Works

Various intrusion detection methods (e.g., arrival time (Song et al., 2016), entropy (Wu et al., 2018a), and voltage (Kneib and Huth, 2018)), and authentication mechanisms (Pesé et al., 2021; Kurachi et al., 2014) 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.

2.2.1 Protections Implemented on a Bus

Wu *et al.* proposed an ID-hopping moving target defense (Humayed and Luo, 2017; Woo et al., 2019; Wu et al., 2018b) 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 car-

ried 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 (Groza et al., 2021) 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.

2.2.2 IDS-side Protection

The frame corruption approach (Takada et al., 2019) 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 updates. 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.

2.2.3 ECU-side Protections

Unlike the protections of IDS-side and on-bus, the ECU-side protection has an advantage in unharmed for messages on the bus when its protection is triggered. Elend *et al.* developed a secure CAN transceiver (Elend and Adamson, 2017) 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 (Mishra et al., 2020), 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 un-harming 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 (Machiry et al., 2017; Guilbon, 2018). 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 1 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 6.1. To solve these problems, we propose an isolable, traceable, and deployable kernel-level protection in the next section.

3 CAN-Bus KERNEL-LEVEL PROTECTION: IVNPROTECT

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.

3.1.1 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).

3.1.2 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.

3.2 Problem Statement

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

3.2.1 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 case the 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.

3.2.2 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.

Table 1: 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	ID-Hopping	Frame Corruption	Elend <i>et al.</i>	TEECheck	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

3.2.3 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

3.2.4 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 μ s which is the transmission overhead by TEECheck (Mishra et al., 2020).

3.3 Overview of IVNPROTECT

IVNPROTECT consists of four stages, allowlist and similarity analysis-based detection modules, security error states, sending function and discarding function, as shown in Fig. 2. In the allowlist and similarity analysis-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 the AGL based in-vehicle infotainment system (Sivakumar et al., 2020), be-

cause such ECUs with entry points to external networks have been compromised in the actual hacking (Nie et al., 2017; Miller and Valasek, 2015). In the following sections, we describe each stage in order.

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.

3.4.1 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.

3.4.2 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 (Ohira et al., 2020) 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

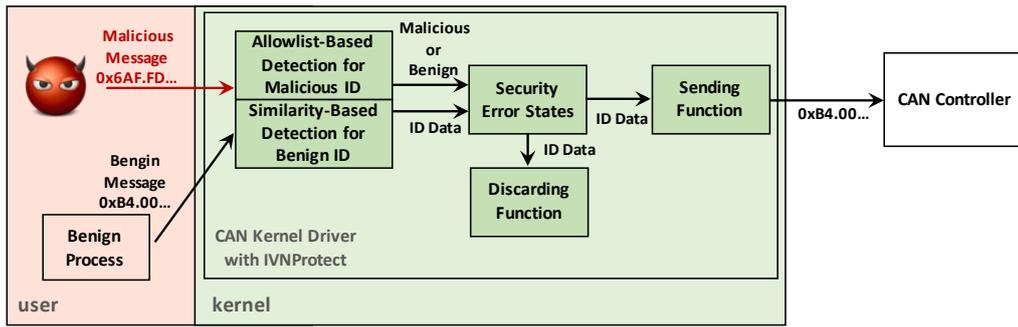


Figure 2: Diagram of the proposed IVNProtect.

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 Simulated Annealing (SA) before the detection-phase. As one example, we show the pre-defined parameters calculated from our evaluation dataset using the SA-based algorithm (Ohira et al., 2020) as follows.

$$W = 7,$$

$$\sigma_s = 0.3414,$$

$$CIDs = \{0x3b8, 0x3b9, 0x3ba, 0x3bb, 0x3bc, 0x3bd, 0x463\}$$

where σ_s is the deviation of benign similarity.

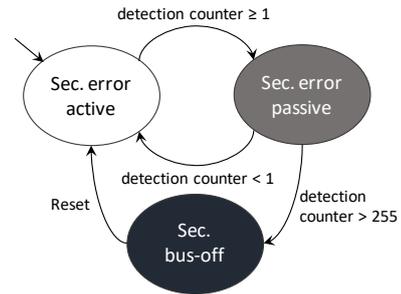
This method also requires defining these parameters in the CAN-bus kernel driver as with the allowlist-based detection module. These parameters are calculated and pre-defined before factory shipment for each ECUs. Thus, as with the allowlist-based detection module, these parameters related to similarity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

3.5 Security Error State Mechanism

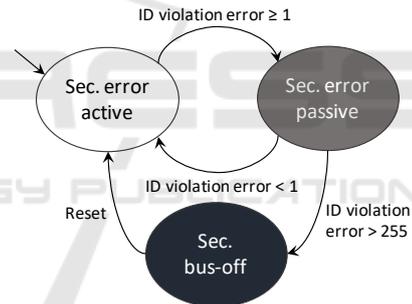
To meet **P2** and **P3**, we employ a security error state mechanism (Fig. 3) for the isolation of a compromised ECU, inspired by the fault error state mechanism specified in CAN. This section explains the functions and the statements of transition in each state.

3.5.1 Security Error Active State

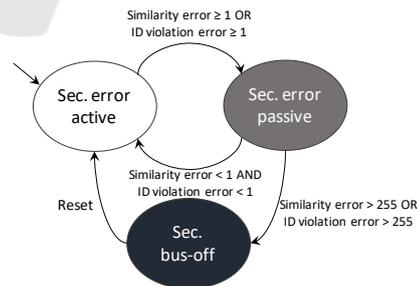
Security error active state implies that there is no security incident so that IVNPROTECT does not execute any functions in this state. As shown in Fig. 3 (a), if a detection module (e.g., allowlist-based detection) finds some malicious activity in this state, the ECU immediately transfers to the following security error passive state.



(a) General security error states.



(b) Security error states for malicious ID DoS.



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

Figure 3: Security (Sec.) error states on IVNPROTECT.

3.5.2 Security Error Passive State

Security error passive state expresses that DoS attack activity has begun to be observed. For example, as shown in Fig. 3 (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. 3 (c). As the same as Fig. 3 (b), this state is entered when an ID outside the allowlist is sent (i.e., ID violation error). In addition, in the case of 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.

3.5.3 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. 6.2.

4 IMPLEMENTATION

In this section, we describe the implementation of IVNPROTECT. We implemented the procedures related to IVNPROTECT to the CAN-bus kernel driver² 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 1 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 mali-

cious transmission and then `netif_wake_queue()` to immediately resume next sending for benign messages. By the way, note that the other CAN kernel driver also has the message transmission function like `mcp251x_tx_work_handler()`. Therefore, IVNPROTECT can be implemented on various CAN peripherals.

Algorithm 1: CAN message transmission algorithm in a kernel driver with IVNPROTECT.

Input: `can_priv, WIDs, CIDs, W, allowlist`

Output: None

```

1: function calc_similarity(set1, set2, set_size)
2:   return  $\frac{|set1 \cap set2|}{set\_size}$ 
3: end function
4:
5: can_frame  $\leftarrow$  can_priv->packet_data;
6: similarity  $\leftarrow$  0;
7: is_ID_violation_error  $\leftarrow$  False;
8: is_similarity_error  $\leftarrow$  False;
9:
10: // calculate similarity
11: can_frame->can_id is added to WIDs
12: if ++window_idx  $\geq$  W then
13:   similarity  $\leftarrow$  calc_similarity(WIDs, CIDs, W);
14:   window_idx  $\leftarrow$  0;
15:   memset(WIDs, 0, sizeof(WIDs)); // initialize WIDs
16: end if
17:
18: // validate CAN ID and similarity
19: mutex_lock(&can_priv)
20: if can_frame->can_id is not in allowlist then
21:   can_priv->can_device_sec_stats->ID_violation_error++;
22:   is_ID_violation_error  $\leftarrow$  True;
23:   update can_priv->sec_state
24: else if similarity exceeds a benign range defined by  $\sigma_s$  then
25:   can_priv->can_device_sec_stats->similarity_error++;
26:   is_similarity_error  $\leftarrow$  True;
27:   update can_priv->sec_state
28: end if
29:
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)

```

²<https://github.com/raspberrypi/linux/blob/4a1f59200d36993f6b32742c03c154ae275bd89c/drivers/net/can/spi/mcp251x.c>

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). And, 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 malicious 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.

5 EVALUATION

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.

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. 4. As shown in Fig. 4 (a), in the case without IVNPROTECT, the busload reached over 45 % during DoS attacks.

In contrast, as shown in Fig. 4 (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, IVNPROTECT inserts a delay to mitigate flooding. As shown in Fig. 4 (b), we confirm that the busload is reduced to about 12 % by inserting the delay.

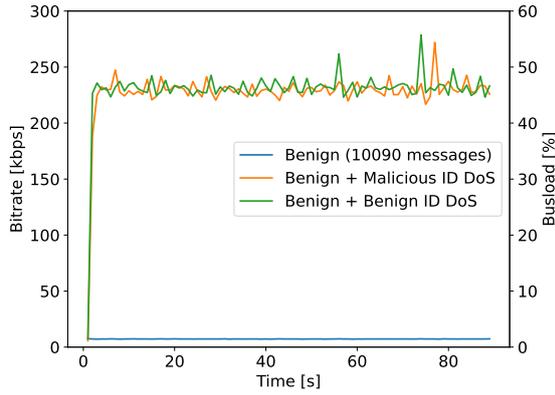
Considering the busload of only benign messages is about 1 %, IVNPROTECT allows the benign ID DoS to increase the busload by 11 %. In general, the busloads of the real-world CAN-buses are limited to about 20 % in order to avoid unacceptable delays for low-priority messages. Thus, when an attacker executes 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 %.

5.3 Isolation Time

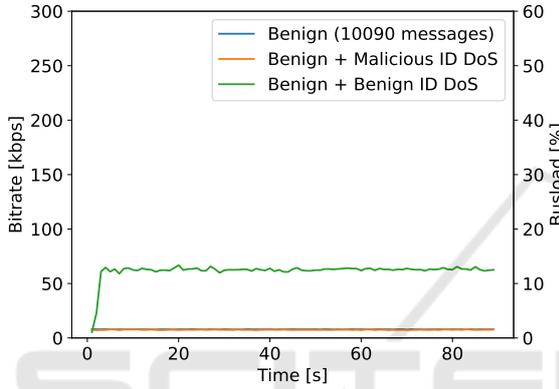
Next, we evaluate the isolation time, which expresses from starts from the DoS attacks to isolation by IVNPROTECT. Like the previous experiment, we had the Raspberry Pi 3 Model B send 10090 benign messages and DoS attacks simultaneously. In Fig. 5, we show the isolation times with different thresholds of the detection counter for transferring the security bus-off state. As shown in Fig. 5 (a), we confirm that the maximum isolation time is 170 ms if the threshold is 255. And, as shown in Fig. 5 (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.

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



(a) Without IVNPROTECT.



(b) With IVNPROTECT.

Figure 4: Comparison of bitrate and busload between with and without IVNPROTECT.

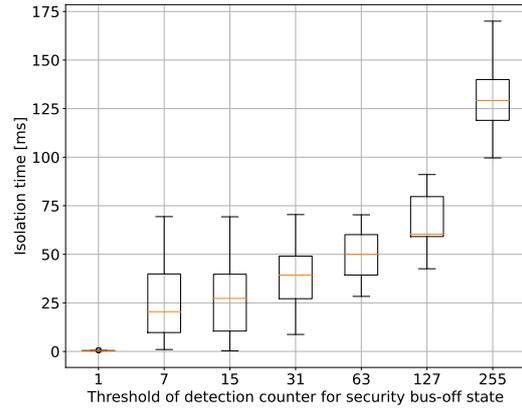
benign messages during DoS attacks. Fig. 6 shows the benign transmission loss rate of IVNPROTECT during DoS attacks.

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.

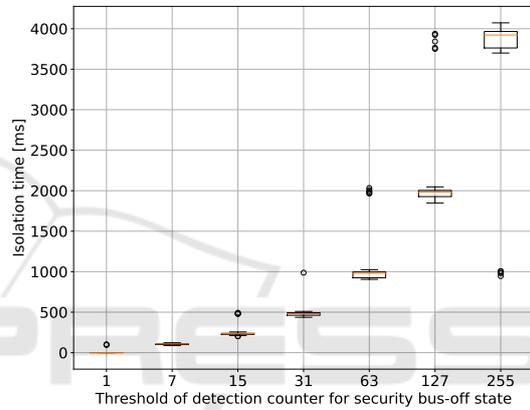
5.5 Benign Transmission Delay Under DoS Attacks

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. Table 2 shows the arrival time of benign messages during various DoS attacks.

The benign traffic in Table 2 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 μ s) represents the arrival time of benign messages during ma-



(a) Malicious ID DoS.



(b) Benign ID DoS.

Figure 5: Isolation time under different thresholds of detection counter for security bus-off state.

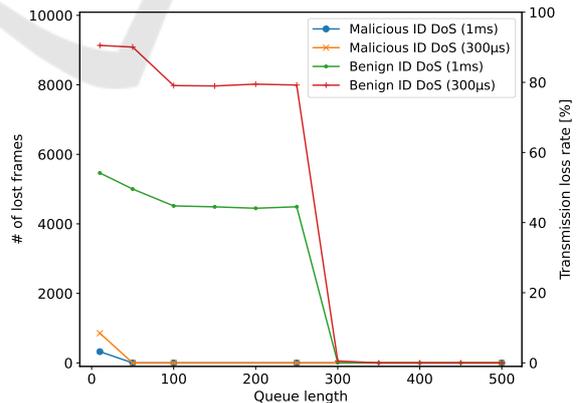


Figure 6: Benign transmission (10090 messages) loss rate under different tx queue size during DoS attacks.

licious DoS attacks (300 μ s), which is the maximum transmission rate.

Comparing benign with benign + malicious ID DoS in Table 2, we confirm that IVNPROTECT does

Table 2: 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 (300 μ s)	99.977	5.055	110.976
benign+benign ID DoS (1ms)	100.040	75.665	199.989
benign+benign ID DoS (300 μ s)	99.989	115.637	333.057

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 6.2.

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 20.090193 ms, while in the case with IVNPROTECT, it was 20.099242 ms. The overhead with IVNPROTECT is 9.049 μ s, which is less than the TEECheck's (494 μ s), so we confirm that IVNPROTECT does not affect on benign messages.

6 DISCUSSION

6.1 Comparison Among Previous Protections

In this section, we compare IVNPROTECT with previous protections. As described in Sec. 2.2, IVNPROTECT has advantages in terms of unharmed 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 3 shows a comparison among the ECU-side protections and IVNPROTECT. As described in Sec. 2.2, the secure CAN transceiver

Table 3: 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 <i>et al.</i>	TEECheck	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%

proposed by Elend *et al.* (Elend and Adamson, 2017) 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. And, it causes the transmission loss rate of benign messages during DoS attacks to increase. Therefore, the secure CAN transceiver cannot report warning messages to other ECUs on the bus while being forced to send DoS attacks by an attacker.

TEECheck (Mishra *et al.*, 2020) 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 (Ohira *et al.*, 2020). 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 has advantages in isolability, deployability, and adaptation to aperiodic tasks.

6.2 Warning Response Using IVNPROTECT

For **P3** described in Sec. 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. 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. 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.

6.3 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 . So, 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. And, 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.

7 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. In our future work, we will try to implement a host-based intrusion detection module to IVNPROTECT, which IVNPROTECT can prevent the attacker's activities before starting some attacks. 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.

REFERENCES

- Elend, B. and Adamson, T. (2017). Cyber Security Enhancing CAN Transceivers. In *Proceedings of the 16th International CAN Conference*.
- Groza, B., Popa, L., Murvay, P.-S., Elovici, Y., and Shabtai, A. (2021). {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.

- Guilbon, J. (2018). Attacking the ARM's TrustZone. <https://blog.quarkslab.com/attacking-the-arms-trustzone.html>. (Accessed on 02/17/2022).
- Humayed, A. and Luo, B. (2017). 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.
- Kneib, M. and Huth, C. (2018). Scission: Signal Characteristic-Based Sender Identification and Intrusion Detection in Automotive Networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 787–800. ACM.
- Kneib, M., Schell, O., and Huth, C. (2020). 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.
- Kurachi, R., Matsubara, Y., Takada, H., Adachi, N., Miyashita, Y., and Horihata, S. (2014). CaCAN-Centralized Authentication System in CAN (Controller Area Network). In *14th Int. Conf. on Embedded Security in Cars (ESCAR 2014)*.
- Linux-Foundation (2019). Automotive Grade Linux. <https://www.automotivelinux.org/>. (Accessed on 07/08/2019).
- Machiry, A., Gustafson, E., Spensky, C., Salls, C., Stephens, N., Wang, R., Bianchi, A., Choe, Y. R., Kruegel, C., and Vigna, G. (2017). BOOMERANG: Exploiting the Semantic Gap in Trusted Execution Environments. In *NDSS*.
- Marchetti, M. and Stabili, D. (2017). Anomaly Detection of CAN Bus Messages through Analysis of ID Sequences. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1577–1583. IEEE.
- Miller, C. and Valasek, C. (2015). Remote Exploitation of An Unaltered Passenger Vehicle. *Black Hat USA*, 2015:1–91.
- Mishra, T., Chantem, T., and Gerdes, R. (2020). 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.
- Müter, M., Groll, A., and Freiling, F. C. (2010). A Structured Approach to Anomaly Detection for In-Vehicle Networks. In *2010 Sixth International Conference on Information Assurance and Security*, pages 92–98.
- Nie, S., Liu, L., and Du, Y. (2017). Free-Fall: Hacking Tesla from Wireless to CAN Bus. *Black Hat USA*, 2017:1–16.
- Ohira, S., Araya, K. D., Arai, I., and Fujikawa, K. (2021). 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.
- Ohira, S., Araya, K. D., Arai, I., Inoue, H., and Fujikawa, K. (2020). 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.
- Pesé, M. D., Schauer, J. W., Li, J., and Shin, K. G. (2021). 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. Association for Computing Machinery.
- Robert-Bosch-GmbH (1991). CAN Specification Version 2.0. <http://esd.cs.ucr.edu/webres/can20.pdf>. (Accessed on 07/08/2019).
- Sivakumar, P., Devi, R. S., Lakshmi, A. N., VinothKumar, B., and Vinod, B. (2020). Automotive grade linux software architecture for automotive infotainment system. In *2020 International Conference on Inventive Computation Technologies (ICICT)*, pages 391–395. IEEE.
- Song, H. M., Kim, H. R., and Kim, H. K. (2016). 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.
- Takada, M., Osada, Y., and Morii, M. (2019). Counter Attack against the Bus-off Attack on CAN. In *2019 14th Asia Joint Conference on Information Security (Asia-JCIS)*, pages 96–102. IEEE.
- Woo, S., Moon, D., Youn, T.-Y., Lee, Y., and Kim, Y. (2019). CAN ID Shuffling Technique (CIST): Moving Target Defense Strategy for Protecting In-Vehicle CAN. *IEEE Access*, 7:15521–15536.
- Wu, W., Huang, Y., Kurachi, R., Zeng, G., Xie, G., Li, R., and Li, K. (2018a). Sliding Window Optimized Information Entropy Analysis Method for Intrusion Detection on In-Vehicle Networks. *IEEE Access*, 6:45233–45245.
- Wu, W., Kurachi, R., Zeng, G., Matsubara, Y., Takada, H., Li, R., and Li, K. (2018b). IDH-CAN: A Hardware-Based ID Hopping CAN Mechanism With Enhanced Security for Automotive Real-Time Applications. *IEEE Access*, 6:54607–54623.