In-Vehicle Network Intrusion Detection System Using CAN Frame-Aware Features

Yeonseon Jeong, Hyunghoon Kim, Seyoung Lee, Wonsuk Choi, Dong Hoon Lee, *Member, IEEE*, and Hyo Jin Jo

***Abstract*— With the advancement of connected and automated vehicles (CAVs), drivers now have access to convenient features such as lane-keeping, cruise control, and more. The electronic control units (ECUs) equipped within vehicles communicate with each other through the controller area network (CAN). However, since the CAN does not possess any security mechanisms, it becomes a target for adversaries to attack. In light of this, a significant amount of research regarding intrusion detection systems (IDSs) has focused on detecting such maliciously injected CAN packets. Nevertheless, most existing machine learning-based IDSs neither calculate the exact time intervals of the CAN packets nor utilize the counter information. Precise timing intervals are a crucial feature for detecting spoofing, fuzzing, and replay attacks, and counter information is also a significant feature that can detect fuzzing and replay attacks. Therefore, in this paper, we propose a methodology for extracting two detection features that are aware of CAN frame characteristics: the interframe space (IFS) between two consecutive CAN packets, and the counter information of a CAN data payload (i.e., data field). Using these features, we introduce decision tree-based IDS. We evaluate the proposed features with popular decision tree-based models such as random forest and extreme gradient boosting (XGBoost). The results show that our proposed IDS can detect maliciously injected CAN packets with an F1 score of 99.54% in binary classification and 97.99% in multi-class classification, which are higher scores than what existing machine/deep learning-based IDSs achieve. Additionally, we measure the detection time of our proposed IDS in both online and offline testing environments.**

***Index Terms*— Controller area network, in-vehicle network, intrusion detection system, machine learning.**

I. Introduction

CONNECTED and automated vehicles (CAVs) are equipped to provide drivers and passengers with new convenient driving functions. In order to provide advanced driver assistant systems (ADAS) including the lane keeping assist system (LKAS), anti-lock braking system (ABS), and smart cruise control (SCC), vehicles need to be controlled electronically, not manually. For this reason, a number of electronic control units (ECUs) have been installed in modern vehicles [1]. A controller area network (CAN) is an in-vehicle network defined in the 1993 ISO 11898 standard [2] that provides efficient communication among ECUs. CAN is known to be robust against noisy environments and adopts broadcast communications based on a bus topology. However, CAN does not incorporate any security mechanisms, including confidentiality, authentication, and access control. Therefore, an attacker can access the CAN bus and then inject CAN packets that control safety-critical functions such as the emergency brake control, acceleration control, steering control, etc. In addition, with the development of CAVs with additional driving interfaces, the possible surfaces for CAN attacks have expanded.

Many recent studies have demonstrated an exponential growth of cyber attacks on CAVs [3]. In 2008, Hoppe et al. initially demonstrated the weaknesses of the CAN bus [4]. While that study [4] focused on testing the automotive hardware, Koscher et al. took this a step further to conduct an attack on a real vehicle, including controlling the brake, light, and dashboard [5]. Furthermore, Miller and Valasek hacked a Jeep Cherokee and then remotely controlled the telematics display as well as braking and steering [6]. Most recently, the Tencent Keen security lab discovered multiple security vulnerabilities on Tesla's models, including but not limited to the Autopilot system [7], [8]. Since these attacks threaten the safety and life of drivers, automotive security for CAVs is an urgent topic that needs to be addressed.

In light of this, many studies on CAN intrusion detection systems (IDSs) have been proposed to detect a CAN packet injection attack. Most existing machine/deep learning-based IDSs make use of packet transmission frequency-related features [9], [10], [11] (e.g., the sequence of CAN identifiers, frequencies of CAN packets, etc.) or data-related features [12], [13], [14], [15], [16] (e.g., abnormal physical values of RPM, vehicle speed, etc.). However, no existing work looks at accurate time intervals between two consecutive CAN packets and the counter information included in the CAN data payload to detect the injected CAN packets more accurately. In this paper, we propose new two features for CAN IDSs: the interframe space (IFS) between two consecutive CAN packets and the counter information of a CAN packet. By using two machine learning models, random forest and extreme

Manuscript received 27 October 2022; revised 20 June 2023 and 10 August 2023; accepted 4 October 2023. Date of publication 18 October 2023; date of current version 13 May 2024. This work was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) Grant funded by the Korean Government Ministry of Science and ICT (MSIT) (Vulnerability analysis for secure smart home Internet of Things (IoT) service) under Grant 00218853 and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) under Grant 2021R1C1C1011960. The Associate Editor for this article was X. Li. *(Corresponding authors: Dong Hoon Lee; Hyo Jin Jo.)*

Yeonseon Jeong, Seyoung Lee, Wonsuk Choi, and Dong Hoon Lee are with the School of Cybersecurity, Korea University, Seoul 02841, Republic of Korea (e-mail: ys_jeong@korea.ac.kr; seyoung0131@korea.ac.kr; beb0396@korea.ac.kr; donghlee@korea.ac.kr).

Hyunghoon Kim and Hyo Jin Jo are with the School of Software, Soongsil University, Seoul 06978, Republic of Korea (e-mail: axolotl0210@gmail.com; hyojin.jo@ssu.ac.kr).

Digital Object Identifier 10.1109/TITS.2023.3323622

Fig. 1. The structure of CAN data frame.

gradient boosting (XGBoost), we evaluate how important these two features are to detect cyber attacks on the CAN. The contributions of this paper are described as follows.

- The proposed system introduces new two features: the IFS between two consecutive CAN packets and the counter information of a CAN packet. These are derived from the CAN packet's arrival timestamps, CAN packet's bit-length and CAN data payloads. We evaluated the effectiveness of these features by using two machine learning models, random forest and XGBoost.
- The proposed IDS was tested under five attack scenarios—flooding, spoofing, replay, fuzzing, and masquerade—using an open dataset and a self-created dataset recently collected on a real vehicle. The experimental results show that the proposed IDS leveraging these two features detects cyber attacks on the CAN with an F1 score of 99.54% in binary classification and 97.99% in multi-class classification, which is the highest score compared to the existing machine/deep learning-based IDSs.
- In addition, it has been confirmed that our proposed method detects attacks faster than the selected IDSs used for comparison. The processing time, which measures both the feature engineering and attack detection speed, demonstrates that our proposed IDS outperforms the others, with a maximum time difference of 23.576ms in offline testing, and 290.195ms in online testing.

This paper is organized as follows: Section II describes the CAN, machine learning models, and potential CAN attack scenarios. Section III introduces the related literature on identification methods for CAN signals and machine/deep learning-based intrusion detection systems. Section IV describes the assumptions and attack scenario for the system model. In Section V, we propose new features and machine learning models to detect intrusions on automotive CAN. To evaluate these, we test our intrusion detection models using an open dataset and compare the performance with existing IDSs in Section VI. Finally, Section VII presents our conclusion.

II. Background

A. Controller Area Network

The CAN adopts a bus topology (the CAN bus) to support communications among ECUs. The CAN specification version 2.0 defined by Bosch in 1986 [17] has been used in modern vehicles. The CAN supports two different formats, a standard format and an extended format. The standard format provides an 11-bit identifier, while the extended format provides a 29-bit identifier which is composed of both an 11-bit identifier and an 18-bit extended identifier. In general, modern commercialized vehicles adopt the standard format, and the 29-bit extended format is mainly used for specialized vehicles such as agricultural machines, semi trucks, etc.

Fig. 1 shows the standard structure of a CAN data frame. Each field is described as follows.

- SOF and Identifier: The start of frame (SOF) field denotes the start of a message, and the arbitration identifier (ID) field is used for arbitration, which is the media access control procedure for dealing with messages transmitted at the same time. In this arbitration, the lower CAN ID has the higher priority.
- RTR: The remote transmission request (RTR) field defines four types of CAN frames: data frame, remote frame, error frame, and overload frame.
- IDE: The identifier extension (IDE) field is used to distinguish between the standard and extended formats.
- R and DLC: The control field consists of the reserved (R) bits for the extended frame (i.e., the 29-bit identifier) and the data length code (DLC) indicating the number of bytes in the data field.
- Data: The data field contains the actual vehicle's sensor data or control data to be transferred, and can consist of 0 to 8 bytes.
- CRC: The cyclic redundancy check (CRC) field is used to confirm the integrity of the SOF, ID, control, and data fields.
- ACK and EOF: The acknowledgment (ACK) field is composed of two bits—the ACK slot and the ACK delimiter—and the end of frame (EOF) field denotes the end of the message.
- IFS: The interframe space (IFS) field refers to the time required for the controller to move the correctly received frame to its suitable position in a message buffer.

B. Machine Learning Models

This subsection introduces the two decision tree-based machine learning models.

1) Random Forest: Random forest is an ensemble machine learning algorithm used for classification and regression problems. It consists of numerous decision trees, which are like a tree structure flowchart, and learns simple decision rules inferred from the data features. Because decision trees have an overfitting tendency, random forest's ensemble model is proposed to use the average of a large number of decision trees. This leads to higher accuracy but is comparatively slower than a decision tree alone.

2) XGBoost: XGBoost [18], as another ensemble model for decision trees, is designed to be efficient, flexible, and portable. While random forest is a bagging-based ensemble model, XGBoost uses a boosting method, especially gradient boosting [19]. The high speed dominates in performance compared to other gradient boosting models.

III. Related Works

A. Identification Methods for CAN Signals

The CAN data field contains several signals (e.g., physical sensor values, counter values, CRC values, etc.). Such information is specified in the CAN database files (DBC files) designed by vehicle manufacturers. However, since the DBC files are confidentially managed by the manufacturers, it is difficult to identify the signal information in the data field. For this reason, existing works propose methods to extract the

signal boundaries and label the types of the signals using a bit flip.

Marchetti and Stabili proposed an automatic reverse engineering tool called READ to analyze signals encoded in CAN bus packets [20]. READ is the first method that uses the total bit flip rate of the data field to identify and label different types of signals. They created trace files of CAN packets sorted by each CAN ID and calculated the changing rate of each bit. Signal boundaries are determined by the bit flip rate distribution of the data field, and signals are categorized into physical sensor values, counter values, CRC values, etc. In LibreCAN [21], after classifying the signal boundaries using the bit flip rate, the meanings of each signal that is related to powertrain CAN packets are identified through the smartphone's sensors and OBD-II protocol.

However, the identification of signal boundaries based on the total bit flip rate has the limitation that identification accuracy varies depending on how many CAN traffics are monitored in various driving tests. To improve the accuracy of bit flip rate-based works [20], [21], Choi et al. analyzed the time series values of bit flip rate and were able to identify signal boundaries more clearly [22]. Recently [23], endianness and signedness have also been evaluated for decoding the CAN signals.

B. Machine Learning-Based CAN IDS

Several methods have recently been proposed to detect malicious CAN packets using machine learning techniques. Minawi et al. introduced a machine learning-based intrusion detection system that uses only the CAN ID and data fields as detection features [24]. Their method employs four machine learning algorithms: random tree, random forest, stochastic gradient descent, and naive bayes. On the other hand, several studies [25], [26], [27], [28] have utilized deep learning models for the CAN network. In [25], a generative adversarial nets (GAN) model was used to detect intrusions on in-vehicle networks. Song et al. proposed an IDS based on a deep convolutional neural network (DCNN), which was built on the customized inception-ResNet model, to achieve high intrusion detection accuracy [26]. Hossian et al. utilized a long short-term memory (LSTM) model and demonstrated the effectiveness of specific hyper-parameter values such as batch size, learning rate, and some of the units are effective [27]. Amato et al. introduced an IDS based on multi-layer perceptrons (MLP) [28].

IV. System Model

A. Assumptions

1) System Assumption: The IDS is assumed to be appropriately positioned in the CAN bus—for instance, at a gateway ECU, which would enable it to monitor CAN traffic.

2) Adversary Assumption: The objective of the adversary is to compromise driving safety by injecting malicious CAN packets. To achieve this, the adversary must be aware of the vulnerable access points (e.g., OBD-II port, USB, Bluetooth, WiFi, etc.) on a target vehicle [4], [5], [6], [7], [8]. Therefore, in this paper, we assume that the attacker can access the CAN network via these vulnerable access points and inject CAN packets, thereby posing a threat to safety-critical driving functions.

B. Attack Scenario

In this subsection, the five types of CAN attack scenarios (i.e., a flooding attack, a spoofing attack, a replay attack, a fuzzing attack, and a masquerade attack) are described as follows.

- Flooding attack: An adversary aims to perform a denial of service (DoS) on the transmission of benign CAN packets by sending a large number of CAN packets with a high priority (i.e., CAN ID of 0×00).
- Spoofing attack: To cause the malfunction of certain functions of a vehicle (e.g., gear, RPM, etc.), an adversary injects control packets by using prior knowledge of the target vehicle.
- Replay attack: An adversary records normal CAN bus traffic and replays it into the CAN bus.
- Fuzzing attack: An adversary randomly generates CAN packets. This attack may cause unexpected misbehavior of an attack target vehicle.
- Masquerade attack: In a masquerade attack, the transmission of a normal ECU is suspended, and then a compromised ECU impersonates the CAN IDs and the corresponding transmission cycles of the suspended ECU.

V. The Proposed System

A. Design Challenges and Approach

1) How to Calculate the Accurate Time Interval Between CAN Packets?: In order to calculate the accurate time interval between consecutive CAN packets, IFS must be obtained. However, the interval between the time stamps of the $i-1$-th packet and the i-th packet obtained through CAN packet monitoring includes the transmission time of the i-th packet. Therefore, in order to calculate the IFS between two consecutive packets, the packet length of the i-th packet must be calculated. Once the packet length of a CAN packet is calculated, the transmission time of the corresponding packet can be easily estimated by utilizing CAN bus speed (e.g., 500 Kbits).

However, in general, partial information of CAN packets such as ID, DLC, and Data fields is obtained through a commercialized CAN packet monitoring tool. Therefore, we propose a method of calculating the total length of a CAN packet ① by generating the undisclosed fields, that are not obtained by the CAN packet monitoring tool, such as the control field and CRC field, and ② by calculating the stuffing bit, which is the rule of CAN.

2) How to Find the Sequence of CAN Packets?: In general, CAN packets transmitted by safety-critical ECUs may contain counter information in the data field. However, since the CAN DBC file, which specifies how CAN packets convey sensor and control information, is managed confidentially by the car manufacturers, it is necessary to analyze the information contained in the CAN packet's data field. Therefore, we use the bit flip rate to identify counter information included in the CAN packets.

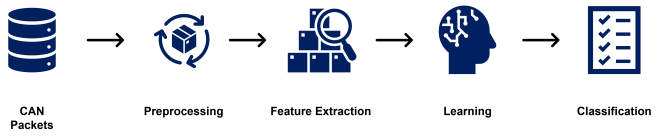


Fig. 2. System overview of the proposed IDS.

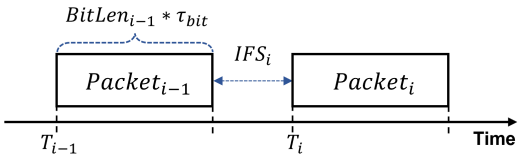


Fig. 3. Interframe space (IFS) between two consecutive CAN packets.

B. Overview

Fig. 2 shows an overview of our proposed IDS, which consists of four phases: a preprocessing phase, a feature extraction phase, a learning phase and a classification phase.

In the preprocessing phase, new two data values, counter and IFS values, used for the feature extraction are derived after identifying the counter field of CAN packets and calculating the IFS between consecutive CAN packets. In the feature extraction phase, a total of 13 features are extracted from CAN packets and the preprocessed data values. In the learning phase, two machine learning models (i.e., random forest and XGBoost) are used to train features of normal CAN packets and attack CAN packets. Finally, the classification phase is used to detect maliciously injected CAN packets by an attacker. Each phase is explained as follows.

C. The Proposed IDS

1) Preprocessing: Existing machine learning-based IDSs have mainly focused on frequency-related features (e.g., CAN ID sequences, frequencies of CAN packets, etc.) or data-related features (e.g., abnormal physical values of RPM, vehicle speed, etc.). However, these methods lack an accurate measure of the time interval between two consecutive CAN packets and do not consider the counter information included in the CAN data field.

To utilize these two characteristics as intrusion detection features, the IFS value between two consecutive CAN packets and the counter value in the counter field of a CAN packet is preprocessed as follows.

Interframe Space Identification. The IFS is the interval between two consecutive data frames transmitted on the CAN bus (i.e., between the last EOF's bit of the $i-1$-th packet and the SOF bit of the i-th packet). According to the CAN protocol, to transmit successive data frames, there needs at least the 3-bit time of the IFS, which is 6μs on a CAN bus with 500 kbps.

In addition to the interval between two consecutive data frames, the IFS is used to check whether two consecutive CAN packets are transmitted by one ECU. For example, if two CAN packets with CAN ID A and CAN ID B are transmitted from two different ECUs, ECU A and ECU B, and have the same transmission period, the IFS of the corresponding packets will be a variable value rather than a constant value due to the other packet's transmission. On the other hand, there will be a pair of CAN packets with a constant IFS if they are successively transmitted from the same ECU. Therefore, the IFS can be used as an intrusion detection feature to check whether two consecutive CAN packets come from one ECU or not.

The IFS can be calculated with the following equation:

$$IFS_i = T_i - T_{i-1} - (BitLen_{i-1} * \tau_{bit}), \tag{1}$$

where T_i and T_{i-1} means the timestamp of the CAN packet i and $i-1$, and $BitLen_{i-1}$ is the bit-length of the CAN packet $i-1$, and τ_{bit} is the time to be required to transmit one bit on the CAN bus, which is 2μs on a CAN bus with 500 kbps. Fig. 3 intuitively shows the IFS between two consecutive CAN packets.

To calculate the bit length, $BitLen_{i-1}$, the number of data frame bits shown in Fig. 1 are derived using the given packet information (ID, DLC, Data) and the CRC bits.[1] Then, the stuffing bits are calculated to find the accurate bit length; the bit-stuffing rule is the insertion of non-information bits into a CAN packet when there are five consecutive bits of the same logic level. For example, if there are five consecutive bits, 00000, in a CAN packet, the 1 is followed by five consecutive bits.

CAN Signal Identification: In CAN traffic, some CAN packet have the counter signal in their data field, which is increased by 1. To identify whether a CAN ID includes the counter signal in its own data field, we first find signal boundaries and label the signal type using the bit flip rate [20], [21]. At the same time, we also consider the endianness (byte ordering) of the signal [23].

The bit flip rate is the rate of bit change (0 from 1, vice versa) in each bit of the data payload. The bit flip rate of the sensor (physical value) signal decreases from the least significant bit (LSB) to the most significant bit (MSB). On the other hand, the counter signal's bit flip rate is one at the LSB and decreases by half as it moves to the MSB; for example, if the bit length of a signal is 4, and the bit flip rates of the corresponding signal are 1, 0.5, 0.25, 0.125 from the LSB to the MSB, this signal is identified as the counter signal. Based on the above-described bit flip rate property, we can identify the counter signal from CAN packets and utilize the counter value as the detection feature.

The bit flip rate can be calculated with the following equation:

$$BFR_{c,k} = \frac{1}{n_c - 1} \sum_{i=2}^{n_c} k_{i-1} \oplus k_i \tag{2}$$

where c, k indicates the k^{th} bit position in the data field of a specific CAN ID c. n_c is the number of CAN packets of c, and the bit flip is counted using the XOR operation on the bit k^{th} of the i^{th} and $i-1^{\text{th}}$ packet on c. The bit flip rate is subsequently calculated by dividing the total number of bit flips by the total number of packets minus 1, represented as $n_c - 1$.

[1] In an automotive CAN, a 15-bit CRC polynomial, $x^{15}+x^{14}+x^{10}+x^8+x^7+x^4+x^3+x^0$, is used.

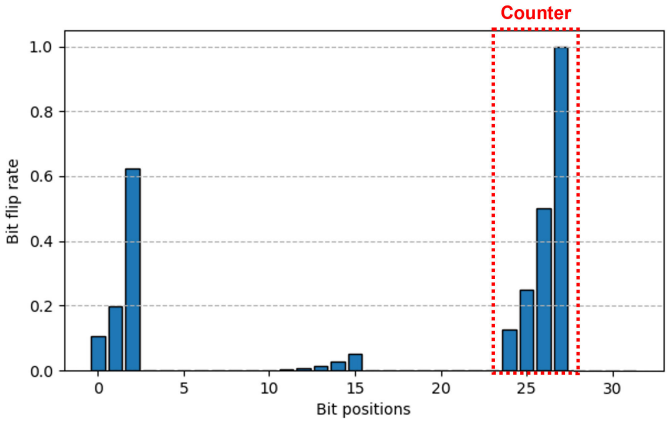


Fig. 4. Example of bit flip rate.

TABLE I
EXAMPLE OF CAN TRAFFIC

Timestamp	CAN ID	DLC	Data	Class	SubClass
1597759837.502286	394	8	00 40 08 00 0F 34 1D FC	Normal	Normal
1597759837.5024571	164	4	00 08 01 B6	Normal	Normal
1597759837.502677	412	8	38 D7 D0 FE 6B F2 1A 48	Attack	Fuzzing
1597759837.502927	387	8	FC B0 35 A4 00 89 04 00	Normal	Normal
1597759837.5039651	381	8	80 A0 3C 00 00 F5 24 05	Normal	Normal

TABLE II
OUR FEATURES FOR INTRUSION DETECTION SYSTEM

Features	Description
ID_i	CAN ID of the i-th CAN packet
DLC_i	Data length code of i-th CAN packet
$Prev\_ID_i$	CAN ID of $i-1$-th packet
$Next\_ID_i$	CAN ID of $i+1$-th packet
$Former\_ID_i$	CAN ID of $i-2$-th packet
$Latter\_ID_i$	CAN ID of $i+2$-th packet
$Prev\_Interval_i$	The time interval between the $i-1$-th and i-th packets
$Next\_Interval_i$	The time interval between the i-th and $i+1$-th packets
$ID\_Prev\_Interval_i$	The difference between the timestamp of i-th packet and the timestamp of the previous packet with the same CAN ID as $\mathbf{ID}_i$
$ID\_Next\_Interval_i$	The difference between the timestamp of i-th packet and the timestamp of the next packet with the same CAN ID as $\mathbf{ID}_i$
$Prev\_Counter_i$	The difference between the counter value of i-th packet and the counter value of the previous packet with the same CAN ID as $\mathbf{ID}_i$
$Next\_Counter_i$	The difference between the counter value of i-th packet and the counter value of the next packet with the same CAN ID as $\mathbf{ID}_i$
IFS_i	Interframe space between the $i-1$-th and i-th packets
$Class_i$	Labeling of the packet type (Normal/Attack) of i-th packet
$SubClass_i$	Labeling of the attack type (Flooding/Spoofing/Replay/Fuzzing) of i-th packet

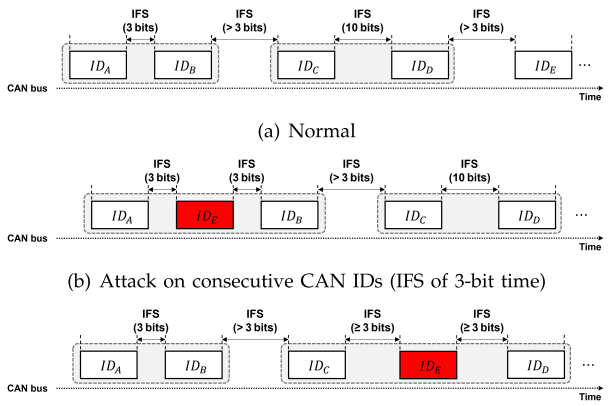
Fig. 4 shows the bit flip rate on each bit position of the data field within the data field for a CAN ID 0×4F1, which has 4 bytes data payload.

2) Feature Extraction: Table I shows the example of CAN traffic from the car hacking dataset [29]. The dataset consists of the *Timestamp, ID, DLC, Data, Class*, and *SubClass*. *Class* shows the packet type (i.e., a normal or attack type) in binary, and *SubClass* specifies the attack scenarios (i.e., a flooding attack, a spoofing attack, a replay attack, and a fuzzing attack) in detail.

For the feature extraction of i-th CAN packet, 13 features are derived from the CAN packets and the preprocessed information as shown in Table II.

Our features can be divided into CAN ID sequence features, a DLC feature, packet interval features, counter features, and an IFS feature.

- CAN ID sequence features: In a situation where there is no attack on the CAN bus, several patterns of the CAN ID sequence are monitored on the CAN bus according to the packet transmission cycles and priority of the CAN packets. However, when an attack occurs, new CAN ID sequence patterns may appear.
  In regards to the CAN ID sequence of the i-th CAN packet, the five features are defined as follows: ID_i, $Prev\_ID_i$, $Next\_ID_i$, $Former\_ID_i$, and $Latter\_ID_i$. ID_i means the identifier of the i-th CAN packet. $Prev\_ID_i$ and $Next\_ID_i$ of the i-th CAN packet are the CAN ID of the $i-1$-th and $i+1$-th CAN packets, respectively. In addition, $i-2$-th and $i+2$-th CAN IDs are set to the $Former\_ID_i$ and $Latter\_ID_i$, respectively.
- DLC feature: Since a fuzzing attack can modify the DLC as well as data payload and CAN ID, the data length code (i.e., DLC_i) is considered to be one detection feature.
- Packet interval features: In general, when a CAN attack occurs, the CAN packet transmission cycle may change significantly.
  For packet interval related features of i-th CAN packet, four time interval features are defined: $Prev\_Interval_i$, $Next\_Interval_i$, $ID\_Prev\_Interval_i$ and $ID\_Next\_Interval_i$. $Prev\_Interval_i$ and $Next\_Interval_i$ are simply calculated through the timestamp differences between the $i-1$-th and $i+1$-th CAN packets. $ID\_Prev\_Interval_i$ and $ID\_Next\_Interval_i$ are the time differences between the timestamp of the i-th CAN packet and the timestamp of the previous/next packet with the same CAN ID as ID_i.
- Counter features: For CAN IDs that have a counter signal, the counter value included in the counter signal can be used to detect an attack packet that violates the sequence of the counter (e.g., $Counter1$, $Counter2$, $Counter3$, $Counter\mathbf{1}$, $Counter4$). Even if an attacker successfully manipulates the counter signal, the sequence of the counter values will be broken (e.g., $Counter1$, $Counter2$, $Counter\mathbf{3}$, $Counter3$, $Counter4$) unless the attacker compromises the ECU and masquerades it.
  For counter features, CAN IDs that have a counter signal are identified by analyzing the bit flip rate of the CAN traffic. Then, the corresponding counter signal's bit position, size, and endianness are preprocessed. After obtaining the value of the counter signal, the counter features, $Prev\_Counter_i$ and $Next\_Counter_i$, are defined as follows. $Prev\_Counter_i$ is the difference between the counter value of the i-th CAN packet and the counter value of the previous packet with the same CAN ID as ID_i.
  $Next\_Counter_i$ is the difference between the counter value of the i-th CAN packet and the counter value of the next packet with the same CAN ID as ID_i.
- IFS: When there are two consecutive IDs with IFSs of constant value, they can be considered to be transmitted



(a) Normal

(b) Attack on consecutive CAN IDs (IFS of 3-bit time)

(c) Attack on consecutive CAN IDs (IFS of constant value more than 3-bit time)

Fig. 5. Examples of IFS in normal and attack situations.

Fig. 6. Experimental Setup.

TABLE III

SUMMARY OF CAR HACKING: ATTACK & DEFENSE CHALLENGE DATASET [29] AND SELF-CREATED DATASET

	Open dataset					
	Normal	**Flooding**	**Spoofing**	**Replay**	**Fuzzing**	**Total**
Training	3,372,743	154,180	7,756	47,593	89,879	3,672,151
Summission	3,358,210	191,679	42,583	62,881	96,693	3,752,046
	Self-created dataset I			**Self-created dataset II**		
	Normal		**Masquerade**	**Normal**		
Training	142,058		2,664	742,280		
Testing	141,962		2,666			

from one ECU. If an injected CAN packet increases or decreases an IFS of constant value, the injected packet can be classified as an attack.

Fig. 5 shows examples of IFS in normal and attack situations. Fig. 5(a) is an example of an IFS that appeared in normal CAN traffic. In this example, the IFS between ID_A and ID_B, which are the consecutive CAN packets transmitted by one ECU, is a constant value, i.e., 3-bit time. ID_C and ID_D are also transmitted by one ECU because the IFS between them is always a constant value, but more than 3-bit time (e.g., 10-bit time). Fig. 5(b) shows an attack example that the attack packet is injected between ID_A and ID_B. If an attack packet including higher priority ID_E than ID_B is injected, the transmission of the packet with ID_B is delayed due to the arbitration, and the IFS between ID_A and ID_B, which should always be 3-bit time, is observed as 6-bit time. In addition, Fig. 5(c) shows an example where the constant IFS between ID_C and ID_D is violated. Therefore, IFS can be used as an intrusion detection feature.

3) Learning and Classification: In these phases, the two decision tree-based machine learning models described in Section II-B in detail, random forest and XGBoost, are used to train 13 features derived from the training dataset. For the evaluation, each machine learning model is built in two versions: a binary classification model and a multi-class classification model. Since the binary classification model is trained by using 13 features and the corresponding label $Class_i$, it can classify a CAN packet into a normal packet or an attack packet. On the other hand, the multi-class classification model can classify a CAN packet into five classes, i.e., a normal packet, a flooding packet, a spoofing packet, a replay packet, and a fuzzing packet, by using 13 features and the corresponding labels $Class_i$ and $SubClass_i$.

VI. EXPERIMENT AND EVALUATION

A. Environmental Setup

For evaluation of the proposed IDS, we prepared two testbeds: one for offline testing that utilizes a powerful computational resource and another for online testing that employs a lightweight computational resource. Furthermore, for comparison between the proposed IDS and existing IDSs, we implemented the existing machine learning-based and deep learning-based IDSs using Python 3.8 with the Scikit-learn, Tensorflow [30], and Keras libraries.

1) Offline Testing: For offline testing, we used a PC with an Intel Core i7-9700 CPU @ 3.00Hz, 48GB RAM, NVIDIA RTX A5000 graphics card, and Ubuntu 20.04 LTS operating system. Offline testing makes the classification of multi-class attacks possible, so it can be used for post-analysis of cyber attacks on the CAN bus (for instance, digital forensics).

2) Online Testing: In order to evaluate the real-time performance of IDSs, we conducted experiments on the Raspberry Pi 4 with PiCAN, which is a CAN bus interface for Raspberry Pi. To evaluate online testing, we measured processing time, which includes feature extraction and model inference time for a single CAN packet. Fig. 6 shows the experimental setup for online testing. We used a real vehicle, a Hyundai Avante CN7, for these tests. To collect the CAN packets, we used a CAN bus monitoring tool called neoVI FIRE2.

B. Dataset

For evaluation, we make use of three datasets. An open dataset [29] is utilized to measure the detection accuracy of the proposed IDS. In contrast, self-created datasets, generated in a real vehicle environment, are used to evaluate the two newly proposed features and to measure the processing time of our model, respectively.

1) Open Dataset: The "Car Hacking: Attack & Defense Challenge 2020" dataset is provided by the Hacking and Countermeasure Research Lab (HCRL) [29]. This dataset was created from the CAN traffic of a Hyundai Avante CN7 and contains four attack scenarios (i.e., a flooding attack, a spoofing attack, a replay attack, and a fuzzing attack) as described in Section IV-B. The dataset consists of a preliminary and a final round. The preliminary round includes the training and submission set, which can be used for the training and testing, respectively.

2) Self-Created Datasets: The self-created dataset I is a dataset in which a masquerade attack was carried out after suspending a specific ECU in the Hyundai Avante CN7. On the other hand, self-created dataset II consists of normal data collected over five minutes from the Hyundai Avante CN7.

Table III shows the number of CAN packets for both the open dataset [29] and the self-created dataset.

C. Preprocessing and Feature Extraction

In order to test the proposed IDS, we performed preprocessing and feature extraction as defined in Sections V-C.1 and V-C.2 to generate the 13 features of Table II. Then, we performed normalization on the extracted features to convert the feature values into a value between 0 and 1. Feature extraction and normalization are completed as follows. The CAN ID sequence features (ID, $Prev\_ID$, $Next\_ID$, $Former\_ID$, and $Latter\_ID$) and the DLC feature (DLC) are divided by the maximum value in a range of numbers that can be represented in each field, e.g., the CAN ID is divided by $0 \times 7FF$, and the DLC is divided by 8. The packet interval features ($Prev\_Interval$, $Next\_Interval$, $ID\_Prev\_Interval$, and $ID\_Next\_Interval$) are calculated as the time difference between CAN packets, as mentioned above. In the case of the counter features ($Prev\_Counter$ and $Next\_Counter$), we first identified CAN IDs that contain the counter signal in the CAN data field by using the bit flip rate to calculate the counter difference for each CAN ID. When the value of the counter signal changes from the maximum value to the minimum value, we converted the counter difference to 1, e.g., if a counter field is 4-bit and the value of the counter changes from 15 to 0, the counter difference is set to 1. On the other hand, there are CAN IDs that do not have the counter signal; in such cases, the values of the counter features are set to 0. IFS is calculated by using Equation 1. Furthermore, we used $Class$ for the binary classification models and labeled normal as 0 and attack as 1, respectively. For multi-class classification models, we used $SubClass$ as well as $Class$ and labeled normal as 0, flooding as 1, spoofing as 2, replay as 3, and fuzzing as 4, respectively. After a label encoding, we applied a one-hot encoding to the integer labeling values.

D. Evaluation Metrics

We calculated accuracy, precision, recall, and F1 score to measure the performance of the existing IDSs and the proposed model. Accuracy is the rate of correct predictions of normal or attack packets compared to the total traffic. Precision is the rate of the attack traffic compared to the traffic predicted as an attack. Recall is the rate of correct predictions of an attack compared to the attack traffic. The F1 score is the weighted average between precision and recall. The 4 metrics are calculated with the following equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F1score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6}$$

where true-positive (TP) means it is correctly classified as an attack, and true-negative (TN) means it is correctly classified as normal. False-positive (FP) means it is misclassified as an attack, and false-negative (FN) means it is misclassified as normal.

E. Evaluation Results

1) Comparison of Overall Performance: For testing our proposed IDS in offline and online environments, the proposed IDSs have two configuration settings, random forest/XGBoost with full features (i.e., w/ F.F) and without reduced features (i.e., w/o R.F). Random forest/XGBoost with full features means a model that uses all our features described in Table II for offline detection. On the other hand, random forest/XGBoost without reduced features means the model that does not use features related to futures (e.g., next or latter) packets (i.e., $Next\_ID_i$, $Latter\_ID_i$, $Next\_Interval_i$, $ID\_Next\_Interval_i$, $Next\_Counter_i$) for the online testing.

We evaluated the proposed IDS in terms of binary and multi-class classification from the dataset [29], and the results are shown in Table IV. In binary classification, the two machine learning models with full features show high performance for detection with all four metrics over 99%. Likewise, in multi-class classification, our two models show high performance with all metrics over 97%, even though the overall detection rate was around 2% lower than in binary classification. Table IV shows that random forest with our features performed slightly better than the XGBoost algorithm with our features in both classifications overall.

Additionally, the proposed IDS was compared with one machine learning-based IDS [24] and four deep learning-based IDSs [25], [26], [27], [28]. Since the comparison target IDSs do not provide the source codes, we implemented them. The implementation results are verified by checking whether their original accuracy presented by [24], [25], [26], [27], and [28] for their test dataset was reproduced or not[2]. For deep learning-based IDSs [25], [26], [27], [28], we conducted the experiment using the best hyper-parameters that they set. The comparison results are shown in Table IV. Comparing ours with existing machine learning-based IDSs, we observed that machine learning models with our features have higher

[2]Minawi et al. [24] adopt four machine learning models: random tree, random forest, stochastic gradient descent, and naive bayes. However, in this comparison, we look at the results using random forest and XGBoost with features suggested by [24].

TABLE IV
COMPARISON OF DETECTION PERFORMANCE BETWEEN OUR PROPOSED IDS AND OTHER IDSS

	Model		Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)
Binary Classification	Machine Learning	Random forest with [24]	97.28	97.21	76.3	85.5
		XGBoost with [24]	97.27	99	74.78	85.2
		Random forest w/ F.F	**99.86**	**99.37**	**99.3**	**99.33**
		XGBoost w/ F.F	**99.9**	**99.6**	**99.48**	**99.54**
		Random forest w/o R.F	**99.4**	**96.96**	**97.33**	**97.14**
		XGBoost w/o R.F	**99.34**	**97.14**	**96.56**	**96.85**
	Deep Learning	GAN[25]	75.7	77.73	75.6	76.65
		DCNN[26]	74.33	71.56	76.29	73.85
		LSTM[27]	97.12	99.76	72.69	84.1
		MLP[28]	94.94	76.81	74.19	75.48
Multi-class Classification	Machine Learning	Random forest with [24]	97.3	95.88	97.3	96.32
		XGBoost with [24]	97.34	96.1	97.34	96.2
		Random forest w/ F.F	**98.31**	**98.32**	**98.31**	**97.99**
		XGBoost w/ F.F	**98.35**	**97.54**	**98.35**	**97.9**
		Random forest w/o R.F	**97.88**	**97.68**	**97.88**	**97.51**
		XGBoost w/o R.F	**97.94**	**97.1**	**97.94**	**97.49**
	Deep Learning	GAN[25]	71.16	70.33	71.66	61.02
		DCNN[26]	64.09	61.76	64.09	62.09
		LSTM[27]	97.18	95.93	97.18	95.88
		MLP[28]	94.88	94.13	94.88	93.3

accuracy, precision, recall, and F1 score by around 2%, 2%, 23%, and 14% respectively than [24] in binary classification. Likewise, we found ours to perform better than [24] in multi-class classification.

Among existing deep learning-based IDSs [25], [26], [27], [27], [28] shows the best detection performance. The F1 score of [27] is 84.1% in binary classification and 95.88% in multi-class classification. However, we found ours to perform even better; the machine learning models with our features have higher accuracy, recall, and F1 score by around 2%, 23%, and 15% respectively compared to [27] in binary classification, and also have a higher score (by around overall 2%) than [27] in multi-class classification. The experiment shows that our proposed IDSs have the best performance compared to other IDSs. This means that the new two features suggested by ours are useful as detection features of cyber attacks on the CAN bus.

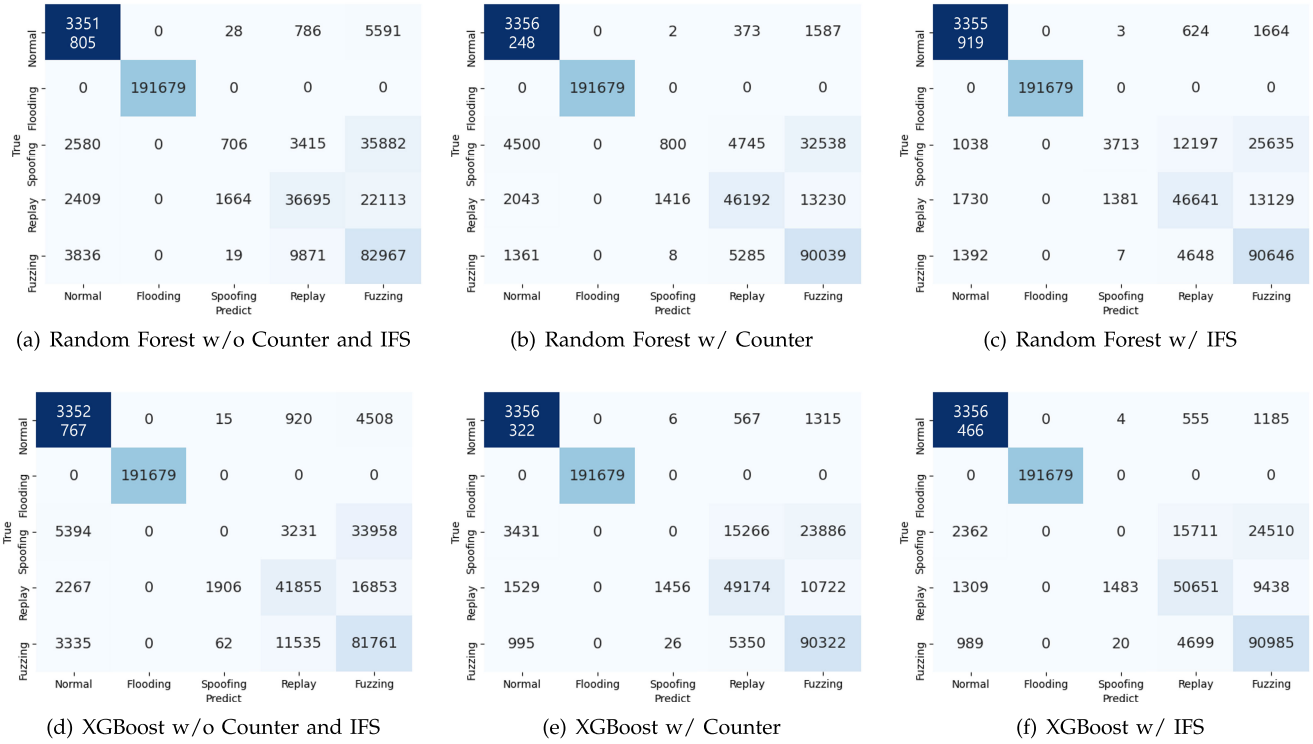
2) Impact of Features: Fig. 8 shows the feature importance for each of our models. We utilized the Gini importance as a measure of feature importance in decision tree-based models. This is a commonly used metric that quantifies the average reduction in impurity brought about by a given feature. Accordingly, a feature that results in a greater reduction in impurity is considered more important. In the two models, the $ID\_Prev\_Interval$ has the most significant effect on attack detection. The CAN ID sequence features and packet interval features also have a significant effect. In particular, while $Prev\_ID$, $Next\_ID$, $Former\_ID$, $Latter\_ID$ and DLC have the least effect, the new features, $Prev\_Counter$, $Next\_Counter$ and IFS, have a greater influence on the proposed IDSs.

In addition, we can see that the feature importance differs slightly between the two models. XGBoost shows that IFS and $Counter$ have more effect than the other features except for $ID\_Next\_Interval$, $Next\_Interval$, and ID. On the other hand, random forest shows that the $Prev\_Interval$ has a higher feature importance score than the $Counter$-related features. The Fig. 7 presents the results of multi-class detection based on these features. Fig. 7(a) and Fig. 7(b) are the results of the model without the counter and IFS, which excludes the two features from with the F.F model using 13 features. We then evaluated the detection results of each model by separately adding the counter and IFS features. In random forest models, the number of packets correctly detected as a spoofing attack increased from 700 to 800 with the use of the counter (Fig. 7(d)), and from 700 to 3,713 with IFS (Fig. 7(c)). Also, these two features show effectiveness against replay and fuzzing attacks. In XGBoost models, Fig. 7(e) and Fig. 7(f) show performance improvements for replay and fuzzing attacks.

Thus, although the feature importance can differ depending on the machine learning models, we confirmed that counter and IFS related features have a significant effect on attack detection in both random forest and XGBoost.

3) Correlations of Features: The correlation between features can be demonstrated as shown in Fig 9. We used the Pearson correlation coefficient that returns a value between -1 and 1. A coefficient of -1 means a strong negative relationship, and 1 indicates a strong positive relationship. The Fig 9 shows that the counter features (i.e., $Prev\_Counter$ and $Next\_Counter$) have independent values without any strong correlations to other features. On the other hand, the IFS shows a strong positive correlation with $Prev\_Interval$ because these features exhibit similar patterns of increases and decreases. However, IFS indicates more accurate time interval information between packets, so it is more helpful in detecting attacks. We demonstrated this experimentally by using self-created dataset I. Table VI shows the performance of detecting masquerade attacks in the self-created dataset I. The model using two features, ID and IFS, shows a 2% and 5% higher recall in the XGBoost and Random Forest models, respectively, compared to the model using ID and $Prev\_Interval$. Therefore, the model using IFS has a lower probability of failing to detect an attack compared to the model using $Prev\_Interval$.

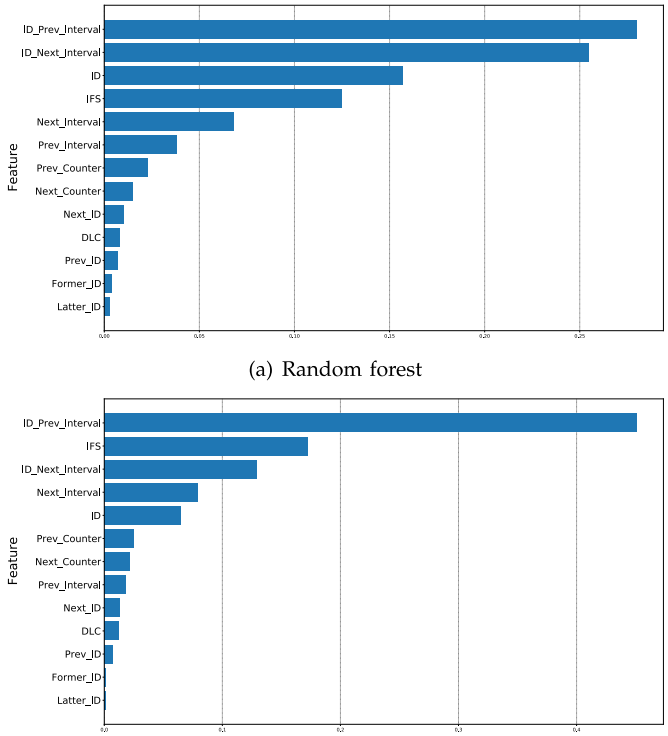
4) Processing Time: In this experiment, we measured the processing time of our models and the existing IDSs. The



(a) Random Forest w/o Counter and IFS (b) Random Forest w/ Counter (c) Random Forest w/ IFS

(d) XGBoost w/o Counter and IFS (e) XGBoost w/ Counter (f) XGBoost w/ IFS

Fig. 7. Heatmap on multi-class classification.



(a) Random forest

(b) XGBoost

Fig. 8. Feature importance for each model.

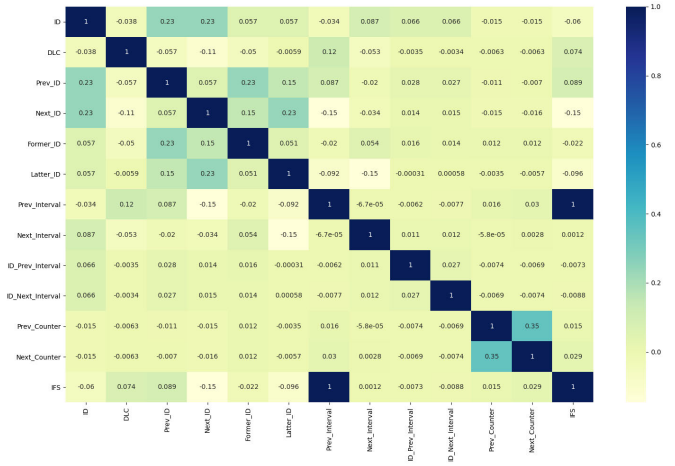


Fig. 9. The correlation coefficient between features.

processing time in the machine/deep learning model is the time between requesting for data and receiving a response; in this measurement, feature engineering time (ms) that contains preprocessing and feature extraction in our phases, and inference time (ms) of IDS's inference were measured. In general, the offline testing is used for post-analysis (e.g., digital forensics) of cyber attacks on the CAN bus, and the online testing is used for real-time attack detection.

Thus, the offline testing was experimented on the PC environment and measured the time required to perform multi-class classification models, while the online testing was experimented on a Raspberry Pi4 and measured the time required to perform binary classification models. Additionally, full features (i.e., w/ F.F) were utilized in the offline testing, while reduced features (i.e., w/o R.F) were utilized in the

TABLE V
PROCESSING TIME FOR A PACKET ON EACH IDS

	Model	Offline Testing Time (ms)		Online Testing Time (ms)	
		Feature extraction Time	**Inference Time**	**Feature extraction Time**	**Inference Time**
Machine Learning	Random forest with [24]	0.008	0.012	0.155	34.034
	XGBoost with [24]	0.008	0.001	0.156	34.325
	Random forest with our features	**0.003**	**0.012**	**0.145**	**33.744**
	XGBoost with our features	**0.003**	**0.001**	**0.145**	**33.599**
Deep Learning	GAN[25]	23.578	0.002	27.765	296.174
	DCNN[26]	3.867	0.013	5.344	179.146
	LSTM[27]	0.729	0.008	4.347	147.054
	MLP[28]	0.008	0.008	0.11	38.03

TABLE VI
DETECTION PERFORMANCE ON THE MASQUERADE ATTACK

Model		Accuracy	Precision	**Recall**	F1 score
Selected features (ID, IFS)	Random forest	99.42	89.34	**57.03**	69.62
	XGBoost	99.51	92.92	**52.78**	67.32
Selected features (ID, $Prev\_Interval$)	Random forest	99.39	89.28	**52.92**	67.32
	XGBoost	99.38	91.14	**50.86**	65.29

online testing. As shown in Table V, the feature extraction time of our proposed features is the shortest in offline testing. In addition, XGBoost shows the fastest inference time among IDSs. It takes about $0.001ms$ in the offline testing and about $33ms$ in the online testing. In the case of the GAN [25], there is a big difference in inference time between binary classification in the offline testing and multi-class classification in the online testing.

VII. CONCLUSION

With the development of CAVs that provide convenient driving functions to drivers and passengers, automotive security to protect driver safety is becoming more critical. However, the automotive internal network (i.e., CAN) has security vulnerabilities like a lack of authentication for access control, allowing an attacker to inject attack packets to control the vehicle's safety-critical functions, such as acceleration, braking, etc. In order to detect the injection of malicious CAN packets, new two detection features—the IFS and counter—have been introduced, and we applied them to two machine learning models, random forest and XGBoost. The performance evaluation was conducted on the dataset from a real vehicle containing five attack scenarios: flooding, spoofing, replay, fuzzing, and masquerade. Our proposed IDSs resulted in a better performance than existing machine/deep learning-based IDSs in terms of binary and multi-class classification. Furthermore, we evaluated the processing time and the importance of the proposed detection features. In future work, we plan to further analyze CAN frame structure awareness features and how these can be used for more advanced IDSs for in-vehicle networks. For instance, there are signal fields in a normal CAN data field that represent physical sensor values, such as vehicle speed, which usually change continuously within a certain range. If the value in this area suddenly increases or decreases significantly within over short period, it could be detected as a maliciously injected attack. In light of this, not only the counter signal but also various physical sensor signals contained in the CAN data field can be utilized as features of an IDS.

REFERENCES

[1] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in automotive communication systems," *Proc. IEEE*, vol. 93, no. 6, pp. 1204–1223, Jun. 2005.

[2] *11898-1: 2003-Road Vehicles–Controller Area Network*. Int. Org. Standardization, Geneva, Switzerland, 2003.

[3] M. Jedh, L. Ben Othmane, N. Ahmed, and B. Bhargava, "Detection of message injection attacks onto the CAN bus using similarities of successive messages-sequence graphs," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4133–4146, 2021.

[4] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive can networks–practical examples and selected short-term countermeasures," in *Proc. Int. Conf. Comput. Saf., Rel., Secur.* Cham, Switzerland: Springer, 2008, pp. 235–248.

[5] K. Koscher et al., "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Secur. Privacy*, May 2010, pp. 447–462.

[6] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. S 91, pp. 1–91, 2015.

[7] S. Nie, L. Liu, and Y. Du, "Free-fall: Hacking Tesla from wireless to CAN bus," *Briefing, Black Hat USA*, vol. 25, pp. 1–16, 2017.

[8] S. Nie, L. Liu, Y. Du, and W. Zhang, "Over-the-air: How we remotely compromised the gateway, BCM, and autopilot ECUs of Tesla cars," *Briefing, Black Hat USA*, vol. 91, 2018.

[9] A. K. Desta, S. Ohira, I. Arai, and K. Fujikawa, "ID sequence analysis for intrusion detection in the CAN bus using long short term memory networks," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2020, pp. 1–6.

[10] M. Nam, S. Park, and D. S. Kim, "Intrusion detection method using bi-directional GPT for in-vehicle controller area networks," *IEEE Access*, vol. 9, pp. 124931–124944, 2021.

[11] M. Han, P. Cheng, and S. Ma, "PPM-InVIDS: Privacy protection model for in-vehicle intrusion detection system based complex-valued neural network," *Veh. Commun.*, vol. 31, Oct. 2021, Art. no. 100374.

[12] V. K. Kukkala, S. V. Thiruloga, and S. Pasricha, "INDRA: Intrusion detection using recurrent autoencoders in automotive embedded systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3698–3710, Nov. 2020.

[13] P. Freitas De Araujo-Filho, A. J. Pinheiro, G. Kaddoum, D. R. Campelo, and F. L. Soares, "An efficient intrusion prevention system for CAN: Hindering cyber-attacks with a low-cost platform," *IEEE Access*, vol. 9, pp. 166855–166869, 2021.

[14] H. Qin, M. Yan, and H. Ji, "Application of controller area network (CAN) bus anomaly detection based on time series prediction," *Veh. Commun.*, vol. 27, Jan. 2021, Art. no. 100291.

[15] X. Duan, H. Yan, D. Tian, J. Zhou, J. Su, and W. Hao, "In-vehicle CAN bus tampering attacks detection for connected and autonomous vehicles using an improved isolation forest method," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2122–2134, Feb. 2023.

[16] H. Sun, M. Chen, J. Weng, Z. Liu, and G. Geng, "Anomaly detection for in-vehicle network using CNN-LSTM with attention mechanism," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10880–10893, Oct. 2021.

[17] R. Bosch, "Can specification version 2.0," *Rober Bousch GmbH, Postfach*, vol. 300240, p. 72, Sep. 1991.

[18] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.

[19] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.

[20] M. Marchetti and D. Stabili, "READ: Reverse engineering of automotive data frames," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 1083–1097, Apr. 2019.
[21] M. D. Pesé, T. Stacer, C. A. Campos, E. Newberry, D. Chen, and K. G. Shin, "LibreCAN: Automated CAN message translator," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 2283–2300.
[22] W. Choi, S. Lee, K. Joo, H. J. Jo, and D. H. Lee, "An enhanced method for reverse engineering CAN data payload," *IEEE Trans. Veh. Technol.*, vol. 70, no. 4, pp. 3371–3381, Apr. 2021.
[23] M. E. Verma, R. A. Bridges, J. J. Sosnowski, S. C. Hollifield, and M. D. Iannacone, "CAN-D: A modular four-step pipeline for comprehensively decoding controller area network data," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 9685–9700, Oct. 2021.
[24] O. Minawi, J. Whelan, A. Almehmadi, and K. El-Khatib, "Machine learning-based intrusion detection system for controller area networks," in *Proc. 10th ACM Symp. Des. Anal. Intell. Veh. Netw. Appl.*, 2020, pp. 41–47.
[25] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN based intrusion detection system for in-vehicle network," in *Proc. 16th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2018, pp. 1–6.
[26] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Veh. Commun.*, vol. 21, Jan. 2020, Art. no. 100198.
[27] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "LSTM-based intrusion detection system for in-vehicle can bus communications," *IEEE Access*, vol. 8, pp. 185489–185502, 2020.
[28] F. Amato, L. Coppolino, F. Mercaldo, F. Moscato, R. Nardone, and A. Santone, "CAN-bus attack detection with deep learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5081–5090, Aug. 2021.
[29] H. Kang, B. I. Kwak, Y. H. Lee, H. Lee, H. Lee, and H. K. Kim, "Car hacking and defense competition on in-vehicle network," in *Proc. 3rd Int. Workshop Automot. Auto. Vehicle Secur.*, 2021, p. 25.
[30] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Des. Implement. (OSDI)*, 2016, pp. 265–283.

Yeonseon Jeong received the B.S. degree in computer engineering from Hallym University, Chuncheon, South Korea, in 2021. She is currently pursuing the M.S. degree in information security with Korea University, Seoul, South Korea. Her research interests include applied cryptography, security and privacy for ad-hoc networks, automotive security, and the IoT/CPS security.

Hyunghoon Kim received the B.S. degree in computer engineering from Hallym University, Chuncheon, South Korea, in 2019, and the M.S. degree from the Graduate School of Software, Soongsil University, Seoul, in 2021, where he is currently pursuing the Ph.D. degree. His research interests include applied cryptography, security and privacy for ad-hoc networks, and the IoT/CPS security.

Seyoung Lee received the B.S. degree in mathematics (major) and computer science (bimajor) from the University of Seoul, Seoul, South Korea, in 2014, and the M.S. degree in information security from Korea University, Seoul, in 2016, where he is currently pursuing the Ph.D. degree in information security with the Graduate School of Information Security. His research interests include applied cryptography, automotive security, and senor security.

Wonsuk Choi received the B.S. degree in mathematics from the University of Seoul, Seoul, South Korea, in 2008, and the M.S. and Ph.D. degrees in information security from Korea University, Seoul, in 2013 and 2018, respectively. He was a Post-Doctoral Researcher with the Graduate School of Information Security, Korea University. In 2023, he joined the School of Cybersecurity, Korea University, as an Assistant Professor, where he was a Post-Doctoral Researcher with the Institute of Cyber Security and Privacy. His research interests include security for body area networks, usable security, applied cryptography, and smart car security.

Dong Hoon Lee (Member, IEEE) received the B.S. degree from the Department of Economics, Korea University, Seoul, South Korea, in 1985, and the M.S. and Ph.D. degrees in computer science from The University of Oklahoma, Norman, OK, USA, in 1988 and 1992, respectively. Since 1993, he has been with the Faculty of Computer Science and Information Security, Korea University. He is currently a Professor and the Director of the Graduate School of Information Security, Korea University. His research interests include cryptographic protocol, applied cryptography, functional encryption, software protection, mobile security, vehicle security, and ubiquitous sensor network (USN) security.

Hyo Jin Jo received the B.S. degree in industrial engineering and the Ph.D. degree in information security from Korea University, Seoul, South Korea, in 2009 and 2016, respectively. From 2016 to 2018, he was a Post-Doctoral Researcher with the Department of Computer and Information Systems, University of Pennsylvania, Philadelphia, PA, USA. He is currently an Assistant Professor with the School of Software, Soongsil University, Seoul. His research interests include applied cryptography and vehicle security.