Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.039992

*Article*

# FIDS: Filtering-Based Intrusion Detection System for In-Vehicle CAN

**Seungmin Lee, Hyunghoon Kim, Haehyun Cho and Hyo Jin Jo***

Department of Software, Soongsil University, Seoul, 06978, Korea

*Corresponding Author: Hyo Jin Jo. Email: hyojinjo86@gmail.com

Received: 28 February 2023; Accepted: 24 May 2023; Published: 11 September 2023

**Abstract:** Modern vehicles are equipped with multiple Electronic Control Units (ECUs) that support various convenient driving functions, such as the Advanced Driver Assistance System (ADAS). To enable communication between these ECUs, the Controller Area Network (CAN) protocol is widely used. However, since CAN lacks any security technologies, it is vulnerable to cyber attacks. To address this, researchers have conducted studies on machine learning-based intrusion detection systems (IDSs) for CAN. However, most existing IDSs still have non-negligible detection errors. In this paper, we propose a new filtering-based intrusion detection system (FIDS) to minimize the detection errors of machine learning-based IDSs. FIDS uses a whitelist and a blacklist created from CAN datasets. The whitelist stores the cryptographic hash value of normal packet sequences to correct false positives (FP), while the blacklist corrects false negatives (FN) based on transmission intervals and identifiers of CAN packets. We evaluated the performance of the proposed FIDS by implementing a machine learning-based IDS and applying FIDS to it. We conducted the evaluation using two CAN attack datasets provided by the Hacking and Countermeasure Research Lab (HCRL), which confirmed that FIDS can effectively reduce the FP and FN of the existing IDS.

**Keywords:** Controller area network; machine learning; intrusion detection system; automotive security

## 1 Introduction

A modern vehicle is not just a means of transportation, but also a large mobile computer integrated with software. With the development of software technology, a variety of Advanced Driver Assistance Systems (ADAS) technologies, such as Lane Departure Warning (LDS), Adaptive Cruise Control (ACC), and Lane Keeping Assist (LKA), have been installed in vehicles. To efficiently control the numerous Electronic Control Units (ECUs) of vehicles, numerous in-vehicle networks have been studied. Controller Area Network (CAN), which was registered as the ISO 11898 standard in 1993, is used as the de facto standard in the automotive industry [1].

CAN is an in-vehicle communication protocol developed by Robert Bosch that allows ECUs to communicate with each other. However, when CAN was initially developed, security was not considered at all, and therefore there were no encryption, authentication, or integrity verification technologies available. Generally, there are two ways to access the CAN network: the first is through a physical access point using the On-Board Diagnostic (OBD-II) port installed on every vehicle, and the second is through a remote access point such as Wi-Fi, Bluetooth, and Cellular, etc. A malicious attacker can control a vehicle through these access points by injecting CAN messages that can manipulate various vehicle operations.

As research on cyber-attacks on the in-vehicle CAN has progressed, research on defense solutions such as IDS has also been actively conducted. In-vehicle IDS is typically divided into two types: a packet-based IDS and a window-based IDS, depending on the number of packets required for attack detection. The packet-based IDS has the advantage of a relatively fast intrusion detection speed because the number of packets required for attack detection is small. However, it has the disadvantage of difficulty in analyzing the correlation between consecutive packets. On the other hand, the window-based IDS can analyze the correlation of successive packets. Recently, IDSs using deep learning technology have been proposed. However, the proposed IDSs have limitations in that they often result in FP and FN. Therefore, this paper proposes a study of a Filtering-based IDS (FIDS) that utilizes whitelist and blacklist to improve the detection rate of existing IDSs.

The main contributions of this paper are as follows:

- In this paper, a filtering-based IDS (FIDS) is proposed to minimize the detection errors of machine learning-based IDSs. FIDS is composed of two parts: a whitelist and a blacklist. The whitelist stores the cryptographic hash information of normal packet windows containing regular CAN packets, while the blacklist contains information about abnormal packet windows resulting from attacks, such as the number of CAN IDs in a packet window and the presence of unused CAN IDs. In addition, FIDS can be used in conjunction with any other packet window-based IDSs.
- Through experiments using CAN datasets from two real vehicles, it has been confirmed that FIDS can reduce the detection errors of machine learning-based IDSs.

The structure of this paper is as follows. In Section 2, related works on different defense solutions for detecting cyber-attacks on CAN are presented. Section 3 provides background knowledge on CAN. Section 4 introduces the proposed FIDS. Section 5 evaluates the performance of FIDS through experimental results. Finally, Section 6 presents the conclusion of the paper.

## 2 Related Work

When the CAN was originally developed, security was not a priority. Consequently, research has been conducted to identify various vulnerabilities and potential cyber-attacks on the CAN, leading to the development of IDS as a defense solution. As outlined in [2], IDSs for CAN can be classified into three metrics, the number of frames required to detect an attack, the type of data used for detection, and the model used to detect an attack. In this paper, we focus on classifying IDSs for CAN into two types based on the number of frames required to detect an attack: packet-based IDS and window-based IDS.

### *2.1 Packet-Based IDS*

Packet-based IDS detects cyber-attacks based on information obtained from a single CAN packet. Packet-based IDS detects cyber-attacks by analyzing information from a single CAN packet. This section outlines several studies on packet-based IDS. Kang et al. proposed a method for distinguishing normal packets from attack packets using a DNN classifier trained on features extracted from a single CAN packet [3].

Lee et al. proposed an IDS that utilizes CAN remote frames and analyzes the offset ratio and time interval between the remote frame and its response frame [4]. Groza et al. proposed an IDS that utilizes bloom filtering based on message identifiers and parts of data frame fields to test frame periodicity. The IDS utilized the fact that most traffic on the CAN network is inherently cyclical and that the format of data fields is fixed due to strict signaling assignments [5]. Cho et al. proposed a Clock-based IDS (CIDS) that uses clock skew, a unique hardware characteristic of ECU, to detect attacks. The interval of the CAN messages in the vehicle was periodically measured, and the Recursive Least Squares (RSL) algorithm was used to construct a baseline of the ECU's clock operation. Based on this baseline, CIDS uses the Cumulative Sum (CUSUM) algorithm to detect attacks by identifying abnormal changes in identification errors [6]. Ying et al. proposed a new Masquerade Attack called a Cloaking Attack, which is not detected by Cho et al.'s CIDS. This attack evades detection by emulating the desired clock skew through manipulation of the message transmission time of the spoofed message. In addition, official models for the State-Of-The-Art (SOTA) IDS and the Network Time Protocol (NTP) were proposed to detect cloaking attacks [7]. Choi et al. proposed VoltageIDS, which utilizes the hardware-specific characteristics of the ECU. The IDS uses the unique characteristics of the CAN signal as a fingerprint for the ECU and detects attack messages based on the voltage signal of the ECU [8].

In general, the Packet-based IDS is relatively fast at detecting intrusions because the number of packets required for attack detection is small. However, there is a limitation in that it is difficult to analyze the correlation of sequential packets. Therefore, research has been conducted on Window-based IDS to utilize the correlation of sequential packets.

### *2.2 Window-Based IDS*

A Window-based IDS uses a packet window containing multiple CAN packets to detect an attack. There are two ways to define a window: one includes a predefined number of packets, and the other includes packets generated during a predefined period. Muter et al. proposed an Entropy-based IDS, considering that IDS for vehicles is a constrained environment in which attacks must be detected in real-time. This study defined entropy as a measure of how many different CAN packets existed in each dataset, and through this, attack messages are detected [9]. However, in an experiment to evaluate the efficiency of the Entropy-based IDS by Marchetti et al., it was proven that the Entropy-based IDS can only detect attacks composed of many manipulated CAN messages and not attacks injected with a few manipulated CAN messages [10]. Taylor et al. proposed a frequency-based IDS since most attacks inject malicious packets into the network, and CAN messages are generally transmitted at a strict frequency. The IDS detects an attack by finding the average time using an algorithm that measures the timing between packets in the sliding window and then compares it with the past average [11]. However, a limitation was found in that the frequency-based IDS could not distinguish between valid aperiodic CAN packets without attacks and valid aperiodic CAN packets under attacks [12]. Islam et al. proposed a Graph-based IDS. The IDS constructs a graph of the CAN message with a predefined number of packets and distinguishes the attack window from the normal window through a chi-square test. However, the proposed IDS has a limitation in that it requires too many packets to detect an attack [13].

Recently, Window-based IDS using deep learning has also been studied. Song et al. proposed a DCNN-based IDS using Deep Convolutional Neural Network (DCNN), which is mainly used in image classification. In their paper, a 2D grid frame of $29 \times 29$ size is created by converting the hexadecimal CAN ID into 29-bit binary for the input of the model. The generated input is used as DCNN's learning data, and spatial correlation features are learned to detect attack messages and normal messages. In the case of the model, a lightweight version of Inception-RestNet called Reduced Inception-RestNet was created and used [14,15]. Hossain et al. proposed an LSTM-based IDS using two deep learning model, Vanilla LSTM and Stacked LSTM [16]. In their paper, ID, Data, and DLC Field of CAN Data Frame were used for model training. The input data of the model was extracted from the CAN Data Frame, and preprocessing was performed to convert ID and Data Field, which are hexadecimal numbers, to binary numbers. Seo et al. proposed a GAN-based IDS (GIDS) using a Generative Adversarial Network (GAN) [17,18]. GIDS consists of two models: a classifier to detect existing attacks and a model to detect new, unknown attacks. In their paper, a CAN ID is created as a 2D grid frame using one-hot vector encoding, and preprocessing is performed to convert it into an image. The classifier for detecting existing attacks creates a real image by performing a preprocessing process on the collected dataset and learns the spatial correlation feature of the real image. A model for detecting a new attack learns to create a fake image like a real image using randomly generated noise. Then, with the fake images generated by the model, the classifier learns to distinguish between real and fake images. Finally, the learned classifier and the threshold of the model are adjusted to detect the attack message. Amato et al. proposed an attack detection and classification methodology for CAN Bus using deep learning-based Neural Networks (NN) and Multi Layer Perceptrons (MLP). In their work, the distribution of normal messages and attack messages was used as feature vectors to classify normal and attack messages [19]. Javed et al. proposed CANintelliDS to prevent attacks on CAN Bus. CANintelliDS is based on a combination of a Convolutional Neural Network (CNN) and an Attention-Based Gated Recurrent Unit (GRU) model. It can detect both single and mixed intrusion attacks on the CAN Bus [20].

All previous proposed IDS studies are used independently. This does not solve detection errors such as FP and FN that exist in the results detected by IDS. However, the FIDS proposed in this paper can be used independently, and the detection results of the IDS can be filtered secondarily. Therefore, the ensemble effect can be expected by combining the previously studied IDS and FIDS, and through this, the detection performance of the IDS can be improved.

## 3 Backgrounds

### *3.1 Controller Area Network*

The CAN protocol is an in-vehicle network that was developed by Robert Bosch in 1986. It supports efficient communication among ECUs in a bus network structure. The CAN frame consists of two types of bits: a dominant bit (0) and a recessive bit (1). These bits are expressed using the voltage difference between two copper wires, CAN-H (high) and CAN-L (low), of the CAN bus. For example, if CAN-H and CAN-L have voltages of 3.5 and 1.5 V, respectively, the difference between the two voltages (2.0 V or more) is expressed as a dominant bit (0). Similarly, if both CAN-H and CAN-L have voltages of 1.5 V, the difference between the two voltages is expressed as a recessive bit (1).

Since the CAN bus operates as an AND logic gate if one ECU transmits a recessive bit (1) and another ECU transmits a dominant bit (0) simultaneously, a dominant bit (0) can flow through the CAN bus. Therefore, the CAN bus determines the priority of the message through the arbitration field for the simultaneously transmitted CAN data frame. A message with a smaller arbitration field value has a higher priority. There are four types of CAN frames: data frame, remote frame, error frame, and overload frame. The data frame is used for transmitting data, the remote frame is used for requesting the transmission of a specified message, the error frame is used for indicating an error detected through an error flag, and the overload frame is used for injecting delay to adjust the speed between frames.

## *3.2 CAN Data Frame*

A data frame, which is one type of CAN frame, is used for transmitting and receiving data, and there are two types available: CAN 2.0A and CAN 2.0B. CAN 2.0A has an ID length of 11 bits, while CAN 2.0B has an extended ID field with a length of 29 bits. The remaining fields in both types of frames have the same length. The basic format for a CAN 2.0A data frame is shown in Fig. 1, and the description of each field is as follows:

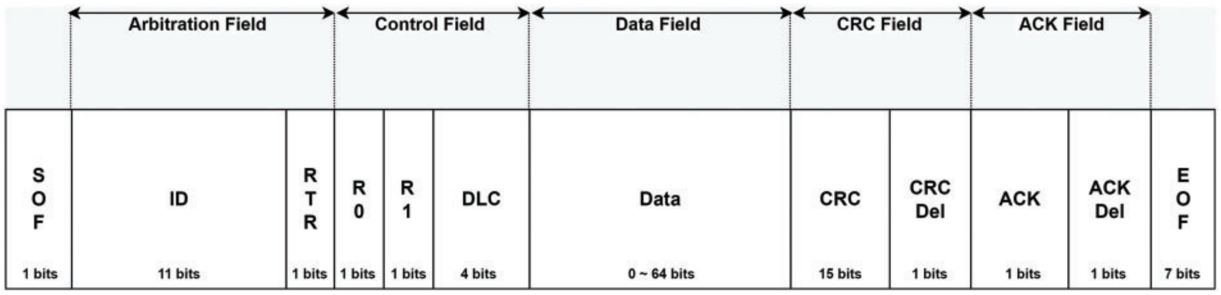

**Figure 1:** CAN 2.0A data frame format

Start of frame (SOF) consists of 1 dominant bit (0) and indicates the beginning of a CAN data frame. The arbitration field includes an Identifier (ID) and a Remote Transmission Request (RTR) value. The ID is an 11-bit value that determines the priority of the CAN data frame, with a lower ID indicating higher priority. The RTR value distinguishes between a CAN data frame and a remote frame, with a dominant bit (0) representing a data frame and a recessive bit (1) representing a remote frame. If the IDs are the same, priority is determined by comparing RTR values, with the data frame having higher priority than the remote frame. The control field includes a total of 6 bits, including R1, R0, and Data Length Code (DLC). R1 and R2 are reserved bits of 2 bits and are used when the ID value is extended to 29 bits. DLC is a 4-bit value that determines the size of the frame data. The data field contains the actual transmitted message data and can include up to 64 bits. The Cyclic Redundancy Check Delimiter (CRC) is a field that uses a cyclic redundancy check to detect errors in the CAN data frame. The Acknowledgement (ACK) field confirms whether the transmission of the CAN data frame was successful. Finally, the End of Frame (EOF) is a value that indicates the end of the CAN frame and consists of 7 recessive bits (1).

## *3.3 CAN Attack Scenarios*

There are four types of attacks on the open dataset used in this paper: Flooding, Spoofing, Fuzzing, and Replay. The following is a description of each attack technique:

- Flooding Attack: As a Denial of Service (DoS) attack, this method involves injecting many CAN messages with a higher priority ID than the CAN IDs used by an attack target vehicle. It is mainly executed by setting the CAN ID to “$0 \times 000$”.
- Spoofing Attack: This attack aims to control a specific function of an attack target vehicle as desired by the attacker. It is executed by analyzing the CAN message used by the vehicle, generating a CAN message responsible for a particular function, and injecting it.
- Fuzzing Attack: This attack is aimed at causing unexpected vehicle malfunctions. It is executed by injecting randomly generated CAN messages into an attack target vehicle.

- Replay Attack: This attack involves re-injecting the CAN message used by an attack target vehicle. It is executed by observing and collecting CAN traffic and re-injecting the recorded CAN traffic at a specific time.

## 4 Proposed System Model

### *4.1 Overview*

FIDS is designed to minimize the false positives and false negatives of existing deep learning-based IDSs by filtering misclassified results. As shown in Fig. 2, FIDS consists of three steps. The first step is Data Preparation, in which CAN packets are collected from CAN datasets, and then a preprocessing process is performed to convert the collected CAN packets into the IDS's input format. The second step is the construction of a White/Blacklist. The whitelist stores valid patterns of normal CAN packet windows, and the blacklist stores invalid patterns of abnormal CAN packet windows. The third step is the filtering step, which corrects the detection errors of the existing IDS by using the whitelist and blacklist.

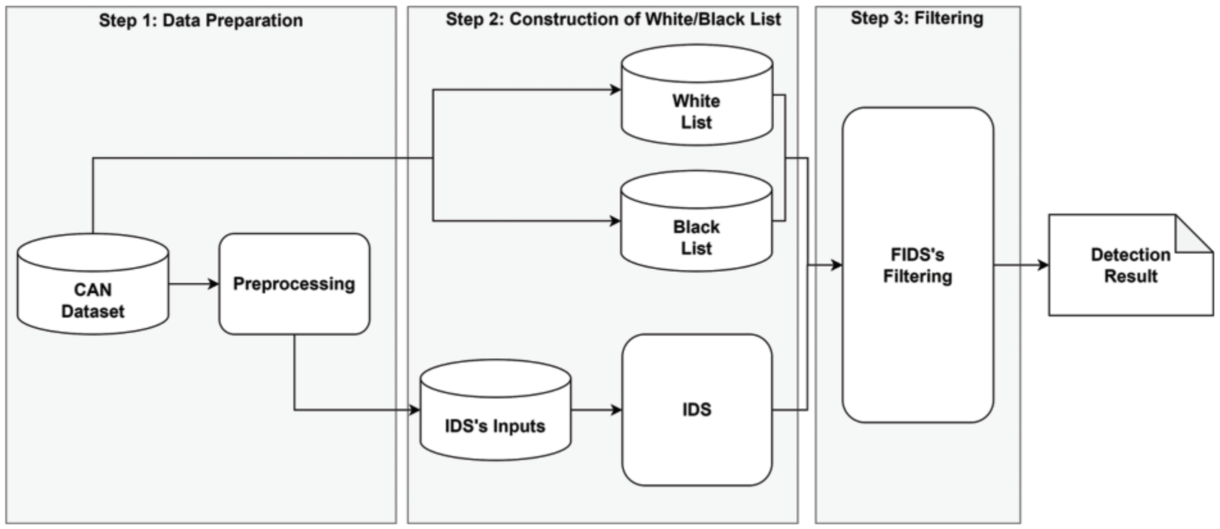

**Figure 2:** Overview of FIDS

### *4.2 Data Preparation*

In general, CAN packets are collected by connecting a monitoring device to the OBD-II port installed on the vehicle. The collected data contains the CAN ID field, DLC field, data field, and timestamp. Since the CAN packets cannot be directly entered into an existing IDS without preprocessing, they should be converted to the required input format of the target IDS. For example, a CNN-based IDS, a predetermined number of CAN IDs are stacked and formed into a 2D grid image. This preprocessing is necessary to train the IDS's AI model.

### *4.3 Construction of White/Black List*

FIDS consists of a whitelist to allow normal messages and a blacklist to detect attack messages. The whitelist contains the cryptographic hash values of normal CAN packet windows without attacks. In addition, the blacklist includes the characteristics of abnormal CAN packet windows to detect an attack. This section describes the process of creating whitelists and blacklists in detail.

### *4.3.1 Generating Whitelist*

A whitelist is created from a dataset labeled as normal without any attack packets because it is used to identify normal messages. The process of creating a whitelist is shown in Fig. 3. First, the ID field of each CAN packet is extracted from a normal dataset. Then, the extracted IDs, which are in hexadecimal format, are converted into binary format with 11 bits. After that, a 2D grid frame is created according to the predefined window size. However, since 2D grid frames are difficult to manage, they are inputted into a cryptographic hash function (e.g., MD5, SHA256), which produces a fixed-size output. The generated cryptographic hash value is stored in the whitelist and is used for detecting a normal packet window.

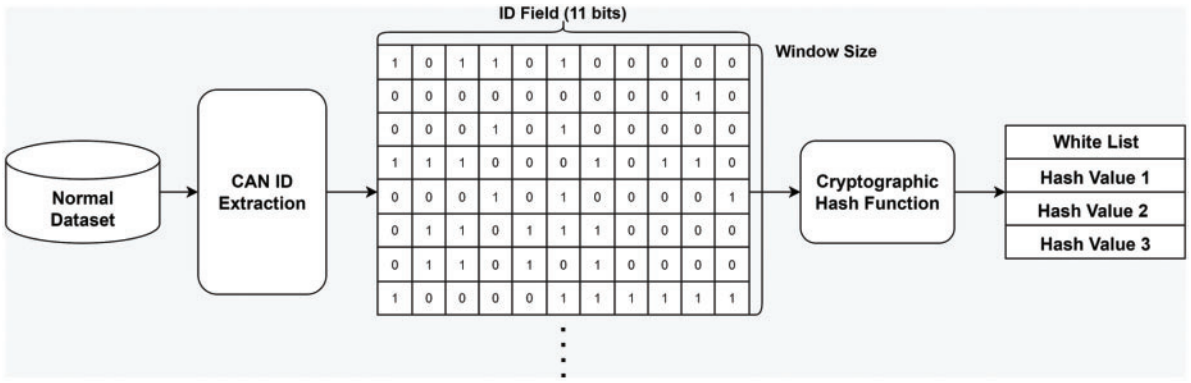

**Figure 3:** Generation of the whitelist

### *4.3.2 Generating Blacklist*

The blacklist contains rules that represent the characteristics of attack messages to detect them. FIDS defines two rules. The first rule, named $Rule_{twice}$, uses the feature that CAN messages are transmitted periodically. Through $Rule_{twice}$, when a CAN ID is found more than twice in one packet window, the corresponding window is judged as an attack window containing attack messages. The second rule, named $Rule_{random}$, uses the feature of fuzzy attacks, in which there are randomly generated and injected CAN IDs. By using $Rule_{random}$, if a packet window containing an unknown CAN ID that is not found in the normal dataset is identified, the corresponding window is regarded as an attack window.

## *4.4 Filtering*

This section describes the filtering step for misclassified results of existing IDSs using FIDS. This filtering step consists of two sub-steps: filtering with the whitelist and filtering with the blacklist. Filtering with the whitelist corrects false positive windows to normal windows if attack windows classified by existing IDSs are found in the whitelist. This reduces the false positive rate. In addition, filtering with the blacklist reduces false negatives by checking $Rule_{twice}$ and $Rule_{random}$ for every normal window classified by the existing IDSs. In conclusion, FIDS, composed of the whitelist and blacklist, improves the detection accuracy of existing IDSs.

## 5 Experiment Results and Evaluation

In this section, we select the existing IDSs and implement it, and then evaluate the performance of FIDS using open CAN datasets.[1]

### 5.1 Experimental Environment and Evaluation Metrics

#### 5.1.1 Experimental Environment

In this paper, the performance of FIDS was evaluated by selecting two IDS models: DCNN-based IDS (DIDS) by Song et al. [15] and LSTM-based IDS (LIDS) by Hossain et al. [16]. The rationale behind choosing these two models is as follows: DIDS represents the pioneering application of a CNN model in automotive IDS, and it can be easily integrated with FIDS to enhance performance. On the other hand, LIDS excels in capturing various patterns and dependencies within a sequence of packets due to the LSTM feature, which enables it to effectively memorize past information and incorporate it with the current input to predict the next state. The source codes of the models used in these studies were not available as open source. Therefore, we developed our own implementation. For the implementation of DIDS, we utilized Python 3.8 and various AI libraries such as Scikit-learn, TensorFlow, and Keras. The hyperparameters employed in both models were set to match those specified in their respective papers [15,16]. The experiments were conducted on a PC equipped with an Intel (R) Core (TM) i7-10750H CPU @ 2.60 GHz, NVIDIA GeForce GTX 1650 Ti, 16 GB RAM, and the Windows 10 Pro operating system.

#### 5.1.2 Confusion Matrix

The confusion matrix is used to analyze the detection results of IDS for CAN. It consists of four values: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). TP occurs when an attack is correctly detected as an attack, while TN occurs when normal behavior is correctly detected as normal. On the other hand, FP occurs when normal behavior is incorrectly detected as an attack, and FN occurs when an attack is incorrectly detected as normal. The confusion matrix table is shown in Table 1.

**Table 1:** Confusion matrix

| Confusion matrix | | Predicted | |
|---|---|---|---|
| | | Negative (Attack) | Positive (Normal) |
| Actual | Negative (Attack) | TN (True Negative) | FP (False Positive) |
| | Positive (Normal) | FN (False Negative) | TP (True Positive) |

#### 5.1.3 Evaluation Metrics

In this paper, four evaluation metrics were selected to evaluate the performance of FIDS: accuracy, precision, recall, and F1-score. Accuracy represents the ratio of correctly predicted normal and attack messages to total messages. Precision represents the ratio of actual attack messages among messages predicted to be actual attack messages. Recall refers to the ratio of correctly predicted attack messages among actual attack messages. F1-score represents the harmonic mean of precision and recall. The

[1] https://ocslab.hksecurity.net/Datasets

equations for calculating these evaluation metrics are as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$\text{F1} - \text{score} = 2 \times \frac{\text{Precision} \ \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

## *5.2 Data Set*

FIDS was evaluated using two datasets provided by the Hacking and Countermeasure Research Lab (HCRL): "Car Hacking: Attack & Defense Challenge 2020" dataset (Dataset I) and the dataset (Dataset II) used in [17]. In the data set provided by HCRL, there are no error frames; it consists only of non-error packets. Therefore, the whitelist and blacklist of FIDS consider only normal CAN packets.

The Dataset I was collected from a Hyundai Avante CN7 and is organized as shown in Table 2. Messages in the CAN dataset include Timestamp, Arbitration ID, DLC, Data, Class (Normal, Attack), and Subclass (attack type), with the attack type being the same as the four attack scenarios described in Section 3.3. Dataset II was collected from a Hyundai YF Sonata and consists of a normal dataset and four attack datasets as shown in Table 3. Messages in the CAN dataset include Timestamp, Arbitration ID, DLC, Data, and Flag, with Flag consisting of "R" for a general message and "T" for an injected message. The four attack scenarios used to construct the Hyundai YF Sonata attack dataset are as follows: for the DoS Attack, a message with a CAN ID of "$0 \times 00$" is injected every 0.3 ms; for the Fuzzy Attack, a randomly generated CAN ID and data message is injected every 0.5 ms; for the Spoofing Attack (RPM/GEAR), a message of a specific CAN ID related to RPM/GEAR information is injected every 1 ms. The dataset serves as input for training and testing the DIDS model and LIDS model, as well as for generating the FIDS whitelist. While Dataset I is partitioned into separate training and testing sets for model training, Dataset II lacks such a division. Thus, in this paper, Dataset II is divided into training and testing sets based on the packet count, as shown in Table 4.

**Table 2:** Construction of Hyundai Avante CN7 dataset (Dataset I)

| Round | Type | # Normal | # Attack | # Total |
|---|---|---|---|---|
| Preliminary | Submission | 3,358,210 | 393,836 | 3,752,046 |
| | Training | 3,372,743 | 299,408 | 3,672,151 |

**Table 3:** Construction of Hyundai Sonata YF dataset (Dataset II)

| Attack type | # Normal | # Attack | # Total |
|---|---|---|---|
| DoS attack | 3,078,250 | 587,521 | 3,665,771 |
| Fuzzy attack | 3,347,019 | 491,847 | 3,838,860 |

(Continued)

**Table 3 (continued)**

| Attack type | # Normal | # Attack | # Total |
|---|---|---|---|
| Spoofing the RPM gauze | 3,845,890 | 597,252 | 4,443,142 |
| Spoofing the drive GEAR | 3,966,805 | 654,897 | 4,621,702 |
| Attack-free normal | 988,872 | - | 988,987 |

**Table 4:** Construction of training and test datasets for Hyundai Sonata YF

| Attack type | Train | Test | # Total |
|---|---|---|---|
| DoS attack | #1∼#2,456,066 | #2,456,067∼#3,665,771 | 3,665,771 |
| Fuzzy attack | #1∼#2,572,036 | #2,572,037∼#3,838,860 | 3,838,860 |
| Spoofing the RPM gauze | #1∼#3,096,540 | #3,096,541∼#4,621,702 | 4,621,702 |
| Spoofing the drive GEAR | #1∼#2,976,905 | #2,976,906∼#4,443,142 | 4,443,142 |

### *5.3 Experiment Result*

In the case of Dataset I, DIDS and LIDS were trained using the "Training Type" dataset and tested using the "Submission Type" dataset. In the case of Dataset II, DIDS and LIDS were trained and tested using the training and test datasets. Finally, the detection results predicted by DIDS and LIDS are filtered through the FIDS proposed in this paper. The performance of FIDS is evaluated by comparing four metrics with and without FIDS.

#### *5.3.1 Packet Window's Patterns according to the Window Size*

To construct the whitelist, a specific pattern in the packet window is required. Therefore, this paper conducted an experiment to determine the optimal window size where these patterns exist. The patterns, represented by unique cryptographic hash values, were extracted from a predefined number of CAN packets (i.e., the window size) in the normal datasets of Dataset I and Dataset II.

During the experiment, the window pattern rate, which measures how many patterns exist based on the window size, was calculated. The window pattern rate is calculated by dividing the number of window patterns by the total number of windows. The equation for obtaining the window pattern ratio is shown in Eq. (5). In this experiment, a pattern is defined as a case where the cryptographic hash value calculated for every window overlaps more than 30 times.

$$Winodw\ Pattern\ Rate = \frac{\text{Duplicated Window}}{\text{Total Window}} \tag{5}$$

The experiment results showed that there were no patterns when the window size was 13 or greater in Dataset I, and no patterns when the window size was 16 or greater in Dataset II, as shown in Table 5. We conducted an experiment using a heuristic method to determine the optimal window size. As shown in Table 5, we found that smaller window sizes provide better performance. This is because smaller window sizes create more distinguishable patterns between normal packets and attack packets. Therefore, window sizes of 3, 5, and 10 were used in the experiment.

**Table 5:** Window pattern rate by window size

| Type | Dataset I (Avante CN7) | Dataset II (Sonata YF) |
|---|---|---|
| Window size | Window pattern rate | |
| 2 | 0.937 | 0.998 |
| 3 | 0.728 | 0.971 |
| 4 | 0.501 | 0.866 |
| 5 | 0.346 | 0.747 |
| 6 | 0.210 | 0.589 |
| 7 | 0.108 | 0.443 |
| 8 | 0.058 | 0.301 |
| 9 | 0.027 | 0.182 |
| 10 | 0.019 | 0.102 |
| 11 | 0.011 | 0.042 |
| 12 | 0.010 | 0.017 |
| 13 | 0.0 | 0.014 |
| 14 | 0.0 | 0.001 |
| 15 | 0.0 | 0.001 |
| 16 | 0.0 | 0.0 |

### 5.3.2 *Comparison*

In this experiment, the performance evaluation of FIDS involved filtering the erroneous detection results of both DIDS and LIDS for each dataset. The window size used in the experiment was determined based on the outcomes of the experiments mentioned in Section 5.3.1. The results of the experiment are presented in Table 6. In the case of Dataset I, both DIDS and LIDS initially exhibited low detection performance. However, their performance significantly improved after applying FIDS filtering, with LIDS achieving particularly high results. For Dataset II, there was an improvement in the F1 scores after the FIDS filtering process. Notably, when the window size was set to 3 (DIDS with FIDS), there was a significant increase in the F1-score, approximately 12%. Conversely, when the window size was set to 10, the increase in the evaluation metric was minimal, reaching only 6%. This highlights the importance of selecting an appropriate window size for FIDS in order to maximize the performance improvement of the existing IDS.

**Table 6:** Comparison of evaluation metrics before and after filtering using FIDS

| Type | Dataset I | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | DCNN based IDS | | | | | | LSTM based IDS | | | | | |
| Window size | 3 | | 5 | | 10 | | 3 | | 5 | | 10 | |
| w/o, w/FIDS | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ |
| Accuracy | 0.867 | 0.881 | 0.796 | 0.810 | 0.741 | 0.756 | 0.892 | 0.929 | 0.813 | 0.858 | 0.740 | 0.789 |
| Precision | 0.756 | 0.795 | 0.683 | 0.703 | 0.698 | 0.708 | 0.834 | 0.930 | 0.752 | 0.820 | 0.769 | 0.797 |

(Continued)

**Table 6 (continued)**

| Type | Dataset I | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | DCNN based IDS | | | | | | LSTM based IDS | | | | | |
| Window size | 3 | | 5 | | 10 | | 3 | | 5 | | 10 | |
| w/o, w/FIDS | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ |
| Recall | 0.771 | 0.772 | 0.809 | 0.816 | 0.774 | 0.807 | 0.761 | 0.807 | 0.715 | 0.775 | 0.627 | 0.729 |
| F1-score | 0.764 | 0.783 | 0.741 | 0.755 | 0.734 | 0.754 | 0.796 | 0.864 | 0.733 | 0.797 | 0.691 | 0.762 |
| Type | Dataset II (DoS Attacks) | | | | | | | | | | | |
| Model | DCNN based IDS | | | | | | LSTM based IDS | | | | | |
| Window Size | 3 | | 5 | | 10 | | 3 | | 5 | | 10 | |
| w/o, w/FIDS | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ |
| Accuracy | 0.980 | 0.999 | 0.987 | 0.999 | 0.989 | 0.994 | 1.000 | 1.000 | 0.999 | 1.000 | 1.000 | 1.000 |
| Precision | 0.777 | 0.990 | 0.847 | 0.984 | 0.862 | 0.917 | 0.998 | 1.000 | 0.995 | 0.999 | 0.998 | 0.999 |
| Recall | 1.000 | 1.000 | 1.000 | 1.000 | 0.999 | 1.000 | 1.000 | 1.000 | 0.997 | 1.000 | 0.997 | 1.000 |
| F1-score | 0.874 | 0.995 | 0.917 | 0.992 | 0.926 | 0.957 | 0.999 | 1.000 | 0.996 | 0.999 | 0.997 | 0.999 |
| Type | Dataset II (Fuzzy Attacks) | | | | | | | | | | | |
| Model | DCNN based IDS | | | | | | LSTM based IDS | | | | | |
| Window Size | 3 | | 5 | | 10 | | 3 | | 5 | | 10 | |
| w/o, w/FIDS | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ |
| Accuracy | 0.979 | 0.998 | 0.985 | 0.997 | 0.987 | 0.992 | 0.999 | 1.000 | 0.999 | 1.000 | 0.999 | 1.000 |
| Precision | 0.806 | 0.979 | 0.881 | 0.974 | 0.901 | 0.934 | 0.998 | 1.000 | 0.995 | 0.997 | 0.998 | 0.998 |
| Recall | 0.993 | 0.994 | 0.988 | 0.997 | 0.994 | 0.999 | 0.992 | 1.000 | 0.993 | 1.000 | 0.989 | 1.000 |
| F1-score | 0.89 | 0.987 | 0.932 | 0.985 | 0.946 | 0.966 | 0.995 | 1.000 | 0.994 | 0.999 | 0.994 | 0.999 |
| Type | Dataset II (Spoofing the RPM) | | | | | | | | | | | |
| Model | DCNN based IDS | | | | | | LSTM based IDS | | | | | |
| Window Size | 3 | | 5 | | 10 | | 3 | | 5 | | 10 | |
| w/o, w/FIDS | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ |
| Accuracy | 0.957 | 0.972 | 0.971 | 0.981 | 0.990 | 0.993 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| Precision | 0.891 | 0.994 | 0.946 | 0.994 | 0.965 | 0.980 | 0.999 | 1.000 | 0.999 | 1.000 | 0.999 | 1.000 |
| Recall | 0.835 | 0.835 | 0.913 | 0.913 | 0.991 | 0.991 | 1.000 | 1.000 | 1.000 | 1.000 | 0.999 | 1.000 |
| F1-score | 0.862 | 0.908 | 0.929 | 0.952 | 0.978 | 0.986 | 1.000 | 1.000 | 0.999 | 1.000 | 0.999 | 1.000 |
| Type | Dataset II (Spoofing the GEAR) | | | | | | | | | | | |
| Model | DCNN based IDS | | | | | | LSTM based IDS | | | | | |
| Window Size | 3 | | 5 | | 10 | | 3 | | 5 | | 10 | |
| w/o, w/FIDS | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ | w/o | w/ |
| Accuracy | 0.965 | 0.983 | 0.978 | 0.989 | 0.989 | 0.993 | 1.000 | 1.000 | 0.995 | 1.000 | 0.999 | 0.999 |
| Precision | 0.864 | 0.991 | 0.928 | 0.991 | 0.956 | 0.974 | 0.999 | 1.000 | 0.995 | 0.999 | 0.998 | 0.998 |
| Recall | 0.880 | 0.880 | 0.942 | 0.942 | 0.989 | 0.989 | 1.000 | 1.000 | 1.000 | 1.000 | 0.996 | 0.997 |
| F1-score | 0.872 | 0.932 | 0.935 | 0.966 | 0.972 | 0.982 | 1.000 | 1.000 | 0.997 | 0.999 | 0.997 | 0.998 |

## 6 Conclusion

The advancement of vehicle technology plays a role in providing driving convenience. As a result, more and more IT technologies are being installed in modern vehicles. However, since security technology has not been considered for the CAN protocol in vehicles, it has been attacked in various ways. To deal with these threats, machine learning-based IDSs have been studied extensively. However, machine learning-based IDSs have inherent limitations regarding identifying attack packets as normal or identifying normal packets as attacks. This is particularly problematic when it comes to CAN packets related to controlling a safety-critical ECU because misidentifying them can lead to serious consequences. Therefore, this paper proposes FIDS to correct the misidentified results of the existing machine learning-based IDSs. For the performance evaluation of FIDS, we directly implemented two machine learning models, CNN-based IDS and LSTM-based IDS, and conducted experiments using publicly available open datasets.

**Funding Statement:** This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. NRF-2021R1A4A1029650).

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

[1] ISO, "ISO 11898-1: 2003-road vehicles–controller area network," in *International Organization for Standardization*. Geneva, Switzerland, ISO, 2003.

[2] G. Dupont, J. den Hartog, S., Etalle and A. Lekidis, "A survey of network intrusion detection systems for controller area network," in *2019 IEEE Int. Conf. on Vehicular Electronics and Safety (ICVES)*, Cairo, Egypt, pp. 1–6, 2019.

[3] M. J. Kang and J. W. Kang, "A novel intrusion detection method using deep neural network for in-vehicle network security," in *2016 IEEE 83rd Vehicular Technology Conf. (VTC Spring)*, Nanjing, China, pp. 1–5, 2016.

[4] H. Lee, S. H. Jeong and H. K. Kim, "OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame," in *2017 15th Annual Conf. on Privacy, Security and Trust (PST)*, Calgary, AB, Canada, pp. 57–5709, 2017.

[5] B. Groza and P. S. Murvay, "Efficient intrusion detection with bloom filtering in controller area networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1037–1051, 2018.

[6] K. T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *25th USENIX Security Symp.*, vol. 40, pp. 911–927, 2016.

[7] X. Ying, S. U. Sagong, A. Clark, L. Bushnell and R. Poovendran, "Shape of the cloak: Formal analysis of clock skew-based intrusion detection system in controller area networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2300–2314, 2019.

[8] W. Choi, K. Joo, H. J. Jo, M. C. Park and D. H. Lee, "VoltageIDS: Low-level communication characteristics for automotive intrusion detection system," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2114–2129, 2018.

[9] M. Muter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *2011 IEEE Intelligent Vehicles Symp. (IV)*, Baden-Baden, Germany, pp. 1110–1115, 2011.

[10] M. Marchetti, D. Stabili, A. Guido and M. Colajanni, "Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms," in *2016 IEEE 2nd Int. Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI)*, Bologna, Italy, pp. 1–6, 2016.

[11] A. Taylor, N. Japkowicz and S. Leblanc, “Frequency-based anomaly detection for the automotive CAN bus,” in *2015 World Congress on Industrial Control Systems Security (WCICSS)*, London, UK, pp. 45–49, 2015.

[12] H. J. Jo and W. Choi, “A survey of attacks on controller area networks and corresponding countermeasures,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6123–6141, 2021.

[13] R. Islam, R. U. D. Refat, S. M. Yerram and H. Malik, “Graph-based intrusion detection system for controller area networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 1727–1736, 2020.

[14] K. He, X. Zhang, S. Ren and J. Sun, “Deep residual learning for image recognition,” in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, USA, pp. 770–778, 2016.

[15] H. M. Song, J. Woo and H. K. Kim, “In-vehicle network intrusion detection using deep convolutional neural network,” *Vehicular Communications*, vol. 21, no. 100198, 2020.

[16] 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.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley *et al.,* “Generative adversarial networks,” *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[18] E. Seo, H. M. Song and H. K. Kim, “GIDS: GAN based intrusion detection system for in-vehicle network,” in *2018 16th Annual Conf. on Privacy, Security and Trust (PST)*, Belfast, Ireland, pp. 1–6, 2018.

[19] F. Amato, L. Coppolino, F. Mercaldo, R. Nardone and A. Santone, “CAN-bus attack detection with deep learning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5081–5090, 2021.

[20] A. R. Javed, S. Ur Rehman, M. U. Khan, M. Alazab and T. Reddy, “CANintelliIDS: Detecting in-vehicle intrusion attacks on a CAN using CNN and attention-based GRU,” *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1456–1466, 2021.