Hindawi
Security and Communication Networks
Volume 2022, Article ID 4322148, 11 pages
https://doi.org/10.1155/2022/4322148

Research Article

Vehicular Multilevel Data Arrangement-Based Intrusion Detection System for In-Vehicle CAN

Wansoo Kim,[1] **Jungho Lee**,[2] **Yousik Lee**,[3] **Yoenjin Kim**,[4] **Jingyun Chung**,[4] **and Samuel Woo**[1]

[1]*Department of Software Science, Dankook University, Yongin 16890, Republic of Korea*
[2]*HYUNDAI AutoEver, Seoul, Republic of Korea*
[3]*ETAS Korea, Seongnam 13494, Republic of Korea*
[4]*Division of Electronic Engineering, IT Convergence Research Center, Jeonbuk National University, Jeonju 54896, Republic of Korea*

Correspondence should be addressed to Samuel Woo; samuelwoo@dankook.ac.kr

Received 11 November 2021; Revised 14 December 2021; Accepted 18 December 2021; Published 20 January 2022

Academic Editor: Ilsun You

Modern vehicles are equipped with various types of electrical/electronic (E/E) systems. Electronic control units (ECUs) are used to control various E/E systems in the vehicle. For efficient information exchange between ECUs, most vehicle manufacturers use the Controller Area Network (CAN) protocol. However, CAN has security vulnerabilities because it does not have an authentication or encryption method. Since attacks on in-vehicle networks affect the safety of drivers, it is essential to develop a technology to prevent attacks. The intrusion detection system (IDS) is one of the best ways to enhance network security. Unlike the traditional IDS for network security, IDS for the in-vehicle network requires a lightweight algorithm because of the limitation of the computing power of in-vehicle ECUs. In this paper, we propose a lightweight IDS algorithm for in-vehicle CAN based on the degree of change between successive data frames. In particular, the proposed method minimizes the load on the ECU by using the CAN data frame compression algorithm based on exclusive-OR operations as a tool for calculating the degree of change.

1. Introduction

Modern vehicles are converging with advanced information and communication technology (ICT) to effectively cope with automotive emission regulations while providing users with a comfortable driving experience. This automotive-ICT convergence fusion is a new paradigm for developing next-generation automobiles [1, 2]. With the development of automobile-ICT convergence, various types of electrical/electronic (E/E) systems have begun to be installed in automobiles. The automotive E/E system consists of one or more sensors, actuators, and electronic control units (ECUs). The key element is the ECU [3]. Initially, ECU was introduced in automobiles to precisely control the engine's core functions. However, in recent years, the ECUs have been expanded to improve vehicle performance and efficiently control various safety devices and amenities [4]. ECUs mounted inside the car have steadily increased in demand since their inception in the 1980s [5]. Various communication methods such as Controller Area Network (CAN), Local Interconnect Network (LIN), Media Oriented System Transport (MOST), and FlexRay were developed for efficient communication between ECUs as the number of ECUs mounted inside cars increased [6]. In particular, the CAN protocol was developed by BOSCH to support bus network topologies as the most representative of automotive in-vehicle network technologies [7].

The CAN protocol, established in 1993 as the "ISO 11898" standard, is an identity-based broadcast communication protocol as a serial communication technique that supports real-time communication [8, 9]. Bus-based CAN is used in many industries such as automobiles, robots, factory automation, airplanes, and medical devices with noise-resistant properties and low cost. However, CAN is vulnerable to cyberattacks because no authentication or

encryption technique is applied. Because CAN operates in a multimaster manner, an attacker can read all signals sent and received on the connected bus through the approached node and can also collect and retransmit signals from other nodes [10]. Also, frequent Denial of Service attacks and Radio Frequency Jamming attacks that have often appeared in traditional IT environments can also be carried out against cars [11, 12]. In recent years, the intelligent transportation system has evolved into an Internet of Things (IoT) device where cars communicate with nearby vehicles or objects [13, 14]. As cars were recognized as IoT devices, various cyberattacks that have occurred in traditional ICT environments were transferred to the automotive environment.

In 2010, Koscher et al. [15] conducted an attack on a real car that forced the control of the ECU. In 2015, Miller and Valasek [16] published their findings on hacking connected cars at BlackHat, a global hacking conference. These findings show that cars can be targeted by cyberattacks. Automotive hacking studies conducted over the past decade point to CAN's vulnerabilities as the biggest problem.

As CAN's security concerns arise, various studies are being conducted to address them. There are studies that add authentication or encryption protocols [17]. Some of the proposed security techniques include the use of Identity-Based Encryption (IBE) [18]. However, applying these techniques requires modifying the CAN protocol of the ECU installed on all vehicles already on the market, which can be costly. Also, cryptographic techniques such as IBE are not suitable for applications to ECUs that own low-performance microcontroller units.

Other studies for automotive in-vehicle network security include the automotive intrusion detection system (IDS), which detects abnormal data frames in automotive in-vehicle networks. Automotive IDS analyzes the data frames sent and received from the in-vehicle network to determine whether they are in a normal or abnormal state and detects data frames that are maliciously injected due to vehicle hacking. Furthermore, automotive IDS has the advantage of not changing the system in the current automotive in-vehicle network structure.

Various types of automotive IDS techniques have been proposed. There are techniques that monitor the contents and the periodicity of the data frame. ECUs installed in the vehicle use a unique cycle to transmit data frames [19]. When attackers perform a replay or an impersonation attack, the ECU's data frame transfer cycle changes. These characteristics allow us to detect malicious activity when an attack is carried out. However, it is not advisable to use only the data frame transfer cycle as a parameter for IDS, because if a remote frame occurs, the data frame transfer cycle may change. Several techniques utilize the properties of electric signals on the physical layer. ECUs communicate with each other via CAN transceivers. CAN transceivers of the same model have subtle differences when generating analog signals. This characteristic allows us to identify the ECUs mounted in the vehicle. That is, it can detect the data frame transmitted by the malicious module in addition to the normal ECU. These IDS techniques require hardware devices such as oscilloscopes to be installed to analyze analog signals. Adding expensive hardware devices to vehicles creates problems that increase the price of the vehicle.

This study proposes the IDS technique that can be mounted on low-performance ECUs. Our proposed IDS technique uses CAN data compression algorithms based on exclusive-OR operations to detect replay and impersonating attacks. The IDS technique we propose uses only exclusive-OR operations; therefore, it can be mounted on low-performance ECUs and does not use expensive hardware. Also, if a remote frame occurs and the data transfer cycle changes, there would be no problem performing the IDS function because it does not affect the data payload change.

The main contributions of this study are as follows:

(1) Designing and developing Vehicular Multilevel Data Arrangement (V-MLDA) algorithms: we have upgraded the Multilevel Data Arrangement (MLDA) technique to use it for IDS for in-vehicle CAN.

(2) V-MLDA-based IDS (V-M-IDS) design and development: we designed a V-M-IDS that uses V-MLDA to detect replay attacks and impossible attacks performed on in-vehicle CAN. A V-M-IDS algorithm design is mounted on Arduino.

(3) V-M-IDS Performance Assessment: we conducted a three-step performance evaluation experiment.

  (i) Step 1: verify the V-M-IDS algorithm using CAN data set obtained from real vehicles
  (ii) Step 2: evaluate V-M-IDS performance using embedded devices
  (iii) Step 3: evaluate V-M-IDS performance based on automotive security living lab

This paper is organized as follows. In Section 2, we introduce the background of our work. In Section 3, we describe the related works. Section 4 presents the details of the proposed vehicular-MLDA-based IDS (V-M-IDS) technique. In Section 5, we describe the performance analysis of V-M-IDS.

2. Background

2.1. Automotive Electrical/Electronic System. The automotive electrical/electronic (E/E) system consists of one or more sensors, actuators, and ECUs. Typical automotive E/E system includes powertrain, chassis, body, and infotainment. E/E systems such as powertrains constitute subnetworks. The gateway ECU performs router functions among physically separated subnetworks. Figure 1 shows the in-vehicle subnetwork. ECUs in the E/E system use communication protocols such as CAN, LIN, FlexRay, MOST, and Ethernet to send and receive data to and from each other. Dual CAN is the most common automotive network protocol. CAN is used by powertrains and chassis subsystems that require real-time data processing.

2.2. CAN. The CAN protocol is a broadcast communication technique based on sender ID that supports bus network topology. By supporting the bus network topology, CAN

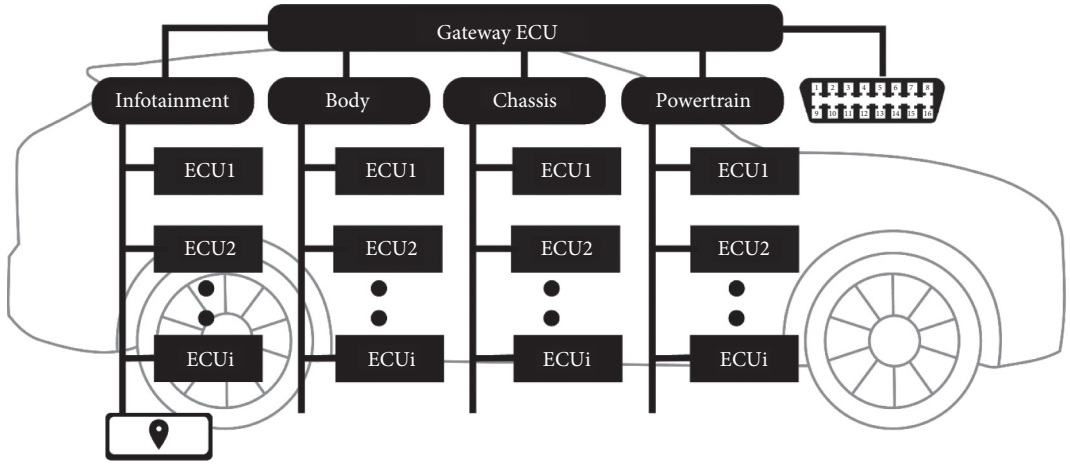

Figure 1: In-vehicle network environment.

dramatically reduces the complexity and length of communication lines between ECUs in the vehicle. For this reason, the automotive industry quickly introduced the CAN bus system. In 1993, it was established as the international standard of ISO (ISO11898). The CAN bus system is divided into two modes, standard CAN 2.0A (11-bit ID) and extended CAN 2.0B (29-bit ID) according to the length of the ID of the data frame.

The data frame structures of standard CAN 2.0A and extended CAN 2.0B are shown in Figure 2. Extended CAN 2.0B has an identifier of 29 bits. Unlike CAN 2.0A, CAN 2.0B has two ID fields. The IDE field separates the two ID fields. The Data Length Code (DLC) indicates the number of bytes in the data field. The data field contains data to be transmitted to other nodes and can use up to 8 bytes.

2.3. CAN Compression Algorithm. CAN data compression makes use of the fact that successive CAN data frames with the same ID do not change rapidly. ECUs use unique IDs to transmit data frames. By observing the data frame being transmitted periodically using a specific ID, it was confirmed that the CAN data field variation in the $n-1$st transmitted data frame and the nth transmitted data frame was not significant. Using this characteristic, CAN compression algorithms were developed. The CAN compression algorithms developed to date have been classified into two main techniques.

The first technique is a compression technique that uses the predicted maximum measurement value (PMDV) between CAN data fields. The data compression ratio of the PMDV algorithm showed improved performance based on the accuracy of the maximum value prediction of the variation. Algorithms using PMDV had adaptive data reduction (ADR)/improved adaptive data reduction (IADR) [20, 21], enhanced data reduction (EDR) [22], and boundary of fifteen compression (BFC) algorithms [23].

The ADR/IADR algorithm transmits compressed data if the difference value of the continuous data frame is above the PMDV value, and if the PMDV value is above, the original data is transmitted. The difference between ADR and IADR algorithms is the number of IDs. ADR algorithms use two data frame IDs to distinguish between the compression data ID and the uncompressed ID. IADR algorithms represent compressed data if the first bit of the data field is one, and zero represents the original data [20, 21] to use only one data frame ID.

The EDR algorithm separates signals into two forms. Signals with a data length of five bits or more are called SDN type (signal, delta, no-change), and signals with data lengths shorter than five bits are called SN type (signal, no-change). In EDR algorithms, the first byte of a data field is used as data compression code (DCC). In DCC, zero means that the data is not compressed, and one means delta compression or it is fully compressed [22].

The BFC algorithm selects PMDV as ±15. Signals below five bits are defined as nonboundary of fifteen (NBF) signals, and signals above six bits are defined as boundary of fifteen (BF) signals. Each NBF signal is assigned a bit parameter compress (BPC) bit. The BPC is one when the NBF signal is compressed, and the BPC is zero when uncompressed. BF signals are assigned a parameter compression of two bits. It is a BF signal that is not compressed when the PC bit is 00, the BF signal is fully compressed when the PC bit is 01, the variation is positive BF signal when the PC bit is 10, and the variation is negative when the PC bit is 11 [23].

The second technique is a compression technique that uses a compression area selection (CAS) map [24–27]. By using a CAS map, we avoid predicting the maximum value of the change. Also, the [25–27] algorithm avoids the use of sign bits by using bitwise XOR operation when obtaining the variation between the old and current data. Section 3 describes the use of CAS maps in detail.

3. Related Works

3.1. MLDA CAN Data Compression Algorithm. We proposed a MLDA CAN data compression algorithm to effectively reduce the amount of transmitted data [26]. In MLDA, the compression efficiency varies depending on how the CAN data is grouped and arranged in the 64-bit data field. Determination of the CAS map arrangement order to maximize

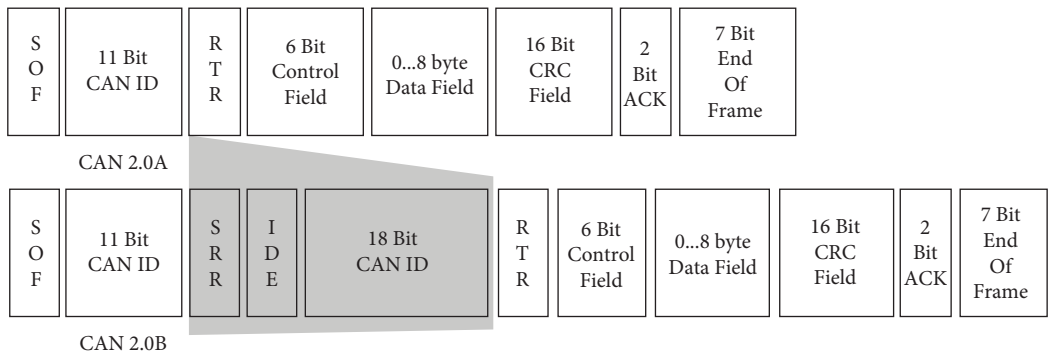

Figure 2: The structure of the CAN data frame.

the compression ratio is a necessary process in the early stage of the system, and data are transmitted/received according to the determined arrangement during system operation.

To determine the CAS map arrangement order in the MLDA algorithm, 64 bits of the CAN data field are grouped into 24-bit Sig A, Sig B, and 16-bit Sig C. When grouping CAN data, the high-order bits should consist of data with a small variation and the low-order bits should consist of data with a large variation to improve the compression ratio. When grouping three signals, it is possible to group signals in units of 8 bits, units of 4 bits, units of 2 bits, and units of 1 bit by analyzing the variation of 64 bits of the CAN data field. For example, when arranging in units of 8 bits, 64 bits of the CAN data field are expressed as $B(n)$ $(0 \le n \le 7)$. The XORed values of the previous and the current data frames are defined as a magnitude value (MV). The frequency value (FV) is calculated by MV. If MV is not 0, FV is 1; if MV is 0, FV is 0. $S_m(n)$ is the sum of all the nth columns of the MV matrix, and $S_f(n)$ is the sum of all the nth columns of the FV matrix.

Figure 3 shows a byte-level data arrangement map. $S_{fm}(n)$ is a value that determines the arrangement position, and $B(n)$ is arranged in $(u0 \longrightarrow m0 \longrightarrow d0 \longrightarrow u1 \longrightarrow m1 \longrightarrow d1 \longrightarrow u2 \longrightarrow m2)$ in the order in which the value of $S_{fm}(n)$ is large. The value of $S_{fm}(n)$ is defined as follows:

$$S_{fm}(n) = S_f(n) + \lambda \times S_m(n). \quad (1)$$

λ is the weight factor between $S_f(n)$ and $S_m(n)$. Optimized λ is chosen from 0 to 3 by simulation. The 4-bit unit, 2-bit unit, and 1-bit unit arrays are extensions of the 8-bit unit array. The scope of $B(n)$ is expanded and the process is conducted in the same way.

After dividing the CAN data into three signals, the XOR value between the previous data frame and the current data frame of each signal is calculated. If the calculated XOR value of a signal is 0, the corresponding header bit is set to 0. If the calculated XOR value of a signal is not 0, the corresponding header bit is set to 1. When composing the memory map of the CAN data field, the transmitting ECU places the 3 header bits in the least significant bit as shown in Table 1. The XOR values of the signal whose header bit is 1 are alternately arranged from the least significant. The length of the signal transmitted is the maximum value of the length of signals A, B, and C, starting at MSB, with bits having a value of 0

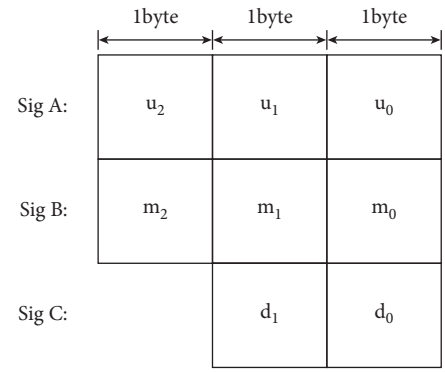

Figure 3: Byte-level data arrangement map.

Table 1: Memory map of a CAN data field.

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit2	Bit0
Byte 0	SB [0]	SA [1]	SC [0]	SB [0]	SA [0]	HC [1]	HB [1]	HA [1]
Byte 1	SA [1]	SC [1]	SB [1]	SA [1]	SC [0]	SB [1]	SA [1]	SC [1]
Byte 2	0	0	0	0	0	0	SC [0]	SB [0]

removed until the first 1 appears. The receiving ECU places the header bit and the received XOR value in the CAS map. The receiving ECU restores the current data by calculating the XOR value between the received XOR value and the previous data.

3.2. Automotive IDS. Automotive IDS detects normal or attack CAN data frames by analyzing various patterns and characteristics generated on the CAN bus. That is, Automotive IDS establishes a training or normal model for system behavior. The IDS then compresses the current system's activity with previously captured normal models to detect changes in behavior and classifies deviations above a certain range as abnormal.

In [28], the authors proposed a technique for detecting an attack and identifying the ECU using clock skew (timing error) that reflects the hardware characteristics of the clock source constituting the ECU. Even ECUs that transmit data

frames in the same period have different unique clock skews due to the characteristics of the hardware clock source. Therefore, when the clock skew of the CAN data frame fluctuates more than a specific value, it is detected as an attack. They also proposed VIDEN (voltage-based attacker identification) based on the characteristic that the CAN signals generated by ECUs are different due to the difference in voltage supplied to each ECU [29]. However, in the case of a technique that utilizes hardware characteristics, there is a limitation that it may be affected by the internal and external temperature of the vehicle or the driving environment. Also, there is a problem that an additional is required to analyze analog signals.

In [30], the authors proposed a technique to detect attacks by calculating the entropy of CAN data frames transmitted to the CAN bus. In an attack situation where a large number of specific CAN data frames are injected, the attack is detected based on the characteristic that the entropy value of the corresponding CAN data frames increases. The authors also showed limitations for the recognition of small-scale attacks which could be part of the normal vehicle or user behavior. In [19], the authors proposed a technique to check the period of the CAN data frame transmitted in the CAN bus. Using the characteristic that CAN data frames are transmitted at regular intervals in a normal state, data frames with a shorter or longer period than the normal period were detected as an attack. In the case of a technique that utilizes the data frame generation period, it may be greatly affected by the remote frame. When a remote frame occurs, the data frame transmission period may be changed.

4. Proposed IDS Technique

This article suggests the IDS technique that is mounted on low-performance ECUs. The IDS technique we proposed detects attacks by measuring the degree of change in the data payload in the CAN data frame that the ECUs transmitted. MLDA technique is used to quickly calculate the degree of change in data payloads, even in low-performance ECUs.

4.1. Attack Model. Figure 4 represents the attack model in this paper. The threat agent, Node B, is capable of sniffing data frames sent by other ECUs. Node A is a normal ECU. In this paper, it was assumed that the data frames transmitted by Node A were used to control the vehicle. Node B then sniffed the data frames transmitted by Node A and used them for retransmission attacks. When Node B performs the retransmission attack, it uses the same cycle as the data frame transmission cycle of Node A.

4.2. MLDA CAN Data Compression Algorithm. The MLDA method is an efficient compression algorithm that uses only exclusive-OR operations. However, the MLDA technique cannot be used in vehicle environments. The MLDA technique does not transfer data frames when the data payloads of the n-1st data frame and the nth data frame are the same; i.e., if the compression ratio is 100%, it does not transmit the data frame. However, in an automotive environment, CAN data frames must be transmitted at a fixed interval. We designed the vehicular-MLDA (V-MLDA) technique, which upgraded the basic MLDA technique to apply the MLDA technique to the vehicle environment.

The MLDA algorithm does not transmit data when the variation between the previous and current data is zero. This can reduce the busload. However, if the data do not change for a long time, the receiving ECU does not know if it is disconnected from the transmitting ECU. To solve this problem, a vehicle multilevel data arrangement algorithm is proposed. In the algorithm, when the variation between the previous and current data is zero, the transmitting ECU transmits one-byte data filled with zero bits. Since $x \oplus 0 = x$, the receiving ECU restores the current data through the previous and received data. Through this, the busload increases, but the receiving ECU periodically receives the transmitting ECU data.

4.3. V-MLDA-Based IDS Technique. The ECUs that make up the automotive E/E system configure the network and transmit the information they collect to other ECUs at specific intervals. In general, the CAN protocol is used for network configuration. Therefore, ECUs periodically broadcast CAN data frames to the in-vehicle network using the data frames defined in the CAN protocol. The transmission cycle is between 10 ms and hundreds of milliseconds depending on the importance. For example, ECUs belonging to the powertrain or chassis system (the most important electronic control system in automobiles) transmit the state information they collect to surrounding ECUs every 10 ms.

Considering these characteristics of ECUs participating in the CAN network, we assume that the normal and attack states of the data frame transmitted by the ECU are as follows.

First, the degree of data change between successive data frames is small. Figure 5 illustrates the speed change of a car in seconds and milliseconds. The figure on the left shows the speed change of the car in seconds, and the figure on the right shows it in milliseconds. If an ECU transmits the vehicle's current speed data in a 10 ms cycle, then we will have the image on the left. In the circle in the left picture, the car's speed increases by 10 km/h for 6 seconds. During that time, the ECU divides these changes into 600 data frames and transmits them. Therefore, the difference between successive data frames is about 0.016 km/h; i.e., the data changes in the previous and current frames do not differ significantly. In other words, if the degree of change in the data frame has more variation than the normal data frame, the data frame is determined as an attack.

Second, the car's state does not change in milliseconds, nor does it change by successive data frames. The ECU transmits the state information to the data in the CAN data frame when the car's state changed. The changes in the vehicle's state caused by those in the user's behavior or surroundings are maintained for a period; i.e., a data frame that reflected the vehicle's state change is transferred; a data frame that remained in the same state for a period is transmitted. In other words, if the car's state is changed by successive data frames, the frame can be determined as an attack.

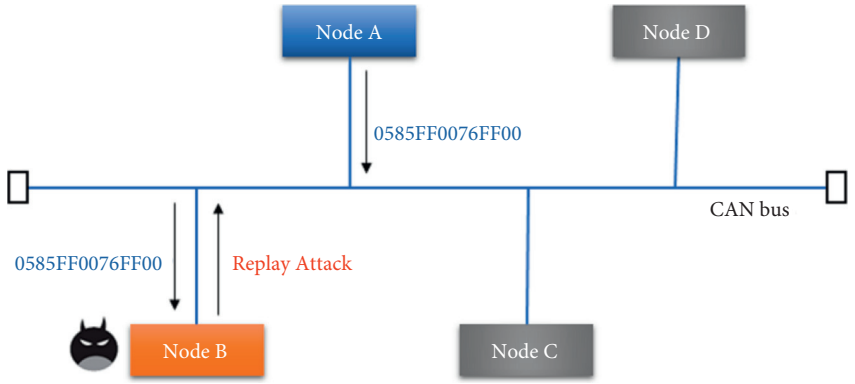

Figure 4: Attack model.

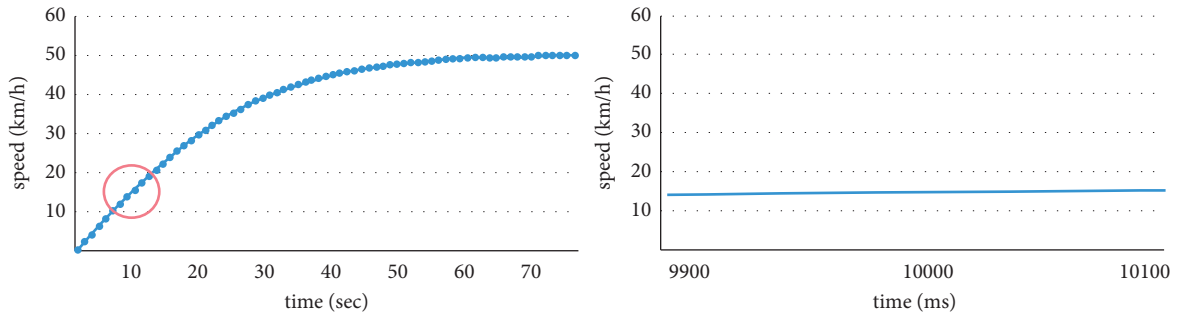

Figure 5: Car speed change graph in seconds and milliseconds.

Third, a replay attack is done by injecting multiple data frames in succession. A replay attack is an attack that captures a normal data frame and sends it back at the time of the attack. In a normal state, the change between data frames is small. Therefore, it is common for an attacker to intensively inject large amounts of attack data to cause a catastrophic malfunction; i.e., a data frame with a large degree of change in the data frame in a normal state may appear once or twice in a row. However, if the data frame with a large degree of change in the data frame appears more than three times in a row, the data frame is determined as an attack.

The V-M-IDS technique detects intrusion into the automotive in-vehicle network by calculating the degree of change between the data that the ECU periodically transmitted based on assumptions about normal and abnormal states. In the foregoing, it was assumed that there is less change between successive data frames in the normal state. Therefore, if the degree of change is below the threshold by calculating the degree of change between the current and the previous data frames, it is considered normal. If the degree of change exceeds the threshold, an attack is assumed. Second, it is assumed that the car's state does not change by successive data frames. Therefore, if the degree of change is below the threshold, the degree of change in the data frame sent immediately after the data frame exceeding the threshold is considered normal. However, if the degree of change in the data frame sent immediately after exceeded the threshold, an attack is assumed. Third, a replay attack is assumed to inject multiple data frames in succession. Therefore, if a data frame that exceeds the threshold in succession appears no more than three times, it was considered normal. However, if it exceeds the threshold more than three times, it is determined that an attack data frame was injected and detects that an intrusion has occurred in the network. Equation (2) shows how the degree of change in the data frame is calculated:

$$\text{degree of change} = \left(\frac{\#\text{ of bits of compressed data}}{\#\text{ of bits of original data}}\right) \times 100. \tag{2}$$

The algorithm of the proposed technique has been shown in Algorithm 1.

Algorithm 1 uses three parameters. The first one is the CAS map data arrangement order for MLDA compression. The second parameter is a threshold value that characterizes the degree of the calculated variation as normal or abnormal. Variations greater than the threshold are considered abnormal. The last parameter is the maximum acceptable limit of abnormal data frames. An attack is detected when the number of abnormal data frames exceeds the maximum acceptable limit, whereby the attack notification function (notify_attack()) is called, which sends an attack detection data frame (ID 0x700: Data payload FF FF FF FF FF FF FF FF).

```
(1) Input:
      cas_map: the data arrangement order for MLDA compression.
      threshold: maximum value of the degree of variation in normal state.
      limit_changes: maximum acceptable limit of abnormal data frames in normal state.
(2) Initialize:
      prev_msg ← 0
      attack_cnt ← 0
(3) while message with ID arrives do
(4)   xor_msg ← prev_msg^message
(5)   comp_msg ← compress_with_v_mlda(xor_msg, cas_map)
(6)   degree ← (bit_length(comp_msg)/bit_length(message)) * 100.
(7)   if degree > threshold then
(8)     attack_cnt ← attack_cnt +1
(9)   else
(10)    attack_cnt ← 0
(11)  if attack_cnt > limit_changes then
(12)    notify_attack()
(13)  prev_msg ← message
```

Algorithm 1: Algorithm for intrusion detection.

5. Experimental Results

To analyze the performance of the proposed V-M-IDS technique, we performed a three-step performance evaluation.

(i) Step 1: verify the V-M-IDS algorithm using the CAN data set obtained from a real vehicle

(ii) Step 2: evaluate V-M-IDS performance using embedded devices

(iii) Step 3: evaluate V-M-IDS performance based on automotive security living lab [w14]

5.1. Step 1. We used a CAN data set obtained from a real vehicle to validate the V-M-IDS algorithm. The V-M-IDS algorithm was implemented in C. The V-M-IDS algorithm was applied to the data set to detect the attack, and all data frames were determined as normal. As for V-M-IDS implementation, the entire data were scanned and the CAS map was configured; then, the MLDA compression technique was applied to calculate the degree of change. The threshold for determining whether an attack was used was 50. Table 2 shows the degree and count of changes in the 199,531 data frames of the ECU, which was the ID 0x123 of the monitoring data. Table 2 confirms that 663 data frames showed a change that exceeded the threshold of 50 but was determined to be a normal state change.

5.2. Step 2. Next, the performance evaluation of V-M-IDS was performed using embedded devices. To this end, V-M-IDS was mounted on Arduino. After implementing an ECU emulator and attack device in the lab, it was confirmed that it detected attacks in the situation of performing CAN communication.

The V-M-IDS mounted on Arduino set the threshold to 50; the same was verified by Step 1. The first 3,000 data frames were used to configure the CAS map and to perform intrusion detection from the 3,001st data frame. When an intrusion was detected, the ID 0x700 was implemented to transmit the attack detection data frame (FF FFFFFFFFFFFFFF) to the ECU. The ECU emulator and attack device were mounted on the NUCLEO board. The ECU emulator was implemented to sequentially transmit eight-byte data frames 00 FF 00 00 D0 11 11 11, 00 FF 00 00 D0 22 22 22, . . ., 00 FF 00 00 D0 FFFFFF in a cycle of 10 ms with ID 0x390. With a click of the black button on the board, the attack device was made to inject eight-byte attack data frames (FF FFFFFFFFFFF FF) 10 times in a cycle of 10 ms with the same ID as the ECU emulator, 0x390. The CAN

Table 2: Degree of change of actual driving data (ID 0x123 199,531 data frames).

Degree of change	Count	Ratio (%)
4.6875	26,161	13.10
6.25	405	0.20
7.8125	18,356	9.20
9.375	30,070	15.10
10.9375	38,322	19.20
12.5	32,509	16.30
14.0625	21,636	10.80
15.625	11,016	5.50
17.1875	7,780	3.90
20.3125	2,672	1.30
23.4375	2,017	1.00
26.5625	1,465	0.70
29.6875	6,412	3.20
32.8125	15	0.00
35.9375	9	0.00
39.0625	3	0.00
42.1875	20	0.00
51.5625	6	0.00
54.6875	651	0.30
57.8125	1	0.00
64.0625	3	0.00
79.6875	2	0.00

Table 3: Embedded devices, H/W and S/W.

No.	Type	Model name	Note
1	HW1	Arduino Uno	V-M-IDS
2	HW2	NUCLEO-F303RE	ECU emulator
3	HW3	NUCLEO-F303RE	Attack device
4	HW4	PCAN-USB	CAN to USB device
5	SW1	PCAN-Explorer 5	CAN packet sniffing

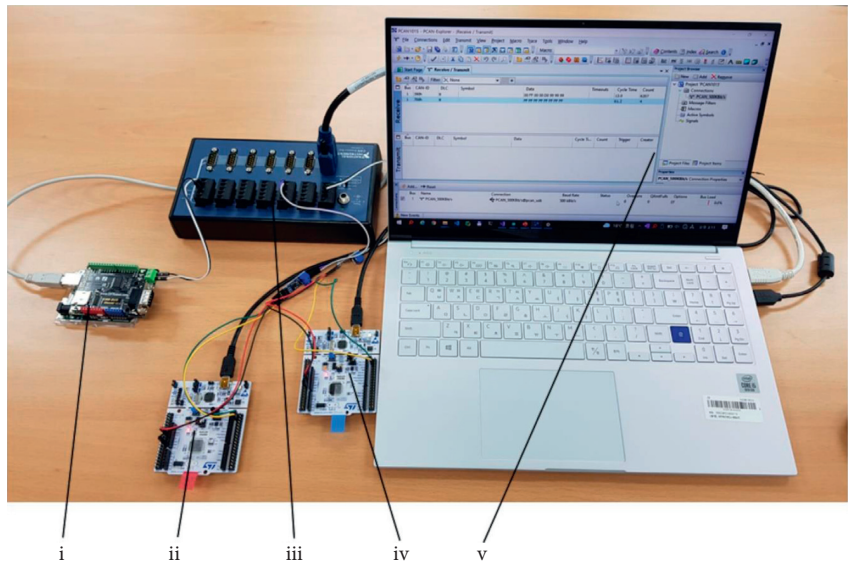

Figure 6: Performance evaluation environment of embedded devices: (i) V-M-IDS device, (ii) attack device, (iii) PCAN-USB, (iv) ECU emulator, and (v) PCAN-Explorer.

network was monitored using PCAN-Explorer. Table 3 and Figure 6 show the configuration of V-M-IDS, ECU emulator, and attack device.

Using embedded devices, Figure 7 shows the performance evaluation results of V-M-IDS. There was no detection of the normal state data frame of the ECU emulator as an attack. The attack was injected at the 101.9925 time. The attack was detected at the 102.0116 time, when there occurred data frames with a degree of change higher than 50 four times in a row. Detecting attacks could be determined using the V-M-IDS implemented on embedded devices and transmitting the attack detection data frame to the ID 0x700 ECU at the 101.0127 time.

5.3. Step 3. Finally, the automotive security living lab performed a V-M-IDS performance evaluation by connecting to the CAN communication of the actual vehicle. An automotive security living lab was established by Korea Internet & Security Agency (KISA) to create a safe use environment through security internalization by preventing and responding to security threats from the development of autonomous vehicles and autonomous driving services. Table 4 shows the main functions of the automotive security living lab.

Figure 8 shows the performance evaluation environment of the automotive security living lab.

The performance evaluation was performed as follows:

(1) Connect the V-M-IDS devices built in Step 2 to the experimental ENVIRONMENT CAN network

(2) Build a CAS map using 3,000 data frames in normal autonomous driving situations

(3) Ensure that V-M-IDS detects normal autonomous driving data frames as attacks

(4) Verify detection after conducting security threats to operate automotive security living lab handles

V-M-IDS was implemented to monitor the data frame of the handle operation ECU (ID 0x390). Embedded devices implemented by V-M-IDS were connected to the CAN network in addition to the CAN interface. Attacks on the CAN network and monitoring the CAN network were performed using the network simulator CANoe. Table 5 shows the automotive security living lab performance evaluation results. During normal autonomous driving, it did not occur to detect normal state data frames as attacks. The attack was injected at the time 705.517106. Detection of the attack occurred at the time 705.573966, the fourth consecutive time of data frames with a change of more than 50. Detecting an attack could be determined using the V-M-IDS that transmitted an attack detection data frame to the ID 0x700 ECU at the time 705.573966.

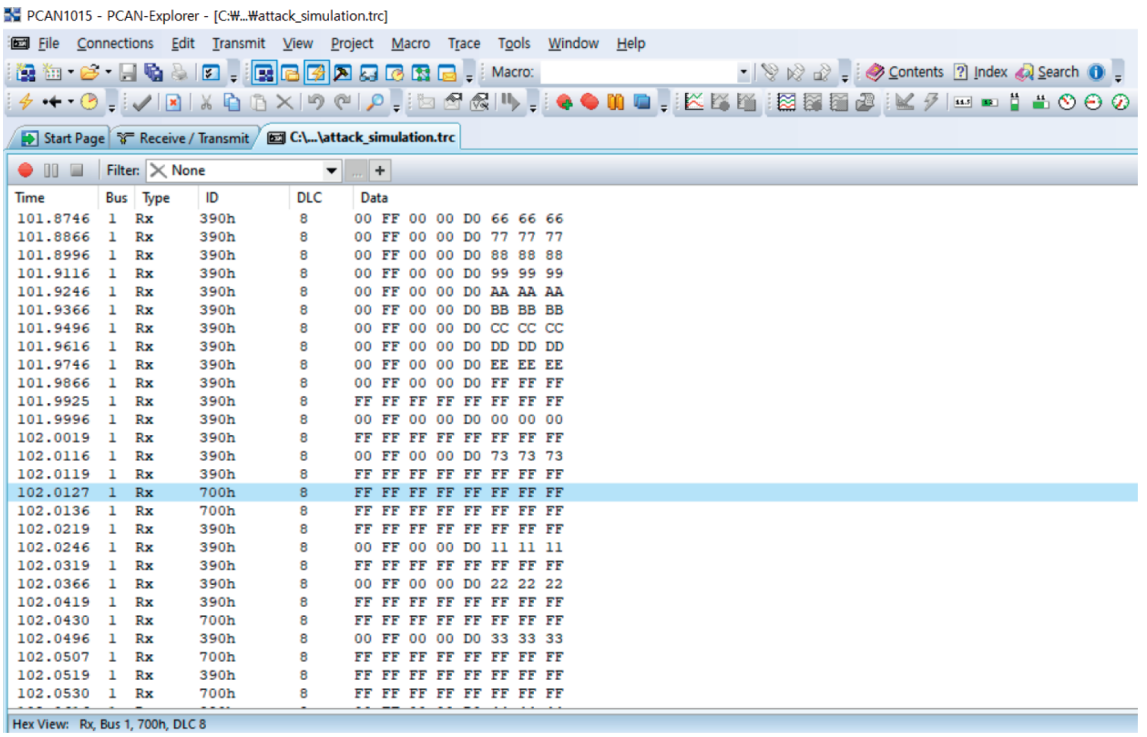

Figure 7: Performance evaluation results of V-M-IDS using embedded devices (PCAN-Explorer).

Table 4: Main functions of automotive security living lab.

Function	Description
Subject to inspection	• Core devices such as automotive ECUs and infotainment (IVI)
Check items	• Controls: service refusal, data frame retransmission, purge test, camouflage attack, etc. • Infotainment: mock hooking, Wi-Fi/Bluetooth purge test, firmware modulation, etc.
Security threat demonstration	• Derailed by handle operation by retransmission of data frame to in-vehicle network+ • Denial of infotainment services through Wi-Fi vulnerabilities

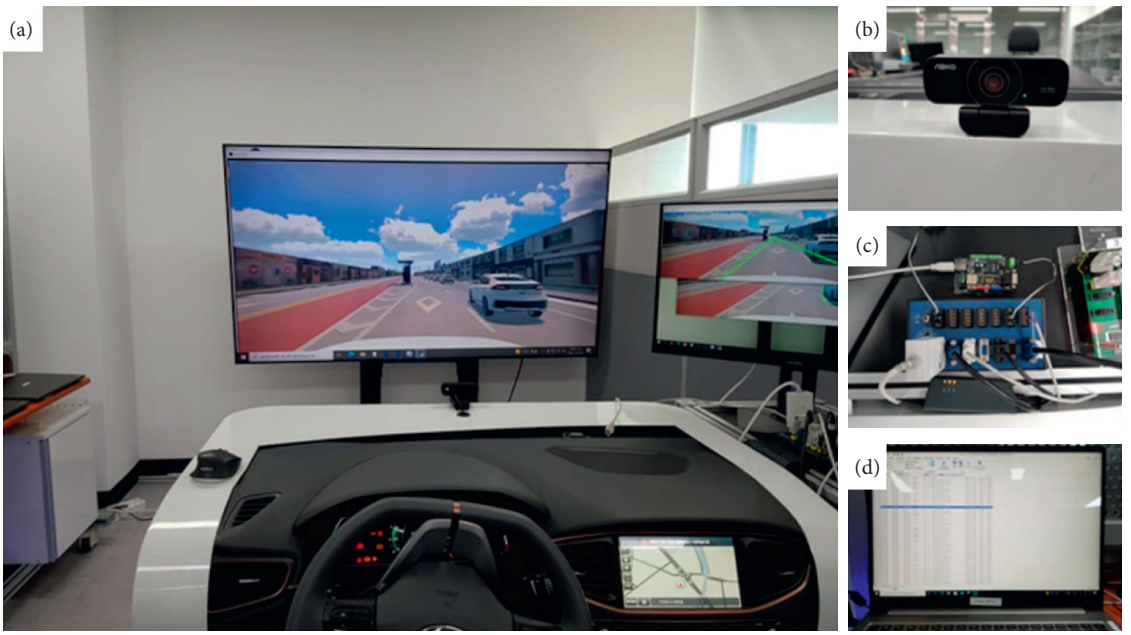

Figure 8: Performance evaluation environment of automotive security living lab: (a) lab car, (b) optical camera, (c) V-M-IDS device and PCAN-USB, and (d) CANoe 1.0.

Table 5: A portion of the output of the CAN data trace block used for CANoe in Figure 8.

ID	Received time (sec)	Data field
⋮	⋮	⋮
390	705.448265	05 82 FF 00 74 FA 00
390	705.479123	05 85 FF 00 75 FE 00
390	705.511216	05 85 FF 00 76 FF 00
390	705.517106	05 AF 00 00 C7 7B 00
390	705.542943	05 84 FF 00 77 FF 00
390	705.549248	05 AF 00 00 C8 7C 00
390	705.573966	05 84 FF 00 78 00 00
700	705.575136	FF FFFFFFFFFFFFF
390	705.581223	05 AF 00 00 C9 7D 00
700	705.582701	FF FFFFFFFFFFFFF
390	705.605282	05 87 FF 00 79 04 00
⋮	⋮	⋮

6. Conclusions

This study proposed an automotive IDS technique using CAN data frame compression algorithms. Performance evaluation of the proposed technique demonstrated that V-M-IDS could be mounted on low-spec ECUs. Three phases of experimentation were conducted for realistic performance evaluation, and final performance verification was completed on automotive security living labs to execute experiments in the same environment as real-world vehicles [31]. If the V-M-IDS technique could be integrated with the existing IDS techniques and applied to in-vehicle CAN, it would increase the detection rate of the existing IDS solutions.

Two additional future studies are needed for the V-M-IDS technique to be applied to the actual vehicle environment. First, in a vehicle environment, what is as important as the detection of cyberattacks is the postdetection response phase [32]. In the future, we plan to conduct further research to bring vehicles into a safe state based on the risk of attacks detected with V-M-IDS.

Second, if an automotive security living lab based on a digital twin [33] was developed, the initial learning process of the V-M-IDS technique would be simplified. We plan to design an automotive security living lab based on a digital twin [33] for the initial learning process of the V-M-IDS technique.

Data Availability

The data used in this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The present research was supported by the research fund of Dankook University in 2019.

References

[1] A. Saad and U. Weinmann, "Automotive software engineering and concepts," in *GI Jahrestagung*, pp. 318-319, GI, Frankfurt, Germany, October 2003.

[2] E. Nickel, "IBM automotive software foundry," in *Proceedings of the Press Conference on Computer Science in Automotive Industry*, Frankfurt University, Frankfurt, Germany, September 2003.

[3] M. Wolf, A. Weimerskirch, and C. Paar, "Security in automotive bus systems," in *Proceedings of the Workshop on Embedded IT-Security in Cars*, Bochum, Germany, November 2004.

[4] B. K. Ramesh and S. Murthy, "In Vehicle Networking," 2004, http://www.sae.org/technical/papers/20%2004-28-0029.

[5] R. Charette, "This Car Runs on Code," 2009, http://www.spectrum.ieee.org/feb09/7649,%20Feb.%202009.

[6] T. Nolte, H. Hansson, and L. L. Bello, "Automotive communications-past, current and future," in *Proceedings of the 2005 ETFA (Emerging Technologies and Factory Automation)*, Catania, Italy, September 2005.

[7] K. H. Johansson, M. Törngren, and L. Nielsen, "Vehicle applications of controller area network," in *Handbook of Networked and Embedded Control Systems*, D. Hristu-Varsakelis and W. S. Levine, Eds., Springer, 2005.

[8] Cia CAN, "CAN in Automation," 2004, http://www.can-cia.org.

[9] Bosch CAN, "CAN IP modules," 2004, http://www.can.bosch.com.

[10] M. Bozdal, M. Samie, and I. Jennions, "A Survey on CAN Bus Protocol: Attacks, Challenges, and Potential Solutions," in *Proceedings of the 2018 International Conference on Computing Electronics & Communications Engineering (ICCECE)*, pp. 201–205, University of Essex, Southend, UK, August 2018.

[11] A. Abhishta, W. van Heeswijk, M. Junger, L. J. Nieuwenhuis, and R. Joosten, "Why would we get attacked? An analysis of attacker's aims behind DDoS attacks," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 11, no. 2, pp. 3–22, 2020.

[12] G. Kasturi, A. Jain, and J. Singh, "Detection and classification of radio frequency jamming attacks using machine learning," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 11, no. 4, pp. 49–62, 2020.

[13] S. Manipriya, C. Mala, and S. Mathew, "A collaborative framework for traffic information in vehicular adhoc network applications," *Journal of Internet Services and Information Security (JISIS)*, vol. 10, no. 3, pp. 93–109, 2020.

[14] M. Alizadeh, K. Andersson, and O. Schel´en, "A survey of secure internet of things in relation to blockchain," *Journal of Internet Services and Information Security (JISIS)*, vol. 10, no. 3, pp. 47–75, 2020.

[15] K. Koscher, S. Savage, F. Roesner et al., "Experimental security analysis of a modern automobile," in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, vol. 447–462, May 2010.

[16] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," 2015, http://illmatics.com/Remote%20Car%20Hacking.pdf.

[17] S. Woo and S.-B. Lee, "Usage techniques of a truncated message authentication code for in-vehicle controller area network," *The Journal of the Institute of Internet, Broadcasting and Communication*, vol. 17, no. 6, pp. 127–135, 2017.

[18] D. H. Duong, W. Susilo, and V. C. Trinh, "Wildcarded identity-based encryption with constant-size ciphertext and secret key," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 11, no. 2, pp. 74–86, 2020.

[19] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *Proceedings of the 2016 International Conference on Information Networking (ICOIN)*, pp. 63–68, Kota Kinabalu, Malaysia, Jan 2016.

[20] P. R. Ramteke and S. M. Mahmud, "An adaptive data-reduction protocol for the future in-vehicle networks," *SAE Technical Paper Series*, vol. 114, no. 7, pp. 1540–1554, 2005.

[21] R. Miucic and S. M. Mahmud, "An improved adaptive data reduction protocol for in-vehicle networks," *SAE Technical Paper Series*, vol. 115, no. 7, pp. 650–658, 2006.

[22] R. Miucic, S. M. Mahmud, and Z. Popovic, "An enhanced data-reduction algorithm for event-triggered networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 6, pp. 2663–2678, 2009.

[23] S. Kelkar and R. Kamal, "Boundary of Fifteen Compression Algorithm for Controller Are Network Based Automotive Applications," in *Proceedings of the International Conference on Circuits, Systems, Communication and Information Technology Applications*, pp. 162–167, CSCITA), Mumbai, India, Apr. 2014.

[24] Y.-j. Wu and J.-G. Chung, "Efficient controller area network data compression for automobile applications," *Frontiers of Information Technology & Electronic Engineering*, vol. 16, no. 1, pp. 70–78, 2015.

[25] Y. Wu and J.-G. Chung, "An improved controller area network data-reduction algorithm for in-vehicle networks," *IEICE - Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E100.A, no. 2, pp. 346–352, 2017.

[26] Y.-J. Kim, Y. Zou, Y.-E. Kim, and J.-G. Chung, "Multi-level data arrangement algorithm for can data compression," *International Journal of Automotive Technology*, vol. 21, no. 6, pp. 1527–1537, 2020.

[27] Y.-J. Kim, S. Woo, and J.-G. Chung, "Triple ID flexible MAC for CAN security improvement," *IEEE Access*, vol. 9, no. 1, Article ID 126399, 2021.

[28] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *Proceedings of the 25th USENIX Security Symposium*, pp. 911–927, Austin, TX, USA, August 2016.

[29] K.-T. Cho and K. G. Shin, "Viden: Attacker identification on in-vehicle networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1109–1123, ACM, Dallas, Texas, USA, October 2017.

[30] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1110–1115, IEEE, Baden-Baden, Germany, June 2011.

[31] KISA, "Self-Driving Vehicle Security Reinforcement," 2014, https://www.kisa.or.kr/business/security/enhancement_sub1.jsp.

[32] O. V. Baranov, N. V. Smirnov, T. E. Smirnova, and Y. V. Zholobov, "Design of a quadrocopter with pid-controlled fail-safe algorithm," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 11, no. 2, pp. 23–33, 2020.

[33] P. Angin, M. H. Anisi, F. G¨oksel, C. G¨ursoy, and A. B. uy ¨ukg¨ulc¨u, "Agrilora: A digital twin framework for smart agriculture," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 11, no. 4, pp. 77–96, 2020.