information

Article

Single-Frame-Based Data Compression for CAN Security

Shi-Yi Jin [1], Dong-Hyun Seo [2], Yeon-Jin Kim [1], Yong-Eun Kim [3], Samuel Woo [4,*] and Jin-Gyun Chung [1,*]

[1] Division of Electronic Engineering, IT Convergence Research Center, Jeonbuk National University, Jeonju 54896, Republic of Korea; jinshiyi@jbnu.ac.kr (S.-Y.J.); genie@jbnu.ac.kr (Y.-J.K.)
[2] Green Mobility R&D Center, Jeonbuk Institute of Automotive Convergence Technology, Gunsan 573450, Republic of Korea; dhseo@jiat.re.kr
[3] Vehicle Electrification R&D Center, Korea Automotive Technology Institute, Cheonan 31471, Republic of Korea; kimye@katech.re.kr
[4] Division of Software Science, Dankook University, Yongin 16890, Republic of Korea
* Correspondence: samuelwoo@dankook.ac.kr (S.W.); jgchung@jbnu.ac.kr (J.-G.C.)

Abstract: To authenticate a controller area network (CAN) data frame, a message authentication code (MAC) must be sent along with the CAN frame, but there is no space reserved for the MAC in the CAN frame. Recently, difference-based compression (DBC) algorithms have been used to create a space inside the frame. DBC has the advantage of being very efficient, but its drawback is that, if an error occurs in one frame, the effects of that error propagate to subsequent frames. In this paper, a CAN data compression algorithm is proposed that compresses the current frame without relying on previous frames. Therefore, an error generated in one frame cannot be propagated to subsequent frames. In addition, a CAN signal grouping technique is proposed based on entropy analysis. To efficiently authenticate CAN frames, the length of the compressed data must be 4 bytes or less (4BL). Simulation shows that the 4BL-compression ratio of a Kia Sorento vehicle is 99.36% in the DBC method, but 100% in the proposed method. In an LS Mtron tractor, the 4BL-compression ratio is 98.58% in the DBC method, but 100% in the proposed method. In addition, the execution time of the proposed compression algorithm is only 27.39% of that of the DBC algorithm. The results show that the proposed algorithm has better compression characteristics for CAN security than the DBC algorithms.

Keywords: CAN; data reduction; entropy; security; signal grouping

check for updates

Citation: Jin, S.-Y.; Seo, D.-H.; Kim, Y.-J.; Kim, Y.-E.; Woo, S.; Chung, J.-G. Single-Frame-Based Data Compression for CAN Security. *Information* **2024**, *15*, 132. https://doi.org/10.3390/info15030132

Academic Editor: Valentina Casola

Received: 7 February 2024
Revised: 24 February 2024
Accepted: 27 February 2024
Published: 28 February 2024

1. Introduction

Although the controller area network (CAN) system has established itself as the most widely used automotive network, CAN does not ensure the confidentiality and authentication of the CAN data frame, opening the way for easy data eavesdropping or the launching of replay attacks [1,2]. The CAN attack surfaces can be categorized into physical-access-based attacks [3–5] and wireless-access-based attacks [6–9]. Wireless-access-based attacks are further classified into two types: (1) attacks that require initial physical access [6,7], and (2) attacks without physical access, exploiting Bluetooth, Wi-Fi, or cellular channels [8,9].

Jo et al. [1] categorize the countermeasures against CAN attacks into four types: (1) authentication of data frames [10–15], (2) preventative protection [16–18], (3) intrusion detection [19,20], and (4) post-protection [21,22]. Authentication of data frames can be an efficient way to improve CAN security, but there is no room for message authentication codes (MACs) in CAN frames. Moreover, the size of the data field of one CAN frame is limited to 8 bytes.

In [10], a truncated 32-bit MAC is inserted inside the data fields in the CAN frames. If the data payload is used to transmit a truncated 32-bit MAC, the total amount of CAN data frame transmission increases at least twice. Therefore, due to the increased data frame overhead, an increase in the bus load follows. In [11], if an electronic control unit (ECU) uses a data field of less than five bytes, the truncated Mini-MAC data can be inserted inside the data field. However, if an ECU uses a data field of five or more bytes, sufficient Mini-MAC data cannot be transmitted.

Data compression techniques for CAN have been developed to reduce the increased bus load caused by the growing number of ECUs. As an application example, Oh et al. propose a CAN/Ethernet gateway system for seamless communication with CAN data compression techniques [23].

Recently, data compression algorithms have been used to create a space to include a MAC inside a data field. In one work, a CAN data compression algorithm is used to reduce the size of the data so that there is space for a 32-bit or larger MAC within the data field. Therefore, all CAN frames are authenticated without changing the original CAN protocol.

In [24], a selective message authentication (SMA) method is proposed specifically for safety-critical CAN frames. SMA can reduce the communication overhead on the CAN bus, but the delay due to the transmission of additional data packets is an unavoidable drawback.

The MAuth-CAN algorithm is resistant to masquerading attacks [25]. The MAuth-CAN algorithm neither uses the full capacity of the CAN network nor requires hardware modifications to the CAN controller. However, the increase in latency still needs to be considered.

A data field in a CAN data frame can hold information about vehicle parameters such as speed and frictional torque. Most CAN data compression algorithms use the difference between the current and previous CAN parameters with the same message ID [26–31]. The reason for this is that the difference in successive CAN parameters is usually very small since the CAN signal generation rate is much higher than the change in the driving environment. The CAN data compression algorithms can be classified into two types: (1) predefined maximum difference value (PMDV)-based algorithms and (2) compression area selection (CAS) map-based algorithms.

In PMDV-based compression algorithms, if the difference between the current and previous frames does not exceed a predefined threshold value (i.e., PMDV), it is used for transmission. If the difference value exceeds the PMDV, the original uncompressed message is sent. Therefore, the choice of PMDV greatly affects the compression efficiency. If the PMDV is too small, many frames are transmitted uncompressed. If it is too large, more bits are needed to represent the compressed data. The adaptive data reduction (ADR) algorithm introduces delta compression (the difference between the current value and the previous value of a parameter). In the ADR algorithm [27], two message IDs are used to distinguish between the compressed data ID and the uncompressed data ID. On the other hand, the improved adaptive data reduction (IADR) algorithm uses a single ID [28]. The first bit of the data field in a CAN message, the data reduction code (DRC), determines if data reduction is used or not. The IADR algorithm allocates two bits per parameter for indicating full compression (zero difference), delta compression, or no compression. The enhanced data reduction (EDR) algorithm [29] uses a method for managing signals of shorter lengths (i.e., <5 bits) by combining them into groups that are handled as single signals. The EDR algorithm uses the first byte of the data field to indicate the compression statuses of the parameters. In the boundary of fifteen compression (BFC) algorithm [30], a CAN signal can be compressed if the absolute value of the difference is within the maximum compression range of fifteen. The BFC algorithm efficiently places one or two compression status bits preceding each parameter. Thus, the BFC algorithm is less computationally intensive than the EDR because it does not use the first byte of the data field to indicate the compression statuses of all the parameters. A performance comparison of PMDV-based algorithms is presented in [31].

In CAS map-based algorithms such as multi-level data arrangement (MLDA) and vehicle MLDA (VMLDA), a CAS map is used to eliminate the need to predict a PMDV, as described in Section 3. In PMDV-based algorithms, the PMDV remains fixed once a PMDV is established during the system design phase. However, in CAS map-based algorithms, the size of the CAS map is adjusted for each CAN frame according to changes in driving conditions. Thus, CAS map-based algorithms achieve higher compression efficiencies than PMDV-based algorithms.

In CAN data compression algorithms, a high compression ratio is obtained by utilizing the difference between two consecutive CAN data frames. However, if a data error, frame loss, replay attack, or ECU reset occurs during the communication process, there is a possibility that all compressed messages after the error may not be properly recovered because the differences between successive frames cannot be correctly calculated. Therefore, it is necessary to develop a compression method that does not propagate the effects of errors to subsequent frames.

In this paper, a new CAN data compression algorithm is proposed that compresses the current frame without calculating the difference between the current message and the previous message. Therefore, the effect of the error does not propagate to the next frame because the difference value is not calculated. In addition, an entropy-based signal grouping method is proposed to easily create space for MAC data in the data field. The proposed entropy-based signal grouping method can be used as a new technique for allocating CAN parameters for each ID to increase the security of the CAN system. Combining the proposed compression method with entropy analysis, it is shown that the proposed algorithm can be used efficiently for data authentication.

The main contributions of this work are as follows:

- By proposing a CAN data compression method using only a single frame, unlike other compression methods, it is possible to prevent the propagation of errors in the CAN system to subsequent frames.
- An entropy-based signal grouping method is proposed to easily create space for MAC data in the CAN data field. Using the proposed entropy-based signal grouping method, the latency for data authentication can be significantly reduced.
- The proposed entropy-based signal grouping method can be used as a new technique for allocating CAN parameters for each ID to increase the security of the CAN system.
- The mapping table of the proposed algorithm is created only once during system design. When driving a car, compression is performed using the prepared mapping table. However, in difference-based compression (DBC) methods, a CAS map must be created every time a frame is sent. Therefore, the actual execution time of the proposed algorithm during online operation is only about 27.39% of that of the CAS map-based algorithm.

In Section 2, hacking and security solutions for In-Vehicle CAN are reviewed. In Section 3, CAS map-based compression algorithms are briefly reviewed. In Section 4, a single-frame (SF)-based compression algorithm and an entropy-based signal grouping method are proposed. Section 5 presents the simulation results using actual CAN data. Section 6 describes the security analysis of the proposed solution. Finally, the conclusions are summarized in Section 7.

2. Related Work

2.1. Research on Vehicular Hacking

With the development of automotive ICT convergence, instances of hacking into an automotive electrical/electronic (E/E) system have increased. This section presents the analysis results of the existing research on automotive E/E system hacking. Vehicular hacking research may be classified into three fields:

(a) Hacking into a vehicle's E/E system (ECU forced actuation attack);
(b) Hacking into a vehicle's smart key (smart key copy);
(c) Hacking into a vehicle's sensor (sensor malfunction).

The research field of (a) introduces techniques for hacking vehicles using the vulnerabilities of In-Vehicle CAN. This section describes the characteristics of research area (a). Based on this, the attack surfaces of In-Vehicle CAN are defined.

In 2010, K. Koscher et al. presented the results of hacking experiments using a real vehicle [13]. Based on their work, various vehicular hacking studies have been conducted, the most representative of which are listed in Table 1.

Table 1. Representative hacker research.

No.	Publication	Wired/Wireless	Research Content	Vulnerability
1	K. Koscher et al. [13]	Wired	Conducted the first hacking experiment using a real car; first published an analysis of vulnerabilities of In-Vehicle CAN	In-Vehicle CAN
2	S. Checkoway et al. [4]	Wireless	Proposed a wireless attack model targeting a vehicle; conducted a wireless attack experiment using a real car	In-Vehicle CAN
3	C. Valasek et al. [3]	Wired	Researched based on Study 1; disclosed In-Vehicle CAN hacking methods in detail	Wireless interface
4	C. Valasek et al. [9]	Wireless	Succeeded in wireless attack targeting a Jeep Cherokee; introduced the most practical hacking technique	In-Vehicle CAN
5	S. Woo et al. [15]	Wireless	Research based on Study 1; succeeded in wireless attack using the driver's smartphone	In-Vehicle CAN

Every study specified in Table 1 conducted vehicular hacking using the vulnerability of In-Vehicle CAN. In the present study, the attack surfaces of In-Vehicle CAN based on previous studies are classified into three types. Figure 1 shows the attack surfaces that can be caused by the vulnerability of CAN:

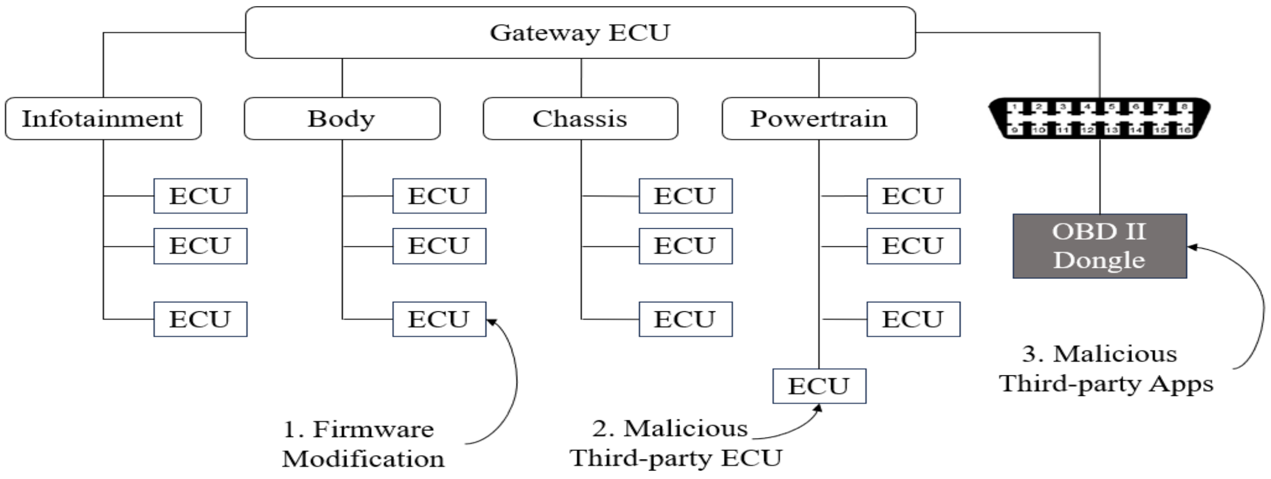


Figure 1. The attack surfaces that can be caused by the vulnerability of CAN.

- Firmware modification of a certain device composing the automotive E/E system [3,9,13];
- Fabrication and installation of a malicious third-party ECU [3,13];
- Wired/wireless access via OBD2 terminal [3,4,13,15].

2.2. *Research on Security Solutions for In-Vehicle CAN*

While research related to vehicle hacking is being actively conducted, so are studies on CAN security in vehicles. The existing studies published in the last 10 years have research themes as follows:

- Message authentication code (MAC);
- Intrusion detection system (IDS);
- Moving target defense (MTD).

IDS, MTD, and MAC have different purposes and functions. This subsection presents previous studies related to MAC. Section 6 describes the results of the comparative evaluation between the studies in this subsection and the proposed method:

- Truncated MAC [15]: Woo et al. proposed a method using a 32-bit truncated MAC for CAN data frame authentication. They used an Extended ID field and a CRC field to transmit the 32-bit truncated MAC. Their proposed data frame authentication method does not generate an additional data frame, so the bus load does not increase. However, the CAN standard must be modified to use their proposed method.
- Mini-MAC [11]: Jackson et al. proposed a method for CAN data frame authentication using a truncated MAC. Their proposed data frame authentication method uses a portion of the data field for MAC transmission. They suggested a method where the unused portion of the 8-byte data field is employed for MAC transmission. They analyzed CAN data frames generated during the driving of a Toyota Prius. Their analytic findings showed that approximately 40% of the total data frames in the analysis used a greater-than-4-byte data field. Hence, their proposed method increases the amount of data transmitted and can increase the bus load.
- SecOC [14]: In AUTOSAR (Automotive Open System Architecture), SecOC (Secure Onboard Communication) was suggested for the security of the In-Vehicle Network. In SecOC, a truncated MAC and counter are used to construct a secure communication environment. SecOC has the same problem as Mini-MAC.

A CAN data frame can only transmit 8 bytes of data at a time. The precedent studies on CAN data frame authentication recommend using a MAC of 32 bits or more for data frame authentication in In-Vehicle CAN [11,15]. They use an additional data frame or modify the CAN standard to transmit the MAC of 32 bits or more. Using an additional data frame is not easy in an automotive environment due to the increased latency. Modification of the CAN standard cannot be applied to existing vehicles. In addition, the data frame authentication methods for In-Vehicle CAN must support real-time authentication.

3. CAS-Map-Based Compression Algorithm and Data Authentication

Since CAS map-based compression algorithms outperform PMDV-based algorithms, only CAS map-based algorithms are reviewed in this section. In addition, CAN data authentication techniques are briefly reviewed.

3.1. ICANDR Algorithm

In the ICANDR algorithm, the 8-byte CAN data field is randomly divided into three signals: Sig A (3-byte), Sig B (3-byte), and Sig C (2-byte). Figure 2 is an example of a signal arrangement. The signal arrangement is determined before the installation of the CAN system and remains the same during system operation.

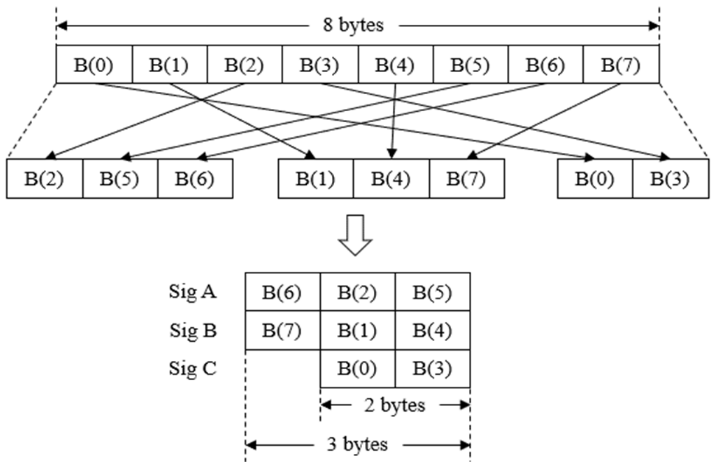


Figure 2. An example of byte-level CAN signal arrangement.

Then, bitwise XOR (Exclusive-OR) values between the current and previous signals are computed for Sig A, Sig B, and Sig C, as shown in the example in Table 2. If the computed XOR value of a signal is nonzero, the corresponding header bit is set to one. Otherwise, the corresponding header bit is set to zero.

Table 2. Example of bitwise XOR and header bit determination.

Signal	Sig A	Sig B	Sig C
Previous frame (F_{i-1})	$1A\ 2B\ CA_{16}$	$9A\ EC\ 96_{16}$	$74\ E3_{16}$
Current frame (F_i)	$1A\ 2B\ A1_{16}$	$9A\ EC\ 92_{16}$	$74\ E3_{16}$
XOR ($F_{i-1} \oplus F_i$)	$00\ 00\ 6B_{16}$	$00\ 00\ 04_{16}$	$00\ 00_{16}$
Header bit	1	1	0

After the determined header bits are placed in the last column beginning in the first row in the CAS map shown in Table 3, the XORed values are placed in bits 23 through 0 for Sig A and Sig B, and bits 15 through 0 for Sig C. If the header bit corresponding to a signal is zero, the row corresponding to the header bit is emptied. Then, starting from the leftmost column of Table 3, if all elements of a column are zero, that column is deleted. This process repeats until the first occurrence of a column with a nonzero element. Then, the remaining data in the shaded area in Table 3 are arranged in the order shown in Table 4. Thus, in this example, 8 bytes of data are compressed into 3 bytes.

Table 3. CAS map for the data in Table 2 (shaded area: selected area).

Signal	Bit23–Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Header (HA)								1
Header (HB)								1
Header (HC)								0
SA ($\text{Sig}A_i \bigoplus \text{Sig}A_{i-1}$)	$0 \cdots 0$	1	1	0	1	0	1	1
SB ($\text{Sig}B_i \bigoplus \text{Sig}B_{i-1}$)	$0 \cdots 0$	0	0	0	0	1	0	0
SC ($\text{Sig}C_i \bigoplus \text{Sig}C_{i-1}$)	-	-	-	-	-	-	-	-

Table 4. Memory map for the CAS map in Table 3.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	SA [2]	SB [1]	SA [1]	SB [0]	SA [0]	HC	HB	HA
Byte 1	SA [6]	SB [5]	SA [5]	SB [4]	SA [4]	SB [3]	SA [3]	SB [2]
Byte 2	0	0	0	0	0	0	0	SB [6]

The amount of compressed data is determined by the size of the selected area. If all three header bits are 0, maximum compression efficiency (100%) is achieved.

3.2. MLDA Algorithm

In addition to the byte-level arrangement in the ICANDR algorithm, the MLDA algorithm presents four-bit-level, two-bit-level, and one-bit-level arrangements. The compression ratio increases gradually as the arrangement bit-level decreases from eight to one. The MLDA algorithm also provides a systematic procedure to place the CAN data bits using multi-level arrangement maps to obtain the highest achievable compression efficiency.

As an example of the byte-level arrangement with three consecutive CAN frames, eight bytes of each data field are denoted as $B(n)(0 \leq n \leq 7)$, as shown in Table 5. The bitwise XORed values between two successive frames are called magnitude values (MVs).

If an MV is nonzero, the corresponding frequency value (FV) is defined as one. Otherwise, it is defined as zero.

Table 5. Example of byte-level arrangement.

Frame		$B(0)$	$B(1)$	$B(2)$	$B(3)$	$B(4)$	$B(5)$	$B(6)$	$B(7)$
F_0		AF	FF	BF	CF	FF	FF	3F	FF
F_1		AD	FD	BC	CB	FA	F9	3E	FF
F_2		AC	FF	BF	CF	FF	FF	3E	FF
$F_0 \oplus F_1$	MV	02	02	03	04	05	06	01	00
	(FV)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(0)
$F_1 \oplus F_2$	MV	01	02	03	04	05	06	00	00
	(FV)	(1)	(1)	(1)	(1)	(1)	(1)	(0)	(0)
$S_m(n)$		3	4	6	8	10	12	1	0
$S_f(n)$		2	2	2	2	2	2	1	0
$S_{fm}(n)$		8	10	14	18	22	26	3	0

For each $B(n)$, $S_m(n)$ is the sum of the MVs and $S_f(n)$ is the sum of the FVs. Then, $S_{fm}(n)$ is defined as follows:

$$S_{fm}(n) = S_f(n) + \alpha \times S_m(n) \tag{1}$$

where α is a weighting factor. The optimal value of α is selected through simulation using actual CAN signals. Typical α values are between 0 and 3.

To reduce the selected area in the CAS map, it is necessary to place slowly changing data in the most significant part of the three signals and frequently changing data in the least significant part. Therefore, since the value of $S_{fm}(n)$ indicates the degree of change in data, $S_{fm}(n)$ can be used to determine the position of each data byte in the arrangement map. Figure 2 shows the byte-level arrangement according to Table 5. The arrangement procedure for smaller bit-levels can be obtained easily by extending the byte-level procedure.

3.3. VMLDA Algorithm

In the MLDA algorithm, the sender ECU does not transmit CAN frames when the difference values between consecutive frames are zero. This procedure reduces the bus load. However, when the difference values between consecutive frames are zero for some time, the receiver ECU cannot be sure whether the transmitter ECU is disconnected or not.

To overcome this problem, if the CAN data is compressed to 0 bytes, the VMLDA algorithm transmits 1 byte of zero data in the data field. This allows the receiver ECU to verify that the transmitter ECU is not disconnected while minimizing the increase in the bus load.

3.4. Error Propagation by Difference-Based Compression

Most CAN data compression algorithms exploit the difference between two consecutive CAN data frames to achieve high compression ratios. However, if a data error, frame loss, replay attack, or ECU reset occurs during the communication process, there is a possibility that all subsequent compressed messages may not be properly recovered.

Figure 3 shows the error propagation process by a replay attack, where the error affects all data received after the replay attack of Δ_3. Therefore, it is necessary to develop a compression method that does not propagate the effects of errors to subsequent frames.

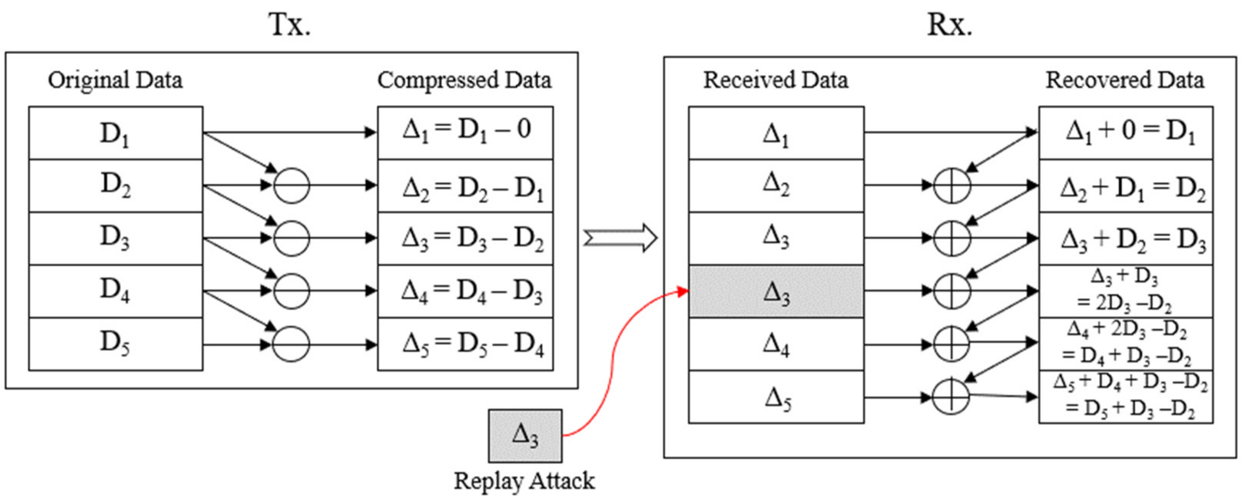


Figure 3. Error propagation by difference-based data compression.

3.5. CAN Data Authentication

Among the CAN data authentication methods introduced in Section 1, only the triple ID flexible MAC (TIFM) approach using VMLDA is reviewed in this subsection. This is because the VMLDA algorithm frees up more space for the MAC compared with other compression methods.

When the CAN data are compressed to less than five bytes, both the compressed data and truncated MAC are transmitted in the same data field using ID_i. Otherwise, the compressed data (or original data) are transmitted using ID_{i+1} and the MAC data are transmitted separately using ID_{i+2}. Therefore, the performance of the data compression is very important in order to reduce the CAN transmission latency and bus load.

When $H_k(\cdot)$ denotes a keyed-hash function using the authentication key k, the truncated MAC tag is computed as follows:

$$\text{Truncated MAC} = Trunc[s, H_k(C_n \parallel CTR_{ECUs} \parallel C_{n-1} \parallel \cdots \parallel C_{n-\lambda})] \tag{2}$$

where the $Trunc[s, \cdot]$ function extracts the s most significant bits of its input, C_n is the current encrypted compressed message, $\{C_{n-1}, \cdots, C_{n-\lambda}\}$ are the previously transmitted λ messages, CTR_{ECUs} is the message counter value of the sender ECU, and $\parallel$ denotes a concatenation operation.

4. SF-Based Compression and Entropy-Based Signal Grouping

In this section, an SF-based CAN data compression algorithm and an entropy-based signal grouping technique are proposed.

4.1. Basic Concept of the Proposed Algorithm

Figure 4 shows the basic concept of the proposed CAN data compression algorithm. When an 8-byte data field in a CAN frame is expressed in the form of an 8×8 matrix, as shown in Figure 4, if all bits in Bytes 4, 5, 6, and 7 are 0, these four bytes can be removed and the data length code (DLC) indicating the length of the data field of a CAN frame is changed from 8 to 4. Thus, the size of the transmitted data is reduced by half.

Based on this observation, the proposed algorithm focuses on the following two points to improve the compression ratio:

- The occurrence of bit 1 should be suppressed if possible.
- Bit 1 should be placed as far as possible from the bottom of the 8×8-bit arrangement matrix.

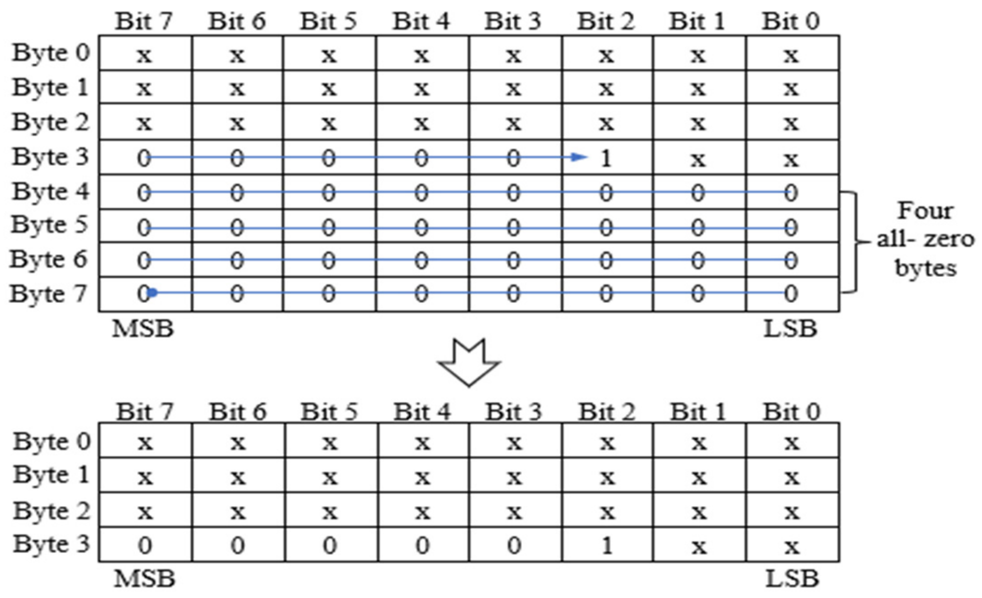


Figure 4. Basic concept of the proposed CAN data compression method.

4.2. Mapping and Inverse Mapping Rules

Mapping and inverse mapping rules need only be determined once during the system design phase. Once the CAN system is put into actual operation, data compression continues according to the determined mapping rule. The determination of the proposed mapping rule consists of the following five steps:

Step 1: Conversion of CAN data field to matrix form

The 8-byte CAN data field can be expressed in the form of an 8 × 8 matrix, as shown in Figure 5. In the matrix, each row number (i) represents a byte number and each column number (j) represents a bit number $(0 \leq i, j \leq 7)$. If the number of CAN frames to be transmitted is N, N matrices are obtained.

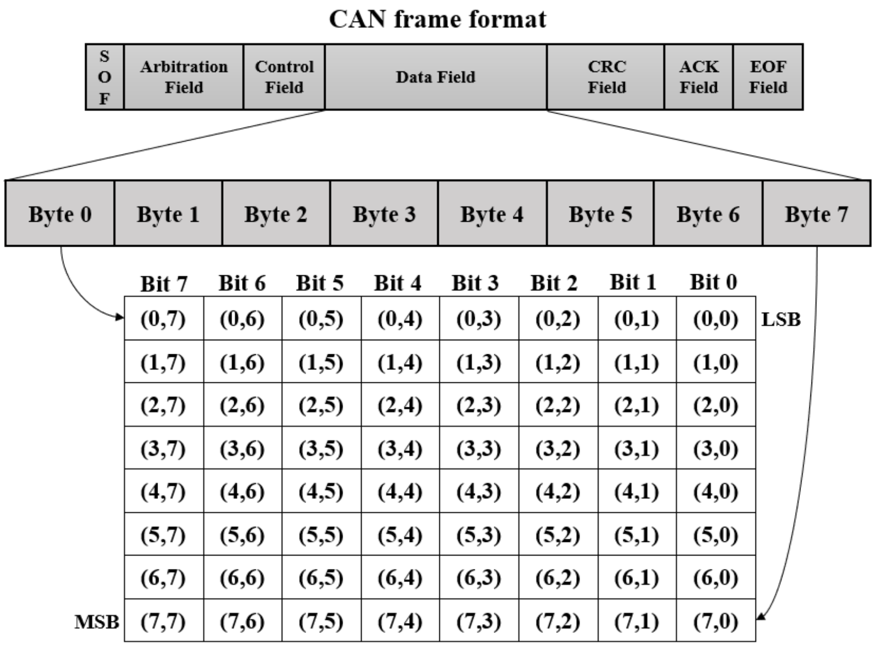


Figure 5. Matrix representation of CAN frame data field.

Step 2: Calculation of the probability of 1 for each bit position

For a sufficient number of CAN frames obtained from actual operation, the probability of the occurrence of 1 for each bit position is calculated. Table 6 is an example showing the probability of the occurrence of 1 for each bit position as a percentage (%). The shaded portions indicate bit positions where the probability of the occurrence of 1 is 50% or more.

Table 6. Probability of occurrence of 1 for each bit position (%).

	b7	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
B0	0	0	0	0	0	0	0	0
B1	0	0	0	0	0	0	0	0
B2	0	23.55	42.09	44.16	41.84	31.99	32.16	95.50
B3	0	0	0	0	0	0	0	100
B4	99.98	99.98	99.99	99.99	99.99	99.99	100	100
B5	0	24.94	36.58	42.10	45.27	47.43	48.06	96.08
B6	0	0	3.86	43.34	44.18	38.08	53.74	96.53
B7	0	0	0	0	0	0	0	0

Step 3: Inversion of bits with a bit 1 probability of 50% or more

To increase the compression ratio, the probability of the occurrence of 1 should be small. To this end, in Table 6, the input bits at the bit positions where the probability of the occurrence of 1 is greater than 50% are inverted. In the decompression process, the inverted bits are inverted again to restore the original data. Table 7 shows the occurrence probability of 1 and the bit inversion position after the bit inversion step. In Table 7, the highest probability of occurrence of 1 is 48.06% at (5,1).

Table 7. Probability of occurrence of 1 for each bit position after inversion step (%) (circled number is the highest probability; shaded area: bit inversion).

	b7	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
B0	0	0	0	0	0	0	0	0
B1	0	0	0	0	0	0	0	0
B2	0	23.55	42.09	44.16	41.84	31.99	32.16	4.50
B3	0	0	0	0	0	0	0	0
B4	0.02	0.02	0.01	0.01	0.01	0.01	0	0
B5	0	24.94	36.58	42.10	45.27	47.43	48.06	3.92
B6	0	0	3.86	43.34	44.18	38.08	46.26	3.47
B7	0	0	0	0	0	0	0	0

Step 4: Data sorting according to the modified probability

The input bits are rearranged in descending order of the modified probability obtained in Step 3. That is, the bit with the highest probability is mapped to (0,0), the bit with the next highest probability is mapped to (0,1), and so on. When the bit mapping to the first row is completed, the mapping to the next row is performed. In the same way, the mapping proceeds to the last row. Bits with the same probability are mapped randomly.

Table 8 shows the occurrence probability of 1 of the rearranged bits. It has a maximum value of 48.06% at the bit position (0,0) and gradually decreases thereafter, showing that all probability values are 0 after the bit position (3,1).

Step 5: Mapping rule determination

The mapping rule summarizes steps 1 to 4 and indicates the new bit position for each input bit and whether the bit is inverted or not. Table 9 shows the mapping rule determined for the probability distribution shown in Table 6.

Table 8. Probability of occurrence of 1 for each bit position after data sorting (%) (circled number is the highest probability; shaded area: bit inversion).

	b7	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
B0	42.10	43.34	44.16	44.18	45.27	46.26	47.43	48.06
B1	23.55	24.94	31.99	32.16	36.58	38.08	41.84	42.09
B2	0.01	0.01	0.02	0.02	3.47	3.86	3.92	4.50
B3	0	0	0	0	0	0	0.01	0.01
B4	0	0	0	0	0	0	0	0
B5	0	0	0	0	0	0	0	0
B6	0	0	0	0	0	0	0	0
B7	0	0	0	0	0	0	0	0

Table 9. Mapping rule for Table 6.

	b7	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
B0	(5,4)	(6,4)	(2,4)	(6,3)	(5,3)	(6,1)	(5,2)	(5,1)
B1	(2,6)	(5,6)	(2,2)	(2,1)	(5,5)	(6,2)	(2,3)	(2,5)
B2	(4,3)	(4,2)	(4,6)	(4,7)	(6,0)	(6,5)	(5,0)	(2,0)
B3	(0,4)	(0,3)	(0,2)	(0,1)	(0,0)	(4,1)	(4,5)	(4,4)
B4	(1,4)	(1,3)	(1,2)	(1,1)	(1,0)	(0,7)	(0,6)	(0,5)
B5	(3,3)	(3,2)	(3,1)	(3,0)	(2,7)	(1,7)	(1,6)	(1,5)
B6	(6,7)	(6,6)	(5,7)	(4,0)	(3,7)	(3,6)	(3,5)	(3,4)
B7	(7,7)	(7,6)	(7,5)	(7,4)	(7,3)	(7,2)	(7,1)	(7,0)

As an example of the mapping rule determination, consider the probability distribution in Table 6. After inversion, the bit with the highest probability in Table 7 is located at (5,1). Therefore, the input bit located at (5,1) is mapped to the position (0,0) in Table 9. Next, the bit with the second highest probability in Table 7 is located at (5,2). Therefore, the bit located at (5,2) is mapped to the position (0,1) in Table 9. The bit with the third highest probability is located at (6,1), which is mapped to the position (0,2) in Table 9 with a shaded area. By the same principle, the mapping rule in Table 9 can be obtained. Algorithm 1 shows the pseudocode expression of the proposed mapping rule determination procedure.

The inverse mapping rule can be created by simply reversing the mapping order. For example, the compressed bit located at (0,0) is mapped to the original position (5,1). The compressed bit at (0,2) is mapped to (6,1) with inversion. Notice that inverted bits must be inverted again at the receiver.

The mapping rule in Table 9 can also be used for inverse mapping by reversing the mapping order. Unlike other compression algorithms, the proposed method is advantageous in increasing the compression ratio by not using header bits.

As mentioned in Step 2, a sufficient number of CAN frames are needed to calculate the probability of the occurrence of 1 for each bit position. For a sufficient number of frames for probability calculation, the moments of normal driving, high-speed driving, low-speed driving, sudden acceleration, and sudden stop of the vehicle must be appropriately included.

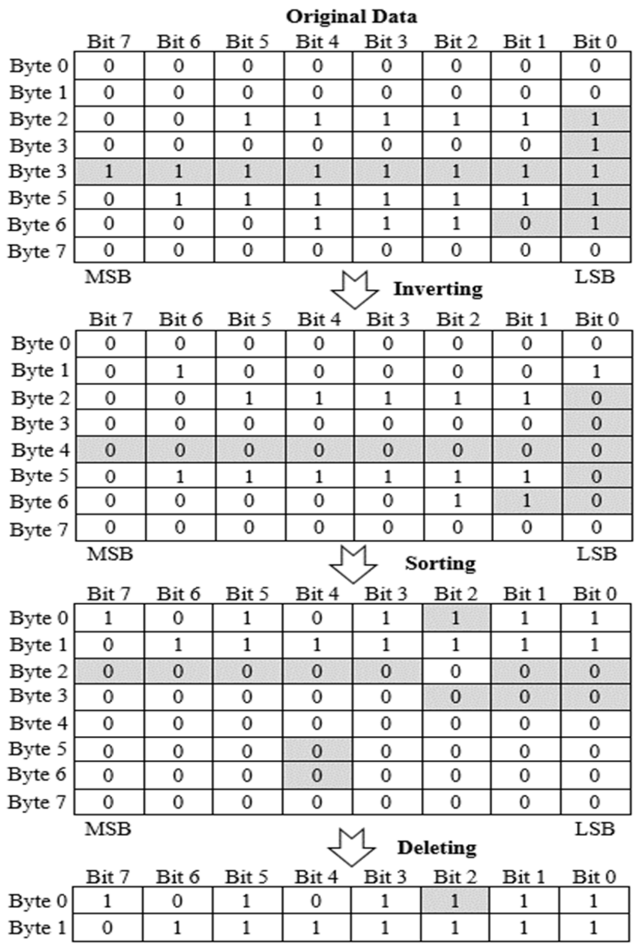
Considering these points, more than 30 min of driving data were collected from a vehicle and a tractor, respectively, for the simulation in Section 5. Since the mapping rule is determined at the time of system design, the number of frames used for probability calculation does not affect the real-time performance of the vehicle.

Algorithm 1 CAN data frame mapping algorithm (0x43F: Frame ID)

```
No.
1    T = Threshold (T)
     [Sufficient number of CAN frames obtained from actual operation]
2    CTR = CAN data frame receive counter
3    0x43F = CAN data frame ID used by the sender ECU
4    **procedure**
5    **initialize** Conversion of CAN data field to matrix form as in Figure 5
6    **while**($CTR_{0x43F} < T$)
7                  receives CAN Data Frame
               **if**(ID = 0x43F) **then** increments $CTR_{0x43F}$ ($CTR_{0x43F} + 1$)
8    **end while**
9    calculation of the probability of 1 for each bit position as in Table 6
10   inversion of bits with a bit 1 probability of 50% or more as in Table 7)
11   data sorting according to the modified probability as in Table 8
12   mapping rule determination as in Table 9
13   End
```

4.3. Operation of the Proposed Compression Algorithm

Once the mapping rule is obtained offline, data compression can be performed in online operation. Figure 6 shows an example of applying the proposed compression algorithm based on the mapping rule in Table 9. Notice that most bits in the shaded area are 1s since the occurrence probabilities of 1 at these locations are very high. It can be seen that the proposed method compresses 8 bytes of input data into 2 bytes.



Original Data

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	0	0	0	0	0	0	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	1	1	1	1	1	1
Byte 3	0	0	0	0	0	0	0	1
Byte 3	1	1	1	1	1	1	1	1
Byte 5	0	1	1	1	1	1	1	1
Byte 6	0	0	0	1	1	1	0	1
Byte 7	0	0	0	0	0	0	0	0

Inverting

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	0	0	0	0	0	0	0	0
Byte 1	0	1	0	0	0	0	0	1
Byte 2	0	0	1	1	1	1	1	0
Byte 3	0	0	0	0	0	0	0	0
Byte 4	0	0	0	0	0	0	0	0
Byte 5	0	1	1	1	1	1	1	0
Byte 6	0	0	0	0	0	1	1	0
Byte 7	0	0	0	0	0	0	0	0

Sorting

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	1	0	1	0	1	1	1	1
Byte 1	0	1	1	1	1	1	1	1
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0
Bvte 4	0	0	0	0	0	0	0	0
Byte 5	0	0	0	0	0	0	0	0
Byte 6	0	0	0	0	0	0	0	0
Byte 7	0	0	0	0	0	0	0	0

Deleting

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	1	0	1	0	1	1	1	1
Byte 1	0	1	1	1	1	1	1	1

Figure 6. An example of applying the proposed compression algorithm.

To send a truncated MAC of 4 bytes in the space obtained by compression, a 4-byte MAC can be added after the 2-byte compressed data, as shown in Figure 7. At this time, the DLC is changed to 6. The receiving side regards the last 4 bytes of the received data as MAC and uses them for authentication. The remaining 2 bytes are treated as compressed data and restored according to the inverse mapping rule.

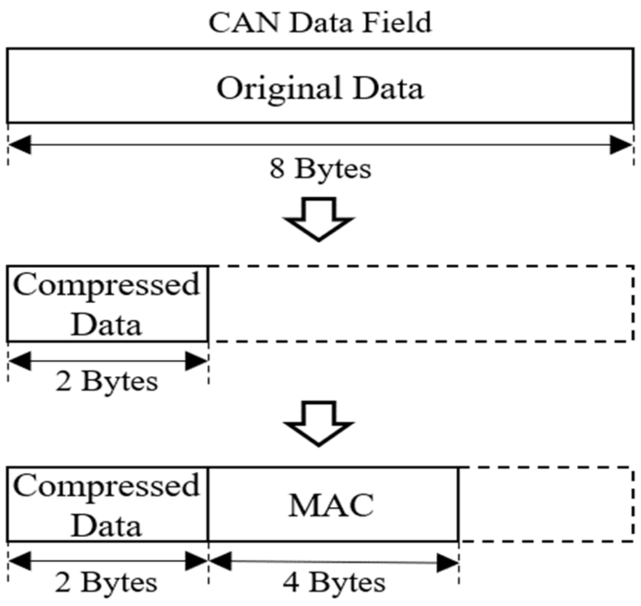


Figure 7. An example of compressed data and a 4-byte MAC transmitted in the same data field of a CAN frame.

4.4. Entropy-Based Signal Grouping

In [10], it is shown that a four-byte MAC provides sufficient security in a vehicular environment. Thus, in the proposed method, a four-byte truncated MAC is transmitted in the space secured by the SF-based compression method.

Assume that the 64-bit data field of a CAN frame is represented as follows:

$$M = b_0 b_1 \cdots b_{63} \tag{3}$$

If p_i is the probability of $b_i = 1$, the entropy of b_i can be calculated as follows [32]:

$$H(b_i) = -p_i log_2 p_i - (1-p_i) log_2 (1-p_i),\ i = 0, \cdots 63 \tag{4}$$

If $p_i = 0.5$, then $H(b_i) = 1$, which means one bit is required to represent b_i. For simplicity, it is assumed that each bit b_i is independent of each other. Then, the message entropy $H(M)$ of 64-bit CAN messages with a specific ID can be calculated as follows:

$$H(M) = \sum_{i=0}^{63} H(b_i) \tag{5}$$

The maximum entropy of a 64-bit CAN message is 64 bits when p_i is 0.5 for $i = 0, \cdots 63$. As p_i approaches 0 or 1, the entropy value decreases.

For CAN frames with a specific ID, the probability p_i for each bit can be calculated from the collected CAN signals. The length of the compressed data is closely related to the entropy of the message. As shown in Section 5, the compressed data length $L(M_C)$ in bytes can be estimated as follows:

$$L(M_C) = \lceil Entropy/8 \rceil \tag{6}$$

where $\lceil t \rceil$ means the smallest integer greater than or equal to t. From (6), it can be seen that the expected length of the compressed data is greater than 4 bytes when the entropy is greater than 32.

To efficiently authenticate a CAN frame, it is desirable to limit the message entropy of an ID to less than 32. Moreover, if possible, it is desirable to equally distribute the total entropy among IDs as shown in Figure 8a. An analysis of automotive CAN signals shows that most CAN messages have an entropy of less than 32. Otherwise, the ID needs to be split into two IDs so that the entropy of each split ID is less than 32, as shown in Figure 8b.

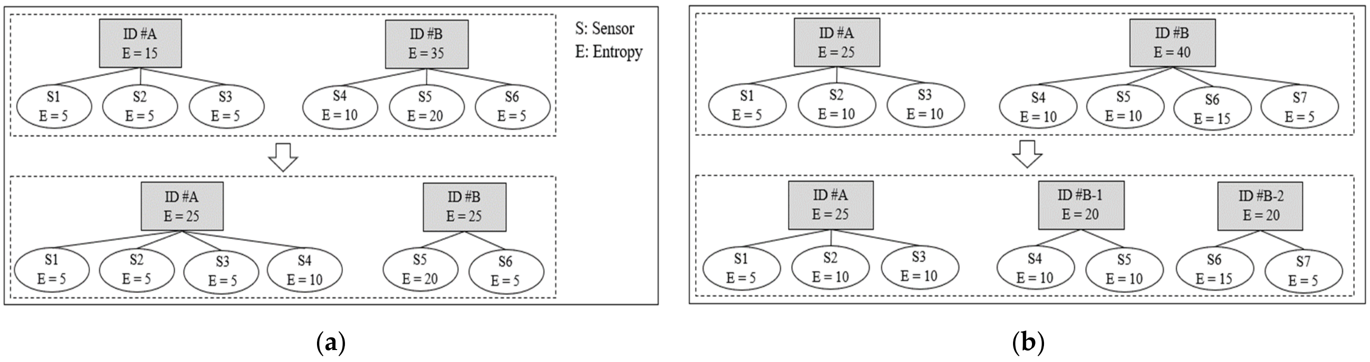


Figure 8. Application examples of signal grouping techniques: (**a**) distribution of the total entropy between IDs; and (**b**) ID split.

By the TIFM approach, two frames are transmitted if the compressed data length is greater than four bytes, as shown in Figure 9a. The first frame is not authenticated until the second frame arrives. However, although the proposed ID split method also transmits two frames when the entropy of an ID is greater than 32, the proposed method has a significant advantage. That is, since each frame contains MAC data, it is authenticated directly without waiting for an additional frame, as explained by Figure 9b.

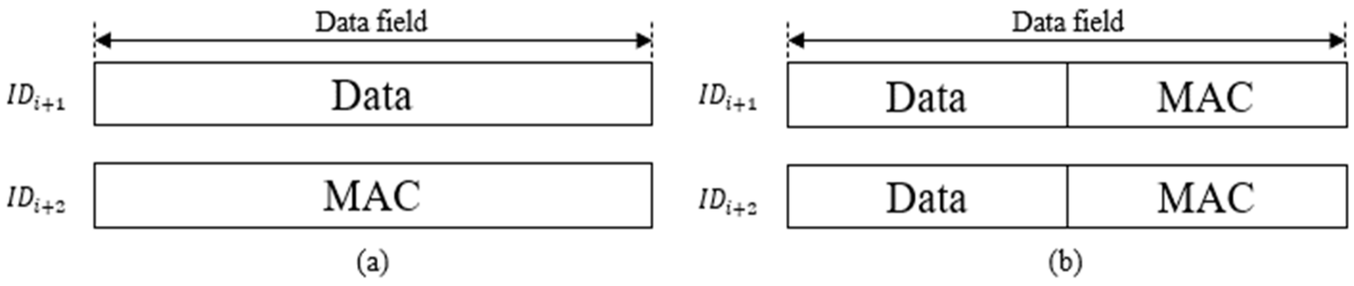


Figure 9. (**a**) TIFM for the case of compressed data length greater than four bytes; and (**b**) proposed method for the case of split ID.

ID split is determined during the system design phase. During system operation, the transmitted data type is determined based on the compressed data length, as shown in Table 10. If entropy-based signal grouping is used, the compressed data length rarely exceeds 4 bytes. However, when this happens, ID_{i+1} and ID_{i+2} are used to transmit the original data and 4-byte MAC, respectively. When the compressed data length is zero, only 4-byte MAC is transmitted in the data field, as opposed to the TIFM method.

If the probability of the occurrence of 1 in a specific bit is close to 50%, the number of occurrences of 1 in that bit increases even through the bit inversion process. Therefore, the compression efficiency of the proposed method is reduced in an environment with many such bits.

Table 10. Transmitted data type according to the compressed data length: $L(M_C)$.

Compressed Length (Byte)	ID	Transmitted Data Type
$L(M_C) \leq 4$	i	Compressed data and 4-byte MAC transmitted in the same data field
$4 < L(M_C)$	$i+1$	Original data transmitted
	$i+2$	4-byte MAC transmitted

5. Simulation Using Automobile and Tractor CAN Signals

In this section, simulation results using actual CAN signals are presented. The compression ratio is defined as follows:

$$\text{Comp.ratio} = \left(1 - \frac{\text{\# of bytes of compressed data}}{\text{\# of bytes of original data}}\right) \times 100 \tag{7}$$

Table 11 shows the data compression results of a CAN system with eight IDs in a Kia Sorento by the proposed method. To show the relationship between compression ratio and entropy, the entropy values for each ID are also shown in the table.

Table 11. Number of occurrences by the length of compressed data in a KIA Sorento by the proposed method.

ID	Number of Occurrences by Compressed Length (Byte)					Entropy	Comp. Ratio (%)
	1	2	3	4	5 or More		
0x260	163,078 (100%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	7.44	87.50%
0x2A0	163,078 (100%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	5.04	87.50%
0x316	0 (0.00%)	0 (0.00%)	0 (0.00%)	4313 (2.64%)	158,765 (97.36%)	46.25	27.60%
0x329	7553 (4.63%)	18,959 (11.63%)	135,917 (83.34%)	649 (0.40%)	0 (0.00%)	19.15	65.06%
0x43F	6086 (3.05%)	61,990 (31.07%)	131,141 (65.72%)	314 (0.16%)	0 (0.00%)	19.76	67.13%
0x440	0 (0.00%)	55,508 (27.82%)	143,950 (72.14%)	73 (0.04%)	0 (0.00%)	20.74	65.97%
0x545	636 (3.70%)	157,042 (96.30%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	13.31	75.46%
0x580	3112 (3.82%)	1444 (1.77%)	5197 (6.38%)	37,535 (46.07%)	34,180 (41.96%)	34.65	47.43%
Total (Avg.)	348,943 (26.93%)	294,944 (22.76%)	416,205 (32.12%)	42,884 (3.30%)	192,945 (14.89%)	19.89	66.65%

As an example of using entropy to estimate the compressed data length, consider ID 0x2A0 with an entropy of 5.04. By (6), the compressed data length is estimated as 1 byte. Notice that the simulated compression ratio of ID 0x2A0 is 87.50% in Table 11, which corresponds to the compressed data length of 1.0 bytes (8 $\times$ 12.5% = 1). For ID 0x316, the entropy is 46.25. Then, the estimated length of the compressed data is 6 bytes by (6). The simulated compression ratio of ID 0x316 is 27.60% in Table 11, which corresponds to the compressed data length of 5.79 bytes (8 $\times$ 72.4%). Therefore, it can be concluded that the compression results by the proposed SF-based compression method closely follow the estimated results by (6). In addition, Table 11 shows that a signal with low entropy has a high compression ratio, while a signal with high entropy has a low compression ratio.

For six IDs (0x260, 0x2A0, 0x329, 0x43F, 0x440, and 0x545) in Table 11, the length of each piece of compressed data is less than 5 bytes, so compressed CAN data and MAC can be transmitted in the same CAN frame. For IDs 0x316 and 0x580, the entropy values are greater than 32. Thus, 32-bit MAC cannot be transmitted in the same CAN frame. If efficient frame authentication is desired, an ID split is required for IDs 0x316 and 0x580. As an example of an ID split, ID 0x316 can be divided into IDs 0x317 and 0x319. Then, IDs 0x317 and 0x319 cut the entropy of ID 0x316 in half. Notice that, if the total entropy is equally divided among 8 IDs, the average entropy is 19.89. Therefore, if entropy-based signal grouping was performed properly for each ID in the system design phase, an ID split would not be necessary. Table 12 shows the occurrences based on a compressed data length of four bytes for a Kia Sorento by the proposed method. Notice that the ID split was applied to IDs 0x316 and 0x580. The length of each piece of compressed data is less than 5 bytes for all IDs due to ID split.

Table 12. Number of occurrences based on a compressed data length of four bytes for KIA Sorento by the proposed method.

ID	Less than 5 Bytes	5 or More Bytes	Total Frames
0x260	163,078 (100.00%)	0 (0.00%)	163,078
0x2A0	163,078 (100.00%)	0 (0.00%)	163,078
0x316 (ID split)	321,843 (100.00%)	0 (0.00%)	321,843
0x329	163,078 (100.00%)	0 (0.00%)	163,078
0x43F	199,531 (100.00%)	0 (0.00%)	199,531
0x440	199,531 (99.98%)	0 (0.00%)	199,531
0x545	163,078 (100.00%)	0 (0.00%)	163,078
0x580 (ID split)	115,648 (100.00%)	0 (0.00%)	115,648
Total	1,488,865 (100.00%)	0 (0.00%)	1,488,865

Table 13 shows the occurrences based on a compressed data length of four bytes for a Kia Sorento by VMLDA. Of a total of 1,287,683 frames, 8237 frames (0.64%) cannot be compressed to less than 5 bytes, so additional frames are required to transmit the MAC data.

Table 14 compares the occurrences based on a compressed data length of four bytes. By the proposed method with ID split, all transmitted frames can be authenticated with a 32-bit MAC without waiting for additional frames. If ID split is not used, VMLDA performs better than the proposed method in terms of the number of occurrences based on a compressed data length of 4 bytes. This is because ID 0x316 and ID 0x580 each have an entropy value greater than 32. In difference-based methods such as VMLDA, the difference in consecutive frames is calculated and compressed, so the compression performance is not significantly reduced even for signals with an entropy value greater than 32. However, difference-based methods have a fundamental problem in that errors in the previous frame are propagated to the next frame during the compression process of calculating the difference, as shown in Figure 3.

Table 13. Number of occurrences based on a compressed data length of four bytes for KIA Sorento by VMLDA algorithm.

ID	Less than 5 Bytes	5 or More Bytes	Total Frames
0x260	163,078 (100.00%)	0 (0.00%)	163,078
0x2A0	163,078 (100.00%)	0 (0.00%)	163,078
0x316	156,250 (95.81%)	6828 (4.19%)	163,078
0x329	163,074 (100.00%)	4 (0.00%)	163,078
0x43F	199,507 (99.99%)	24 (0.01%)	199,531
0x440	199,489 (99.98%)	42 (0.02%)	199,531
0x545	163,078 (100.00%)	0 (0.00%)	163,078
0x580	80,129 (98.36%)	1339 (1.64%)	81,468
Total	1,287,683 (99.36%)	8237 (0.64%)	1,295,920

Table 14. Comparison results of the number of occurrences based on a compressed data length of four bytes of KIA Sorento between the proposed method and VMLDA.

	Less than 5 Bytes	More than 4 Bytes
VMLDA	99.36%	0.64%
Proposed (w/o ID split)	85.11%	14.89%
Proposed (w/ ID split)	100.00%	0.00%

A simulation using a tractor was performed to show that the proposed algorithm can be applied to various industrial machines. Table 15 shows the data compression results of a CAN system with thirteen IDs in an LS Mtron tractor. The entropy values for each ID are also shown in the table. As with the KIA Sorento, we can conclude from Table 15 that signals with low entropy have high compression ratios, and signals with high entropy have low compression ratios. Compared to the KIA Sorento, the CAN signals of the LS Mtron tractor have lower entropy as they operate in a more restricted environment. Therefore, as shown in Table 15, the average compression ratio of LS Mtron (75.77%) is higher than that of Kia Sorento (66.65%).

Table 16 shows the comparison of occurrences based on a compressed data length of four bytes. In the case of the LS Mtron, the proposed compression method without ID split shows slightly better (0.22% higher) performance than VMLDA. This is due to the low average entropy of 12.77 of the LS Mtron tractor. Notice that ID 0xCF00400 is the only ID with entropy close to 32 bits. Therefore, if the ID split method is applied to ID 0xCF00400, as shown in Table 17, all frames can be authenticated directly without transmitting two frames in the proposed method. If VMLDA is used, CAN signals of four IDs (0xCF00300, 0xCF00400, 0x18FEDF00, and 0x18FF2100) may need to transmit two frames for authentication. This simulation shows that the proposed method can be used very efficiently to improve the security of CAN systems with low entropy values.

Table 15. Number of occurrences by the length of compressed data in an LS Mtron tractor by the proposed method.

ID	Number of Occurrences by Compressed Length (Byte)					Entropy	Compression Ratio (%)
	1	2	3	4	>4		
0xC000027	0 (0.00%)	33,064 (80.82%)	7847 (19.18%)	0 (0.00%)	0 (0.00%)	15.64	72.60%
0xC000127	40,900 (100.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	5.86	87.50%
0xCF00300	0 (0.00%)	610 (7.41%)	3329 (40.43%)	4295 (52.16%)	0 (0.00%)	27.22	56.91%
0xCF00400	0 (0.00%)	0 (0.00%)	103 (0.50%)	18,295 (88.86%)	2190 (10.64%)	31.63	48.73%
0xCFF0027	8182 (100.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0.01	87.50%
0x18F02300	1571 (19.08%)	5917 (71.86%)	746 (9.06%)	0 (0.00%)	0 (0.00%)	14.31	76.25%
0x18FEDF00	0 (0.00%)	46 (0.56%)	8182 (99.37%)	6 (0.07%)	0 (0.00%)	22.65	62.56%
0x18FF2100	8234 (100.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0.23	87.50%
0x18FF6121	8181 (100.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0.01	87.50%
0x18FF9521	4090 (100.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0.01	87.50%
0x18FF9E21	7967 (97.37%)	215 (2.63%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	4.66	87.17%
0x19FFA000	65 (0.64%)	6351 (62.06%)	3817 (37.30%)	0 (0.00%)	0 (0.00%)	18.43	70.42%
0x19FFA010	7853 (95.70%)	353 (4.30%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	4.41	86.96%
Total (Average)	87,043 (47.72%)	46,557 (25.52%)	24,024 (13.17%)	22,596 (12.39%)	2190 (1.20%)	12.77	75.77%

Table 16. Comparison results of the number of occurrences based on a compressed data length of four bytes of LS Mtron between the proposed method and VMLDA.

ID	Entropy	Less than 5 Bytes		Total Frames
		VMLDA	Proposed (w/o ID Split)	
0xC000027	15.64	40,911 (100%)	40,911 (100%)	40,911
0xC000127	5.86	40,900 (100%)	40,900 (100%)	40,900
0xCF00300	27.22	7227 (87.77)	8234 (100%)	8234
0xCF00400	31.63	19,119 (92.87%)	18,397 (89.36%)	20,588
0xCFF0027	0.01	8182 (100%)	8182 (100%)	8182

Table 16. *Cont.*

ID	Entropy	Less than 5 Bytes		Total Frames
		VMLDA	Proposed (w/o ID Split)	
0x18F02300	14.31	8234 (100%)	8234 (100%)	8234
0x18FEDF00	22.65	8127 (98.70%)	8234 (100%)	8234
0x18FF2100	0.23	8234 (99.99%)	8234 (100%)	8234
0x18FF6121	0.01	8181 (100%)	8181 (100%)	8181
0x18FF9521	0.01	4090 (100%)	4090 (100%)	4090
0x18FF9E21	4.66	8182 (100%)	8182 (100%)	8182
0x19FFA000	18.43	10,233 (100%)	10,233 (100%)	10,233
0x19FFA010	4.41	8206 (100%)	8206 (100%)	8206
Total (Average)	12.77	179,826 (98.58%)	180,218 (98.80%)	182,409

Table 17. Comparison results of the number of occurrences based on a compressed data length of four bytes of LS Mtron tractor between the proposed method and VMLDA.

	Less than 5 Bytes	5 or More Bytes
VMLDA	98.58%	1.42%
Proposed (w/o ID split)	98.80%	1.20%
Proposed (w/ ID split)	100%	0%

The peak load of the CAN bus was calculated using CANoe. The simulation results are obtained from Kia Sorento CAN data at 500 kbps used for the simulation in Table 11. The bus load is defined as follows:

$$\text{Busload}(\%) = \frac{\#\,\text{bits sent}}{\text{speed}} \tag{8}$$

The peak load is defined as the maximum bus load. In Figure 10, VN1630 (Vector CAN/CANFD IP Core) is used as TX HW, and VH6501 (Vector CAN/CANFD IP Core) is used as RX HW. A Pico Scope 5444D is used for CAN wave monitoring.

The simulation results in Table 18 show that the peak load of the original CAN system is reduced by 7.58% by using the proposed compression algorithm without MAC. This means that the peak load of the compressed system is only 60.8% of that of the original system. For the simulation, 795 frames are transmitted per second, and the total number of transmitted frames is 1,295,928 (frames transmitted in approximately 27 min).

To authenticate a CAN frame, the ID split method is used. In this case, as can be inferred from Table 12, the number of IDs increases from 8 to 10. Thus, the number of transmitted frames is increased from 795 to 914. Nevertheless, the peak load of the proposed system is almost the same as that of the original system without MAC, mainly due to the data length reduction by the compression algorithm.

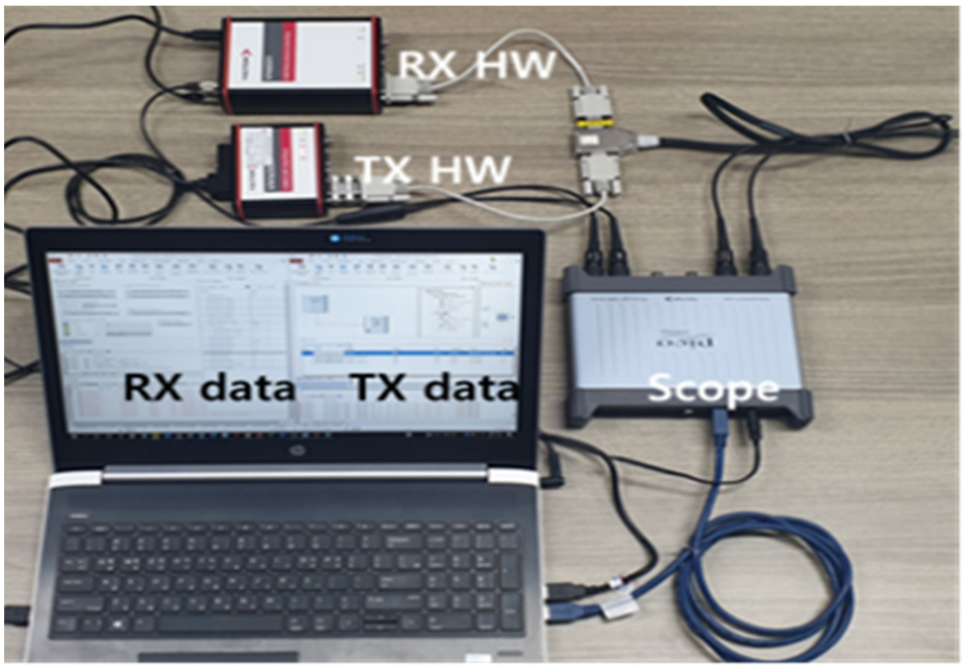


Figure 10. Simulation using the CANoe system.

Table 18. Peak load comparison.

	Peak Load	**Frames/s**	**Total Frames**
Original without MAC	19.38% (1)	795 (1)	1,295,928 (1)
Proposed without MAC	11.80% (0.608)	795 (1)	1,295,928 (1)
Proposed with MAC (ID split)	19.45% (1.004)	914 (1.14)	1,488,875 (1.14)

Table 19 compares the execution time (required number of clock cycles) for the data compression of KIA Sorento using a 32-bit MCU (TMS320F28335PGFA) operating at 100 MHz. In the proposed algorithm, IDs with higher entropy values require more clock cycles for data compression. The proposed algorithm requires an average of 190.75 clock cycles (1.9075 µs) for the data compression of one frame, whereas the VMLDA algorithm requires 696.5 clock cycles (6.965 µs). Therefore, the execution time of the proposed algorithm is only 27.39% of that of the VMLDA algorithm, indicating that the proposed algorithm is easier to process in real time.

Table 19. Comparison of execution time (required number of clock cycles) for data compression of KIA Sorento using 32-bit MCU (TMS320F28335PGFA) operating at 100 MHz.

ID (Entropy)	**Proposed**		**VMLDA**	
	Average Clock Cycles	**Compression Time (µs)**	**Average Clock Cycles**	**Compression Time (µs)**
0x260 (7.44)	105	1.05	555	5.55
0x2A0 (5.04)	119	1.19	536	5.36

Table 19. *Cont.*

ID (Entropy)	Proposed		VMLDA	
	Average Clock Cycles	**Compression Time (µs)**	**Average Clock Cycles**	**Compression Time (µs)**
0x316 (46.25)	299	2.99	752	7.52
0x329 (19.15)	172	1.72	683	6.83
0x43F (19.76)	213	2.13	628	6.28
0x440 (20.74)	220	2.20	726	7.26
0x545 (13.31)	158	1.58	1063	10.63
0x580 (34.65)	240	2.40	629	6.29
Average	190.75	1.9075	696.5	6.965

6. Security Analysis

In this section, the proposed method is compared to the security solutions proposed in Truncated-MAC, Mini-MAC, and SecOC. Table 20 shows the results of a comparative evaluation. All four methods provide data frame authentication and can block impersonation attacks and replay attacks.

Table 20. Comparison of security solutions. (Y: Yes. N: No).

	Proposed	**Truncated-MAC**	**Mini-MAC**	**SecOC**
No standard change	Y	N	Y	Y
No communication overhead	Y	Y	Y	N
Prevent impersonation attack	Y	Y	Y	Y
Prevent replay attack	Y	Y	Y	Y

The three studies described in Section 2.2 use ID, Data, and CRC fields for MAC transmission. While the use of the data field, as shown in Mini-MAC and SecOC, is common, it is not recommended due to the data frame overhead. The CAN standard must be modified to repurpose the CRC field. Accordingly, similar to Truncated-MAC, a method to use the CRC field cannot be applied to a real automotive environment.

The proposed compression-based data authentication is a practical method that guarantees real-time data processing. Since MAC is delivered using the extra space generated by the compression of data frames, data frame overhead does not occur. Furthermore, the proposed method can be used without modification of the CAN standard since it can be implemented only with modification of the software.

7. Conclusions

In this paper, an SF-based CAN data compression algorithm was proposed that compresses the current frame without relying on previous frames. Therefore, the proposed compression method does not propagate the effects of errors caused by data errors, frame loss, replay attacks, or ECU resets to subsequent frames, unlike difference-based compression methods. To authenticate CAN data with 4-byte MAC, a signal grouping technique was also proposed based on entropy analysis.

Simulation using actual CAN signals showed that the proposed compression algorithm can guarantee a compression ratio of more than 50% (32 bits). Real-time CAN authentication is easily achievable, unlike other compression methods. Simulation results using CANoe show that the proposed method can achieve data authentication in the CAN system with almost the same peak load as the original CAN system without authentication. In addition, the execution time of the proposed algorithm is only 27.39% of that of the VMLDA, indicating that the proposed algorithm is easier to process in real time. In addition, taking a tractor as an example, it was shown that the proposed method can be successfully applied to other industrial machines.

The compression performance of the proposed method depends on the data used to create the mapping table. Therefore, the data should be collected carefully through several test runs to represent typical characteristics of the vehicle during driving. If the probability of occurrence of 1 in a specific bit of the data is close to 50%, the number of occurrences of 1 in that bit increases even through the bit inversion process. Therefore, the compression efficiency of the proposed method is reduced in an environment with many such high-entropy bits.

Countermeasures for CAN attacks such as authentication, preventative protection, intrusion detection, and post-detection impose large overhead on the availability of the existing resources of automobiles. Therefore, it is necessary to develop CAN security enhancement methods that do not require heavy computational power. The co-development of hardware and software to eliminate the need for extremely high-speed ECUs would be a good research topic. No single countermeasure can provide complete security. The development of algorithms that efficiently combine two or more countermeasures is also a promising future topic.

Author Contributions: Data curation, Y.-E.K. and D.-H.S.; formal analysis, S.-Y.J. and Y.-J.K.; methodology, J.-G.C., S.W. and Y.-E.K.; resources, J.-G.C. and D.-H.S.; supervision, J.-G.C.; writing—original draft, J.-G.C. and S.-Y.J.; writing—review and editing, J.-G.C. and S.-Y.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) funded by the Korea Government (MIST), Development of Collection and Integrated Analysis Methods of Automotive Inter and Intra System Artifacts through Construction of Event-based experimental system, under Grant 2022-0-01022.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Jo, H.J.; Choi, W. A survey of attacks on controller area networks and corresponding countermeasures. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 6123–6141. [CrossRef]
2. Ben, L.N.M.; Nasri, O.; Adouane, L. Controller area network reliability: Overview of design challenges and safety related perspectives of future transportation systems. *IET Intell. Transp. Syst.* **2020**, *14*, 1727–1739.
3. Valasek, C.; Miller, C. Adventures in Automotive Networks and Control Units. Def. Con. Available online: https://ioactive.com/pdfs/IOActive_Adventures_in_Automotive_Networks_and_Control_Units.pdf (accessed on 1 February 2024).
4. Checkoway, S.; McCoy, D.; Kantor, B.; Anderson, D.; Shacham, H.; Savage, S. Comprehensive experimental analyses of automotive attack surfaces. In Proceedings of the 20th USENIX Security Symposium, San Francisco, CA, USA, 8–12 August 2011; pp. 8–12.
5. Evenchick, E. An Introduction to the CANard Toolkit. Black Hat USA. Available online: https://blackhat.com/docs/asia-15/materials/asia-15-Evenchick-Hopping-On-The-Can-Bus-wp.pdf (accessed on 1 February 2024).
6. Wen, S.; Zhao, Q.; Chen, Q.A.; Lin, Z. Automated cross-platform reverse engineering of CAN bus commands from mobile apps. In Proceedings of the 2020 Network and Distributed System Security Symposium, San Diego, CA, USA, 23–26 February 2020; pp. 1–17.
7. Mandal, A.K.; Panarotto, F.; Cortesi, A.; Ferrara, P.; Spoto, F. Static analysis of Android Auto infotainment and on-board diagnostics II apps. *Softw. Pract. Exp.* **2019**, *49*, 1131–1161. [CrossRef]

8. Nie, S.; Liu, L.; Du, Y. Free-fall: Hacking TESLA from Wireless to CAN Bus. Black Hat USA. Available online: https://www.blackhat.com/docs/us-17/thursday/us-17-Nie-Free-Fall-Hacking-Tesla-From-Wireless-To-CAN-Bus-wp.pdf (accessed on 1 February 2024).
9. Miller, C.; Valasek, C. Remote Exploitation of an Unaltered Passenger Vehicle. Black Hat USA. Available online: https://www.blackhat.com/us-15. (accessed on 1 February 2024).
10. Cui, J.; Chen, Y.; Zhong, H.; He, D.; Wei, L.; Bolodurina, I.; Liu, L. Lightweight encryption and authentication for controller area network of autonomous vehicles. *IEEE Trans. Veh. Technol.* **2023**, *11*, 14756–14770. [CrossRef]
11. Schmandt, J.; Sherman, A.T.; Banerjee, N. Mini-MAC: Raising the bar for vehicular security with a lightweight message authentication protocol. *Veh. Commun.* **2017**, *9*, 188–196. [CrossRef]
12. Kim, Y.J.; Woo, S.; Chung, J.G. Triple ID flexible MAC for CAN security improvement. *IEEE Access* **2021**, *9*, 126388–126399. [CrossRef]
13. Koscher, K.; Czeskis, A.; Roesner, F.; Patel, S.; Kohno, T.; Checkoway, S.; Savage, S. Experimental security analysis of a modern automobile. In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA, 16–19 May 2010; pp. 447–462.
14. Bella, G.; Biondi, P.; Costantino, G.; Matteucci, I. CINNAMON: A module for AUTOSAR secure onboard communication. In Proceedings of the 16th European Dependable Computing Conference (EDCC), Munich, Germany, 7–10 September 2020; pp. 103–110.
15. Woo, S.; Jo, H.J.; Lee, D.H. A practical wireless attack on the connected car and security protocol for in-vehicle CAN. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 993–1006. [CrossRef]
16. Fowler, D.S.; Bryans, J.; Shaikh, S.A.; Wooderson, P. Fuzz testing for automotive cyber-security. In Proceedings of the 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), Luxembourg, 25–28 June 2018; pp. 239–246.
17. Yu, L.; Deng, J.; Brooks, R.R.; Yun, S.B. Automobile ECU design to avoid data tampering. In Proceedings of the 10th Annual Cyber and Information Security Research Conference, New York, NY, USA, 7–9 April 2015; pp. 10:1–10:4.
18. Cheng, K.; Bai, Y.; Zhou, Y.; Tang, Y.; Sanan, D.; Liu, Y. CANeleon: Protecting CAN bus with frame ID chameleon. *IEEE Trans. Veh. Technol.* **2020**, *69*, 7116–7130. [CrossRef]
19. Olufowobi, H.; Young, C.; Zambreno, J.; Bloom, G. SAIDuCANT: Specification-based automotive intrusion detection using controller area network (CAN) timing. *IEEE Trans. Veh. Technol.* **2020**, *69*, 1484–1494. [CrossRef]
20. Katragadda, S.; Darby, P.J.; Roche, A.; Gottumukkala, R. Detecting low-rate replay-based injection attacks on in-vehicle networks. *IEEE Access* **2020**, *8*, 54979–54993. [CrossRef]
21. Lee, S.; Choi, W.; Jo, H.J.; Lee, D.H. T-Box: A forensics-enabled trusted automotive data recording method. *IEEE Access* **2019**, *7*, 49738–49755. [CrossRef]
22. Steger, M.; Boano, C.A.; Niedermayr, T.; Karner, M.; Hillebrand, J.; Roemer, K.; Rom, W. An efficient and secure automotive wireless software update framework. *IEEE Trans. Ind. Inform.* **2018**, *14*, 2181–2193. [CrossRef]
23. Oh, S.B.; Do, Y.S.; Lee, M.J.; Kim, J.H.; Jeon, J.W. Performance enhancement of CAN/Ethernet automotive gateway with a CAN data reduction algorithm. *Electronics* **2023**, *12*, 2777. [CrossRef]
24. Mun, H.; Han, K.; Lee, D.H. Ensuring Safety and security in CAN-based automotive embedded systems: A combination of design optimization and secure communication. *IEEE Trans. Veh. Technol.* **2020**, *69*, 7078–7091. [CrossRef]
25. Jo, H.J.; Kim, J.H.; Choi, H.Y.; Choi, W.; Lee, D.H.; Lee, I. MAuth-CAN: Masquerade-attack-proof authentication for in-vehicle networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 2204–2218. [CrossRef]
26. Misbahuddin, S.; Mahmud, S.M.; Holou, N.A. Development and performance analysis of a data-reduction algorithm for automotive multiplexing. *IEEE Trans. Veh. Technol.* **2001**, *50*, 162–169. [CrossRef]
27. Ramteke, P.R.; Mahmud, S.M. An adaptive data-reduction protocol for the future in-vehicle networks. *SAE Trans.* **2005**, *114*, 1540–1554.
28. Miucic, R.; Mahmud, S.M. An improved adaptive data reduction protocol for in-vehicle networks. *SAE Trans.* **2006**, *115*, 650–658.
29. Miucic, R.; Mahumd, S.M.; Popovic, Z. An enhanced data-reduction algorithm for event-triggered networks. *IEEE Trans. Veh. Technol.* **2009**, *56*, 2663–2678. [CrossRef]
30. Kelkar, S.; Kamal, R. 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 (CSCITA), Mumbai, India, 4–5 April 2014; pp. 162–167.
31. Oh, S.B.; Kim, J.H. Comparison and Analysis of Controller Area Network Compression Algorithms. *Trans. Korean Soc. Automot. Eng.* **2020**, *28*, 629–636. [CrossRef]
32. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]