applied sciences

MDPI

Article

MAC-Based Compression Ratio Improvement for CAN Security

Jinhui Piao [1], Shiyi Jin [1], Dong-Hyun Seo [2], Samuel Woo [3] and Jin-Gyun Chung [1,*]

[1] Division of Electronic Engineering, IT Convergence Research Center, Jeonbuk National University, Jeonju 54896, Republic of Korea
[2] Green Mobility R&D Center, Jeonbuk Institute of Automotive Convergence Technology, Gunsan 54158, Republic of Korea
[3] Department of Software Science, Dankook University, Yongin 16891, Republic of Korea
* Correspondence: jgchung@jbnu.ac.kr

Abstract: Information security in a controller area network (CAN) is becoming more important as the connections between a vehicle's internal and external networks increase. Encryption and authentication techniques can be applied to CAN data frames to enhance security. To authenticate a data frame, a message authentication code (MAC) needs to be transmitted with the CAN data frame. Therefore, space for transmitting the MAC is required within the CAN frame. Recently, the Triple ID algorithm has been proposed to create additional space in the data field of the CAN frame. The Triple ID algorithm ensures every CAN frame is authenticated by at least 4 bytes of MAC without changing the original CAN protocol. However, since the Triple ID algorithm uses six header bits, there is a problem associated with low data compression efficiency. In this paper, we propose an algorithm that can remove up to 15 bits from frames compressed with the Triple ID algorithm. Through simulation using CAN signals of a Kia Sorento vehicle and an LS Mtron tractor, we show that the generation of frames containing compressed messages of 4 bytes or more is reduced by up to 99.57% compared to the Triple ID method.

Keywords: CAN; security; data compression; authentication; header bits; Triple ID; MAC

Citation: Piao, J.; Jin, S.; Seo, D.-H.; Woo, S.; Chung, J.-G. MAC-Based Compression Ratio Improvement for CAN Security. *Appl. Sci.* **2023**, *13*, 2654. https://doi.org/10.3390/app13042654

Academic Editors: Ireneusz Kubiak, Tadeusz Wieckowski and Yevhen Yashchyshyn

Received: 22 December 2022
Revised: 15 February 2023
Accepted: 17 February 2023
Published: 18 February 2023

1. Introduction

In recent years, more and more electronic control units (ECUs) have been connected to controller area networks (CANs) to extend automotive capabilities. As the number of in-vehicle systems and external network connections increases, the need for information security becomes even greater.

A comprehensive survey of state-of-the-art attack surfaces of automotive CAN and corresponding protection mechanisms is presented in [1]. In [1], the attack surfaces are categorized into three types: (1) physical access-based attacks exploiting the OBD-II port or USB ports [2,3], (2) wireless access-based attacks requiring initial physical access [4,5], and (3) wireless access-based attacks without physical access exploiting Bluetooth, WiFi or cellular channels [6,7].

The countermeasures against CAN attacks are categorized into four types: (1) preventative protection (fuzzing methods [8] and anti-analysis methods [9]), (2) intrusion detection [10,11], (3) authentication [12–16], and (4) post-protection (attack identification methods [17] and secure patch methods [18]).

One way to efficiently improve CAN security is to apply authentication and encryption methods to CAN frames [12]. Each frame must be encrypted to prevent the leakage of information. During transmission, a message authentication code (MAC) must be generated and transmitted to authenticate the transmitted frame. Therefore, additional data space must be reserved in the CAN frame to transmit the MAC.

The size of the data field in a CAN frame is limited to 8 bytes. The Practical Security algorithm [12] uses 32-bit MAC for transmission due to the size limitation of the data

field. Therefore, the 16-bit extended ID field and 16-bit CRC field of CAN 2.0B are used for MAC transmission. However, variations in the CAN protocol can cause issues such as compatibility.

In the Mini-MAC algorithm [13], if most ECUs connected to the CAN system send data smaller than 5 bytes, MAC can be transmitted using the remaining area of the data field. However, data may not be authenticated if sufficient space is not secured.

The Triple ID algorithm compresses data using a compression algorithm and adds a MAC of 4 bytes or more to the secured space [14]. Therefore, the Triple ID algorithm ensures every CAN frame is authenticated without changing the original CAN protocol. However, if the compressed data length is greater than 4 bytes, one frame cannot contain both compressed data and MAC since the length of MAC should be at least 4 bytes to ensure sufficient authentication performance. In this case, the MAC is sent using a separate frame. Therefore, it is important to ensure that the compressed data length is not larger than 4 bytes to reduce bus load and latency.

In [15], a selective message authentication method is proposed specifically for safety-critical CAN packets. Selective message authentication can reduce the communication overhead on the CAN bus, but the delay stemming from the transmission of additional data packets is an unavoidable drawback.

The MAuth-CAN algorithm presented in [16] is resistant to masquerading attacks. The MAuth-CAN algorithm neither uses the full capacity of the network nor requires hardware modifications to the CAN controller. However, latency in real-time applications still needs to be considered.

A compression algorithm can be used to overcome the limitations of the short data field of CAN frames. CAN data compression uses the difference between two frames for transmission based on the fact that the data fields of successive CAN frames with the same message ID do not change rapidly. CAN data compression can be divided into two categories: methods using a predefined maximum difference value (PMDV) [19–23] and methods using compression area selection (CAS) maps [24–26].

In PMDV-based methods, if the difference does not exceed some predefined maximum value, only the differences between the current and preceding CAN messages are transmitted. In [19], if the value of the delta (difference) exceeds the length of the assigned delta field, the current CAN message is transmitted rather than the delta compressed version of the message. Therefore, two message IDs are used to distinguish between the compressed data and the uncompressed data. In [20], to use a single ID, the first bit of the data field is assigned as a data reduction code (DRC) indicating whether data reduction is used or not. By using data length code (DLC) in CAN frame format, the enhanced data reduction (EDR) algorithm [21] eliminates the difficulties in the identification of compressed messages, such as the use of the reserved bit [22]. In the boundary of the fifteen compression (BFC) algorithm, if the current value of a CAN signal has changed within the maximum compression rage of ± 15, then the CAN signal can be compressed [23].

The compression efficiency of PMDV-based methods depends on the accuracy of the PMDV chosen for the application. The best compression efficiency is achieved when the PMDV adapts to changes in the vehicle operating environment [24]. However, it is difficult to continuously maximize compression efficiency since PMDV is difficult to adjust once determined for an application.

In CAS map-based algorithms, a compression area selection map eliminates the need to predict the maximum difference value as described in Section 2.3. In [24], the difference between the previous and the current data field is calculated and each difference value is represented using a modified sign-magnitude number which denotes the sign by the least significant bit. In [25], the bit-wise exclusive-OR values between successive data fields are computed instead of calculating the difference values. Thus, the use of a sign bit is not necessary. In CAS map-based algorithms, the 64-bit CAN data field is arranged as three signals having 24, 24, and 16 bits. In [26], a systematic way is presented to place the CAN data bits using multi-level arrangement maps to obtain the best compression efficiency.

In CAS map-based algorithms, the size of the CAS map varies according to vehicle driving conditions. Since the change of the CAS map corresponds to the change in the PMDV, the compression algorithms based on the CAS map show better compression efficiency than the PMDV-based algorithms.

Recently, CAN ID shuffling (hopping or obfuscation) techniques have been proposed to prevent DoS attacks and replay attacks. Abdulmalik et al. proposed a DoS defense technique using CAN ID hopping [27]. This method proposes a CAN ID hopping technique using the offset generated by the trusted party when detecting a DoS attack. Martin et al. proposed a CAN ID obfuscation scheme [28]. In this method, a CAN ID obfuscation technique is used, which uses a different CAN ID system for each car, even for cars of the same model. While these methods are effective in not increasing the amount of data to be transmitted, they have the drawback that they cannot prevent impersonation and replay attacks.

In this paper, we propose an algorithm that significantly reduces the amount of transmission data of the Triple ID method by combining CAS map and MAC. By the proposed method, up to 15 bits can be further reduced from the compressed data obtained by the Triple ID method. Using the proposed method, we show that the occurrence of compressed messages exceeding 4 bytes is reduced by up to 99.57% compared to the Triple ID method.

Section 2 introduces existing data authentication methods for CAN security. A new compression efficiency improvement method is proposed in Section 3. Simulation results using vehicles are presented in Section 4. Finally, a brief conclusion is provided in Section 5.

2. Existing CAN Data Authentication Methods

2.1. Practical Security Algorithm

In the Practical Security algorithm, AES-128 and hash-based message authentication (HMAC) are used for CAN data encryption and authentication, respectively [12]. The ciphertext C is obtained from the plaintext M as

$$C = E_{EK_k}(CTR_{ECU_s}) \oplus M, \tag{1}$$

where $E_{EK_k}(\cdot)$ is the AES-128 encryption function using the k-th session encryption key EK_k, CTR_{ECU_s} is the message counter value of the transmitting ECU ($ECUs$), and $\oplus$ is the XOR operation. Each ECU must manage data frame counter values for all relevant incoming and outgoing data.

AES-128 encryption produces an output of 128 bits, but since the maximum size of the CAN data field is 64 bits, only the first 64 bits are used to generate the ciphertext in (1).

The MAC data of the Practical Security algorithm is generated as follows:

$$MAC_{PS} = H_{AK_k}(ID_s \| C \| CTR_{ECU_s}), \tag{2}$$

where $H_{AK_k}(\cdot)$ denotes the keyed hash function using the k-th session authentication key AK_k, ID_s denotes the ID of the sending ECU, and $\|$ denotes a concatenation operation.

In the Practical Security algorithm, 32-bit MAC is used, and the 32-bit MAC provides sufficient authentication for CAN systems. Since there is no room for MAC in the data field, the 16-bit extended ID field and 16-bit CRC field of CAN 2.0B are used for MAC transmission as shown in Figure 1. However, variations in the CAN protocol can cause issues such as compatibility.

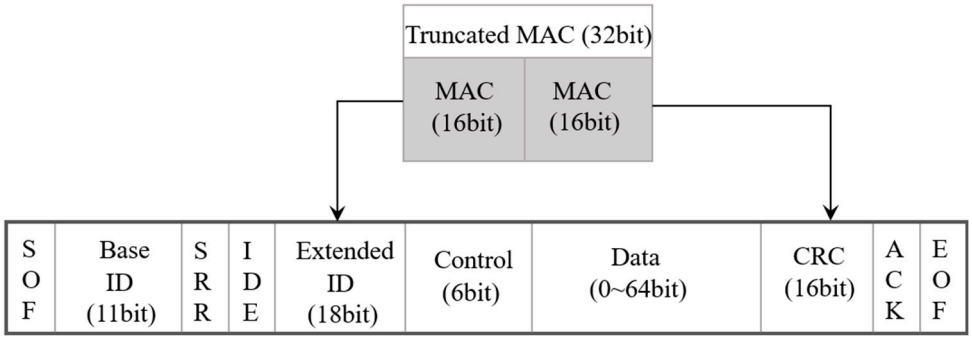

Figure 1. 32-bit MAC of the Practical Security algorithm in CAN 2.0B.

2.2. Mini-MAC Algorithm

The Mini-MAC algorithm generates a MAC as follows [13]:

$$MAC_{Mini} = T[s, H_k(k \| M_n \| CTR_{ECU_s} \| M_{n-\lambda} \| \cdots M_{n-1})], \quad (3)$$

where $T[s, \cdot]$ is a function that extracts the first s bits from input, $H_k(\cdot)$ denotes a hash function using the authentication key k, M_n is the current message to send, and $\{M_{n-\lambda} \| \cdots M_{n-1}\}$ is the concatenation of the most recent λ messages. The hash function used by Mini-MAC is Message-digest algorithm 5 (MD5) which takes 64-byte input.

For the 64-byte input in (3), the counter and key need 8 and 16 bytes, respectively. Therefore, for $\lambda = 5$, 40 bytes are allocated for 6 messages $(M_n \cdots M_{n-5})$, and each message is expected to consume an average of 6.67 bytes.

If most ECUs use data fields smaller than 5 bytes, the remaining space in the data field can be used to transmit at least 4 bytes of truncated Mini-MAC data. Otherwise, a Mini-MAC algorithm does not provide sufficient authentication for CAN systems.

2.3. Modified MAC Algorithm Using ICANDR

In the ICANDR algorithm, the 8-byte CAN data is first divided into three signals: Sig A (3 bytes), Sig B (3 bytes), and Sig C (2 bytes). Next, the bit-wise XOR values between the current signal and the previous signal are computed as shown in Table 1. The difference between two successive CAN signals with the same message ID is usually close to zero. This is because the rate of generation of CAN signals is much greater than the rate of change in car driving. If the calculated XOR value is nonzero, the corresponding header bit is set to one. Otherwise, the corresponding header bit is set to zero.

Table 1. Header bit calculation in the ICANDR algorithm.

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)	$1A\ 2B\ E1_{16}$	$9A\ EC\ 92_{16}$	$74\ E3_{16}$
XOR $(F_{i-1} \oplus F_i)$	$00\ 00\ 2B_{16}$	$00\ 00\ 04_{16}$	$00\ 00_{16}$
Header bit	1	1	0

The calculated header bits are arranged in the last column of the CAS map in sequence, as shown in Table 2. Starting in the next row, the XORed values for Sig A and Sig B are placed in bits 23 through 0. However, the XORed values for Sig C are placed in bits 15 through 0 because Sig C is a two-byte signal. If a header bit is 0, the corresponding row is emptied, as shown in Table 2.

Table 2. CAS map corresponding to Table 1 (shaded area: selection area).

Signal	Bit 23—Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Header (HA)							1
Header (HB)							1
Header (HC)							0
SA $(\text{Sig } A_i \oplus \text{Sig } A_{i-1})$	$0 \cdots 0$	1	0	1	0	1	1
SB $(\text{Sig } B_i \oplus \text{Sig } B_{i-1})$	$0 \cdots 0$	0	0	0	1	0	0
SC $(\text{Sig } C_i \oplus \text{Sig } C_{i-1})$	-	-	-	-	-	-	-

Next, starting from the leftmost column, 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 area is designated as a selection area as shown in Table 2.

Finally, the selection area in Table 2 is arranged using the memory map in Table 3. Thus, in this example, eight-byte data are compressed into only two bytes.

Table 3. Memory map corresponding to Table 2.

	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	0	SB [5]	SA [5]	SB [4]	SA [4]	SB [3]	SA [3]	SB [2]

In the modified MAC algorithm [29], data compression is performed using the ICANDR algorithm, but the compression ratio is further improved by displaying the header bits in the last byte of the MAC instead of the compression area. For example, if the header bits are "101", the decimal value of the header bits is "5". Then, the fifth bit from the LSB of the MAC is inverted as shown in Figure 2. The receiving ECU calculates the MAC independently, compares it with the last byte of the received MAC, and determines the header bit value by the inverted bit position.

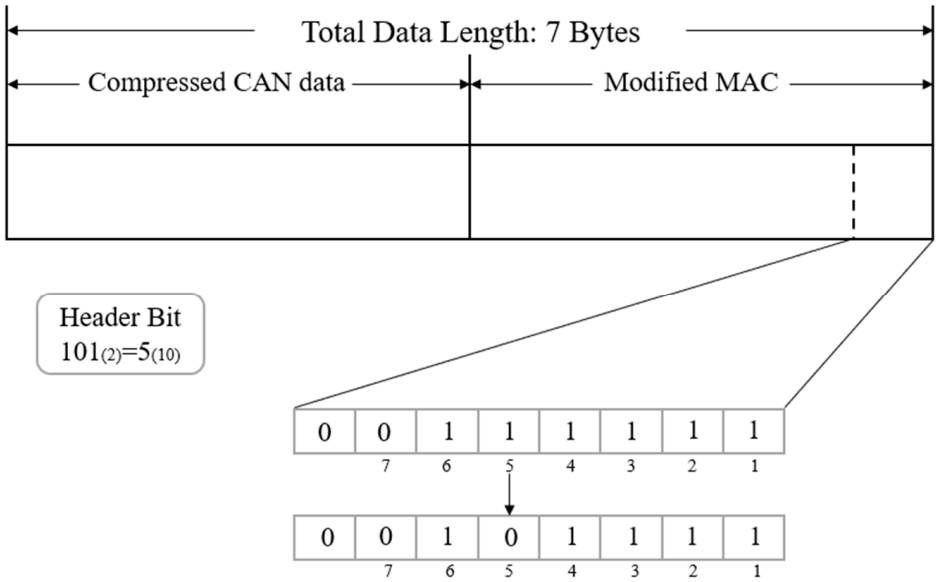

Figure 2. Header bit representation by modified MAC.

The modified MAC algorithm can be applied when the length of compressed data is 0 to 6 bytes. The length of MAC_{mod}, the MAC data of the modified MAC algorithm, is determined as

$$L(MAC_{mod}) = 7 - L(M_c), \tag{4}$$

where $L(x)$ means the length of x in bytes, and M_c means compressed data. In (4), the largest value of $L(MAC_{mod})$ is 7. The 8-byte case is not used to avoid confusion with uncompressed 8-byte data.

Using a modified MAC algorithm, 3 bits can be further reduced compared to the ICANDR algorithm. If the length of the compressed data is 6 bytes, the length of the MAC is limited to 1 byte. However, it is difficult to expect sufficient authentication performance with a 1-byte MAC. In addition, the modified MAC algorithm cannot be applied if the compressed data length is greater than 6 bytes.

2.4. Triple ID Algorithm

Unlike the Mini-MAC or modified MAC algorithms, the Triple ID algorithm always transmits a MAC of at least 4 bytes to ensure sufficient authentication performance of CAN frames [14]. The Triple ID algorithm uses a CAS map-based data compression technique similar to the ICANDR algorithm.

Figure 3 shows the MAC generation procedure of the Triple ID algorithm. The generated MAC including the authentication key k is

$$MAC_{TI} = T\left[s, H_k\left(k \| C_{C_n} \| CTR_{ECU_s} \| C_{C_{n-1}} \| \cdots C_{C_{n-\lambda}}\right)\right], \tag{5}$$

where C_{C_n} is the current encrypted compressed message, and $\left\{C_{C_{n-1}}, \cdots C_{C_{n-\lambda}}\right\}$ are the previous encrypted compressed messages. Therefore, malicious attackers attempting to intrude into the CAN system should collect previous valid messages in addition to the authentication key. (Equations (5) and (6) of [14] may be referred to in order to calculate a compressed ciphertext and retrieve the compressed data from the compressed ciphertext.)

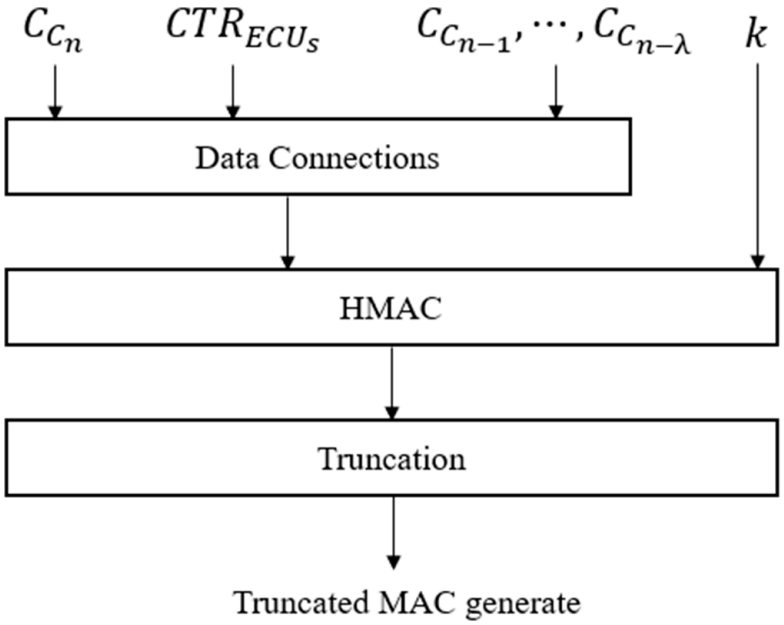

Figure 3. MAC generation procedure of Triple ID algorithm.

In the Triple ID algorithm, when the compressed data length is less than 4 bytes, the length of the MAC is calculated as

$$L(MAC_{TI}) = 8 - L(M_c). \tag{6}$$

Then, the compressed data and the MAC are transmitted as one frame using ID i as shown in Table 4.

Table 4. ID assignment of the Triple ID algorithm.

Compressed Data Length	ID	Type of Data
$L(M_c) \leq 4$	i	Compressed data and MAC in the same data field
$4 < L(M_c) \leq 7$	$i+1$	Compressed data
	$i+2$	8 bytes of MAC
$7 < L(M_c)$	$i+1$	Uncompressed data
	$i+2$	8 bytes of MAC

If the compressed data length is greater than 4 bytes, one frame cannot contain both compressed data and MAC since the length of MAC should be at least 4 bytes. If the compressed data length is greater than 4 bytes and less than 7 bytes, the compressed data is transmitted with ID $i + 1$ and the 8-byte MAC is transmitted with ID $i + 2$. If the compressed data length is greater than 7 bytes, the uncompressed CAN data is transmitted using ID $i + 1$, and the 8-byte MAC is transmitted using ID $i + 2$.

The Triple ID algorithm uses 6 header bits. As in the ICANDR algorithm, the 3 header bits of HA, HB and HC indicate the presence of Sig A, Sig B and Sig C, respectively. The other three bits LC[2], LC[1], and LC[0] indicate the compressed data length in bytes and the length of the MAC in the data field can be determined using this information.

The Triple ID algorithm does not require any changes to the CAN protocol to provide authentication. In addition, the Triple ID algorithm always transmits a MAC of at least 4 bytes to provide sufficient authentication performance for CAN frames. However, two frames must be transmitted if the compressed data length is larger than 4 bytes. Therefore, it is important to compress CAN data to 4 bytes or less for communication efficiency such as low bus load and short latency.

3. Proposed CAN Security Method

The Triple ID algorithm provides good authentication performance by always transmitting a MAC of 4 bytes or more. However, two frames must be transmitted if the com-pressed data length is greater than 4 bytes. In this section, we propose an algorithm to reduce the probability of the case where the compressed data length is greater than 4 bytes.

Our proposed security architecture is divided into three phases. Phase 3 (Section 3.3 Session Key Management) uses a well-known security method (Authenticated Key Exchange Protocol 2 (AKEP2)) to construct a session key distribution process. In the security architecture we propose, the main novel contributions lie in Phases 1 (Section 3.1 Bit Replacement Algorithm) and 2 (Section 3.2 Data Transmission and Reception).

3.1. Bit Replacement Algorithm (BRA)

In the Triple ID algorithm, the CAN data field consists of compressed data and a truncated MAC of 4 bytes or more. The compressed data is encrypted using the AES-128 algorithm, but MAC is not encrypted. MAC generation of the proposed method uses the same scheme as the Triple ID method. However, in the proposed algorithm, the length of the truncated MAC is fixed at 4 bytes. This is because a 4-byte MAC provides sufficient authentication for CAN systems [12]. Therefore, since the compressed data length can be determined directly, 3 bits (LC[2], LC[1], LC[0]) can be removed from the 6 header bits of the Triple ID method.

The Bit Replacement Algorithm is summarized as follows:

① Let X and Y denote 3-bit and 8-bit data, respectively.
② If the decimal value of X is i, X can be expressed using Y by inverting the i-th bit from the LSB of Y.
③ X can be restored from the inverted bit position information of Y.

- BRA can be applied repeatedly. If Y is (4×8)-bit data, (4×3)-bit data can be expressed using Y as shown in Figure 4. In Figure 4, since the last 3 bits of the compressed data are '010', the second bit from the LSB of the last byte of the MAC is inverted. In a similar way, 12 bits of compressed data can be expressed using 4 bytes of MAC.
- Compressed data of the proposed method is obtained in the same way as the Triple ID method. If the length of the compressed data is 0, the 4-byte MAC is transmitted without any change. Otherwise, BRA is applied starting from the last 3 bits of the compressed data. Each application of BRA removes 3 bits from the compressed data. BRA can be applied up to 4 times, but the application of BRA stops when there is no compressed data left. By the proposed method, a maximum of 15 bits (12 bits by BRA and 3 header bits) can be further reduced from the compressed data by the Triple ID method.

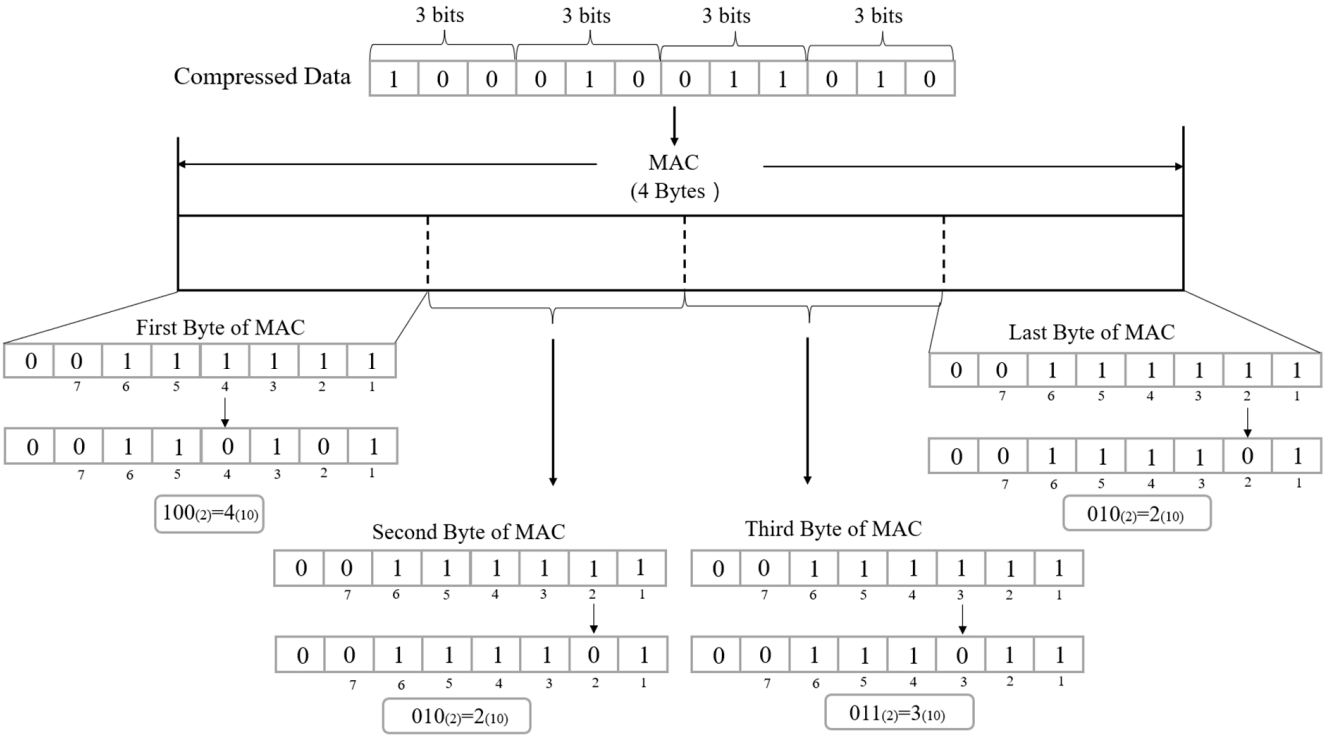

Figure 4. 12-bit data representation using 4-byte MAC with the proposed method.

If the compressed data length is still greater than 4 bytes after applying the proposed algorithm, the same procedure as the triple ID algorithm (i.e., using ID_{i+1} and ID_{i+2}) is applied except that the length of MAC is 4 bytes. The receiver compares the independently generated MAC with the received MAC. After examining the inverted bit locations, the compressed data can be restored.

Proposed method differs from the modified MAC in that it improves compression performance by moving not only header bits but also part of data information to MAC.

3.2. Data Transmission and Reception

Data transmission by the proposed algorithm can be defined by two cases based on the BRA-compressed data length.

- Case I ($0 \leq$ BRA-compressed data length ≤ 4): BRA-compressed data and 4-byte MAC are transmitted in the same data field using ID_i.
- Case II ($4 <$ BRA-compressed data length ≤ 7): BRA-compressed data and 4-byte MAC are transmitted using ID_{i+1} and ID_{i+2}, respectively.

The maximum length of the BRA-compressed data is 7. Therefore, unlike the Triple ID method, there is no need to consider an 8-byte uncompressed message. Figures 5 and 6 show the proposed data transmission and reception flow charts, respectively.

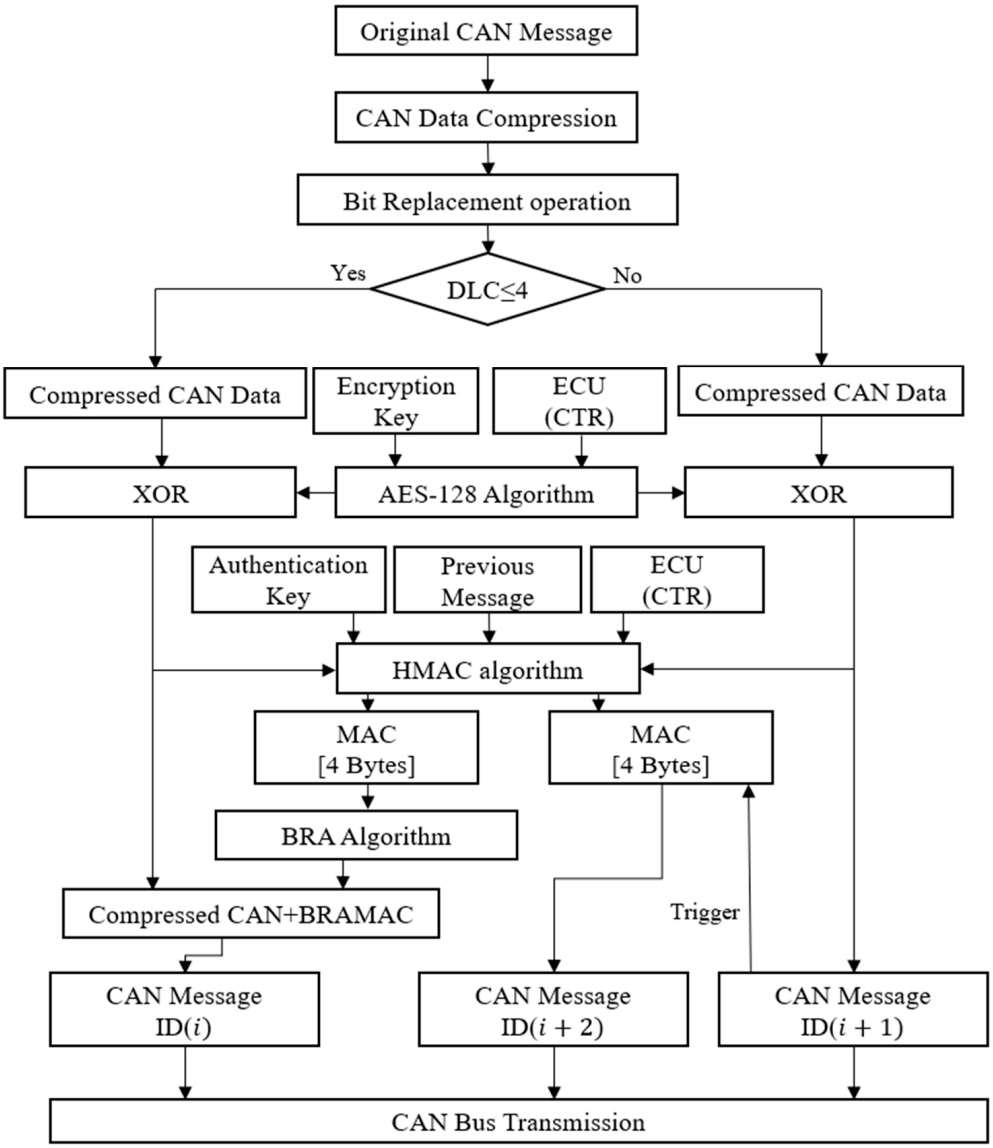

Figure 5. Proposed data transmission flow chart.

3.3. Session Key Management

After starting a vehicle, every ECU performs a session key derivation process with a Gateway ECU (GECU) in a fixed order. While GECU derives the session keys for a particular ECU, other ECUs belonging to the same subnetwork do not communicate, i.e., they wait for their turn. We used AKEP2 to construct a secure and efficient key derivation process in the in-vehicle CAN environment; it provides mutual entity authentication and implicit key distribution [12,30].

The process of AKEP2 performance consists of a 3-way handshake. The distribution of the session keys is shown in Figure 7. The output size of HKDFx() is 256 bits. The leftmost 128 bits are used as an authentication key (used for HMAC algorithm) and the rightmost 128 bits as an encryption key (used for AES algorithm).

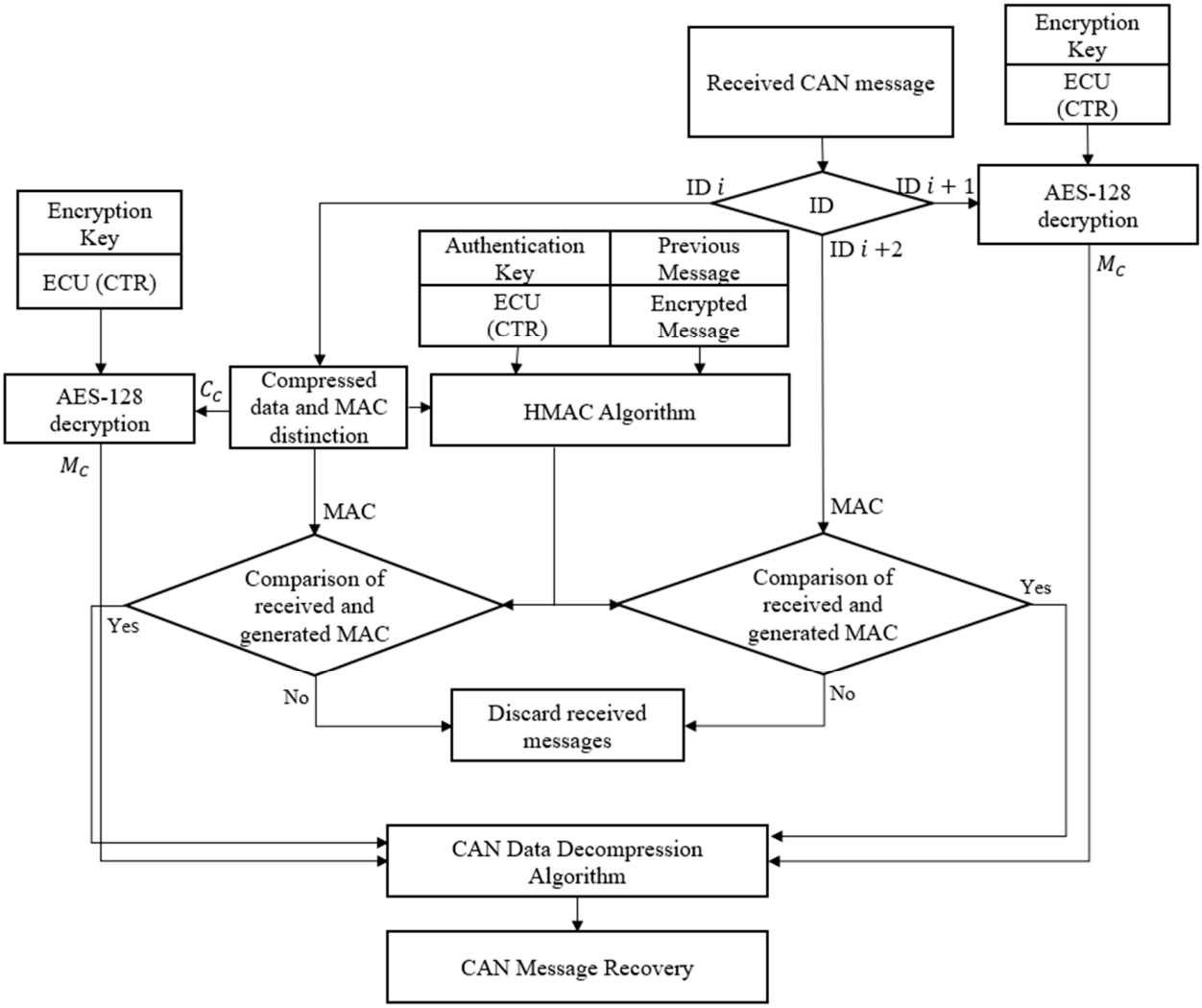

Figure 6. Proposed data reception flow chart.

In the k_{th} session

Handshake 1 : ECU_i ●——— R_i ———→ GECU (1) $MAC_1 = H_{SKi}(ECU_i \,||GECU||\, R_i \,||\, Seed_K)$

Handshake 2 : ECU_i ←— $MAC_1||Seed_k$ ——● GECU (2) $HKDF_{KGK}(Seed_K) = AK_K \,||\, EK_K$

Handshake 3 : ECU_i ●——— MAC_2 ———→ GECU (3) $MAC_2 = H_{SKi}(ECU_i \,||\, Seed_{1K})$

Notation

ECU_i	ECU using identity "i"	$HKDF_x(\,)$	H_X : {0,1}* x KEY ➔ {0, 1}^{256}
R_i	Random number	AK_K	Authentication key of k_{th} session
$Seed_K$	Seed value of k_{th} session	EK_K	Encryption key of k_{th} session
SKi	Long-term symmetric key between ECU_i and GECU(authentication key)	KGK	Long-term symmetric key between ECU_i and GECU(key generation key)

Figure 7. 3-way handshake for the session keys (authentication key and encryption key).

4. Simulation

In this section, we present the MATLAB simulation results using CAN data from Kia Sorento vehicle and an LS Mtron tractor. MATLAB simulations were performed on a computer equipped with an Intel(R) i5-9600KF CPU @3.70Ghz. The transmission and reception flows shown in Figures 5 and 6 were programmed in MATLAB. The automobile dataset was used for the AI/ML based driver classification challenge track in '2018 Information Security R&D dataset challenge' in South Korea and can be downloaded from the site introduced in [31].

The compression ratio in this simulation is defined as follows:

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

Tables 5 and 6 show the length (in bytes) of compressed data using the Triple ID algorithm and the proposed algorithm in a Kia Sorento, respectively. A total of 1,295,920 frames with 8 CAN IDs were used for the simulation. The simulation results in Tables 5 and 6 are summarized in Tables 7 and 8.

Table 5. Compressed data length for a KIA SORENTO using Triple ID algorithm.

ID	No. of Frames	No. of Occurrences by Compressed Data Length								
		0 Byte	1 Byte	2 Byte	3 Byte	4 Byte	5 Byte	6 Byte	7 Byte	8 Byte
260	163,078	132,141	0	30,601	336	0	0	0	0	0
2A0	163,078	162,433	0	645	0	0	0	0	0	0
316	163,078	55,299	0	64,884	30,078	5989	3433	1640	1690	65
329	163,078	130,940	0	23,671	8307	156	4	0	0	0
43F	199,531	110,188	0	87,408	1891	20	24	0	0	0
440	199,531	26,161	0	155,111	17,362	855	42	0	0	0
545	163,078	34,328	0	51,314	77,426	10	0	0	0	0
580	81,468	56,799	0	16,050	5471	1809	1001	338	0	0
Total	1,295,920	708,289 (54.66%)	0 (0.00%)	429,684 (33.16%)	140,871 (10.87%)	8839 (0.68%)	4504 (0.35%)	1978 (0.15%)	1690 (0.13%)	65 (0.00%)

Table 6. Compressed data length for a KIA SORENTO using the proposed algorithm.

ID	No. of Frames	No. of Occurrences by Compressed Data Length								
		0 Byte	1 Byte	2 Byte	3 Byte	4 Byte	5 Byte	6 Byte	7 Byte	8 Byte
260	163,078	162,380	698	0	0	0	0	0	0	0
2A0	163,078	163,078	0	0	0	0	0	0	0	0
316	163,078	111,928	34,911	10,410	3607	1047	1115	60	0	0
329	163,078	140,279	22,545	252	2	0	0	0	0	0
43F	199,531	196,204	3122	198	7	0	0	0	0	0
440	199,531	175,780	22,410	1330	11	0	0	0	0	0
545	163,078	62,068	101,002	8	0	0	0	0	0	0
580	81,468	69,972	8020	2446	419	611	0	0	0	0
Total	1,295,920	1,081,689 (83.47%)	192,708 (14.87%)	14,644 (1.13%)	4046 (0.31%)	1658 (0.13%)	1115 (0.09%)	60 (0.00%)	0 (0.00%)	0 (0.00%)

Table 7 shows the difference in compression ratios between the Triple ID algorithm and the proposed algorithm for each ID. It can be seen that a higher compression ratio can be obtained by the proposed algorithm, ranging from a maximum of 17.94% to a minimum of 0.10% (average 8.07%).

In the case of the Triple ID algorithm, a total of 10,367,360 bytes in the data field are compressed into 1,335,718 bytes, showing a compression ratio of 89.54%. On the other hand,

in the case of the proposed algorithm, the same CAN data is compressed into 247,399 bytes, showing a compression ratio of 97.61%. Therefore, an additional compression ratio of 8.07% can be obtained by the proposed algorithm.

Table 7. Comparison of compression ratios for a Kia Sorento.

ID	No. of Original Bytes	Compressed Bytes (Compression Ratio)		Difference
		Triple ID	Proposed	
260	1,304,624	62,210 (95.23%)	1396 (99.89%)	4.66%
2A0	1,304,624	1290 (99.90%)	0 (100%)	0.10%
316	1,304,624	283,313 (78.28%)	76,675 (94.12%)	15.84%
329	1,304,624	72,907 (94.41%)	23,055 (98.23%)	3.82%
43F	1,596,248	180,689 (88.68%)	3539 (99.78%)	11.10%
440	1,596,248	86,738 (94.57%)	25,103 (98.42%)	3.85%
545	1,304,624	334,946 (74.32%)	101,018 (92.26%)	17.94%
580	651,744	62,782 (90.37%)	16,613 (97.45%)	7.08%
Average	1,295,920	135,609 (89.54%)	30,925 (97.61%)	8.07%

Table 8. Comparison based on compressed data length greater than 4 bytes for a Kia Sorento.

ID	Compressed Data Length Greater than 4 Bytes	
	Triple ID	Proposed
316	6828	2222
329	4	0
43F	24	0
440	42	0
580	1339	0
Total	8237 (100%)	2222 (26.98%)

As explained in Section 2.4, it is important to compress CAN data to 4 bytes or less for communication efficiency. Table 8 summarizes the number of occurrences where the compressed data length exceeds 4 bytes. It can be seen that the proposed algorithm reduces the occurrences of compressed data exceeding 4 bytes by 73.02%, compared to the Triple ID algorithm.

A simulation using an LS Mtron tractor was performed to show that the proposed algorithm can be applied to various industrial machines. Table 9 shows that the proposed algorithm can achieve higher compression ratios compared to the Triple ID algorithm, ranging from a maximum of 22.76% to a minimum of 12.50% (average 17.44%).

Table 10 shows a comparison based on the compressed data length greater than 4 bytes for an LS Mtron tractor. It can be seen that the proposed algorithm reduces the occurrences of compressed data exceeding 4 bytes by 99.57%, compared to the Triple ID algorithm.

The bus load is defined as

$$\text{Busload}(\%) = \frac{\text{used capacity}}{\text{maxcapacity}} = \frac{\#\text{ bits sent}}{\text{speed}}. \quad (8)$$

Peak load is defined as the maximum bus load. In this paper, CANoe is used to calculate the peak load of the CAN bus using Kia Sorento CAN data at the speed of 500 kbps.

Figure 8 shows the peak load test environment. In Figure 7, 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 waveform monitoring. Table 11 shows the peak load of the CAN system with the proposed algorithm is only 64.8% of that of the CAN system with the Triple ID algorithm.

Table 9. Comparison of compression ratios for an LS MTRON tractor.

ID	No. of Original Bytes	Compressed Bytes (Compression Ratio)		Difference
		Triple ID	Proposed	
CF00400	164,704	66,346 (59.71%)	28,872 (82.47%)	22.76%
CFE4523	32,904	4163 (87.34%)	4 (99.99%)	12.65%
18FE5600	3288	421 (87.19%)	0 (100.00%)	12.81%
18FF2100	65,872	8238 (87.49%)	0 (100.00%)	12.51%
18FF6121	65,448	8181 (87.50%)	0 (100.00%)	12.50%
18FEF200	32,936	10,781 (67.26%)	4321 (86.88%)	18.02%
18FF9E21	65,456	8497 (87.02%)	13 (99.98%)	12.96%
19FFA010	65,648	14,191 (78.38%)	32 (99.95%)	21.57%
19FFA030	32,736	5153 (84.26%)	530 (98.38%)	14.12%
Average	58,777	13,997 (76.18%)	3752 (93.62%)	17.44%

Table 10. Comparison based on compressed data length greater than 4 bytes for an LS Mtron tractor.

ID	Compressed Data Length Greater than 4 Bytes		No. of Frames
	Triple ID	Proposed	
CF00300	1007	0	8234
CF00400	1469	12	19,119
CFE4523	50	0	4113
18FE4321	6	0	4090
18FEDF00	107	0	8127
18FEF200	137	0	4036
19FFA030	12	0	4092
Total	2788 (100%)	12 (0.43%)	51,811

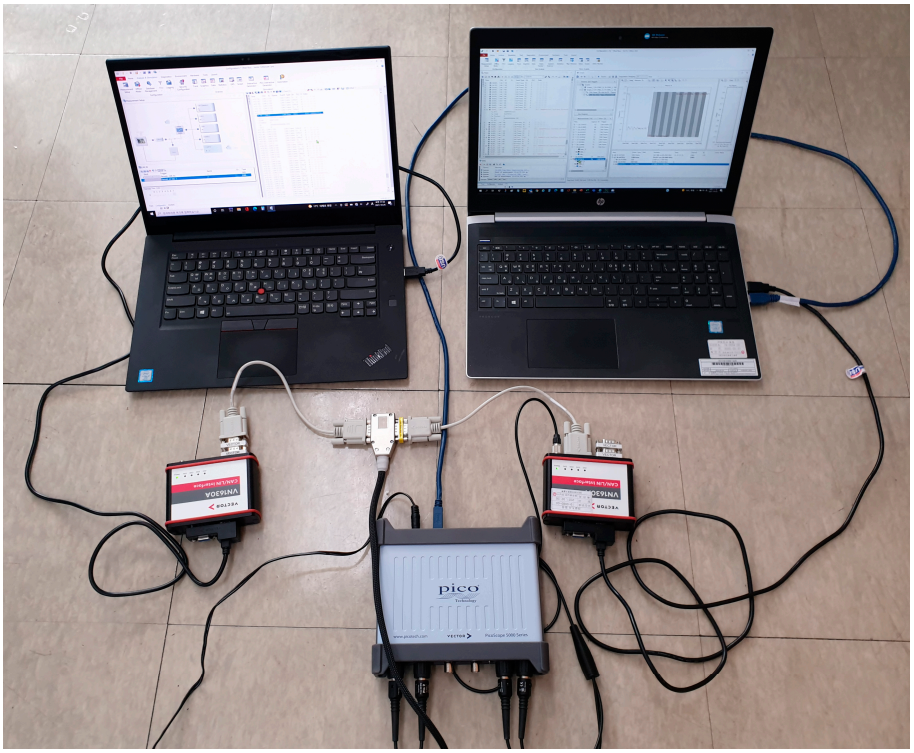

Figure 8. Peak load simulation using CANoe system.

Table 11. Comparison of the peak loads.

	Triple ID	Proposed
Peak load	29.27% (1)	18.99% (0.648)

The transmission part of the proposed algorithm inverts 4 bits in 4 bytes according to the 12-bit value after performing the Triple ID algorithm, and the receiving part performs the opposite process. Therefore, the calculation cost of the proposed algorithm is slightly higher than that of the Triple ID algorithm. However, compared to the entire process of the Triple ID algorithm, the cost of the additional computational process required by the proposed algorithm is almost negligible, so the computational cost of the proposed algorithm is almost the same as that of the Triple ID algorithm.

5. Conclusions

We proposed a bit replacement algorithm that further increases the compression efficiency of the Triple ID method by combining CAS map and MAC. In the Triple ID algorithm, it is very important to compress CAN data to 4 bytes or less for communication efficiency. The proposed method reduced the occurrence of compressed messages exceeding 4 bytes by up to 99.57% compared to the Triple ID method. Simulation using a CANoe system shows that the peak load of the CAN system with the proposed algorithm is only 64.8% of that of the CAN system with the Triple ID algorithm.

The Triple ID algorithm provides sufficient authentication performance without changing the CAN protocol. In addition to the authentication performance, the proposed algorithm further improves the communication efficiency of the Triple ID algorithm. Therefore, the proposed algorithm is expected to be usefully applied to CAN systems, including recent connected cars, where security vulnerabilities are of great concern.

The proposed method differs from the modified MAC method in that it improves compression performance by moving not only header bits but also part of the data information to MAC. This is an important step forward, as the proposed method is applicable to other compression algorithms as well.

Future research should focus on applying the proposed method to other communication systems such as FlexRay.

Author Contributions: S.J. and J.P., writing—original draft; D.-H.S., formal analysis; S.W.—review and editing; J.-G.C.—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: On behalf of all the authors, the corresponding author states that our data are available upon reasonable request.

Conflicts of Interest: The authors declare no conflict 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.* **2021**, *23*, 6123–6141. [CrossRef]
2. Valasek, C.; Miller, C. Adventures in automotive networks and control units. *Def. Con.* **2013**, *21*, 15–31.
3. 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; USENIX Association: San Francisco, CA, USA, 2011; pp. 447–462.
4. Mandal, A.K.; Panarotto, F.; Cortesi, A.; Ferrara, P.; Spoto, F. Static analysis of Android Auto infotainment and on-board diagnostics II apps. *Software: Pract. Exp.* **2019**, *49*, 1131–1161. [CrossRef]

5. Jo, H.J.; Choi, W.; Na, S.Y.; Woo, S.; Lee, D.H. Vulnerabilities of Android OS-Based Telematics System. *Wirel. Pers. Commun.* **2016**, *92*, 1511–1530. [CrossRef]
6. Nie, S.; Liu, L.; Du, Y. Free-fall: Hacking tesla from wireless to CAN bus. *Black Hat USA* **2017**, *25*, 1–16.
7. Miller, C.; Valasek, C. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA* **2015**, *2015*, 9.
8. 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. [CrossRef]
9. 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. [CrossRef]
10. 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]
11. 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]
12. 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]
13. 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]
14. Kim, Y.-J.; Woo, S.; Chung, J.-G. Triple ID flexible MAC for CAN security improvement. *IEEE Access* **2021**, *9*, 126388–126399. [CrossRef]
15. 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]
16. 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]
17. 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]
18. 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. Informatics* **2018**, *14*, 2181–2193. [CrossRef]
19. Ramteke, P.R.; Mahmud, S.M. An Adaptive Data-Reduction Protocol for the Future In-Vehicle Networks. *SAE Tech. Pap.* **2005**, *114*, 519–530. [CrossRef]
20. Miucic, R.; Mahmud, S.M. An Improved Adaptive Data Reduction Protocol for In-Vehicle Networks. *SAE Tech. Pap.* **2006**, *115*, 650–658. [CrossRef]
21. Miucic, R.; Mahmud, S.M.; Popovic, Z. An Enhanced Data-Reduction Algorithm for Event-Triggered Networks. *IEEE Trans. Veh. Technol.* **2009**, *58*, 2663–2678. [CrossRef]
22. Misbahuddin, S.; Mahmud, S.M.; Al-Holou, N. Development and performance analysis of a data-reduction algorithm for automotive multi-plexing. *IEEE Trans. Veh. Technol.* **2001**, *50*, 162–169. [CrossRef]
23. Kelkar, S.; Kamal, R. Boundary of fifteen compression algorithm for controller area 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.
24. Wu, Y.-J.; Chung, J.-G. Efficient controller area network data compression for automobile applications. *Front. Inf. Technol. Electron. Eng.* **2015**, *16*, 70–78. [CrossRef]
25. Wu, Y.; Chung, J.-G. An Improved Controller Area Network Data-Reduction Algorithm for In-Vehicle Networks. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2017**, *100*, 346–352. [CrossRef]
26. Kim, Y.-J.; Zou, Y.; Kim, Y.-E.; Chung, J.-G. Multi-level data arrangement algorithm for can data compression. *Int. J. Automot. Technol.* **2020**, *21*, 1527–1537. [CrossRef]
27. Abdulmalik, H.; Luo, B. Using ID-hopping to defend against targeted DoS on CAN. In Proceedings of the 1st International Workshop on Safe Control of Connected and Autonomous Vehicles, Pittsburgh, PA, USA, 18–21 April 2017; pp. 19–26.
28. Lukasiewycz, M.; Mundhenk, P.; Steinhorst, S. Security-Aware Obfuscated Priority Assignment for Automotive CAN Platforms. *ACM Trans. Des. Autom. Electron. Syst.* **2016**, *21*, 32. [CrossRef]
29. Hwang, D.Y.; Kim, Y.J.; Chung, J.G. CAN security protocol using modified MAC. In Proceedings of the 2020 International SoC Design Conference (ISOCC), Yeosu, Republic of Korea, 21–24 October 2020; pp. 308–309.
30. Bellare, M.; Rogaway, P. Entity Authentication and Key Distribution. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 2001; pp. 232–249. [CrossRef]
31. Available online: https://ocslab.hksecurity.net/Datasets/driving-dataset (accessed on 1 October 2022).