Triple ID Flexible MAC for Can Security Improvement

YEON-JIN KIM\textsuperscript{1}, (Member, IEEE), SAMUEL WOO\textsuperscript{2}, AND JIN-GYUN CHUNG\textsuperscript{1}\textsuperscript{1} Division of Electronic Engineering, IT Convergence Research Center, Jeonbuk National University, Jeonju 54896, South Korea
\textsuperscript{2}Department of Software Science, Dankook University, Yongin, Gyeonggi-do 16891, South Korea
Corresponding author: Jin-Gyun Chung (jgchung@jbnu.ac.kr)

ABSTRACT The development of automotive-information & communication technology (ICT) convergence has resulted in various vehicle-based electrical/electronic (E/E) systems. An automotive E/E system consists of one or more electronic control units (ECUs), sensors and actuators. With the commercialization of connected/autonomous cars, vehicle-based wireless communication systems have appeared, and are expected to grow in popularity. This increases the number of attack surfaces that can potentially threaten in-vehicle controller area network (CAN). To combat the vulnerabilities of the CAN protocol, the CAN data field should be encrypted and transmitted with authentication codes. Recently, a method of transmitting authentication codes by modifying the CAN protocol was proposed. However, changing the original CAN protocol can cause serious problems in CAN systems. In this paper, to enhance CAN security, a data compression algorithm is used to reduce the data frame length so that there is space for a message authentication code (MAC) to be contained inside the data field. The proposed algorithm guarantees that all CAN frames are authenticated by a MAC of at least four bytes without any change of the original CAN protocol. Simulations using CAN data from Kia Sorento, Kia Soul, and LS Mtron vehicles show that the proposed algorithm works successfully with only a slight increase in the peak load.

INDEX TERMS CAN, data reduction, authentication code, vehicle communication, data rearrangement.

I. INTRODUCTION
The CAN system was introduced by Robert Bosch GmbH in 1986 [1]. As the most representative in-vehicle network, the CAN system has become the de facto standard because it dramatically decreases the number of communication lines required and ensures higher data transmission reliability [2], [3].

Recently, various kinds of electrical and electronic systems have been installed in vehicles. More than 70 ECUs are often used in a vehicle. Since the CAN bus load is increased by the large number of ECUs, it is necessary to reduce the CAN frame length. Furthermore, in-vehicle networks are connected to external networks [4], [5]. Unfortunately, information security was not considered in the initial CAN design, although every bit of information transmitted could be critical to driver safety [6]–[11].

To enhance security, encryption and authentication of CAN data frames are necessary [12], [13]. To provide confidentiality for the data frame, it should be encrypted before transmission. In addition, to authenticate a transmitted data frame, a MAC should be generated and transmitted along with the encrypted message.

To transmit the MAC, additional data space is required in the CAN frame. In [14], a truncated 32-bit MAC is inserted inside the extended ID and CRC field. However, this changes the original definition of the CAN protocol, which can cause severe problems in the CAN systems.

In [15], the Mini-MAC algorithm uses part of a data field to transmit a MAC. If an ECU uses a data field of less than five bytes, the truncated Mini-MAC data can be transmitted using the empty space in the CAN data field. However, the Mini-MAC algorithm does not authenticate messages with eight-byte payloads.

Since the CAN data field is limited to 64 bits, it is better to reduce the data size and combine it with the MAC than to modify the CAN frame format.

In [16], after the CAN data field is compressed using the efficient controller area network data compression (ECANDC) algorithm, a one-byte MAC is inserted in the...
TABLE 1. Notation used for the proposed algorithm.

| Notation      | Description                                        |
|---------------|----------------------------------------------------|
| $E_{CU_i}$    | ECU using identity $i$                             |
| $E_{K_k}$     | Encryption key of the $k_{th}$ session             |
| $A_{K_k}$     | Authentication key of the $k_{th}$ session         |
| $H_x$         | Keyed hash function using $x$                      |
| $CTR_{E_{CU_i}}$ | $E_{CU_i}$ data frame counter                      |
| $C$           | Ciphertext                                        |
| $C_c$         | Ciphertext generated from compressed data         |
| $C_{cn}$      | $n$-th frame of ciphertext generated from compressed data |
| $M$           | Plaintext                                         |
| $M_c$         | Compressed data                                   |
| $M_n$         | $n$-th frame of plaintext                          |
| $L(\cdot)$    | Length of $\cdot$                                 |
| $LC[i]$       | $i$-th bit of the compressed message length        |

empty space obtained by the compression algorithm. However, a one-byte MAC does not have sufficient authentication capability for the data.

CAN data compression algorithms are divided into two classes. The first class of algorithms uses a predefined maximum difference value (PMDV) [17]–[21]. The compression efficiency depends on the accuracy of the PMDV selected for an application. However, it is very difficult to optimize the selection of the PMDV. Actually, the PMDV needs to be adjusted depending upon automobile driving conditions to achieve the optimum compression efficiency.

The second class of algorithms uses a compression area selection (CAS) map without using the PMDV [22]–[24]. The size of the CAS map changes depending on the automobile driving conditions, which corresponds to a PMDV change. Thus, this second class of compression algorithms shows better compression efficiencies than the first class.

Recently, a multi-level data arrangement (MLDA) algorithm was proposed based on the CAS map [24]. While the other compression algorithms in this class use an eight-bit signal arrangement, the MLDA algorithm uses a multi-level signal arrangement to increase the compression efficiency.

In the MLDA algorithm, the sender ECU does not transmit CAN frames when the difference between the current and the previous CAN data is zero, which reduces the CAN bus load. However, this method creates a problem in which the receiver ECU is not sure whether the transmitter ECU is disconnected or not. In this paper, we first propose a Vehicle MLDA (VMLDA) algorithm that overcomes some of the disadvantages of the MLDA algorithm in vehicle environments. Based on this VMLDA algorithm, we propose a CAN security improvement method.

By the proposed algorithm, all CAN frames are authenticated by a MAC of at least four bytes, without making any changes to the original CAN protocol.

In Section II, a brief review of the compression algorithms and existing security protocols. In Section III, we propose a flexible authentication method for CAN security improvement. In Section IV, we show the simulation results using actual CAN data. Finally, brief conclusions are given in Section V.

II. PREVIOUS WORK

CAN data compression takes advantage of the fact that successive CAN frames with the same message identifier do not change rapidly [17]–[24]. In PMDV-based CAN data compression algorithms, the difference value is only transmitted if the difference between the current and preceding CAN messages does not exceed the PMDV. If the difference value is greater than the PMDV, the current frame value is transmitted without any modification. The compression efficiency depends on the accuracy of the PMDV for a specific application. However, it is very difficult to choose an optimal PMDV. Indeed, the best compression efficiency can be achieved if the PMDV can be adjusted depending on the operating conditions of a given system. However, since the PMDV is fixed at the installation stage of the CAN system, it is not possible to change the PMDV.

As explained in Section I, CAS map-based compression algorithms outperform PMDV-based algorithms. Thus, in this section, we review only CAS map-based algorithms. Then, existing CAN authentication algorithms are briefly reviewed. The notations used in this paper are listed in Table 1.

A. ICANDR ALGORITHM

In the improved CAN data reduction (ICANDR) algorithm, a CAS map eliminates the need to predict the maximum difference value. When the CAN signal is transmitted, a CAS map is used for compression and decompression. As shown in [23], the ICANDR algorithm achieves a higher compression efficiency than other PMDV-based methods. In the ICANDR algorithm, once the CAN data arrangement procedure is completed at the manufacturing or installation stage of the CAN system, the arrangement for a specific ID is maintained. To improve the compression efficiency, as shown in Fig. 1, the 64-bit CAN data field is divided into three signals: 24-bit (Sig A), 24-bit (Sig B), and 16-bit (Sig C) signals. Table 2 shows an example of bit-wise XOR calculation and header bit allocation. After CAN data are divided into three signals, the bit-wise XOR values between the current and the previous data fields are computed. If the calculated XOR value of a signal is nonzero, the corresponding header bit is set to one. Otherwise, the corresponding header bit is set to zero.
After the three header bits are determined, the header bits are placed in the last column beginning in the first row in a CAS map, as shown in Table 3. Then, starting from the next row, 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 signal is emptied, as shown in Table 2. The CAS map in Table 3 is arranged using the memory map in Table 4.

Thus, in this example, eight-byte data are compressed into only three bytes. If all three header bits are zero, the sender ECU does not transmit the CAN data frame.

**B. MLDA ALGORITHM**

In the ICANDR algorithm, when CAN data bits are arranged using the byte-level arrangement map shown in Fig. 1, slowly changing data should be placed in the most significant parts of the three signals, while frequently changing data should be placed in the least significant parts [24]. Although different arrangements can result in different compression efficiencies, the ICANDR algorithm does not provide any systematic signal arrangement procedure.

In addition to the byte-level arrangement, the MLDA algorithm presents four-bit-level, two-bit-level, and one-bit-level arrangements. Fig. 2 shows the two-bit-level arrangement map. The MLDA algorithm provides a systematic way to place the CAN data bits using multi-level arrangement maps to obtain the best compression efficiency.

As an example of byte-level arrangement, eight bytes of each data field are denoted as $B(n)$ ($0 \leq n \leq 7$) from left to right, as shown in Table 5. The XORed values between two...
successive frames $F_i$ and $F_{i-1}$ are first computed; these are called magnitude values (MV). Then, the frequency values (FV) are computed from the magnitude values. If a magnitude value is nonzero, the corresponding frequency value is one. Otherwise, it is zero. $S(m)$ is the sum of the magnitude values and $S_f(n)$ is the sum of the frequency values for each $B(n)$.

Then, $S_m(n)$ is defined as follows:

$$S_m(n) = S_f(n) + \lambda \times S_m(n), \quad (1)$$

where $\lambda$ is a weighting factor between $S_f(n)$ and $S_m(n)$. Optimal $\lambda$ values are selected by comparing the compression ratios obtained by each $\lambda(0 \leq \lambda \leq 3)$. $S_m(n)$ values are used to determine the position of each data byte on the arrangement map.

### C. PRACTICAL SECURITY ALGORITHM

In [14], AES-128 and hash-based message authentication (HMAC) are used for CAN data encryption and authentication, respectively. The cipher text ($C$) is obtained from plaintext ($M$) as

$$C = E_{EK_i}(CTR_{ECU_i}) \oplus M, \quad (2)$$

where $E_{EK_i}$ represents AES-128 encryption using the $k$-th session encryption key $EK_i$, $CTR_{ECU_i}$ is the message counter value of the sender ECU ($ECU_i$), and $\oplus$ denotes an exclusive OR (XOR) operation. An ECU should manage the data frame counter values for all received and transmitted data frames related to it.

The result of AES-128 encryption $E_{EK_i}(CTR_{ECU_i})$ is 128 bits. As the maximum size of the CAN data payload is 64 bits, only the first 64 bits are used to generate $C$.

$ECUs$ generates MAC data for the CAN data frame as

$$MAC = H_{AK_i} (ID_s \parallel C \parallel CTR_{ECU_i}), \quad (3)$$

where $H_{AK_i}$ denotes a keyed-hash function using the $K_{th}$ session authentication key $AK_i$, $ID_s$ denotes the identity of $ECUs$, and $\parallel$ denotes a concatenation operation.

### D. MINI-MAC ALGORITHM

The Mini-MAC algorithm is a group-keyed, variable-length, truncated HMAC that uses a counter and recent message history [15].

The Mini-MAC tag with authentication key $k$ is computed as

$$\text{Mini-MAC} = \text{Trunc}[s, H_k(k \parallel M_n \parallel CTR_{ECU_i} \parallel M_{n-\lambda} \parallel \cdots \parallel M_{n-1})], \quad (4)$$

where the Trunc[$s$, ·] function extracts the $s$ most significant bits of its input, which is recommended by NIST for truncating a hash tag [25]. Additionally, $(M_{n-\lambda} \cdots M_{n-5})$ is the sequence of the most recent valid $\lambda$ messages.

Message-digest algorithm 5 (MD5) is used in the Mini-MAC algorithm. The input size of MD5 is 64 bytes. Although the recommended $\lambda$ value is 16, the Mini-MAC algorithm actually uses $\lambda = 5$. For 64-byte input, the counter and key require 8 bytes and 16 bytes, respectively. Thus, 40 bytes are allocated to six messages $(M_n, M_{n-1} \cdots M_{n-5})$ for $\lambda = 5$, meaning that each message assumes an average of 6.67 bytes.

The Mini-MAC algorithm uses part of a data field to transmit the MAC. In a Toyota Prius, most ECUs use a data field that is smaller than five bytes. Therefore, the truncated Mini-MAC data can be transmitted using the empty space in the CAN data field.

However, the Mini-MAC algorithm does not authenticate messages with eight-byte payloads. In addition, if the length of the MAC is not sufficiently long, the authentication capability is not strong enough to protect the CAN systems.

### III. PROPOSED CAN SECURITY PROTOCOL

In this section, we propose a flexible MAC algorithm for CAN security improvement. The goals of the proposed algorithm are as follows:

- The original CAN protocol is not changed.
- The length of a MAC is always at least four bytes long.
- The increase in the peak load caused by adding MACs is very small or negligible.

### A. MESSAGE COMPRESSION BY VMLDA AND ENCRYPTION

To transmit a MAC without changing the original CAN frame format, the length of the truncated MAC should be limited by the size of the CAN data in the data field. As an example, if the CAN data size with a specific ID is four bytes, a four-byte MAC can be transmitted. However, if the CAN data size is eight bytes, MAC data cannot be transmitted. Thus, we propose the VMLDA compression method to secure space for MAC.

As indicated in Section I, in the MLDA algorithm, the sender ECU does not transmit CAN frames when the difference between the current and the previous CAN data is zero. It is clear that this procedure reduces the bus load. However, this method creates a problem in which the receiver ECU is not sure whether the transmitter ECU is disconnected or not.
To overcome this problem, in the VMLDA algorithm, when CAN data is compressed to zero bytes, the data length code (DLC) of the CAN frame is set to one and one byte of zero data is sent within the data field. This is reasonable because in the MLDA algorithm, when CAN data is compressed to zero bytes, all header bits become zero and bitwise XOR values are also zero. By doing this, the receiver ECU can be sure that the sender ECU is not disconnected while minimizing the increase in busload.

Fig. 4(a) shows the data compression procedure of the MLDA and ICANDR algorithms. As can be seen from the figure, the first message of each ID is transmitted without compression since there is no previous message.

In VMLDA algorithm, an optimized initial message (OIM) is provided for each ID to compress the first message, as shown in Fig. 4(b). The OIM for each ID should be selected such that the bit-wise XOR value between the first message and the OIM is minimized in magnitude.

The use of an OIM helps to reduce the peak bus load when MAC data are transmitted using a triple ID, which will be explained in Section IV.

As explained in the previous section, the MLDA algorithm uses multiple bit arrangement levels to increase the data compression ratios. Fig. 5 shows typical compression ratios achieved by different arrangement levels [24]. As can be seen from the figure, compression ratios usually improve as arrangement levels change from eight bit to two bit.

However, the improvement obtained by changing arrangement levels from two bit to one bit is very small, while the data processing time increases from 102.8% to 104.1%, as compared with that of the eight-bit level [24]. Thus, in VMLDA algorithm, we adopt two-bit-level arrangement for data compression.

After data compression, the compressed ciphertext \( C_c \) is obtained as follows:

\[
C_c = E_{E_{K_1}}(CTR_{E_{CU_1}}) \oplus M_c, \quad (5)
\]

where \( M_c \) is compressed data using the proposed data compression algorithm. Notice that the length of \( C_c \) is equal to \( M_c \).

In the proposed method, it is assumed that an ECU maintains a message counter for each ID for all transmitted/received messages related to the ECU. Thus, the encrypted counter value \( E_{E_{K_1}}(CTR_{E_{CU_1}}) \) can be generated at all related receivers. Then, from the received cipher text, compressed data can be decrypted as

\[
M_C = C_C \oplus E_{E_{K_1}}(CTR_{E_{CU_1}}). \quad (6)
\]

Thus, in the proposed method, decryption is actually performed via encryption in the receiving ECU.

### B. TRIPLE ID METHOD

When the CAN data field is compressed to less than five bytes, compressed data are transmitted using ID \( i \). In this case,
FIGURE 5. Typical compression ratios with different bit-level arrangements by the MLDA algorithm \((L = \text{bit level})\).

TABLE 6. Triple-ID according to length of the compressed data.

| ID | Compressed data length (byte) | Transmitted data type |
|----|-----------------------------|-----------------------|
| \(i\) | \(L(M_c) = 0\) | DLC of the CAN frame is set to one and one byte of zero data is sent within the data field |
| \(i\) | \(L(M_c) \leq 4\) | Compressed data and MAC data are transmitted in the same data field with \(L(\text{MAC}) = 8 - L(M_c)\) |
| \(i + 1\) | \(4 < L(M_c) \leq 7\) | Compressed data are transmitted |
| \(i + 2\) | \(7 < L(M_c)\) | Original eight-byte data are transmitted |
| \(i + 2\) | \(4 < L(M_c)\) | MAC data are transmitted with \(L(\text{MAC}) = 8\) |

CAN data and MAC data are transmitted in the same data frame. The length of the MAC data is as follows:

\[
L(\text{MAC}) = 8 - L(M_c), \quad (7)
\]

where \(L(M_c)\) and \(L(M_e)\) are the length of the MAC data and compressed data, respectively.

When the length of the compressed data is more than four bytes, four-byte MACs cannot be transmitted in the same data field. To solve this problem, triple IDs are used, as shown in Table 6.

If the length of the compressed CAN data is more than four bytes and less than or equal to seven bytes, compressed data are transmitted using ID \((i + 1)\) and eight-byte MACs are transmitted using ID \((i + 2)\). If the length of the compressed CAN data is more than or equal to eight bytes, the original CAN data are transmitted using ID \((i + 1)\) and eight-byte MACs are transmitted using ID \((i + 2)\).

In the ICANDR algorithm, three signal header (HA, HB, HC) bits are used to distinguish whether Sig A, Sig B, or Sig C are changed or not. In the proposed algorithm, three lengths of compressed message (LC) bits are also used, as shown in Fig. 6. While the data length code (DLC) represents the length of the data in the CAN data field, the number of LC bits represents the length of compressed data in bytes. Thus, the MAC size can be inferred based on the LC bits.

C. AUTHENTICATION CODE

Fig. 7 shows the proposed MAC generation procedure. The proposed MAC tag with authentication group key \(k\) is computed as

\[
\text{Proposed MAC} = \text{Trunc}[s, \text{H}_k(k \parallel C_{\text{ct}n} \parallel C_{\text{ct}n-1} \cdots \parallel C_{\text{ct}n-\lambda})], \quad (8)
\]

where \(C_{\text{ct}n}\) is the current encrypted compressed message and \(\{C_{\text{ct}n-1}, \ldots, C_{\text{ct}n-\lambda}\}\) are the previously transmitted \(\lambda\) messages.
Thus, to intrude into the CAN system, malicious attackers need to collect not only the key but also previous messages. Generally, timestamps, nonces (random sequence number), or counters are used to protect a secure communication environment from a replay attack [14], [26]. In our proposed method, to prevent a replay attack we used the sender’s message counter to generate a MAC tag. Unlike in (4), encrypted compressed messages are used in (8).

As mentioned in Section II.D, the recommended value of \( \lambda \) is 16. However, in [15], the actual \( \lambda \) value is 5, meaning that the most recent sequence of six valid messages can be used to generate a MAC.

Since the CAN data is compressed using the VMLDA algorithm in the proposed method, larger \( \lambda \) values can be used to increase the security level. In Section IV, simulation results using the actual CAN signals from a Kia Sorento show that an 8-byte CAN signal is compressed to an average of 1.61 bytes. Therefore, since \( 40 \) bytes/1.61 bytes = 24.84, the recommended \( \lambda \) value of 16 can be easily met for compressed data.

D. TRANSMISSION AND RECEPTION

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

- Case I (compressed data length = 0): The data length code (DLC) of the CAN frame is set to one and one byte of zero data is sent within the data field
- Case II (0 < compressed data length \( \leq 4 \)): Compressed data and MAC data are transmitted in the same data field using ID \( i \) since the length of the MAC can be larger than or equal to four bytes in this case.
- Case III (4 < compressed data length \( \leq 7 \)): The available space for a MAC is less than four bytes in this case. Thus, eight-byte MAC data are transmitted separately using ID \( (i+2) \), while the compressed data are sent using ID \( (i+1) \).
- Case IV (compressed data length > 7): The compression algorithm cannot reduce the original data in this case. Thus, eight-byte MAC data are transmitted separately using ID \( (i+2) \), while the original data are sent using ID \( (i+1) \).

Fig. 8 and Fig. 9 show the transmission and reception flowcharts of the proposed method, respectively. The proposed CAN message transmission procedure can be summarized as follows:

Step 0: In order to compress the \( N \)-th message, each ECU must store the \( (N-1) \)-th message (the most recently transmitted message) in an uncompressed form. Also, each ECU needs to store the most recently transmitted \( \lambda \) messages in compressed form for MAC generation using (8).

Step 1: (Compression) Data compression is carried out according to the procedure shown in Fig. 4(b). For the first message \( (N = 1) \), the OIM is used as the previous message. For other messages \( (N \geq 2) \), the \( (N-1) \)-th message is used as the previous message to compress the \( N \)-th message.

Step 2: (Encryption) After data compression, the compressed cipher text is obtained by (5). For encryption,
the AES-128 algorithm is used with the sender ECU’s session encryption key and message counter value.

Step 3: (MAC) Based on compressed data length, the corresponding transmission case is selected from among four cases (Cases I, II, III or IV). By (8), the MAC is then generated according to the selected case.

Step 4: (Transmission) Encrypted data and MAC are transmitted using the appropriate ID(s) according to the selected transmission case.

The proposed CAN message reception procedure can be summarized as follows:

Step 0: In order to decompress the N-th message, each ECU must store the (N-1)-th message (the most recently received message) in an uncompressed form. Also, each ECU needs to store the most recently received λ messages in compressed form for MAC generation using (8).

Step 1: (ID Check) Determine whether the ID of the received message belongs to ID i, ID (i + 1), or ID (i + 2).

Step 2: (Decryption) Messages with ID i and ID (i + 1) are decrypted using the process in Fig. 4(c). Then, for messages with ID i, the compressed message and MAC are separated according to the header bits LC[2], LC[1] and LC[0]. Messages with ID (i + 2) transmit MAC only, so no decryption step is required.

Step 3: (MAC) MAC data are generated by (8) and the generated MAC data are compared with the received MAC. Received messages are authenticated only if the received and the generated MAC data are the same.

Step 4: (Decompression) For message authentication, the decompression procedure in Fig. 4(c) is performed to obtain the original message.

As explained in subsection III.A, the first message for each ID i is not compressed in the VMLDA and ICANDR algorithms. In this case, if the proposed triple ID method is applied, two frames with ID (i + 1) and ID (i + 2) need to be transmitted for each ID i. Consequently, the peak load increases abruptly during the initialization stage since two frames are transmitted instead of one for each ID. Thus, by using the proposed OIM, an abrupt increase in the peak load can be avoided.

E. SESSION KEY MANAGEMENT

We suggested the session key management scheme based on Authenticated Key Exchange Protocol 2 (AKEP2) from our previous research [14], [27]. AKEP2 provides mutual entity authentication and key distribution [28]. AKEP2-based mutual authentication and session key distribution is general information, so we do not provide a detailed description.

F. SECURITY ANALYSIS OF FOUR-BYTE MAC

When our proposed compression algorithm is used, from an eight-byte data payload more than four bytes can be used for transmission of the MAC. That is, when a sender ECU transmits a data frame, it is possible to insert more than a four-byte MAC into the data frame.

In our previous research, we analyzed the security of four-byte MAC [14]. We demonstrated that four-byte MAC provides sufficient security in a vehicular environment.

A four-byte MAC seems too short from the perspective of modern cryptography. However, in cryptography, a large number of short secret values have already been utilized as authentication information. For instance, we use five-byte information as passwords in many online services. The size of passwords is very small compared with that of ordinary cryptographic private information, but it is possible to guarantee sufficient security in real applications since a limitation is put on the number of adversarial trials against a password. If an adversary fails to forge a password three or more times, the service provider can guarantee security by blocking use of the password.

We can also use smaller information for authentication. In one-time password (OTP) systems, a random password is used only once, unlike other general passwords. Therefore, based on the literature, six-decimal one-time tokens can be utilized in practical real-world applications, including security-sensitive services such as e-banking [29]. Since $10^{6}$ ($=2^{19}$) < $2^{32}$, CAN communication in the proposed scheme is at a security level sufficient for resisting impersonation attacks.

IV. SIMULATION

In this section, we present the simulation results of the proposed method obtained using actual CAN signals from Kia Sorento, Kia Soul, and LS Mtron vehicles. Kia Sorento data are actual driving data from 30 minutes of driving around the Korea Automotive Technology Institute [24]. The Kia Soul data are actual driving data from 10 drivers driving for 24 hours between Korea University and the Sangam World Cup Stadium [30]. The Kia Sorento and Kia Soul data are automobile CAN signals, while the LS Mtron data are tractor data obtained from driving around the test site of the Korea Automotive Technology Institute.

Table 7 shows the simulation results of a CAN network with eight ECUs in a Kia Sorento. For ID 260, all compressed frames are less than five bytes long for a total of 163,078 frames. Thus, for ID 260, compressed CAN data and MAC can be transmitted in the same CAN frame without any additional frames.

On the other hand, for ID 316, the lengths of 6,828 compressed frames (or 4.19% of the frames) are greater than four bytes for a total of 163,078 frames. Thus, for ID 316, compressed CAN data and MAC can be transmitted in the same CAN frame 95.81% of the time.

For the remaining 4.91%, two frames are required for each CAN data transmission because compressed data are sent using ID 317 (i.e., $i + 1$) and MAC data are sent using ID 318 (i.e., $i + 2$). Table 8 shows the comparison of occurrences based on a compressed data length of four bytes for each ID.
TABLE 7. The number of occurrences by length of compressed CAN data in a Kia Sorento.

| ID | Number of occurrences by compressed length (byte) | Total frames |
|----|-----------------------------------------------|--------------|
|    | 0 | 1 | 2 | 3 | 4 | More than 4 |
| 260 | 132,141 | 30,601 | 336 | 0 | 0 | 163,078 |
| (81.03%) | (18.76%) | (0.21%) | (0.00%) | (0.00%) | (100.00%) |
| 2A0 | 162,433 | 645 | 0 | 0 | 0 | 163,078 |
| (99.60%) | (0.40%) | (0.00%) | (0.00%) | (0.00%) | (100.00%) |
| 316 | 55,299 | 64,884 | 30,078 | 5,989 | 6,828 | 163,078 |
| (33.91%) | (39.79%) | (18.44%) | (3.67%) | (4.19%) | (100.00%) |
| 329 | 130,940 | 23,671 | 8,307 | 156 | 4 | 163,078 |
| (80.29%) | (14.52%) | (5.09%) | (0.10%) | (0.00%) | (100.00%) |
| 43F | 110,188 | 87,408 | 1,891 | 20 | 24 | 199,531 |
| (55.22%) | (43.81%) | (0.95%) | (0.01%) | (0.01%) | (100.00%) |
| 440 | 26,161 | 155,111 | 17,362 | 855 | 42 | 199,531 |
| (13.11%) | (77.74%) | (8.70%) | (0.43%) | (0.02%) | (100.00%) |
| 545 | 34,328 | 51,314 | 77,426 | 10 | 0 | 163,078 |
| (21.05%) | (31.47%) | (47.48%) | (0.01%) | (0.00%) | (100.00%) |
| 580 | 56,799 | 16,050 | 5,471 | 1,809 | 1,339 | 81,468 |
| (69.72%) | (19.70%) | (6.72%) | (2.22%) | (1.64%) | (100.00%) |

TABLE 8. Number of occurrences based on a compressed data length of four bytes for a Kia Sorento.

| ID | Less than five bytes | More than four bytes | Total frames |
|----|-----------------------|----------------------|--------------|
| 260 | 163,078 | 0 | 163,078 |
| (100.00%) | (0.00%) | (100.00%) |
| 2A0 | 163,078 | 0 | 163,078 |
| (100.00%) | (0.00%) | (100.00%) |
| 316 | 156,250 | 6,828 | 163,078 |
| (95.81%) | (4.19%) | (100.00%) |
| 329 | 163,074 | 0 | 163,078 |
| (100.00%) | (0.00%) | (100.00%) |
| 43F | 199,507 | 24 | 199,531 |
| (99.99%) | (0.01%) | (100.00%) |
| 440 | 199,489 | 42 | 199,531 |
| (99.98%) | (0.02%) | (100.00%) |
| 545 | 163,078 | 0 | 163,078 |
| (100.00%) | (0.00%) | (100.00%) |
| 580 | 80,102 | 1,339 | 81,468 |
| (98.36%) | (1.64%) | (100.00%) |

TABLE 9. Total occurrences based on a compressed data length of four bytes for a Kia Sorento.

| No. of occurrences less than five bytes | 1,287,683 (99.36%) |
| No. of occurrences more than four bytes | 8,237 (0.63%) |
| Total frames | 1,295,920 (100.00%) |

for a Kia Sorento. Table 9 shows the total occurrences based on a compressed data length of four bytes. Among a total of 1,295,920 frames, only 8,237 (or, 0.63%) frames require additional frames to transmit MAC data for a Kia Sorento. Thus, the increase in the bus load is very small due to the use of the proposed data compression technique.

Table 10 and Table 11 show the number of occurrences based on a compressed data length of four bytes in a CAN network with 27 ECUs for a Kia Soul. Among a total of 3,523,115 frames, 3,110,943 (88.3%) frames have

TABLE 10. Number of occurrences based on a compressed data length of four bytes for a Kia soul.

| Ratio of less than five bytes | No. of IDs |
|------------------------------|------------|
| 100%                         | 14         |
| 99.99% ~ 95.00%             | 5          |
| 92.06%                       | 1          |
| 89.99% ~ 85.00%             | 2          |
| 64.99% ~ 60.00%             | 2          |
| 48.43%                       | 1          |
| 34.35%                       | 1          |
| 28.05%                       | 1          |
| Total                        | 27         |

TABLE 11. Number of occurrences based on a compressed data length of four bytes for a Kia soul.

| No. of occurrences less than five bytes | 3,110,943 (88.3%) |
| No. of occurrences more than four bytes | 412,172 (11.69%) |
| Total frames | 3,523,115 (100.00%) |
TABLE 12. Number of occurrences based on a compressed data length of four bytes for an LS Mtron.

| Ratio of less than five bytes | No. of IDs |
|-------------------------------|------------|
| 100%                          | 37         |
| 99.99% – 95.00%               | 4          |
| 94.99% – 90.00%               | 2          |
| 87.77%                        | 1          |
| 0.00%                         | 1          |
| Total                         | 45         |

Compressed lengths less than five bytes; consequently, no additional frames are needed to transmit these MAC data. For the remaining 412,172 (11.69%) frames, two frames are required for each CAN data transmission.

Notice that for three IDs in Table 10, the ratios of less than five bytes are less than 50%, which increases the CAN transmission latency. These cases happen when CAN signals change rapidly. Thus, if a CAN frame with a specific ID contains critical components, it is advisable to divide the CAN frame into critical and non-critical messages and limit the frame length of the critical messages to less than five bytes. In this way, we can reduce the latency of sending more important messages.

A simulation using a tractor was performed to show that the proposed algorithm can be used not only for automobiles but also for various industrial vehicles.

Table 12 and Table 13 show the number of occurrences based on a compressed data length of four bytes in a CAN network with 45 ECUs for an LS Mtron tractor. Among a total of 238,997 frames, 235,924 (98.72%) frames have compressed lengths less than five bytes; consequently, no additional frames are needed to transmit these MAC data. For the remaining 1.28%, two frames are required for each CAN data transmission. This simulation shows that the proposed method can be used very efficiently to increase the security of the CAN system in the tractor.

CANoe was used to calculate the peak load of the CAN bus. The bus load is defined as

\[
\text{Busload} (\%) = \frac{\text{used capacity}}{\text{max capacity}} \times \frac{\text{bits sent}}{\text{speed}}, \quad (9)
\]

and the peak load is defined as the maximum bus load. The bus load is updated every 100 ms by the CANoe tool.

Fig. 10 shows the simulation block diagram using CANoe.

In CAN 1, the original (uncompressed) data are transmitted without the MAC while the compressed data obtained by the proposed algorithm are transmitted in CAN 2 with the MAC.

Each ECU sends the CAN data corresponding to the ID assigned to the ECU.

Table 14 shows that the peak load of the proposed system is reduced by 3.89% compared to the conventional method. The simulation results of the conventional CAN and the proposed CAN are obtained from Kia Sorento CAN data at 500Kbps CAN speed. The proposed method transmits more frames than the conventional method. However, the peak load of the proposed method is lower than that of the conventional method since the CAN data are mostly compressed to less than four bytes (99.36%) by the proposed method. If the CAN data is compressed to zero bytes, the sender ECU sends one byte of zero data. Therefore, although the proposed method transmits more frames than the conventional method, the number of transmitted bits of the proposed method is less.

Fig. 11 shows a screenshot captured during the first 100 ms of CAN data transfer with and without an OIM. For data transfer without an OIM, two frames (ID \(i+1\) and ID \(i+2\)) are needed for each ID \(i\) during initial stage to send both non-compressed CAN data and MAC data, as can be seen from Fig. 11(a). This can be avoided by using an OIM (since OIM can be used as the previous data), as can be seen from Fig. 11(b). Since the difference between the current and the previous CAN data is zero due to OIM, only one byte of zero data is transmitted.

Table 15 shows the comparison of the peak loads with and without OIM during the initial stage. Peak load increases as the number of ECUs increases. When OIM is not used, the peak load with 16 ECUs is 68.48%. By using OIM, the peak load is dramatically reduced to 16.32%. Table 14 clearly
TABLE 14. Comparison of the peak load between the original CAN bus and the proposed CAN bus.

|                  | Conventional CAN without MAC | Proposed CAN with MAC |
|------------------|------------------------------|-----------------------|
| Peak load (%)    | 19.35                        | 15.46                 |
| Data (frames/s)  | 794                          | 797                   |
| Data (total frames) | 1,295,928              | 1,297,500             |

TABLE 15. Comparison of the peak loads with and without an OIM in initial stage.

| No. of ECU | Peak load with an OIM (%) | Peak load without an OIM (%) | Diff |
|------------|---------------------------|-------------------------------|------|
| 8          | 8.16                      | 34.24                        | 26.08|
| 16         | 16.32                     | 68.48                        | 52.16|

shows that OIM should be used for CAN frame authentication when the number of ECUs is high.

V. CONCLUSION

In this paper, we proposed a CAN security improvement method based on the VMLDA algorithm with two-bit-level arrangement. A triple ID method was proposed to make sure that the length of the MAC is always greater than or equal to four bytes, without changing the original CAN protocol.

An OIM was proposed to reduce the increase in the peak load caused by the addition of the MAC during the initial stage of the CAN system. In addition, when CAN data is compressed to zero bytes, unlike in the MLDA algorithm, one byte of zero data is transmitted so that the sender and the receiver ECUs can verify the operation of each other.

Simulation results using CANoe show that data authentication in CAN systems can be achieved with a slight decrease in the peak load by the proposed method. We also showed that the proposed method can be successfully applied to other industrial machines, such as tractors.

For future research, it will be interesting to apply the proposed method to other type of in-vehicle network (IVN).

REFERENCES

[1] R. Bosch, CAN Specification Version 2.0. Stuttgart, Germany: Robert Bosch GmbH, 1991.
[2] G. Leen and D. Heffernan, “Expanding automotive electronic systems,” Computer, vol. 35, no. 1, pp. 88–93, Jan. 2002.
[3] M. Desai, R. Shetty, V. Padte, M. Parulekar, and S. Ramrajkar, “Controller area network for intelligent vehicular systems,” in Proc. Int. Conf. Adv. Technol. Eng. (ICATE), Mumbai, India, Jan. 2013, pp. 1–6.
[4] T. Nolte, H. Hansson, and L. L. Bello, “Automotive communications-past, current and future,” in Proc. IEEE Conf. Emerg. Technol. Factory Automat., Catania, Italy, Sep. 2005, pp. 985–992.
[5] E. Ortega, T. Heurung, and R. Swanson, “System design from wires to warranty,” Automot. Electron., vol. 14, no. 8, pp. 14–18, Feb. 2006.
[6] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, “Efficient and secure source authentication for multicast,” in Proc. Netw. Distrib. Syst. Secur. Symp., 2001, pp. 35–46.
[7] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, “The TESLA broadcast authentication protocol,” RSA CryptoBytes, vol. 5, no. 2, pp. 2–13, 2002.
[8] C. Szilagy and P. Koopman, “A flexible approach to embedded network multicast authentication,” in Proc. 3rd Workshop Embedded Syst. Secur. (WESS), Atlanta, GA, USA, 2008, pp. 1–7.
[9] C.-W. Lin and A. Sangiovanni-Vincentelli, “Cyber-security for the controller area network (CAN) communication protocol,” in Proc. Int. Conf. Cyber-Secur., Washington, DC, USA, Dec. 2012, pp. 1–7.
[10] B. Groza and S. Murvay, “Efficient protocols for secure broadcast in controller area networks,” IEEE Trans. Ind. Informat., vol. 9, no. 4, pp. 2034–2042, Nov. 2013.
[11] D. K. Nilsson, U. E. Larson, and E. Jonsson, “Efficient in-vehicle delayed warranty,” in Proc. Int. Conf. Cyber-Secur. Privacy, Oakland, CA, USA, May 2010, pp. 447–462.
[12] K. Koscher, A. Czeskis, F. Roosen, S. Patel, T. Kohno, and S. Checkoway, “Remote exploitation of an unaltered passenger vehicle,” in Proc. IEEE Symp. Security and Privacy, Oakland, CA, USA, May 2010, pp. 447–462.
[13] G. Leen and D. Heffernan, “Expanding automotive electronic systems,” in Proc. IEEE Int. Conf. Veh. Technol., vol. 50, no. 1, pp. 162–169, Jan. 2001.
[14] J. Schmandt, A. T. Sherman, and N. Banerjee, “Mini-MAC: Raising the bar for controller area network using ECANDC compression algorithm,” in Proc. Int. Conf. Adv. Technol. Eng. (ICATE), Mumbai, India, Jan. 2013, pp. 1–6.
[15] S. Kelkar and R. Kamal, “Boundary of fifteen compression algorithm for automobile applications,” in Proc. Int. Conf. Circuits, Syst., Commun. Inf. Technol. Appl. (CSCITA), Mumbai, India, Apr. 2014, pp. 162–167.
[16] Y.-J. Wu and J.-G. Chung, “Efficient controller area network data compression for automobile applications,” Frontiers Inf. Technol. Electron. Eng., vol. 16, no. 1, pp. 70–78, Jan. 2015.
[23] Y. Wu and J.-G. Chung, “An improved controller area network data-reduction algorithm for in-vehicle networks,” *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. 100, no. 2, pp. 346–352, 2017.

[24] Y.-J. Kim, Y. Zou, Y.-E. Kim, and J.-G. Chung, “Multi-level data arrangement algorithm for can data compression,” *Int. J. Automot. Technol.*, vol. 21, no. 6, pp. 1527–1537, Dec. 2020.

[25] Q. Dang, “Recommendation for applications using approved hash algorithms,” NIST, Gaithersburg, MD, USA, Special Publication 800-107, 2012.

[26] S.-P. Wang, Q.-M. Ma, Y.-L. Zhang, and Y.-S. Li, “A HMAC-based RFID authentication protocol,” in *Proc. 2nd Int. Symp. Inf. Eng. Electron. Commerce*, Ternopil, Ukraine, Jul. 2010, pp. 1–4.

[27] S. Woo, H. I. Jo, I. S. Kim, and D. H. Lee, “A practical security architecture for in-vehicle CAN-FD,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2248–2261, Aug. 2016.

[28] M. Bellare and P. Rogaway, “Entity authentication and key distribution,” in *Proc. Annu. Int. Cryptol. Conf.*, 1993, pp. 232–249.

[29] Mirae Technology. Accessed: 2018. [Online]. Available: https://www.cryptopp.com/benchmarks.html

[30] B. I. Kwak, J. Woo, and H. K. Kim, “Know your master: Driver profiling-based anti-theft method,” in *Proc. 14th Annu. Conf. Privacy, Secur. Trust (PST)*, Auckland, New Zealand, Dec. 2016, pp. 211–218.

YEON-JIN KIM (Member, IEEE) received the B.S. and M.S. degrees in electronic engineering at Jeonbuk National University, Jeonju, South Korea, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree in electronic engineering. His research interests include VLSI design of digital signal processing and communication algorithms.

SAMUEL WOO received the Ph.D. degree in information security from Korea University, Seoul, South Korea, in 2016. He was a Senior Researcher with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. He is currently an Assistant Professor with the Department of Software Science, Dankook University, Jukjeon, South Korea. His research interests include cryptographic protocols in authentication, security and privacy in vehicular networks, and controller area network security.

JIN-GYUN CHUNG received the M.S. and Ph.D. degrees in electrical engineering from the University of Minnesota, Minneapolis, MN, USA, in 1991 and 1994, respectively. Since 1995, he has been with the Division of Electronic Engineering, Jeonbuk National University. His research interests include VLSI architectures and algorithms for signal processing and communication systems, including the design of high-speed and low-power algorithms for digital filters, XRF systems, OFDM systems, and ultrasonic NDE systems.