International Journal of Computer (IJC)

ISSN 2307-4523 (Print & Online)

https://ijcjournal.org/index.php/InternationalJournalOfComputer/index

A Lightweight Way to Secure Automotive Networks Using CAN/CAN-FD

Sukhyun Seo*

Department of Electronics Engineering, Tech Uniersity of Korea, Siheung-si, Gyeonggi-do, Korea

Email: shseo@tukorea.ac.kr

Abstract

In-vehicle communication uses the CAN/CAN-FD bus, and communication speed and security are important. As current CAN/CAN-FD communication is used without encryption, many cases of vehicle hacking have been reported over time. With the advent of autonomous driving and connected cars, vehicles are no longer independent; they can be infiltrated from the outside and personal information such as vehicle location and driving habits can be accessed through the vehicle, posing a serious threat to personal privacy and life. Therefore, communication data needs to be encrypted to increase the security of communication. In this paper, data frames are encrypted using a shuffling algorithm in the CAN/CAN-FD communication system environment. We also compare and analyse standardised encryption methods, namely AES and ARIA, and shuffling algorithms, and suggest ways to increase the security and communication speed in the vehicle.

Keywords: Cyber security; Controller area network; Automotive networks.

1. Introduction

For the comfort and safety of the user, the number of electrical and electronic devices in the vehicle has increased [1]. In order to control each device, many control units are installed in the vehicle, making the car's system increasingly complex. Although the commercialization of autonomous vehicles seems to be a thing of the future, threats to vehicle safety are closer than expected. A researcher working with Professor Stefan Savage at the University of California hacked into the vehicle's Onboard Diagnostic System (OBD2) to gain control of a Corvette [2]. The researchers were able to control the car's brakes from outside using a mobile phone. Security experts Charlie Miller and Chris Valasek also hacked a Jeep Cherokee to manipulate the windscreen wipers and stop the car [3]. These hacks happened in 2015, just four years ago, and continue to be reported [4]. Security issues tend to be a problem only in toxic autonomous vehicles and connected cars. However, the issue of security in virtually any vehicle is an important issue that cannot be avoided: every driver wants to protect their vehicle from hacking.

Received: 8/13/2023
Accepted: 9/18/2023
Published: 9/27/2023

----

* Corresponding author.

When the cloud connects to vehicles, personal information such as online services, personal location and driving habits are stored. This information is accessible from the vehicle. Therefore, if the sensitive information is passed to the perpetrators, they can steal control of the driver's vehicle and pose a deadly threat to the driver, such as causing a traffic accident by manipulating the controls. Methods have been proposed to secure the vehicle, such as frame authentication by generating authentication codes, symmetric key encryption by key distribution, and encryption using hash functions. This approach has many disadvantages, such as time consuming, data loss due to key eavesdropping, and memory wastage due to unnecessary data generation. Therefore, in this paper, we use a light and fast shuffling algorithm to achieve high performance and to encrypt the data frame of CAN/CAN-FD so that the data cannot be eavesdropped by external attack.

We propose a CAN/CAN-FD communication security method to compensate for the security weaknesses of the CAN/CAN-FD communication protocol that controls the electronic control unit of the vehicle and the existing weaknesses. Chapter 2 discusses CAN/CAN-FD communication and shuffling, while Chapter 3 describes the application of the proposed shuffling algorithm. Chapter 4 analyses the performance difference between the shuffle algorithm and the existing security algorithm. Chapter 5 describes the performance and development direction of this proposal and concludes the paper.

The technique proposed in this paper is suitable for in-vehicle communication, where communication speed is important, and improves the security compared to the conventional method, which has security vulnerabilities.

2. Related Research

2.1. CAN Communication

CAN/CAN-FD communication is one of the vehicle communication methods and is a standard communication protocol designed for electronic control units in vehicles to communicate with each other [5]. In the past, cars used UART, a serial communication method, for vehicle modules to communicate. UART communication has the characteristic that each module has one-to-one communication, and each module requires numerous connection lines. Many wires are inefficient in terms of space and have increased costs. To solve this problem, Mercedes-Benz commissioned Bosch, an automotive supplier, to develop a vehicle network, which Bosch first developed in 1985. The development of CAN/CAN-FD made it possible to control devices more efficiently, for example, to reduce the weight of vehicles and cut costs. In 1993, ISO defined the international standard. Since then, CAN/CAN-FD communication has become the standard for vehicle communication.

CAN/CAN-FD is a communication method that sends and receives data by connecting an electronic control unit, called an ECU, in parallel to the CAN/CAN-FD bus. CAN/CAN-FD communication does not have a master that controls the entire ECU, as each EUC accesses the CAN/CAN-FD bus to read or write data flowing over it. Therefore, when the EUC is ready to transmit data, it checks the state of the bus and transmits the CAN/CAN-FD frame to the network. The transmitted CAN/CAN-FD frame does not contain a unique send or receive ECU address, but instead has a unique identifier to identify each ECU. Each ECU receives the input if it is a necessary message, analyses the data and ignores unnecessary messages. When multiple data are received, the data identifiers are compared, and the data is prioritised. This can improve the speed and safety of system control.

The CAN/CAN-FD frame consists of the following [5]. The Start-Of-Frame (SOF) bit is used to synchronize the ECU with the start of the message. Identifier (ID) is an identifier used to cope with data collisions when multiple data are received. Control is the length of the data. Data can be up to 8 bytes and is used to store data. CRC is used for transmission error and error detection of a frame. The ACK bit is a bit that determines the error of the message and checks if the message has been sent correctly. End-of-Frame (EOF) is a bit that indicates the end of a message.

The vulnerability of CAN/CAN-FD can be analyzed using three characteristics [6]. CAN/CAN-FD uses the broadcast method to receive messages. As a result, messages from all nodes can be intercepted from the outside. This means that there is a risk of eavesdropping. In addition, CAN/CAN-FD communication is vulnerable to distributed service attacks because the node does not receive the message with the small ID value among the messages on the bus. Finally, there is no separate field in CAN/CAN-FD communication for authenticating the EUC transmitting a frame. In addition to this approach, CAN/CAN-FD vulnerabilities can be analyzed according to five characteristics: confidentiality, integrity, authenticity, availability and non-repudiation.

Many studies have proposed methods, such as control mechanisms that identify authentication or intrusion detection through certificates, for vulnerabilities in CAN/CAN-FD communication. Since this study is a follow-up to an attack, a countermeasure is proposed for a direct complement.

Encrypted data uses an authentication code [7]. After generating message authentication codes for four data frames to be transmitted, the generated message codes are split and inserted into each data frame. When four data frames are received, the data frames are analysed and processed. This approach is time consuming because it has to wait for all four messages to be received. Therefore, the data processing environment of the vehicle, where communication speed is important, cannot be satisfied.

The gateway ECU, which enables secure communication of the vehicle, performs certificate management, key update and key distribution of the entire ECU [8]. Each ECU is equipped with a certificate, and all ECUs register their certificate with the Gateway ECU via external communication before leaving the vehicle. For secure communication, gateway ECUs generate secret values and send them to all ECUs. The ECUs use the secret value to generate an encryption session key, a message authentication key and a key update key. Using this key, the EUC encrypts the data field in the data frame and generates a message authentication code. A data field is regenerated by combining the encrypted data field and the authentication codes. The regenerated data frame is transmitted over the CAN/CAN-FD bus. The message authentication key is used to prevent retransmission attacks that are intercepted and manipulated externally. An authentication key is generated by storing the corresponding ID values of the sending and receiving ECUs. To ensure the stability of the key, the ECU updates the key at regular intervals.

However, the disadvantage of this method is that the data flowing over the CAN/CAN-FD bus is encrypted, which makes the data unintelligible, but the data can be intercepted in the middle. In addition, counter values are used to prevent retransmission attacks, which analyse the pattern of counter values and change only the counter values in the data frame to enable spoofing attacks such as retransmission attacks. In addition, repeated renewal of the certificate may cause inconvenience to the vehicle owner and may cause the vehicle to malfunction due to an error or incorrect certificate at the time of renewal. To prevent the attacker from finding the unique user ID and intercepting the data, encryption is performed using a hash algorithm [9]. The hash algorithm converts an arbitrary bit string into a bit string of fixed length. In other words, the user ID is represented by a bit string of constant size determined by its hash function, regardless of data capacity. It encrypts the Identifier data element in the CAN/CAN-FD frame, and the original User ID value is compared one-to-one with the encrypted value. However, the main disadvantage of the hash algorithm is that since the array is allocated as the maximum size of the key value, the size is very large. When there is less data to store, a lot of space is wasted. If unnecessary data is created in the storage area, unnecessary values are created, which can cause system problems. Also, memory leaks could allow an attacker to attack, crash the system or execute arbitrary code.

2.2. Shuffling Algorithm

Shuffling is similar to shuffling cards in a card game [10]. The shuffling algorithm randomly shuffles the numbers. The result is an unpredictable order that makes it difficult for attackers to find the original form [4]. The advantage of shuffling is that it is performed so quickly that it has little impact on computing speed. Typical algorithms include the Fisher-Yates shuffle, XOR replacement and Blowfish.

The Fisher-Yates shuffle was named by Ronald Fisher and Frank Yates [11]. Generates a random sequence that is reordered when there are numbers from 1 to n. Pick a random number from 1 to n and put it in the first order. Instead of the drawn number, pick one of the remaining numbers and put it in the second order. Repeat this process until there are no more numbers to draw and you have a randomly sorted sequence. Blowfish is a symmetric key cryptography that was designed by Bruce Schneier in 1993 [13]. It has variable keys from (32 to 448) bits and features complex key scheduling and SBox dependencies. It encrypts using XOR and Blowfish functions. The advantage of Blowfish is that it encrypts at 18 clocks per byte in a 32-bit microprocessor, which is much faster than DES or IDEA. In addition, as the key length is variable, the security performance can be adjusted.

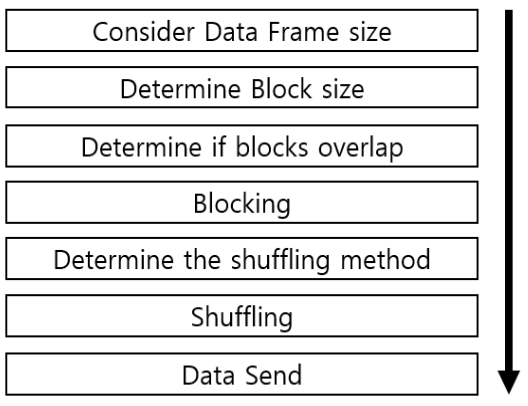

Figure 1: Shuffling encryption process.

3. Security Design in CAN/CAN-FD

3.1. Design Configuration

Car information is sent and received using various sensors, control modules and CAN/CAN-FD communication. In this paper, we focus on improving the security of CAN/CAN-FD communication data frame and fast processing speed by analyzing the encryption performance by applying shuffling algorithm technology to improve the security of CAN/CAN-FD communication.

To apply the shuffling algorithm, the proposed encryption method is to select the blocking method, which represents the bits of the CAN/CAN-FD data frame by changing the columns and rows, and then structuring them into blocks. Shuffling is then performed to hide the data frame. The obfuscation level can be adjusted according to the blocking and shuffling criteria. Figure 1 shows a flowchart of data transmission. When receiving data, a similar process is performed, but back-shuffling must be performed during the shuffling process

The test electronic control unit supports CAN/CAN-FD, LIN, and FlexRay communications, which are widely used in automotive electronics communications. Using two test boards, it is divided into a sending side and a receiving side and performs CAN/CAN-FD communication through CAN/CAN-FD cable. The sender displays the encryption time and the data to be encrypted on the LCD window, and the receiver displays the decrypted data on the LCD window to confirm the communication performance analysis and data accuracy. Fig. 2 shows the system configuration.

The size of the CAN/CAN-FD data frame is 8 bytes, i.e., 64 bits, and the encryption time is measured using a timer/interrupt with a 25 µs cycle. The bits are locked by AND and XOR bit operations. Blocking requires that blocks do not overlap. The CAN/CAN-FD communication flow checks the transmission status and transmits data when it is ready to send. If it is not ready to send, it uses the delay function to wait until it is ready to send, and then sends again. When reception is complete, the data frame is acknowledged by decoding.

For CAN/CAN-FD communication only, it took 72 µs. Since the proposed method can vary the block size and the number of shuffles, the time was measured according to each condition. We analyzed the security strength according to the speed and compared the execution time with existing security algorithms such as AES and ARIA. The unit of time is microseconds (µs).

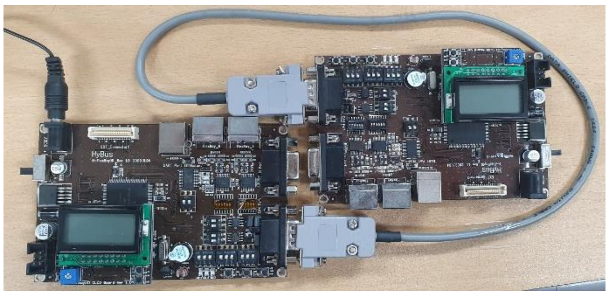

Figure 2: CAN/CAN-FD Test Eectronic control units.

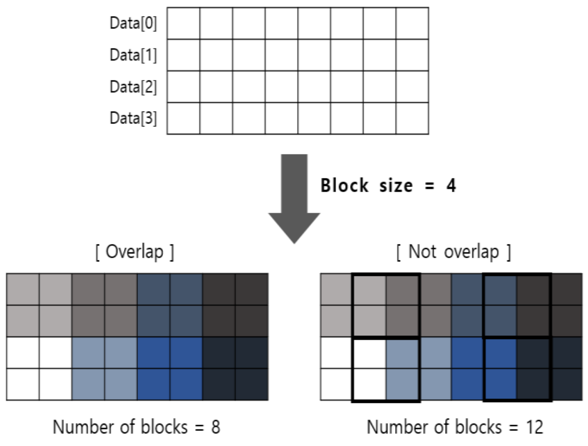

Figure 3: Block Generation with block size is 4.

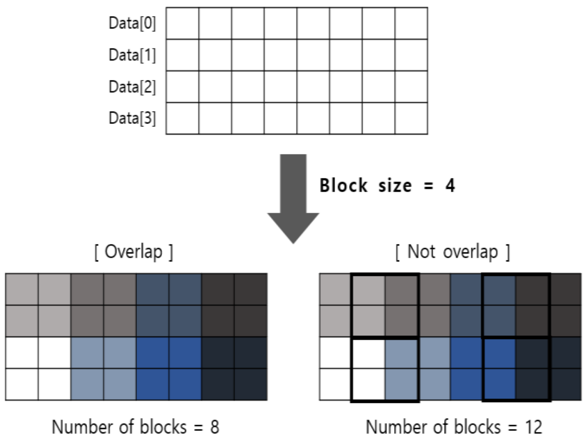

Figure 4: An example for shuffling operation.

3.2. Composition of Blocks

CAN/CAN-FD communication provides hardware packets with up to 8 bytes of data transmission. One byte is eight bits, and the bits are binary, the smallest unit of computer data. The size of the block can be selected according to the size of the data frame. The size of a data frame is between (1 and 8) bytes, and the block size is variable. The order of blocking is as follows:

- Transpose bits of a specific data frame into columns and rows.
- Depending on the size of the data frame, choose a block size so that all data is the same size.
- Structure the bits to the selected block size. (Only blocks may overlap unless there is one block size)

Fig. 3 is a schematic to show the number of blocks that can be created when the data frame size is 4, and the block size is 4. The number of blocks can occur in two cases: when the blocks overlap, and when they do not overlap. There is also a case where the block is structured horizontally and vertically. There are various ways to block like this.

1) If the blocks do not overlap: Block size is determined according to the data frame. However, when all data frames are blocked so that they do not overlap, all bits must be blocked. For example, if the data frame is composed of 8×4 and the block size is 4, the block can be configured to the optimal value because it consists of 8 blocks. Compared to the case of overlapping blocks, the optimized communication speed is the fastest and intuitive, but the number of blocks decreases, so the security is relatively low.

2) If the blocks do overlap: Blocks can be divided over, except in the case of one block size. When blocking with rows of data frames and blocking with overlapping columns, the number of cases increases. When the data frame is 2 bytes and the block size is 2 bits, it is 22 when the blocks overlap, compared to 8 when the blocks do not overlap. Although the number of blocks increases compared to when the blocks do not overlap, security is relatively increased, but communication speed is slow, and blocks cannot be viewed intuitively.

3.3. Shuffling

The shuffling process is a process of replacing bit data with another block that does not overlap a specific block. Shuffling is performed to conceal data frames after the blocking process. Through shuffling, existing data frames are obfuscated. However, while performing a task, all blocks must be shuffled at least once. Shuffling proposes two methods: random method and random function definition.

1) Arbitrary method: This method is used to randomly select a shuffle rule based on code when the block is made according to the data frame size.

- Block the bits
- Select two random blocks and swap.
- When transmission is completed, reverse the shuffling by swapping in reverse order of the encryption swap.

2) Random function definition: This method defines an arbitrary random function with rules. For example, it defines an arbitrary random function with a rule that shuffles the data values of blocks of odd numbers with odd numbers and even numbers with even numbers. It calls the random function defined during shuffling and performs the model year. In the decoding process, a function is called and deshuffled.

Table 1: Number of Reverse-Shuffle Attempts.

Suffling Block Size	m=variable value (1bit at least to data frame size/2bits)
Data Size	S
Shuffle Reverse Trace Attempts	(s/m)!

4. Implementation and Result

4.1. Processing performance by block size

If the number of blocks is 4, it is 118 times faster than AES encryption and 14 times faster than ARIA encryption; if the number of blocks is 8, it is 69 times faster than AES encryption and 8 times faster than ARIA encryption; and if the number of blocks is 32, it is 41 times faster than AES and 5 times faster than ARIA.

When the data frame is 8 bytes, the speed is measured by changing the block size. That is, the number of blocks is changed. In addition, the blocks are locked so that they do not overlap. The possible block size for 8 bytes is between (1 and 32). When (4, 8 and 32) blocks were selected, communication times of (86, 96 and 112) µs were measured. This corresponds to encryption execution times of (14, 24 and 40) µs respectively..

The smaller the block size, the more stable it is, because the position per block unit has to be changed. Fig. 5 compares the speed of only (4, 8, and 32) of the various blocks.

The shuffle block size has a large effect on the number of shuffle reverse tracking attempts [15]. The attacker must attempt the factor of the shuffle block size to obtain the original. The shuffle block size is variable because it varies with the user. The table below shows the number of backtracking attempts for the proposed scheme.

If the shuffle size is 4 bits when the data frame is 8 bytes, then 2,092,278,988,000 reverse shuffle attempts must be made to determine the original text of the shuffle performance. Table 1 details the number of reverse shuffle attempts.

4.2. Shuffling

With 4 shuffles, the performance is approximately 79 times faster than AES encryption and 10 times faster than ARIA encryption [5]. When the number of shuffles is 8, it is 49 times faster than AES encryption and 6 times faster than ARIA encryption; and when the number of shuffles is 10, it is 37 times faster than AES encryption and 4 times faster than ARIA encryption. Communication times of (81, 91, 94, 107, and 118) µs are shown when measuring 8 times data frames by changing the number of shuffles to (2, 4, 6, 8, and 10) times based on an 8 block size. This was measured. This means that the encryption execution time is (8, 18, 21, 34 and 45) µs respectively. As the number of shuffles increases, the decryption rule is unknown, which increases security. Increasing the number of shuffles provides faster performance than the conventional method. Figure 6 shows the results of changing the number of shuffles. The encryption speed depends on the number of shuffles.

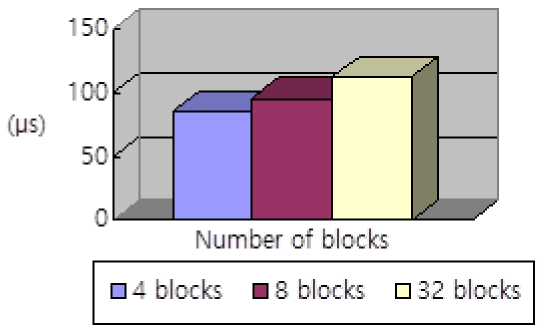

Figure 5: Encryption speed by number of blocks.

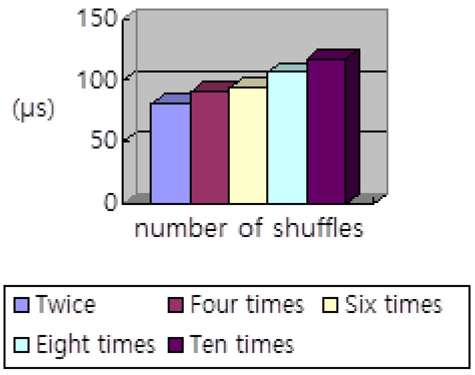

Figure 6: Encryption speed by number of shuffle counts.

5. Conclusion

In this paper, we presented the need for encryption by explaining the security weakness of CAN/CAN-FD communication. In addition, we proposed a shuffling algorithm with faster performance than the existing encryption method. In the CAN/CAN-FD communication environment, we measured the time taken to encrypt 8 bytes of a data frame. As a result of the comparison and analysis with encryption methods, namely AES and ARIA, this proposal provided faster communication performance. Fast performance is essential in vehicle situations where security incidents need to be anticipated. In addition, because it is in the form of a function rather than a traditional key method, a lot of execution is required in the decryption process, which increases security. In addition, security can be tailored to the user. Security can be adjusted depending on how the blocking is done and how the shuffling rules are used.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government, MSIT (No.2020R1F1A1071819). In addition, this research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2023-2018-01426) supervised by the IITP(Institute for Information & communications Technology Promotion.

References

[1] Jae-Hong Min, Hyun-Woo Lee, Jae-Young Kim, “Technical Trend on Embedded Software of Motor Vehicle,” Electronics and Telecommunications Trends, vol. 26, no. 2, pp. 137–147, Apr. 2011.

[2] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, “Experimental security analysis of a modern automobile,” In IEEE Symposium on Security and Privacy (S&P), May. 2010.

[3] C. Miller, C. Valasek, “A survey of remote automotive attack surfaces”, Black Hat USA, 2014.

[4] Woo-Jin Jung, Eun-Min Choi, Sung-Min Han, Ji-Woong Choi, “CNN based Malicious Node Detection in Controller Area Network (CAN/CAN-FD) Systems”, Journal of the Korea Institute of Telecommunications, vol. 50, no. 1, pp.263–264, Jan. 2019.

[5] Farsi, Mohammad, Karl Ratcliff, and Manuel Barbosa, “An overview of controller area network,” Computing & Control Engineering Journal, vol. 10, no. 3, pp.113–120, 1999.

[6] K. Koscher, A. Czeskis, F. Roesner, S. Patel, and T. Kohno, “Experimental security analysis of a modern automobile,” Proceedings of the 2010 IEEE Symposium on Security and Privacy, pp. 447-462, May. 2010.

[7] D. K. Nilsson and U. E. Larson, “A defense-in-depth approach to securing the wireless vehicle

infrastructure,” Journal of Networks, vol. 4, no. 7, pp. 552–564, Sep. 2009.

[8] A-Ram Cho, Hyo-Jin Jo, Samuel Woo, Young-Dong Son and Dong-Hoon Lee, “Secure message authentication and key distribution mechanism against CAN/CAN-FD bus attack,” Journal of the Korea Institute of Information Security and Cryptology, Volume 22 Issue5, pp.1057–1068, Oct. 2012.

[9] Soo-Min Choi and Yong-Tae Shin, “A study on the prevention and verification of CAN/CAN-FD frame bit forgery and modification using hash function,” Korean Institute of Information Scientists and Engineers, pp.1277-1279, Jun. 2019.

[10] Min-ji Yoon, “A study on variable block-based encryption and shuffling for electronic documents,” Master's Thesis, Feb. 2017.

[11] Fisher, Ronald A., and Frank Yates, “Statistical tables for biological, agricultural and medical research,” Oliver and Boyd Ltd, London, pp.26-27, 1943.

[12] Schneier and Bruce, “Description of a new variable-length key, 64-bit block cipher (Blowfish),” International Workshop on Fast Software Encryption, Springer. Berlin. Heidelberg, pp191-204, 1993.

[13] J. Bechennec, M. Briday, S. Faucou, and Y. Trinquet, “Trampoline an open source implementation of the OSEK/VDX RTOS,” in 11th International Conference on Emerging Technologies and Factory Automation (ETFA'06), IEEE Industrial Electronics Society, 2006.

[14] Bo-Jo Hong, In-Chul Han, Dong-Won Jang and Nam-Yong Lee, “An empirical study on extended CAN/CAN-FD bus communication with security algorithm,” Journal of Korean Institute of Communication Sciences, vol.43, no.9, pp.1525-1531, Sep. 2018.