2019 14th Asia Joint Conference on Information Security (AsiaJCIS)

Counter Attack against the Bus-Off Attack on CAN

Masaru Takada
Electrical and Electronic Engineering
Kobe University
Kobe, Japan
e-mail: morii@ieee.org

Yuki Osada
Electrical and Electronic Engineering
Kobe University
Kobe, Japan
e-mail: oosada@stu.kobe-u.ac.jp

Masakatu Morii
Electrical and Electronic Engineering
Kobe University
Kobe, Japan
e-mail: mmorii@kobe-u.ac.jp

***Abstract*—The computerization of vehicles has accelerated and in-vehicle networks connect to external networks. Along with this, cyber security of vehicles becomes a problem. A bus-off attack was proposed by Cho et al. in 2016. The bus-off attack is a kind of denial of service (DoS) attacks. An attacker exploits the error handling function of Control Area Network (CAN) and puts a node into the bus off state in which it cannot transmit/receive any messages. The existing CAN security technology cannot prevent the bus-off attack. In 2018, Soma et al. proposed a counter attack as a countermeasure. However, they showed that the success rate of their counter attack is as low as about 30%, which is not enough as a countermeasure against a bus-off attack. In this paper, we propose a novel counter attack. In order to increase the success rate of the bus-off attack, attackers need to inject the preceding frame. We removed attackers from CAN bus by a bus-off attack to the preceding frame. And we have implemented our counter attack and showed its effectiveness as a countermeasure against the bus-off attack.**

Keywords—CAN (Control Area Network), In-vehicle Network, Bus-off Attack, Countermeasure

I. INTRODUCTION

The computerization of cars has accelerated and in-vehicle network connects to an external network. And, the invasion route of attackers who intend to control vehicles illegally is increasing. Along with this, the cybersecurity of cars has become a problem. In fact, a large number of researchers are reporting the experiments of the illegal remote control of the vehicle. In particular, C.Miller and C.Valasek reported the possibility of the remote control of Cherokee[1], and it caused 1.4 million recall.

CAN(Control Area Network) is a communication protocol widely used in-vehicle network[2]. To control vehicles, ECUs (Electric Control Units) exchange information with each other by using CAN. However, it is known that CAN does not have functions related to encryption and authentication. So attacks such as eavesdropping on CAN communication and unauthorized injection of messages are possible.

In 2016, a bus-off attack, a type of denial of service attack(DoS attack), was reported by Cho and Shin[3]. The bus-off attack is an attack that forces the victim node to transition to the bus off state by exploiting the error handling features of the CAN protocol. In the bus off state, a node cannot transmit/receive any messages. A bus-off attack aims for the frame collisions. Therefore, the bus-off attacks cannot be prevented by measures such as encryption and message authentication by Message Authentication Code(MAC). Cho et al. have reported that resetting the victim node can prevent the bus off state, but it does not prevent the threat of the attack itself.

In 2018, Soma et al. proposed a countermeasure against the bus-off attack[5]. They aim to put only the attacker node into a bus off state. They used the timing that only the attacker transmits. However, this countermeasure has problems with success rate and impact on counter failure. As reported by them, the success rate of the counter attack is about 30%, and the cause of the failure is unknown. Also, they injected a large number of dominant bits, so when the countermeasure fails, all transmitting nodes transition to the bus off state.

In this paper, we propose a counter attack that improves the method proposed by Soma et al. Our counter attack eliminates attackers by adapting a bus-off attack to their preceded frame.

The organization of this paper is as follows. Section 2 describes an overview of the CAN protocol. Chapter 3 describes two types of bus-off attacks and the countermeasure proposed by Soma et al. Chapter 4 explains the principle of our counter attack. Section 5 describes the experiment of the proposed method in detail. And the conclusion is described in Chapter 6.

II. CAN

A. CAN protocol

CAN is a communication protocol for an in-vehicle network that had been developed by Bosch in the 1980s. CAN consists of a bus-type multi-master network. Each node is connected to the CAN bus and performs broadcast communication. The CAN bus consists of two wires (CAN High, and CAN Low), and a node uses the potential difference between them In order to transmit information. The case where the potential difference is above the threshold is assigned to logical value “0” and it is called dominant. The case where the potential difference is below the threshold is assigned to logical value “1” and it is called recessive. If dominant and recessive are simultaneously sent to the bus by multiple nodes, the dominant bit overwrites the recessive bit, and the dominant bit will be sent to the bus. Arbitration described in Section 2.3 uses this feature.

B. Frame

In CAN, data communication is performed in frame units, and four types of frame are defined: data frame, remote frame, error frame, and overload frame. This section describes the data frame and the error frame related to this paper.

First, a data frame (Figure. 2.1) is used for data transmission. The data can be transmitted at most 8 bytes in one data frame,

978-1-7281-2556-5/19/$31.00 ©2019 IEEE
DOI 10.1109/AsiaJCIS.2019.00021

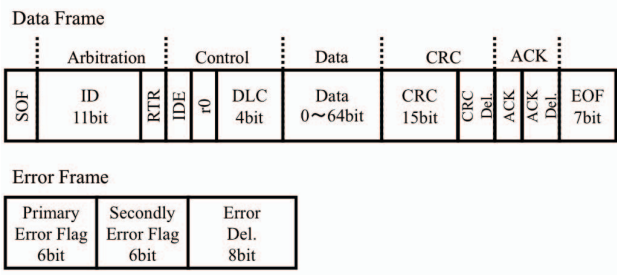

Figure.2.1 Data Frame and Error Frame.

and the data length is stored in the DLC area. The ID area can store 11 bits as the identifier. Each node uses the ID to determine whether the frame is necessary or not, and receives it if it decides necessary. The ID is also used to indicate transmission priority in Arbitration, which is described in Section 2.3.

An error frame (Fig. 2.1) is sent when an error occurs. It consists of a primary error flag, a secondly error flag, and an error delimiter. Normally, when a node detects an error, it transmits 6 consecutive bits as a primary error flag. Since a primary error flag violates the bit stuffing rule described later, 6 bits of the continuous signals are transmitted as secondary error flag from all other nodes. After that, the recessive bits are transmitted as an error delimiter. We explain the error handling function in detail in Section 2.5.

C. Arbitration

In CAN, when the bus is idle, each node can transmit a message. However, CAN has the feature that it cannot send multiple messages simultaneously. Therefore, when one node starts transmission, the others wait for transmission to avoid message collision. Also, when different nodes start transmission at the same time, the ID of each frame contends for the transmission priority. In CAN, "0" is transmitted by overwriting "1", so a frame with a smaller ID acquires the transmission priority.

D. Bit stuffing rule

Each node synchronizes at the timing when the signal changes from recessive to dominant. However, if the same signal continues for a long time, the synchronization may deviate. Therefore, the bit stuffing rule is defined in CAN. The bit stuffing rule is that when the same signal is continued five times, then one bit of the opposite signal is injected. For example, when one node transmits the data with 6 consecutive dominant bits (000000), it injects one recessive bit in the 6th bit (0000010) and transmits it to the bus. Then, the other nodes receive the message as (000000).

E. Error handling

The CAN protocol defines five types of errors: bit error, stuff error, CRC error, form error, and ACK error. When each error is detected, an error frame is sent to the CAN bus immediately. we describe the bit error and the stuff error that are related to this paper.

Bit error

Each transmitting node always compares the bit transmitted by the node with the bit on the bus. If they are different in areas other than the Arbitration area and the ACK area, the node detects a bit error. This is a transmission error that occurs at the transmitting node.

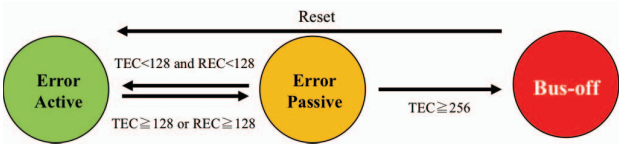

Fig.2.2 State Diagram in CAN.

Stuff error

Normally, when a node transmits the same signal continuously for six or more bits, the CAN controller automatically injects one bit of the opposite signal in accordance with the bit stuffing rule. However, if the same signal continues for 6 bits or more, nodes determine that the CAN controller is not operating and send an error frame. This is the stuff error. This error occurs at the receiving node.

Every node has two error counters: Transmit Error Counter (TEC) and Receive Error Counter (REC). Like the bit error, when an error is detected in the sender node, the TEC of the sender is increased by 8, and the RECs of other nodes are increased by 1. And like the stuff error, when an error is detected in receiving nodes, the RECs of the receivers are increased by 8. Also, when the message is correctly transmitted, both TEC and REC of the sender and the receiver nodes are decreased by 1.

Each node has its state defined by the TEC and REC (Fig. 2.2). There are three states in CAN: error active state, error passive state, and bus off state.

Error active state

The error active state is the normal state, and the counter values are $0 < TEC < 128$ and $0 < REC < 128$. One of the most important features of this state is that when an error occurs, a node transmits 6 consecutive dominant bits as an error flag.

Error passive state

In the error passive state, the error counter values are $128 < TEC < 256$ or $128 < REC < 256$. In this state, when an error occurs, 6 consecutive recessive bits are transmitted as an error flag. In addition, when the node in this state finishes transmitting the frame, it must wait for 11 bits as waiting time. While in error active state, it must wait for only 3 bits.

Bus off state

The bus off state is the state when $TEC > 256$. The node in this state cannot transmit/receive any frames and is disconnected from the bus.

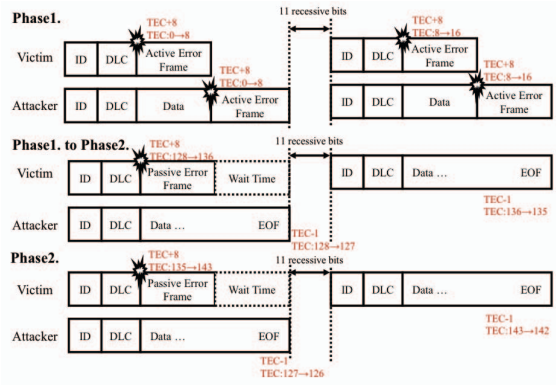

Fig.3.1 Bus-off Attack Proposed by Cho et al.

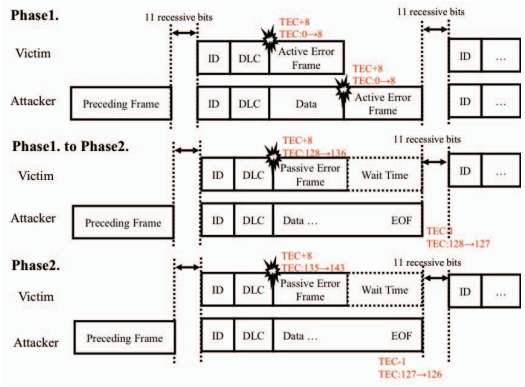

Fig.3.2 Synchronization Using Preceding Frame.

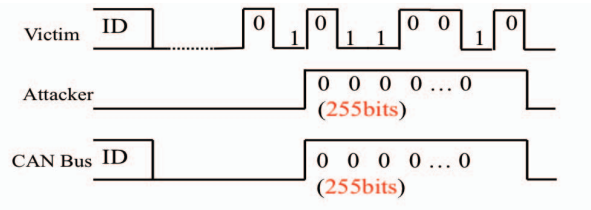

Fig.3.3 Bus-off Attack Proposed by Kameoka et al.

III. BUS-OFF ATTACK AND COUNTERMEASURE

A. Bus-off attack

1) Bus-off attack proposed by Cho et al.

This section describes the bus-off attack proposed by Cho et al. The bus-off attack is a type of DoS attack that exploits CAN error handling to intentionally shift the victim node to the bus off state (Fig.3.1). The attacker causes a bit error of the victim node and a message collision by transmitting the frame at the same timing as the victim node. The attacker repeats it every time the victim node sends a message, and puts the victim node into the bus off state. The bus-off attack consists of two phases.

Phase1

Phase 1 is when both the victim node and the attacker are in the error active state. The attack frame has the same ID as the victim frame. And the attacker sends it at the same time as the victim frame. Then, the attack frame causes a bit error of the victim node in the data area. When an error occurs, because the victim node is in the error active state, it transmits 6 consecutive dominant bits as a primary error flag. The attacker receives the it and sends 6 consecutive dominant bits as a secondly error flag. Both the victim node and the attacker increase their TECs by 8 in this cycle. And they retransmit their frames automatically.

Phase1 to 2

Phase1 continues until both TECs exceed 128. Then, Phase1 to Phase2 proceeds as follows. When both TEC values reach 128, both node transition to the error passive state. The cycle continues as well as Phase1, but when a bit error occurs, the victim node transmits the 6 recessive bits as an error flag. Therefore, the attack message is correctly transmitted without being overwritten, and the TEC value of the attacker decreases by 1. After that, in accordance with the CAN protocol specification, the victim node tries to retransmit the frame. Since the attacker has finished transmitting the attack frame, the victim node also correctly transmits the frame. And the TEC value of the victim node decreases by 1. So, in this phase, the victim node is in the error passive state with TEC:135. On the other hand, the attacker returns to the error active state with TEC:127.

Phase2

Phase 2 is when the victim node is in the error passive state and the attacker is in the error active state. As in Phase 1 to 2, the victim node repeats bit errors and transmission success. So the TEC value rises by 7(+8-1). On the other hand, the attacker reduces the TEC by 1 because of successful transmissions. When the TEC value of the victim node is 256 or more, the victim becomes the bus off state and the attack ends up.

However, in order to perform a bus-off attack, an attacker needs to transmit the attack frame at exactly the same time as the victim frame. It is very difficult to realize this on your own. The method used by CHO et al. to solve this problem is the injection of a preceding frame (Fig. 3.2). According to the CAN protocol, if you try to transmit another frame while transmitting a frame, Arbitration occurs. After the transmission of the first frame is finished, the next frame is started transmitting automatically. Using this mechanism, the preceding frame is injected immediately before the attack frame. Then, after the transmission of the preceding frame is finished, the attack frame and victim frame will be automatically transmitted. The transmission timing will be matched successfully. In this way, an attacker transmits the attack frame at exactly the same time as the victim frame and a bus-off attack begins. Also, the preceding frame needs to be given a smaller ID than the attack frame.

2) Bus-off attack proposed by Kameoka et al.

Kameoka et al. proposed a bus-off attack different from Cho et al. in 2017 [4]. They showed that it is possible to put every node into the bus off state by sending a large number of dominant bits continuously to the CAN bus using a device composed of a raspberry pie and inexpensive chips (Fig. 3.3). As mentioned above, when a bit error occurs, the TEC value is increased by 8. After that, according to the CAN controller

Table 4.1 Comparison of Existing Counter Attack and Proposed Counter Attack.

	Existing Method	**Proposed Method**
Target Frame	Attack frame	Preceding frame
Attack Method	Operation that applies forced electric potentials on the CAN bus	Frame injection in accordance with CAN protocol
Success Rate	Low probability	High probability
Impact on other nodes at the time of failure	Put all other transmitting nodes into the bus off state.	Put the node into the bus off state when a frame has the same ID as the target frame and it is transmitted.

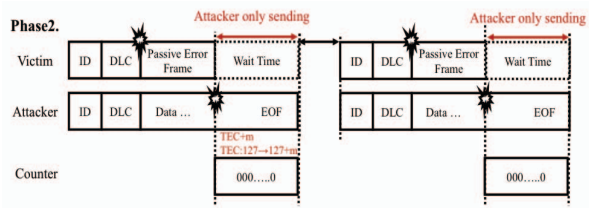

Fig.3.4 Counter Attack Proposed by Soma et al.

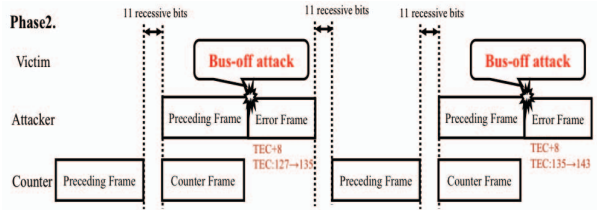

Fig.4.1 Proposed Counter Attack.

specification, each time 14 dominant bits continues, the TEC is increased by 8. Then, each time 8 dominant bits continues, the TEC is increased by 8. That is, in order to increase the TEC value of victim to 256 or more, the attacker only needs to inject at least 255 dominant bits continuously. While the bus-off attack proposed by Cho et al. injects the frame in accordance with the CAN protocol, the bus-off attack by Kameoka et al. requires operations that deviate from the CAN protocol. Kameoka et al. proposed to transmit a large number of dominant bits by keeps applying a forced potential on the CAN bus with raspberry pie.

B. Counter attack proposed by Soma et al.

Soma et al. proposed a counter attack in 2018, which is a countermeasure against the bus-off attack by Cho et al.[5]. They showed that it is possible to transition only the attacker to the bus off state under the bus-off attack (Fig.3.4). First, to detect the bus-off attack, they used a lot of error frames due to consecutive bit errors that occur in Phase 1 of the bus-off attack. Since the probability that consecutive errors occur in a short time is very low, it can be used to detect the bus-off attack.

Next, in order to make the counter attack successful, it is necessary to increase the TEC of only the attacker. For this purpose, Soma et al. proposed that perform the bus-off attack by Kameoka et al. aiming at when only the attackers are transmitting. This timing appears in phase 2 of the bus-off attack. In phase 2, when the victim node detects a bit error, it sends 6 consecutive recessive bits as the error flag and enters a standby state. On the other hand, the attacker continues sending the attack frame. At this time, by applying the bus-off of Kameoka et al., it is possible to attack only the attacker and transition to the bus off state.

IV. DETAILS OF THE PROPOSED COUNTER ATTACK

A. Counter attack to the preceding frame

The counter attack of Soma et al. has two problems: the success rate and the impact on counter failure. In the study of Soma et al., the success rate of counter attack is as low as about 30%. In addition, since a large number of dominant bits are injected, at the time counter attack fails, there is a risk that all nodes transmitting at that time will transition to the bus off state.

Therefore, we propose a counter attack improves the problems of the counter attack by Soma et al. We call our countermeasure a counter attack because it performs a bus-off attack against the preceding frame of the attacker (Fig. 4.1).

As mentioned in Section 3, in order to increase the success rate of the bus-off attack, the attacker injects the preceding frame. So, we thought the bus-off attack to the preceding frame can be a countermeasure against the bus-off attack. While Soma et al. deviate from the CAN protocol, our counter attack is frame injection in accordance with the CAN protocol. The counter attack consists of detection of the bus-off attack and the bus-off attack to the preceding frame. First, in order to detect the bus-off attack, we used the method proposed by Cho et al. [3], as well as Soma et al. In phase 1 of the bus-off attack, a large number of error frames are generated due to consecutive bit errors. Since the probability that consecutive errors occur in a short time is very low, it can be used to detect the bus-off attack. After detecting the bus-off attack, our counter attack starts in Phase 2. In the bus-off attack, it is considered that the frame immediately before the frame which causes the bit error is the preceding frame. So, we performed the bus-off attack on that frame as a counter attack. If this attack is successful, the attacker will transition to the bus off state prior to the victim node, thus protecting the victim node from the bus-off attack.

B. Comparison with existing methods

We compared our counter attack with the proposed counter attack by Soma et al. (Table 4.1).

First, we describe the target frame of the two counter attack. While Soma et al. target the attack frame of the attacker, our counter attack targets the preceding frame of the attacker. Considering the bus-off attack in a real vehicle, the preceding frame is definitely injected to increase the success rate of attacks. So, our method is also effective as a countermeasure against the bus-off attack.

Secondly, we describe the attack method. Both we and Soma et al. performed the bus-off attack against the attacker. However, while Soma et al. use the bus-off attack of Kameoka

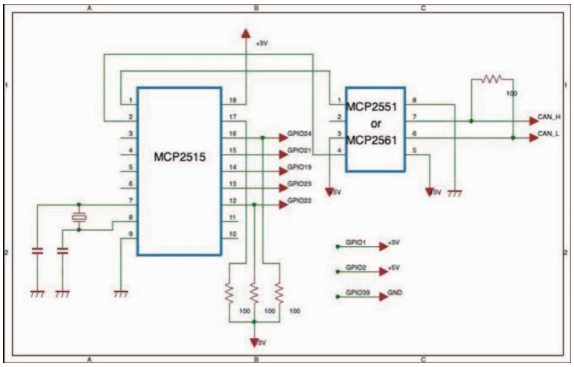

Fig.5.1 The Configured Circuit.

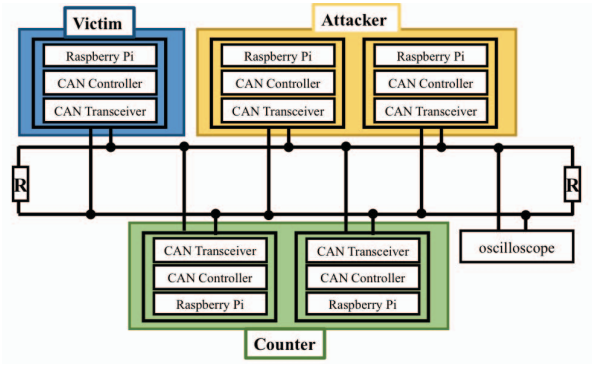

Fig.5.2 Virtual CAN Bus Environment.

et al., we use the one of Cho et al. And, Soma et al. target the attack frame of the attacker, but we target the preceding frame of the attacker. We describe these differences in detail. The bus-off attack of Kameoka et al. needs to inject a large number of dominant bits. For that, Soma et al. showed that it is necessary to keep applying a forced potential on the CAN bus. Since this operation deviates from the CAN protocol, it may be necessary to install a new device on a real vehicle. The bus-off attack by Cho et al. realized by frame injection in accordance with CAN protocol. So, considering that our method is implemented on a real vehicle, it can only be handled by software update of the ECU.

Thirdly, we compare the success rate of the counter attack. According to the study of Soma et al., the success rate of the counter attack is about 30% and they don't identify the causes. In our method, as mentioned later, we performed the experiments ten times in the virtual environment, and all succeeded.

Finally, we compare the impact of the counter attack failure. If the counter attack of Soma et al. falsely detected the bus-off attack and performed the attack, the TECs of all transmitting nodes connected to the same bus are increased and transition to the bus off state. It is because in the method of Kameoka et al., a large number of consecutive dominant bits is transmitted at one time, and TECs increase to 256 or more. In our method, if the bus-off attack is falsely detected and a counter attack is performed, it will be affected only when a node with the same ID as the preceding frame (we call node-A) is transmitting. When a node-A continues to transmit a frame immediately before the victim frame, the node-A transitions to the bus off state. This probability is very low.

V. EXPERIMENTS

In order to confirm the effectiveness of our counter attack, we constructed a virtual CAN bus environment and conducted a counter attack experiment. This chapter describes the environment and the results of the experiment.

A. Experiment environment

In this section, we describe the experimental environment. First, we describe the configuration of CAN bus prototype, which is a device used for the bus-off attack, counter attack of Soma et al., and proposed counter attack. The CAN bus prototype is a CAN simulator used for these experiments. We used [3], and [6] as a reference when creating the CAN bus prototype. The CAN bus prototype consists of three hardware: Raspberry Pi 3 model B, MCP2515(CAN Controller), andMCP2551 or MCP2561(CAN Transceiver).

The configured circuit is shown in Fig. 5.1. Although there are modules on which both MCP 2515 and MCP 2551 are mounted, it was necessary to perform an experiment to apply a forced potential on the bus. So we created the circuits using independent chips each. Then, we describe the configured hardware. We connected Raspberry Pi, MCP 2515 of the CAN controller[7], and the MCP 2551 [8] or MCP 2561 of the CAN transceiver[9]. And, connect the CAN High and CAN Low terminals to the bus.

Next, we describe the software necessary to handle the CAN bus prototype. When conducting simple CAN communication experiments, we used can-utils [10] to confirm the operation of the CAN bus prototype.

Finally, we describe the virtual CAN bus environment. The model of the experimental environment is shown in Fig.5.2. It is assumed that the attacker broke into the CAN bus and installed illegal nodes. we prepared five CAN bus prototypes. One of them is used as a victim node, and two are used as attacker nodes and the rest are the counter nodes. Connect all of them on the same bus. Since the CAN bus is a closed circuit, a termination resistance is required at the CAN High and CAN Low ends of the bus.

B. Experimental method

We describe an experimental method to confirm the effectiveness of the proposed counter attack. We considered the scenario shown in Fig.5.3, Fig.5.4, and Fig.5.5.

Fig.5.3 showed the CAN bus state of normal transmission. Fig.5.4 showed the scenario of the bus-off attack. The victim node transmits some frames, and the attacker nodes inject some attack frames in order to increase the TEC of the victim node. Fig.5.5 showed the scenario of the proposed counter attack. The Counter nodes inject some frames in order to increase the TECs of only the attacker nodes. We describe the movements of these three nodes in detail.

In a vehicle, there are many nodes that transmit some frames periodically [3]. The Victim node is designed to transmit the frames of ID: 7, 8 and 12 once per second. The frame of ID:12

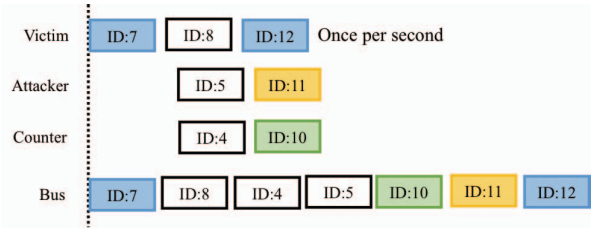

Fig.5.3 CAN Bus in the Normal State.

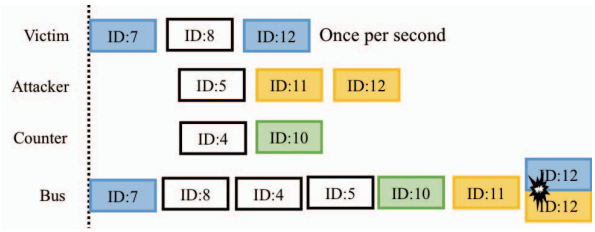

Fig.5.4 Bus-off Attack Scenario.

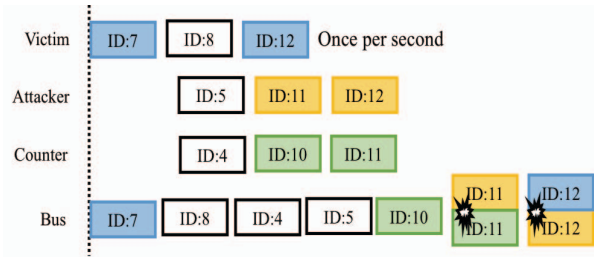

Fig.5.5 Proposed Counter Attack Scenario.

is the victim frame attacked by the Attacker nodes in the experiment. The frame of ID:8 is a delay frame. When a node transmits some frames continuously, there is a problem that the transmission preparation time exceeds the transmission waiting time in CAN. This frame is injected to solve this problem.

The Attacker nodes perform a bus-off attack proposed by Cho et al. against the frame of ID:12 sent by the Victim node. The frame of ID:5 is a delay frame. And the frame of ID:12 is the attack frame. This frame is injected in order to cause some bit errors on the victim frame. As mentioned above, attacker nodes need to inject the preceding frame in order to synchronize the attack frame and the victim frame. In this experiment, the preceding frame for the attacker has the ID:11 (it is smaller than ID:12) and it is transmitted immediately before the attack frame. An attacker node is designed to send the frames of ID:5 and ID:11 after detecting the frame of ID:7 on the CAN bus. And the other attacker node transmits the attack frame after detecting the frame of ID:11. Thus, victim frame and attack frame are transmitted at the same time and some errors occur.

The Counter nodes perform the bus-off attack of Cho et al. against the frame of ID:11, which is the preceding frame of Attacker nodes. Since the Counter nodes also perform the bus-off attack, a preceding frame is required. The frame of ID:10 is the preceding frame for the Counter node, and the frame of ID:11 is the counter frame. The frame of ID:4 is a delay frame. The behavior of the counter nodes is that one node transmits the frames of ID:4 and ID:10 after detecting the frame of ID:7 on

Fig.5.6 Waveform in the Normal State.

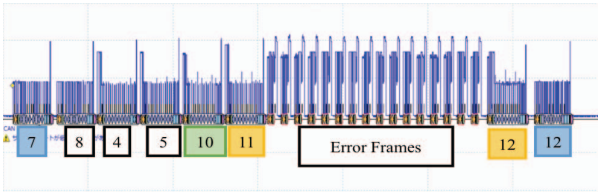

Fig.5.7 Waveform at the time of Bus-off Attack.

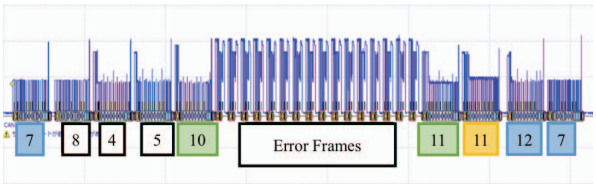

Fig.5.8 Waveform at the time of Counter Attack.

the CAN bus. And the other node transmits the counter frame after detecting ID:10.

C. *Experiment results*

This section describes the results of experiments of the proposed counter attack. The results of the experiment are shown in Fig. 5.6, Fig. 5.7 and Fig.5.8.

Fig.5.6 shows the waveform in the CAN bus of the Fig.5.3. Because of Arbitration, frames are transmitted in the order of ID 7, 8, 4, 5, 10, 11, and 12.

Fig.5.7 shows the waveform when we performed a bus-off attack by Cho et al. At the time of the bus-off attack, the attacker node starts to transmit the frame of ID:12 that is the attack frame. Since the frame of ID:11 is the preceding frame for the attacker node, the attack frame is transmitted at the same time as the victim frame. And as mentioned in Section 3, Phase1 of the bus-off attack starts. The attack frame causes a bit error of the victim node and the victim node generates an error frame. As a result, the TECs of the attacker node and victim node increase by 8. This process continues automatically until Phase2. After detecting the error frames 16 times (it means the transition to Phase2), the frames of ID:12 are transmitted completely. In Phase2, the TEC of the victim node increases by 7, and it transitions to the bus off state.

Fig.5.8 shows the waveform at the time of the counter attack. As mentioned in section 4, we considered a frame causing some bit errors continuously as the victim frame and the frame immediately before the victim frame as the preceding frame. And we performed the counter attack against the preceding frame. In this experiment, the victim frame is considered to be ID: 12, and we performed the bus-off attack against the frame of ID:11. The counter node transmits the counter frame of

ID:11 after detecting the bus-off attack. As a result, we can put the attack node into the bus off state and protect the victim node.

The Counter node succeeded in counter attack against the Attacker node before the Attackers complete the bus-off attack. As a result of performing 10 experiments in the same scenario, they succeeded in all cases. The success rate was higher than the counter attack of Soma et al.

VI. CONCLUSION

In this paper, we pointed out the problems of the counter attack proposed by Soma et al. and proposed the counter attack that improved these problems. The proposed counter attack puts the attacker into the bus off state before the victim node. The difference between the proposed method and the existing method is the target frame of the counter attack. Unlike the existing method, we focused on the preceding frame and performed the counter attack. As a result of the experiments, the success rate of the counter attack is higher than the existing method. In addition, considering the case that the counter attack fails, it is unlikely to affect other nodes. Also, when applied to a vehicle, since the proposed counter attack is realized by frame injection in accordance with CAN protocol, it can only be handled by software update of the ECU. Since one node is protected by two counter nodes, the number of counter nodes increases twice as the number of nodes you want to protect increases. In the future, we will improve it so that we can decrease the number of counter nodes. Also, we will conduct some quantitative experiments to evaluate the proposed method.

ACKNOWLEDGMENT

The authors are grateful to Akira MORI, Ph.D. from National Institute of Advanced Industrial Science and Technology (AIST) for giving the guideline of the work. Also, this work was supported by JSPS KAKENHI Grant Number JP17K00184.

REFERENCE

[1] C.Miller and C.Valasek, “Remote exploitation of an unaltered passenger vehicle”, Black- Hat, USA, 2015.

[2] ISO 11898:2015 Road vehicles - Controller Area Network(CAN), 2015.

[3] K.Cho and K.G.Shin, “Error handling of in-vehicle networks makes them vulnerable”, CCS, pp.1044-1055, 2016.

[4] Ryota Kameoka, Takaya Kubota, Mitsuru Shibasaki, Masayoshi Shirahata, Ryo Kurachi, Ryo Fujichi, “Bus-Off Attack against CAN ECU using Stuff Error injection”, SCIS, 2017.

[5] Daisuke Soma, Akira Mori, Yoichi Hata, “Improvement of Bus Off Counter Attack in Car Network CAN”, CSS, 2018.

[6] https://qiita.com/suzutsuki0220/items/8642b1c3ea51859a95ad, (2019.4.19)

[7] Stand-Alone CAN Controller with SPI Interface, http://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf,(2019.4.19)

[8] High-Speed CAN Transceiver MCP2551 MICROCHIP, https://www.sparkfun.com/datasheets/DevTools/Arduino/MCP2551.pdf, (2019.4.19)

[9] High-Speed CAN Transceiver MCP2561 MICROCHIP, http://hades.mech.northwestern.edu/images/5/5e/MCP2562.pdf, (2019.4.19)

[10] GitHubs can-utils, https://github.com/linux-can/can-utils, (2019.4.19)