OTIDS: A Novel Intrusion Detection System for In-vehicle Network by using Remote Frame

Hyunsung Lee
Korea University
Seoul, Korea
Email: line@korea.ac.kr

Seong Hoon Jeong
Korea University
Seoul, Korea
Email: seonghoon@korea.ac.kr

Huy Kang Kim
Korea University
Seoul, Korea
Email: cenda@korea.ac.kr

***Abstract*—Controller Area Network (CAN) is a bus communication protocol which defines a standard for reliable and efficient transmission between in-vehicle nodes in real-time. Since CAN message is broadcast from a transmitter to the other nodes on a bus, it does not contain information about the source and destination address for validation. Therefore, an attacker can easily inject any message to lead system malfunctions. In this paper, we propose an intrusion detection method based on the analysis of the offset ratio and time interval between request and response messages in CAN. If a *remote frame* having a particular identifier is transmitted, a receiver node should respond to the *remote frame* immediately. In attack-free state, each node has a fixed response offset ratio and time interval while these values vary in attack state. Using this property, we can measure the response performance of the existing nodes based on the offset ratio and time interval between request and response messages. As a result, our methodology can detect intrusions by monitoring offset ratio and time interval, and it allows quick intrusion detection with high accuracy.**

I. Introduction

CAN was developed for efficient in-vehicle communication in 1985 by Bosch. CAN is a broadcast-based communication protocol that supports the maximum baud rate up to 1Mb per second on a single bus. Also, it is designed to operate smoothly in an environment where electromagnetic disturbance factors exist. CAN is proven to be a good fit for in-vehicle networks because it can reduce wiring cost, weight, and complexity. In addition, it provides an efficient error detection mechanism for stable transmission and fast recovery. It does not impair the bus communication by the arbitration process even if several nodes send messages simultaneously. Therefore, all nodes receive messages sequentially, and they can not receive the next message until they process the current received message completely.

In CAN protocol, there are four types of standard CAN frames: (1) Data frame: It is the only frame for the purpose of data transmission. (2) Remote frame: It is used to request a data frame to a destination node. If a node broadcasts a *remote frame* having a particular identifier, a node which generates the corresponding identifier broadcasts a data frame immediately as a response. (3) Error frame: It is used to notify that there is an error in a currently transmitted frame. (4) Overload frame: It is used to delay a start of the next message when CAN controller does not finish processing the current message yet.

Despite various advantages of CAN, modern vehicles are exposed to new threats as they are connected to external networks. Checkoway *et al.* discovered that remote exploitation is possible via a broad range of attack vectors such as CD players, mechanics tools, Bluetooth, and cellular [1], and it can cause remote vehicle control, location tracking, and theft. Kwak *et al.* proposed a machine learning based approach to classify drivers by analyzing features extracted from in-vehicle traffics. Their proposed algorithm could determine whether the driver's vehicle is stolen by another person [2]. In CAN, malicious components can monitor all communications and also transmit any message to existing nodes because there is no way to check the origin of a message. If an attacker injects malicious messages, the other nodes will inevitably deal with them, and it can impair the bus communication.

Attacks on CAN bus. As mentioned earlier, receiving node does not validate the origin of a CAN message, and this property causes a vulnerability in CAN. Wolf *et al.* did highlight vulnerabilities in in-vehicle networks, with a particular focus on the CAN protocol [3]. Some possible attacks on the CAN protocol are summarized as follows. First, there are message injection attacks which are DoS (Denial-of-Service) and fuzzy attack [4]. For the first, the packet flooding by DoS attack can compromise the priority-based arbitration scheme of CAN; and it allows one node to declare dominant status indefinitely on the bus and the other nodes to retreat inevitably. Thus, the CAN protocol is very vulnerable to the traffic volume based DoS attack. Also, the fuzzy attack is indiscriminately injecting messages which are randomly spoofed identifiers having arbitrary data, and it causes unintended behaviors or malfunction of real vehicles. Since a range of valid CAN messages is relatively narrow, a simple fuzzy attack can fill whole range of functional CAN identifiers on the bus, and it causes severe damage with less effort. This attack can override normal outputs of existing nodes or reset some internal conditions of a vehicle. Second, there is an attack to plant an impersonating node to CAN bus, which paralyzes target node and performs the same role of the stopped node [4]. If an impersonating node takes the stopped node's role for malicious purpose, a vehicle will show unintended states without knowing causes.

Findings. Differences in offset ratio and time interval for each state are as follows. First, if *remote frame*s are transmitted in attack-free state, each ratio of instant reply for all identifiers has a fixed value, and ratio significantly drops when message injection attack begins. Second, each ratio of lost reply is about 1% in attack-free state, and ratio increases by from 15% to as much as 70% when message injection attack begins. Third, a correlation coefficient between offset and time interval of response to a request message is also almost about 0.99, but it is broken immediately during attack states. Last, we find that the average of time interval distribution has a high density at a particular time, it is hard to manipulate impersonating node to send responses on time because the performance of nodes varies depending on the crystal oscillator's margin of error and environment of each node such as temperature and location. [5].

Contributions. Our intrusion detection method has following contributions.

1) First, we propose the intrusion detection method based on *remote frame* handling to enhance overall performance and accuracy.
2) Second, we propose a novel algorithm to monitor the change of in-vehicle nodes by using *remote frame* with a particular identifier. This algorithm is beneficial to detect the compromised node with low computation power.
3) Third, our algorithm achieves a successful intrusion detection without modifying the CAN protocol. Therefore, it can be applied to common vehicles using CAN protocol.
4) Finally, we released the dataset used in our experiments to foster further research. We make our dataset available at http://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset.

The rest of this paper is organized as follows. In § II, we introduce the background of the CAN protocol and related works about attack surfaces, authentication method, and Intrusion Detection System (IDS) in CAN bus. In § III, we present attack models. In § IV, we propose our methodologies for the enhanced intrusion detection. In § V, we demonstrate the experiments and evaluation results, and we discuss our methodology in § VI. Finally, we conclude this paper in § VII.

II. Preliminaries

CAN is a message-based protocol. All messages are broadcast to existing nodes on the CAN bus and reception filter of each node decides the selection using arbitration identifier of the message. Each node can extract and use the data after determining whether a message is accepted. Thus, the arbitration identifier is not used as a concept of address but acts as a priority when a collision occurs. The CAN protocol adopts CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) schemes and AMP (Arbitration on Message Priority) for sharing the same bus with multiple nodes. When some nodes attempt to transmit at the same time, these messages should compete to occupy the bus. In this case, arbitration process between messages starts according to a bit level of the identifier. The competition procedure is made by sequentially comparing from first to end bit of arbitration field. It shows a dominant bit when a value is "0" because two bits operate on AND-gate. In the result, a lower value between two identifiers has a higher priority.

A. Standard Frames

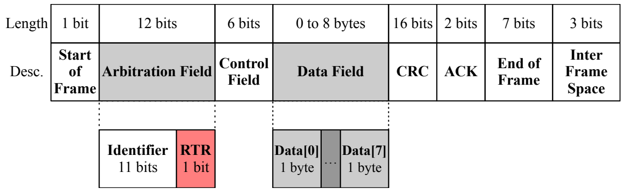


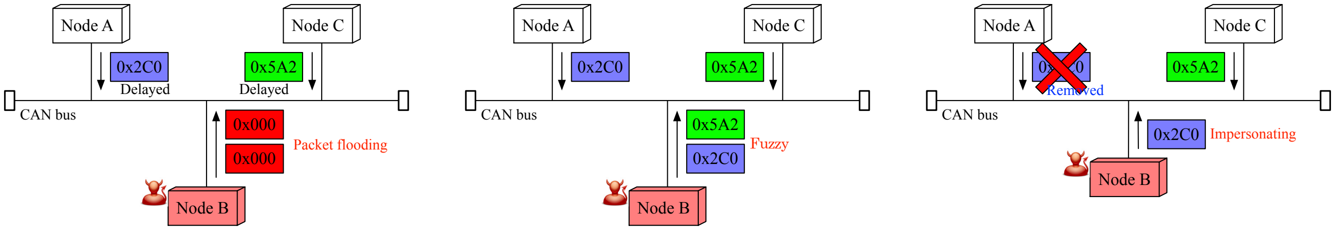
Figure 1: Structure of CAN frame

Figure 1 shows the structure of CAN frame which consists of seven fields. The description of each field in CAN frame is as follows:

- **Start of frame (SOF).** It is a start bit composed of a single dominant bit and informs a start of transmission to all nodes. It can start if the bus is in a rest state.
- **Arbitration Field.** It consists of 11 bits as an identifier and one bit as RTR (Remote Transmission Request). The identifier is used as a priority during the arbitration process, and the RTR is determined according to the kind of CAN frame.
- **Control Field.** It indicates two reserved bits and four bits as DLC (Data Length Code).
- **Data Field.** It refers to actual data for transferring information to another node. Any value from 0 to a maximum of 8 bytes can be made.
- **CRC Field.** It guarantees the validity of a message as CRC (Cyclic Redundancy Code), and all nodes are subjected to verification process of a message generated by a sender using this field.
- **Acknowledge Field.** It consists of two bits as ACK part and ACK delimiter part. If a node receives a valid message normally, it replaces the ACK part, which was a recessive bit, with a dominant bit.
- **End of Frame.** CAN frame is terminated by a flag consisting of seven recessive bits.

Data Frame. The data frame among CAN frames is the only frame for data transmission. Thus, nodes on the bus interact with others by transmitting data frames. If the RTR bit of arbitration field is a dominant bit, it becomes a data frame.

Remote Frame. All nodes periodically broadcast messages regardless of the receiver's status. However, all receiver nodes can require certain kinds of information to run a given task. In this case, a node broadcasts a *remote frame* to all nodes. That is, it requests a data frame having the same value as the identifier of the *remote frame*. Figure 3 shows a response mechanism of the *remote frame* in CAN bus. The *remote frame*'s arbitration field is composed of an identifier and RTR bit which is a recessive bit. The other fields of a *remote frame* are same as a data frame, but it is not intended to transmit information so that the data field can be empty.



(a) DoS attack

(b) Fuzzy attack

(c) Impersonation attack

Figure 2: Three types of attack scenarios on a CAN bus.

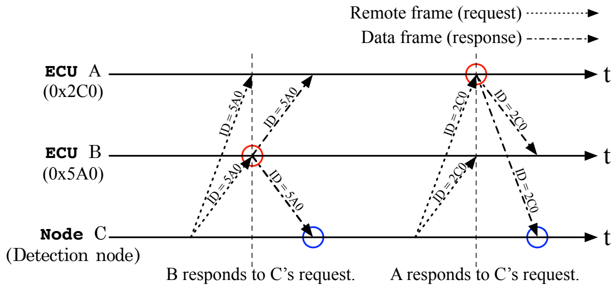


Figure 3: A response mechanism in CAN bus. If node C broadcasts a *remote frame* having a particular identifier, the node which sends the corresponding identifier broadcasts a data frame immediately as a response.

B. Propagation delay time

The propagation delay time is determined by the distance and own clock frequency of end-to-end controller [5]. When a message arrives at an input stage of receiving node after being output from a transmitting node, the following factors determine the delay time: (1) Required by a transmission controller to output a signal. (2) Required by a transceiver to generate a signal on medium. (3) Required by taking to carry a signal along the medium. (4) Required by taking a transceiver to transfer a signal to a receiving controller. (5) Required by a receiving controller to process an input signal. The defined delay time is divided into two layers of CAN, those related to physical properties of the medium and electric characteristics that can occur in a controller and transceiver. Thus, it can be seen that delay time of these two types represents the total time taken to move from one node to another, and messages show different travel time.

C. Related Work

Lilsson and Larson evaluated how current in-vehicle networks to stand attacks by simulating plausible attacks targeting ECUs on the CAN bus [6]. They demonstrated that in-vehicle networks of current vehicles lack security mechanisms against attacks. Therefore, we investigate attack surfaces in in-vehicle networks and how to defend them using authentications and intrusion detection system.

Attack surface of vehicle. Hoppe *et al.* warned basic attack principles and referred the special relevance of intrusions from the black box perspective in the automotive domain [7]. It is demonstrated that CAN protocol major drawback is a lack of security mechanism, and it does not provide protections against attacks such as intrusion, denial of service or impersonation [8]. The OBD-II (On-Board Diagnostics-II) port provides direct and access to in-vehicle networks in modern vehicles. Telematics control unit can be targeted, and compromised by an external attacker. It is demonstrated that such a compromise allows arbitrary remote control of the vehicle [9]. Thus, attack surfaces of either internal or external can be available by which an attacker might compromise a component and gain access to in-vehicle networks [4]. These attackers have an ability to control vehicle functions and completely ignore input that is selectively braking individual wheels on demand, disabling the brakes, stopping the engine, and so on.

Authentication. Herrewege *et al.* proposed a message authentication protocol of the CAN bus [10]. They retrofitted the data frame format of the CAN protocol which adds authentication field using HMAC (Hash-based Message Authentication Code). Lin and Sangiovanni-Vincentelli presented a run-time authentication in the system steady state [11]. Each node stores key elements which are composed of identifier table, receiving and sending counters and the pair-wise symmetric secret keys for the authentication process. Groza and Murvay presented a broadcast authentication protocol based on key-chains and time synchronization, a commonly used mechanism in wireless sensor networks [12]. Oguma *et al.* proposed an attestation-based security architecture that is a master node which can verify the other nodes similar to a verification server [13].

Intrusion detection in CAN bus. Larson *et al.* explained specification based attack detection in in-vehicle networks [14]. They created a set of security specifications from CAN v2.0 and CANopen v3.01. Cho and Shin presented an anomaly-based intrusion detection system using a baseline of ECUs' clock with the recursive least squares algorithm [15]. They acquire fingerprints of ECUs and detect anomaly situations in in-vehicle networks. Toledo and Wang presented a robust nonparametric detection mechanism for the CSMA/CA media-access control layer DoS attack that does not require any modification to protocols [16]. Hoppe *et al.* designed the

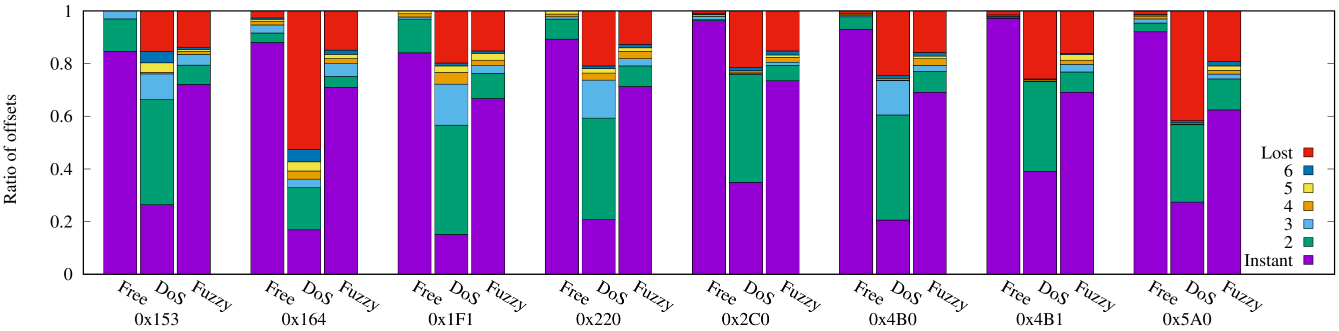


Figure 4: Offset ratio for all identifiers used in our experiment. Instant reply ratio is dominant in attack-free state. However, in attack states, not only instant reply ratio decrease but also lost reply ratio increase.

central gateway and proposed intrusion detection based on IT-forensic measures [17]. Song and Kim discovered that time intervals of CAN messages are changed in attack states. They detect anomaly using analysis of intervals [18]. Alternatively, Müter proposed methodologies of detecting an intrusion which monitors a traffic state [19] or entropy in in-vehicle networks [20].

III. Attack Model

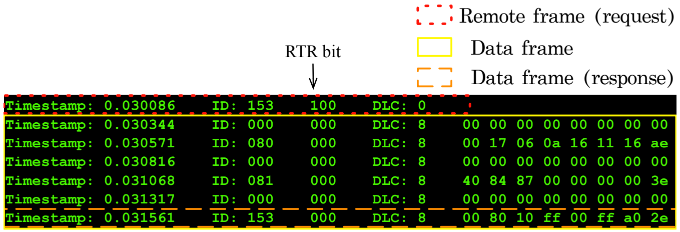
A. Adversary model

An attacker can remotely/physically manipulate in-vehicle nodes via various attack surfaces such as CD players, mechanics tools, Bluetooth, and cellular [1]. In this paper, we present two types of attackers who can directly modify an in-vehicle node of the CAN bus, and who can inject unauthenticated messages via bus access. First, we assume an attacker who is capable of physical modifications can take control of a target node, stopping transmission of messages, and let impersonating node send messages to replace the terminated node. Since the malicious node supersedes the target node, the system can not detect changes of overall messages. We call this type of attack as impersonation attack. Second, we assume an attacker who does not have control of nodes but can inject malicious messages to induce malfunctions and manipulate vehicle to prevent it from functioning. As a result, we will consider two different types of message injection attack: DoS and fuzzy.

B. Attack Scenarios

Based on the presented adversary model, we present three types of attack scenarios, as shown in Figure 2.

DoS attack. An attacker can inject high priority messages in a short cycle on the bus. Since any node is not owned or controlled by a network manager, a malicious node may not comply with protocol rules to obtain unauthorized access. DoS attack messages aim at occupying the bus using the theoretically highest priority identifier as `0x000`. Figure 5 shows messages during the DoS attack mounted state. The first record having the third index as `100` is a *remote frame*, and the last is the corresponding response message. There are three injected `0x000` messages between request and response messages, and they delay reception of response inevitably.



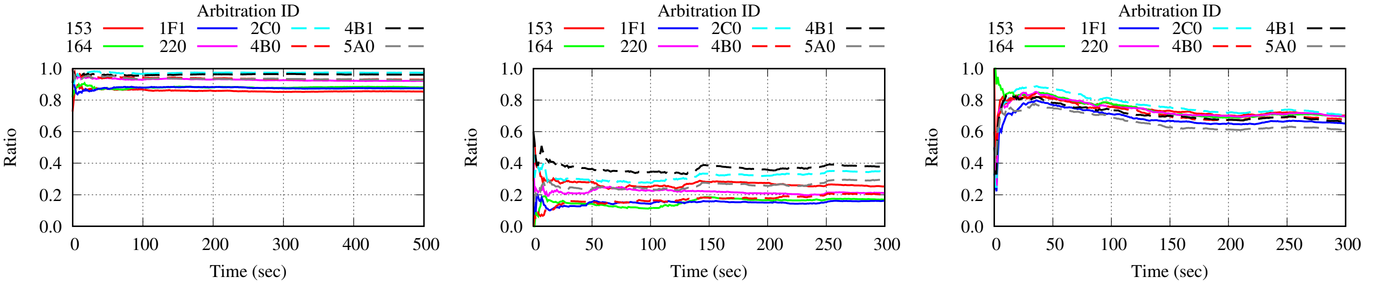
```
Timestamp: 0.030086    ID: 153    100    DLC: 0
Timestamp: 0.030344    ID: 000    000    DLC: 8    00 00 00 00 00 00 00 00
Timestamp: 0.030571    ID: 080    000    DLC: 8    00 17 06 0a 16 11 16 ae
Timestamp: 0.030816    ID: 000    000    DLC: 8    00 00 00 00 00 00 00 00
Timestamp: 0.031068    ID: 081    000    DLC: 8    40 84 87 00 00 00 00 3e
Timestamp: 0.031317    ID: 000    000    DLC: 8    00 00 00 00 00 00 00 00
Timestamp: 0.031561    ID: 153    000    DLC: 8    00 80 10 ff 00 ff a0 2e
```

Figure 5: An example of DoS attack. The first record is a *remote frame* which has the third index as `100` and the DLC as `0`. The last record is a corresponding response message.

Since all nodes share a single bus, increasing occupancy of the bus can produce latencies of other messages and cause threats regarding availability with no response to driver's commands.

Fuzzy attack. An attacker can inject messages of randomly spoofed identifiers with arbitrary data. As a result, all nodes will receive lots of functional messages, and it causes unintended vehicle behaviors. To exploit the fuzzy attack, an attacker observed in-vehicle messages and selected target identifiers to produce unexpected behaviors. When inserting arbitrary data into spoofed identifiers, we found that the steering wheel is shaken tremendously, turn signal lamps light irregularly, instrument panel blinks in countless ways, and gear shift changes automatically. Unlike the DoS attack, it paralyzes functions of a vehicle rather than delaying normal messages via occupancy of the bus.

Impersonation Attack. Cho and Shin introduced an attacker who can stop messages transmission by controlling the target node and can plant/manipulate an impersonating node [15]. If a victim node stops transmitting, all messages sent by the targeted node will be removed from the bus. However, when an ECU (Electronics Control Unit) receive a *remote frame*, it is designed to have to transmit a data frame immediately. If a receiver node does not receive a response to a *remote frame*, it can be known that the existing node is attacked or broken. In this case, an adversary can plant an impersonating node to respond to a *remote frame*. Thus, the impersonating node broadcasts data frames periodically and respond to a *remote frame* like targeted node, to conceal whether each node

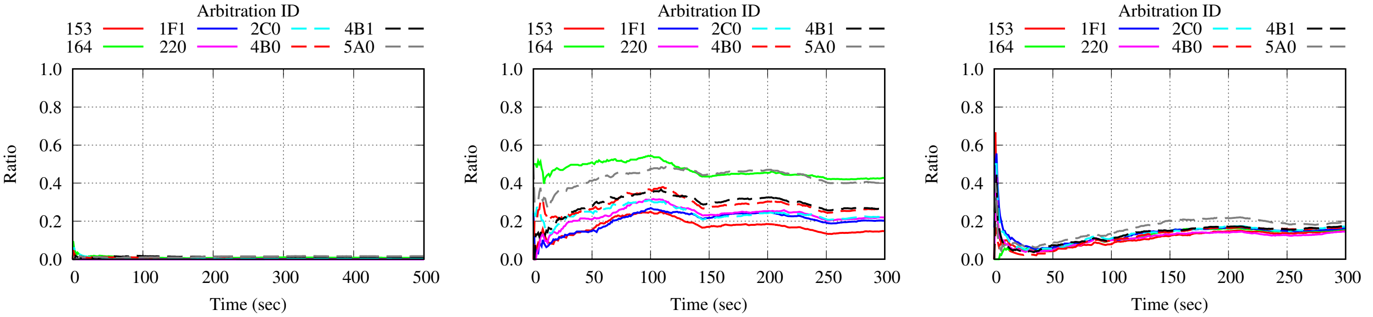


(a) Attack-free

(b) DoS attack

(c) Fuzzy attack

Figure 6: Instant reply ratio for each state. In Figure 6a, all identifiers have a fixed ratio. In Figure 6b and Figure 6c, ratios decrease compared to attack-free state.



(a) Attack-free

(b) DoS attack

(c) Fuzzy attack

Figure 7: Lost reply ratio for each state. In Figure 7a, all identifiers are close to 0%. In Figure 7b and Figure 7c, ratios increase compared to attack-free state.

changes its role.

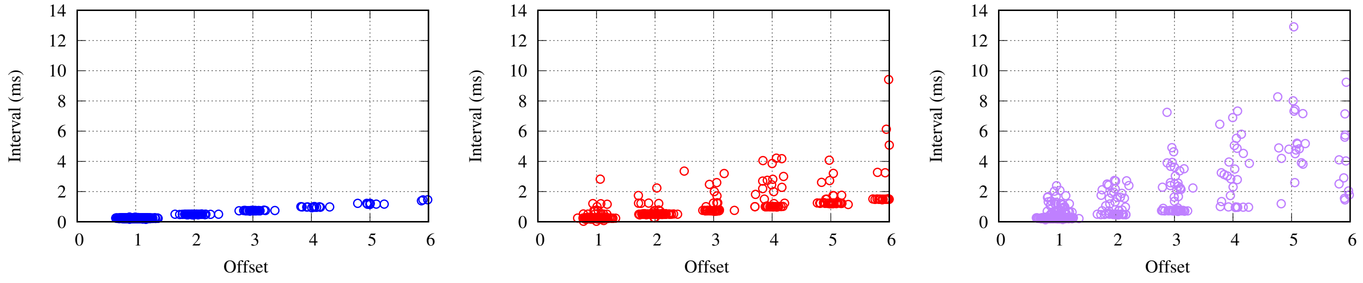
C. Defense against the attacks

Based on attack scenarios described, message injection attacks generate kinds of messages whose frequencies are changed. An adversary injects attack messages in a short cycle, then normal data have no choice but to be delayed. Thus, state-of-the-art IDS mainly used frequency or sequence analysis of overall messages [19], [20]. However, a frequency may not be changed if impersonating node supersedes an existing node. We discovered that response performance is degraded when the clock frequency or location is changed. If we get distributions of response time for all identifiers, our detection system can detect impersonation attack state using time intervals which are getting longer than before. As a result, our IDS can detect not only message injection attack but also detect impersonation attack.

IV. Methodologies

In this paper, we implemented an intrusion detection system, named as OTIDS *(Offset Ratio and Time Interval based Intrusion Detection System)*. OTIDS broadcasts *remote frame*s periodically and receives response messages from sender nodes. In this case, we defines a unit of analysis called "`window`". A window is composed of messages from *remote frame* to response data frame so that it analyzes a small amount of data compared to the total data. If we designed that *remote frame*s are broadcast every 0.01 second during measurements, only 10% of the total data are needed for the analysis. Moreover, because the system acquires an offset and time interval and identifier information of messages, it needs a small resource of the limited environment of in-vehicle networks. We define some features to apply for our detection model as follows.

- **Window.** It is a bundle of messages from request to response message. One window is composed of at least two records and at most seven records.
- **Offset and time interval.** They mean an order and time interval from request to response message. Each node receives messages sequentially in CAN, then it is possible to count an order and to calculate time interval between request and response messages.
- **Instant reply.** It means that receiving a response immediately after sending a *remote frame*. In our methodology, it is a window which has offset as one.
- **Lost reply.** It means that missing response to a *remote frame*. The lost reply criterion is when the offset exceeds the sixth message.



(a) Attack-free

(b) DoS attack

(c) Fuzzy attack

Figure 8: Scatter plots of identifier = `0x164` for each state. In attack states, regularity of time interval about offset is broken.

A. Exploratory data analysis

Our detection method sequentially broadcasts *remote frame*s for all identifiers and independently stores offsets and time intervals of windows. We chose several identifiers and found out functions of identifiers. After that, a thousand of *remote frame*s were sent to measure offset ratio of each identifier. As a result, we discovered a difference between attack-free and attack states, as shown in Figure 4. Numbers from one to six mean offset and "`lost`" which means a response message not coming. If response message offset exceeds the sixth record, it is considered as a response failure. It is uncertain whether a response to the request or a message transmitted periodically by an existing node. Moreover, another reason for failure criterion is that probability of missing a response is less than 0.1% in attack-free. When DoS and fuzzy attacks are mounted, we found not only decrease for a ratio of instant reply but also increase for a ratio of lost reply comparing to attack-free state.

Under message injection attack. We observe changes about three features (ratio of instant reply, lost reply, and correlation coefficient) do detect message injection attacks. In Figure 6, each instant reply ratio converges to a particular value in attack-free state, and it steadily decreases when attacks are mounted. In Figure 6a, there is a difference between convergence values for each identifier, hence our detection method should set thresholds independently for all identifiers. Since ratios can be corrupted easily by message injection attack, it provides clear criteria for detecting attacks. IDS first set margin-based error bounds for thresholds, and it detects attacks when any ratio of all identifiers is out of its range. In addition to instant reply ratio, lost reply ratio also changes when attacks are mounted, as shown in Figure 7. All slopes of lost reply ratios are close to 1% in attack-free, but they increase in DoS and fuzzy attack. DoS attack state shows a more significant difference with attack-free state, but fuzzy attack also shows a difference compared to attack-free state. Last, Figure 8 shows that ID = `0x164` scatter plot which consists of offsets and time intervals for each state.

We use Pearson correlation coefficient to find out relationships between offsets and time intervals. Pearson correlation coefficient can be calculated as follows:

$$\rho(O,T) = \frac{n(\sum_{i=1}^{n} O_i T_i) - \sum_{i=1}^{n} O_i \sum_{i=1}^{n} T_i}{\sqrt{\{n\sum_{i=1}^{n} O_i^2 - (\sum_{i=1}^{n} O_i)^2\}\{n\sum_{i=1}^{n} T_i^2 - (\sum_{i=1}^{n} T_i)^2\}}}$$

where,

O_i : offset
T_i : time interval

In Figure 8a, the correlation coefficient between offsets and time intervals is almost converged to 1 and regression line is close to a straight line. However, when attacks are mounted, it decreases significantly and the regularity of time interval about offset is broken. Thus, it can detect intrusion quickly when data are far from the regression line, or the correlation falls below a threshold.

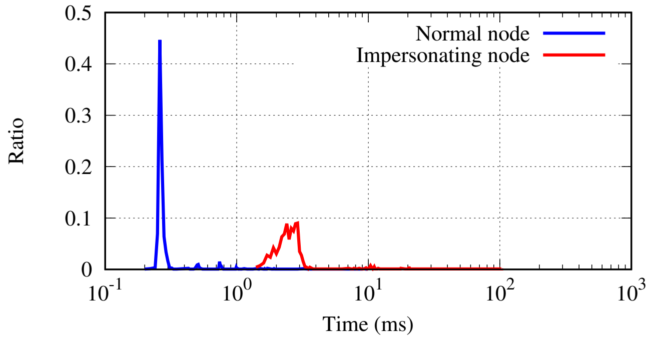


Figure 9: Time interval distribution of normal and impersonating node.

Under impersonation attack. We observe changes in the average of response time interval to detect impersonation attack. In Figure 9, the time interval density graph of impersonating node is more behind than that of the normal node. We found that it is hard to control an arrival of response within a particular time. The impersonating node is programmed to send a reply message immediately without delay, but it responds slower rate than the existing node. The reason is that impersonating node does not have the same clock frequency, and it is a different location where the existing node is

Algorithm 1 Offset and time interval measurement

```
Initialize:
    $Queue_k = []$
    $Instant_k = 0,\ Lost_k = 0$
while message arrives do
    $time, id \leftarrow$ timestamp, identifier of arrival message
    if arrival message is $remote\ frame$ then
        $remote\_flag \leftarrow$ True
        $target \leftarrow id$
        $offset \leftarrow 0$
        $request\_time \leftarrow time$
        continue
    if $remote\_flag$ then
        $offset \leftarrow offset + 1$
        if $offset > 6$ then                          ▷ Lost reply
            increment $Lost_{target}$
            $remote\_flag \leftarrow$ False
            continue
        if $id == target$ then
            $interval \leftarrow time - request\_time$
            enqueue $(offset, interval)$ to $Queue_{target}$
            if $offset == 1$ then                     ▷ Instant reply
                increment $Instant_{target}$
            $remote\_flag \leftarrow$ False
            continue
```

original. In-vehicle nodes' clock frequencies can be operating at 12Mhz, 16Mhz or 11.0592Mhz for functional reasons. However, even if the frequency is the same, the performance varies depending on the crystal oscillator's margin of error or environment of each node such as temperature and location [5]. To detect impersonating node attack, we set the threshold by an average of time intervals as follows:

$$\mu_{id}^{(i)} = \frac{1}{N}\sum_{i=1}^{n} \Delta T_{id}^{(i)}$$

where,

ΔT_{id} : time interval of messages

We use the average of time intervals for detecting impersonation attack because the graph has a multimodal distribution. Figure 9 shows one or more peaks on normal node graph, but data concentrates on the first peak. Thus, the average of cumulative time intervals approximates the first peak having high density. We set thresholds by adding a margin of error to the measured averages. As a result, the attack state is detected when the average of time intervals are out of range.

B. Measurement algorithm

We designed how to measure the offset and time interval like Algorithm 1. While our system collects windows, they are stored in each identifiers' first-in-first-out queue which has a limited size. The reason of limited queue size is that the system calculates ratio and correlation coefficient with cumulative offsets, time intervals. If it stores the cumulative data from the start of the car, it will become burdensome later to store the total data. Thus, we set each queue having a thousand of store space during measurement. Before the detection, thresholds are set by measures of an attack-free state and IDS checks anomaly state once per second. Finally, it can detect the anomaly phenomenon when measures are changed below thresholds.

V. Experiments and Evaluation

Based on the described attack and detection model, we configure several types of equipment with Raspberry Pi and Arduino. They are connected to OBD-II port of a vehicle as attack and detection node.

A. Configuration of equipments

The detection node collects and analyzes offsets and time intervals of windows for each identifier in both cases of with and without attacks. We first applied to the CAN bus implemented on a breadboard, and extend the experiment to a real vehicle, Kia Soul.

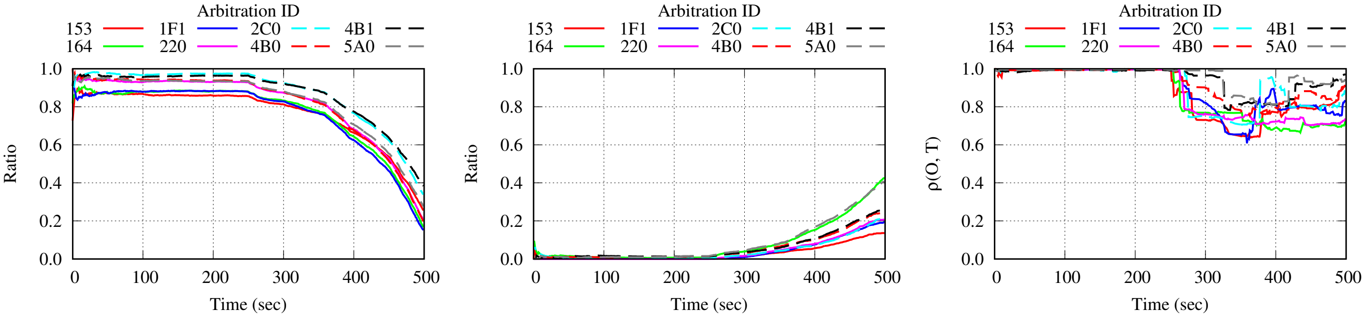
Adversary node. To construct the attack nodes, we configure an Arduino with CAN-shield and a Raspberry Pi3 with PiCAN2. These devices perform attack roles as DoS, fuzzy and impersonation attack. The DoS attack node continuously injects messages with the identifier = 0x000 using a loop, and the fuzzy attack is implemented by a random function to inject identifiers existing in in-vehicle networks with arbitrary data. These attacks are implemented via Raspberry Pi3. Next, the impersonating node chooses a particular identifier, and it is programmed that data frame to be transmitted periodically. In addition, when a *remote frame* is received, it sends a data frame immediately as a response. This attack node is constructed via Arduino with CAN-shield which has a chip as MCP2515 as one of most commercial on CAN bus. Therefore, an adversary can achieve as closely as same possible configuration as in-vehicle ECU.

Detection node. The detection node is constructed by a Raspberry Pi3 with PiCAN2 shield. The node broadcasts *remote frame*s sequentially and collects offsets and time intervals of windows. Then, we applied the data to the detection model utilized by our methodologies to check whether a state is malicious.

B. Defending Against Message Injection Attack

To evaluate the detection model against message injection attack under a real vehicle, we applied it to a dataset in which message injection attack begins at 250 seconds. During measurement of offsets, time interval and correlation coefficient, our model inspects anomaly states of a vehicle continuously in real-time.

DoS attack. When the DoS attack model that is infinitely inserting messages with the identifier = 0x000 initiated, we discover the changes of features via three types of graphs, as shown in Figure 10. First, Figure 10a shows instant reply ratios for all identifiers, and we intuitively understand that all slopes are getting lower as soon as DoS attack are mounted.

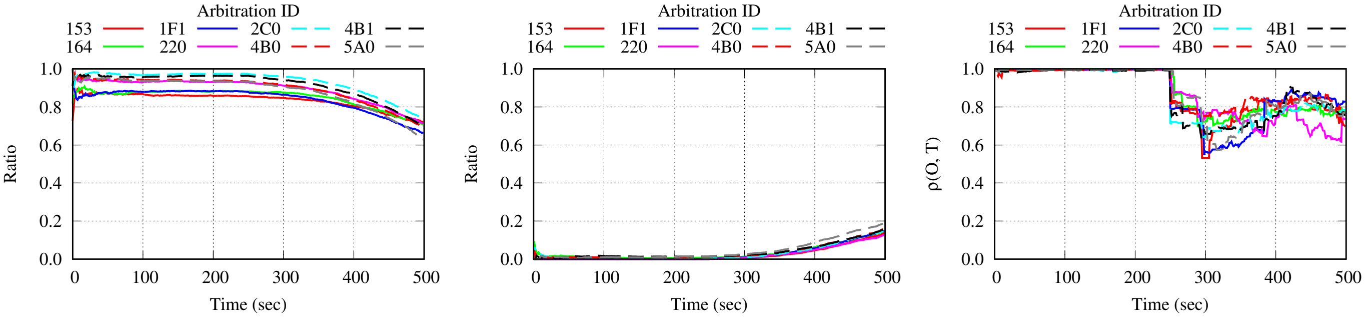


(a) Instant reply ratio

(b) Lost reply ratio

(c) Correlation Coefficient

Figure 10: Time series graphs under the DoS attack (attack starts at 250 seconds).



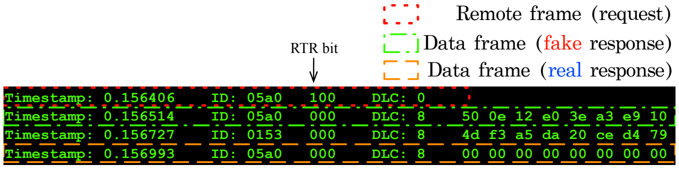
(a) Instant reply ratio

(b) Lost reply ratio

(c) Correlation Coefficient

Figure 11: Time series graphs under the fuzzy attack (attack starts at 250 seconds).

Before the intrusion begins, the rate of identifiers moved within a very narrow range. However, it shows how instant offset ratios are different for both cases of with and without a DoS attack. Therefore, attack messages delay others, and response performance of existing node is degraded. In addition, if we analyze messages of windows in Figure 5, we can see what identifiers used for DoS attack. Second, Figure 10b shows the changes about lost reply ratio. The DoS attack causes that not only messages of existing nodes are delayed, but also the case where unanswered responses increase rapidly. Finally, Figure 10c shows the changes of the correlation coefficient for offsets and time intervals, and we can see that slopes of the graph are changed more dramatically than reply ratio figures.



```
Timestamp: 0.156406    ID: 05a0   100     DLC: 0
Timestamp: 0.156514    ID: 05a0   000     DLC: 8     50 0e 12 e0 3e a3 e9 10
Timestamp: 0.156727    ID: 0153   000     DLC: 8     4d f3 a5 da 20 ce d4 79
Timestamp: 0.156993    ID: 05a0   000     DLC: 8     00 00 00 00 00 00 00 00
```

Figure 12: An example of fuzzy attack. The first record is a *remote frame*, and the last record is a corresponding response message. However, the detection system recognizes the second record (injected message) as a response.

Fuzzy attack. The fuzzy attack is also that injecting attack messages, but it inserts arbitrary data into spoofed identifiers. Thus, changes of three features are a little bit different from the DoS attack state. DoS attack messages delay other messages by occupying the bus communication using a highest priority = 0x000, but some fuzzy attack messages can not get an occupancy when identifier of message defeats in an arbitration process. Therefore, slopes of instant and lost reply ratio do not decrease sharply compared to the DoS attack, as shown in Figure 11a and Figure 11b. However, since the detection system occasionally recognizes attack message as a response to the request, the correlation changes rapidly more than the DoS attack state. In Figure 12, the first record is a *remote frame*, and the last is a real response. However, the second message injected by an attacker supersedes the real answer. As a result, the detection system inevitably collects until the second record as one window. Because the correlation between offset and time interval of the injected message does not have regularity, the correlation is broken rapidly compared to DoS attack state.

C. Defending Against Impersonation Attack

To evaluate detection model against the impersonation attack, we removed targeted messages and planted the impersonating node which has the same role of the terminated. In other words, it can send disguised messages periodically and

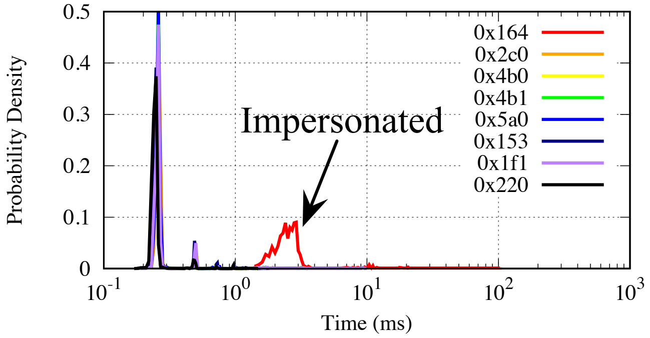


Figure 13: Time interval distribution of impersonating node and normal nodes

respond to *remote frame*s. As a result, the overall data of a bus does not change, and it does not deteriorate the availability of the bus like the message injection attack. Figure 9 shows time interval comparison graphs of impersonating node and existing nodes. When the malicious node starts to transmit responses to requests, the time interval distribution graph of impersonated messages shows a different with normal identifiers' messages. As shown in Figure 13, the distribution of ID = `0x164` is different from existing nodes. Thus, we can quickly discover the impersonating node attack via distribution graphs of reply messages.

VI. Discussion

The methodology proposed in this paper requires additional node arrangement because it sends and receives messages to and from ECUs on the CAN bus. During the experiment, the detection node sends request messages every 0.01 second and receives responses from each ECU. As a result, we found that only 4% of the total messages correspond to the request. Therefore, the overhead of communication due to the deployment of additional hardware is insufficient, and the proposed attack models can be detected through a significantly smaller amount than the total data.

We used instant reply ratio, lost reply ratio, the correlation coefficient between offsets and time intervals, and average response time as features to detect several attacks. These features showed slightly different results depending on the attack model, but as a result, these values helped to catch each attack state quickly. In the case of DoS attack, the ratio of instant reply decreased drastically. In the case of fuzzy attack, fast detection was achieved by the correlation coefficient between offsets and time intervals. Also, we found that the average response time of the impersonating node attack was different from other existing nodes. Therefore, our algorithm can detect the proposed attack models in any state quickly.

We also checked whether an attacker could bypass the detection method. Our algorithm is designed to detect message injection attacks regardless of the change of attack interval. Since the detection node sends request packets (*i.e. remote frame*) in a short cycle, there will always be some windows which have abnormal offset and time interval value if the in-vehicle network (*i.e.* CAN) is under attack. Any small number of irregular windows which have inconsistent offset and time interval can affect the decrease of correlation coefficient; Because of this characteristics, attackers cannot bypass the proposed detection method.

Finally, we connected the Raspberry Pi3 to the OBD-II terminal to place the detection node on the CAN bus. We used Raspberry Pi3 equipped with PiCAN2 to construct a prototype of detection node because it can communicate with ECUs on CAN bus. We implemented it using Python to configure the algorithm of this prototype. If a real detection node is placed on the CAN bus, not the OBD-II terminal, it could be expected better detection efficiency than our prototype of detection node.

VII. Conclusion

As modern vehicles are exposed to a new range of threats, IDS for vehicles becomes one of the most important security components. In this paper, we analyze the response performance of nodes to detect whether a vehicle is under attacks or not. OTIDS can successfully detect the message injection attack and impersonating node attack which can be the most dangerous attacks for vehicles. Moreover, OTIDS can find out what types of messages are injected during message injection attack, and which node is compromised during impersonating node attack. We believe that our detection method contributes to enhancing vehicle security without changing the CAN protocol. In the future, we will apply the proposed algorithm to the other in-vehicle communication protocols including Ethernet, FlexRay, and LIN protocol.

Acknowledgment

This work was supported by Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-TB1403-00.

References

[1] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. of USENIX*, 2011.

[2] B. I. Kwak, J. Woo, and H. K. Kim, "Know your master: Driver profiling-based anti-theft method," in *Privacy, Security and Trust (PST), 2016 14th Annual Conference on*. IEEE, 2016, pp. 211–218.

[3] M. Wolf, A. Weimerskirch, and C. Paar, "Security in automotive bus system," in *Proc. of ESCAR*, 2004.

[4] K. Koscher, A. Czeskis, and F. Roesner, "Experimental security analysis of a modern automobile," in *Proc. of SP*, 2010.

[5] D. Paret, *Multiplexed Networks for Embedded Systems: CAN, LIN, FlexyRay, Safe-by-Wire...* Wiley, 2007.

[6] D. K. Nilsson and U. E. Larson, "Simulated attacks on CAN buses: Vehicle virus," in *Proc. of Aisa CNS*, 2008.

[7] T. Hoppe, S. Kiltz, and J. Dittmann, "Automotive it-security as a challenge: Basic attacks from the black box perspective on the example of privacy threats," in *Proc. of SAFECOMP*, 2009.

[8] A. Boudguiga, W. Klaudel, A. Boulanger, and P. Chiron, "A simple intrusion detection method for controller area networks," in *Proc. of IEEE ICC*, 2016.

[9] I. Foster, A. Prudhomme, K. Koscher, and S. Savage, "Fast and vulnerable: A story of telematic failures," in *Proc. of USENIX*, 2015.

[10] A. V. Herrewege, D. Sigelee, and I. Verbauwhede, “CANAuth - a simple, backward compatible broadcast authentication protocol for CAN bus,” in *Proc. of ECRYPT*, 2011.

[11] C.-W. Lin and A. Sangiovanni-Vincentelli, “Cyber-security for the controller area network (CAN) communication protocol,” in *Proc. of CyberSecurity*, 2012.

[12] B. Groza and S. Murvay, “Efficient protocols for secure broadcast in controller area networks,” *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2034–2042, 2013.

[13] H. Oguma, A. Yoshioka, M. Nishikawa, R. Shigetomi, A. Otsuka, and H. Imai, “New attestation-based security architecture for in-vehicle communication,” in *Proc. of IEEE GLOBECOM*, 2008.

[14] U. E. Larson, D. K. Nilsson, and E. Jonsson, “An approach to specification-based attack detection for in-vehicle networks,” in *Proc. of IEEE Intelligent Vehicles Symposium*, 2008.

[15] K.-T. Cho and K. G. Shin, “Fingerprinting electronic control units for vehicle intrusion detection,” in *Proc. of USENIX*, 2016.

[16] A. L. Toledo and X. Wang, “Robust detection of MAC layer denial-of-service attacks in CSMA/CA wireless networks,” *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 347–358, 2008.

[17] T. Hoppe, S. Kiltz, and J. Dittmann, “Security threats to automotive CAN networks - practical examples and selected short-term countermeasures,” in *Proc. of SAFECOMP*, 2008.

[18] H. M. Song, H. R. Kim, and H. K. Kim, “Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network,” in *Proc. of ICOIN*, 2016.

[19] M. M*ü*ter, A. Groll, and F. C. Freiling, “A structured approach to anomaly detection for in-vehicle networks,” in *Proc. of IAS*, 2010.

[20] M. M*ü*ter and A. Naim, “Entropy-based anomaly detection for in-vehicle networks,” in *Proc. of IEEE IVS*, 2011.