





## Research Article

# A CAN Bus Security Testbed Framework for Automotive Cyber-Physical Systems

Dongxian Shi <sup>1,2</sup>, Liang Kou <sup>1</sup>, Chaobin Huo <sup>3</sup>, and Ting Wu <sup>1</sup>

<sup>1</sup>School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China

<sup>2</sup>College of Information Technology, Zhejiang Institute of Economics and Trade, Hangzhou 310018, China

<sup>3</sup>National Computer System Engineering Research of China, Beijing 100000, China

Correspondence should be addressed to Liang Kou; [kouliang@hdu.edu.cn](mailto:kouliang@hdu.edu.cn)

Received 25 May 2022; Revised 9 July 2022; Accepted 19 July 2022; Published 12 August 2022

Academic Editor: Chenglu Jin

Copyright © 2022 Dongxian Shi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The popularization of electronic devices and the enrichment of external interfaces have increased the attack surface of the automotive cyber-physical system (CPS). As a vital part of the CPS, the controller area network (CAN) is more vulnerable to security threats due to the lack of corresponding security protection mechanisms. This kind of security problem has also attracted extensive attention from academia and industry. Researchers have proposed numerous intrusion detection models for the in-vehicle CAN bus, solving some security problems to a certain extent. However, vehicle manufacturers seldom disclose the internal details of vehicle ECUs due to safety concerns. Thus, it is difficult for researchers to investigate the operation mechanism of the bus. Meanwhile, there is a risk of personal safety in completing attack experiments on real vehicles, which can also lead to the lack and diversification of in-vehicle network data, especially the data of attack behavior. Based on real vehicle data, an open, adaptable, and low-risk CAN bus security testbed framework in the automotive CPS is proposed in this study, aiming to enrich the operation data of the CAN bus and enhance the personal safety of researchers. Besides, the delay of the testbed sending and receiving periodic and aperiodic CAN messages is theoretically explored. The results demonstrate that the generated timestamp in the dataset is mainly associated with the timestamp of the real vehicle data and that the transmission and collection of time series data are completed by Algorithm 1 and Algorithm 2. In the evaluation of the security testbed, the stability of the security testbed is studied from the two indicators of delay and packet loss rate. The experiment reveals that the testbed has a small relative delay difference and a low packet loss rate. Moreover, the DTW algorithm is employed to calculate the distance between the real vehicle and the testbed, and the experimental results demonstrate that the testbed is featured with high similarity and simulation.

## 1. Introduction

With the rapid development of technologies including the Internet of Vehicles, new energy, and artificial intelligence, vehicle manufacturers have configured more external interfaces and electronic devices for consumers, providing a more comfortable driving and entertainment experience. Meanwhile, the security of the CPS has been more threatened [1]. For example, hackers obtain root privileges through Wi-Fi and tamper with the electronic control unit (ECU) firmware on the bus [2, 3]. Therefore, the issue of vehicle security has attracted extensive attention from academia

and industry. As one of the core components of modern automotive electronics, the ECU exchanges information with other ECUs through the bus protocol. Most bus protocols are public, such as CAN which is currently the most widely used bus protocol [4, 5]. However, the vehicle manufacturers or parts manufacturers have not disclosed the details of the internal parameters of the ECU in order to protect intellectual property rights [3], which means that the specific operating conditions on the bus are invisible to vehicle users and that security researchers need to have strong reverse analysis capabilities, thus bringing great difficulties to analyze the internal working mechanism of the ECU. At

the same time, the implementation of bus attack experiments in the running state of the vehicle will result in unpredictable consequences, and it will also bring personal safety risks to researchers. Some researchers have published datasets of real vehicles [6, 7] that contain some common attack behaviors, which is conducive to the study on CAN bus security. However, in the face of complex vehicle environment, there is still a lack of rich attack types and attack behavior data. As a result, it is of practical significance to establish an open, adaptable, high-accuracy, and low-risk in-vehicle bus security testbed [8], providing important support for the theoretical research and real-time detection of vehicle CPS security [9, 10].

At present, the existing vehicle security testbeds are mainly based on the CAN bus, and there are mainly the following problems: (1) The versatility is not very strong [9, 11], and the same testbed cannot be shared between different models. (2) The security testbed is mainly constructed by simulating the vehicle through software, which cannot accurately evaluate the physical characteristics of the CAN bus and the ECU including power consumption and current, lacking certain authenticity. (3) The developed hybrid testbed improves the original vehicle bus structure and environment, but it will influence the security experiment effect of real vehicles [12]. (4) It is difficult to restore the data correlation between ECUs [13]. For example, in the homemade prototype system, the operating rod is used as the throttle to simulate the driver's acceleration and deceleration, which cannot truly represent the data correlation between the throttle and other ECUs. These problems will affect the performance evaluation of CAN bus intrusion detection methods [10], especially the research on content awareness and aperiodic characteristics of the CAN bus.

In this paper, a CAN bus security testbed based on real vehicle data in the automotive CPS is proposed, and the delay between CAN messages sent from the ECU Operation Center and received by the collector is theoretically explored. In the specific implementation process of the platform, by running Algorithm 1, the time series data of the vehicle is sent from the ECU Operation Center to the ECU, and the six designed attacks are employed to simulate the attack on the CAN bus. By running Algorithm 2, the data is sent to the CAN bus. Finally, the entire bus is generated to run time series data. The stability of the security testbed is examined through the two indicators of delay and packet loss rate. The experimental results indicate that the platform has a small relative delay difference and a low packet loss rate. Furthermore, the similarity of the time series between the real vehicle and the testbed is evaluated by the similarity index. Meanwhile, the distance between the two is calculated by the DTW algorithm, revealing that the testbed possesses high simulation.

Contributions of this paper are as follows:

- (1) A CAN bus security testbed framework derived from real vehicle data is proposed. The testbed designs and implements six common attack models, and completes the sending and collection of CAN messages by running two algorithms, thereby ensuring

the high real-time performance of the platform and the accuracy of the generated data

- (2) In this study, the delay of sending and receiving periodic and aperiodic messages on the platform is theoretically explored. The results show that the generated time series data is mainly related to the timestamp sent. At the same time, the relative delay difference and packet loss rate are employed to explore the stability of the overall and single-type CAN messages of the platform. In addition, the experimental results also verify the ability of the platform in this regard
- (3) The DTW algorithm is used to compare the similarity of the time series between the real vehicle and the security testbed. The experimental results demonstrate that the two sequences have a short distance, confirming the high simulation performance of the platform

The remainder of this paper is organized as follows. The background material about the CAN bus and related work are presented in Section 2. The framework, attack scenario and time series data generation method of the CAN bus security testbed in the automotive CPS are illustrated in Section 3. In Section 4, our experiment environment, evaluation metrics and results are elaborated. Besides, conclusions are drawn in Section 5.

## 2. Background

*2.1. Controller Area Network.* The CAN is a field bus with high reliability, high performance, and low cost [10, 14]. It was originally used in the vehicle electronic control network to realize the exchange of information between vehicle ECUs and was later extended and widely used in the field of industrial control. As an important part of the entire in-vehicle network and the automotive CPS, the CAN bus is a peer-to-peer network [15]. Each node on the CAN bus can either receive messages or actively send messages. When multiple nodes send messages to the bus at the same time, the bus adopts an arbitration mechanism to avoid conflicts. The nodes will read the messages on the bus and compare the bits of the arbitration field with the messages sent by themselves one by one. If the dominant bit is 0, it will continue to obtain control on the bus, while if the invisible bit is 1, the arbitration will be lost, and it will change to the receiving state from the next bit until the bus is free to continue sending messages [16].

*2.2. Related Work.* In the research of automotive CPS security, the known vehicle security testbeds mainly adopt the CAN bus protocol. In 2013, HRL and GM developed a set of security testbeds that highly imitates real vehicles [11]. As the platform employs the same ECU as real products, the simulation accuracy is high. Nevertheless, each vehicle and driving environment is complex, and this platform is not suitable for other models. In 2014, Miller and Valasek customized a portable off-road vehicle to build a security

```

INPUT: listR real time series dataset, listA attack time series data to CAN Shields
OUTPUT: listT messages received and preprocessed by CAN Shields from ECU Operating Center
1: /*ECU Operating Center sends real messages to CAN Shields according to a specified dataset*/
2: function  $S_{END}M_{SG}(list)$ 
3:   list P  $\leftarrow list$ 
4:   i  $\leftarrow 0$ 
5: while i < len(listP) do
6:   t  $\leftarrow$  current time
7:   if t == listP[i].ts then
8:     /*ECU Operating Center sends real messages to CAN Shields */
9:     ui  $\leftarrow$  delay for CAN Shield to process the CAN message
10:    listP[i].ts  $\leftarrow listP$ [i].ts + ui
11:    listT.add (listP[i])
12:    i  $\leftarrow i + 1$ 
13:   end if
14: end while
15:   return listT
16: end function
17: /* If real public dataset is not null,ECU Operating Center starts a new thread to send CAN messages*/
18: if len(listR) > 0 then
19:   Thread t1  $\leftarrow Thread(S_{END}M_{SG}(listR))$ 
20:   t1.start( )
21: end if
22: /* If attack dataset is not null, ECU Operating Center starts a new thread to send CAN messages*/
23: if len(listA) > 0 then
24:   Thread t2  $\leftarrow Thread(S_{END}M_{SG}(listA))$ 
25:   t2.start( )
26: end if
27: return listT

```

ALGORITHM 1: ECU Operating Center sends times series data to CAN Shields.

testbed [9]. Although the price of the platform is not high, it is difficult to upgrade it. Moreover, due to the connection with real vehicles, there are certain security risks when researchers conduct attack experiments. There are also some software which can be used to simulate the vehicle ECU and CAN bus, such as CANoe. However, because of vehicle hardware with the software simulation, many physical properties, including power and current, could not be accurately simulated and evaluated. Especially in terms of the complexity of the vehicle system, software also remains stable. In 2016, Tuohy et al. proposed a hybrid testbed for the simulation of in-vehicle automotive networks. The platform incorporates multiple in-vehicle networks and is used via Ethernet in order to assist in the testing and development of automotive video systems and novel Advanced Driver Assistance System (ADAS) algorithms [12]. Although the testbed retains the in-vehicle network, this method changes the structure and environment of the original in-vehicle network, which will influence the effect of security testing.

At the BLACK HAT EUROPE 2018 conference, the Toyota Information Technology Center team proposed an adaptable portable security testbed PASTA [9] which turns the vehicle into a mini car with real vehicle functions through proportional scaling, using 4 ECUs and 1 console module to conduct the attack test of the operating system, without the need to complete the attack test in the real complex environment, lowering the risk of completing the exper-

iment on the real vehicle. In addition, researchers can flexibly customize their own security technology through this platform, and the ECU has programmable capabilities. Meanwhile, some internal operations of PASTA are realized in the following three ways: (1) Displaying the ECU status on the monitor; (2) moving the model car through physical operation; and (3) the software simulator reads CAN messages and visualizes the behavior of the vehicle to the computer. However, there still exists a certain gap between the platform and the complexity of the real vehicle, and the correlation of CAN data generated between ECUs also lacks authenticity and rationality. Moreover, the device is still immature, causing some inconvenience to users.

### 3. Methodology

In this section, firstly, the framework of the CAN bus security testbed in the automotive CPS is presented. Then, the attack scenarios and the method of time series data generation for the security testbed are introduced.

*3.1. Framework Overview.* Figure 1 shows the proposed CAN bus security testbed framework based on real vehicle data. The platform inputs the time series data of each ECU of the real vehicle into the ECU Operation Center, which simulates the start of the vehicle and sends the corresponding data to the relevant ECUs based on the time series

```

INPUT: list messages processed by CAN Shield, type attack type,  $t_1$  attack start timestamp,  $t_2$  attack end timestamp
OUTPUT: list' messages collected from the CAN bus
1: /*Get the delay caused by CAN bus conflict when a CAN Shield sends messages to the CAN bus*/
2: function GET_CONFLICT_DELAY(msg)
3:    $c \leftarrow 0$ 
4:   flag  $\leftarrow$  has any conflict on CAN bus when sending msg?
5:   if flag == TRUE then
6:     /*Self-delayed because other messages with smaller CANIDs are sent to the CAN bus at the same time */
7:      $c \leftarrow$  delay
8:   end if
9:   return c
10: end function
11: listP  $\leftarrow$  list
12: list'  $\leftarrow$  null / * Store in-vehicle messages collected from CAN Bus */
13:  $i, j \leftarrow 0$ 
14: flag  $\leftarrow$  FALSE
15: while  $i < \text{len}(\text{listP})$  do
16:    $t \leftarrow \text{listP}[i].ts$ 
17:   if current time ==  $t$  then
18:     if  $t < t_1$  and  $t < t_2$  then
19:       if  $\text{listP}[i].source == attack$  and ( $type == masquerade$  or  $type \neq suspend$ ) then
20:         flag  $\leftarrow$  TRUE
21:       else
22:         flag  $\leftarrow$  FALSE
23:       end if
24:     else
25:       flag  $\leftarrow$  TRUE
26:     end if
27:     if flag == TRUE then
28:       /*Update timestamp for the conflict delay*/
29:        $\text{listP}[i].ts \leftarrow$  GET_CONFLICT_DELAY
30:       list'.add( $\text{listP}[i]$ )
31:       /*Consider network delay*/
32:        $d_i \leftarrow$  delay for network transmission
33:        $\text{list}'[j].ts \leftarrow +d_i$ 
34:        $j \leftarrow j + 1$ 
35:     end if
36:      $i \leftarrow i + 1$ 
37:   end if
38: end while
39: return list'

```

ALGORITHM 2: Generate in-vehicle dataset collected from the CAN bus.

relationship. However, the ECU sends the CAN messages to the CAN bus in line with the CAN protocol. Besides, there are many interfaces for data interaction with the outside world on the entire security testbed including OBD-II, Bluetooth, and Wi-Fi, which not only facilitates the data exchange between the vehicle and the outside world, but also increases the risk of the vehicle being attacked. Hackers can connect to telematics devices through wireless communication channel and subsequently invade the CAN bus. Therefore, the platform is designed with multiple attack models to study in-vehicle bus security. In addition, the collector module on the security testbed extracts the running data on the CAN bus to form the testbed dataset of the platform that is adopted for subsequent data analysis, attack detection and attack prediction.

The platform simulates an ECU through an Arduino board loaded with CAN Shield. The ECU has programmable capabilities and can recognize commands sent by the ECU Operation Center and perform related operations, for instance, sending CAN messages to the CAN bus and performing spoofing attacks.

**3.2. Attack Scenarios.** According to the current common vehicle CAN bus attacks [13, 17, 18], the platform implements 6 types of attacks, respectively, fuzzy attack, replay attack, spoofing attack, suspend attack, DoS attack, and masquerade attack. The specific description is presented below.

**3.2.1. Fuzzy Attack.** In order to learn the correspondence between ECU and CAN ID or the meaning of CAN message fields, the attacker randomly injects CAN ID and payload

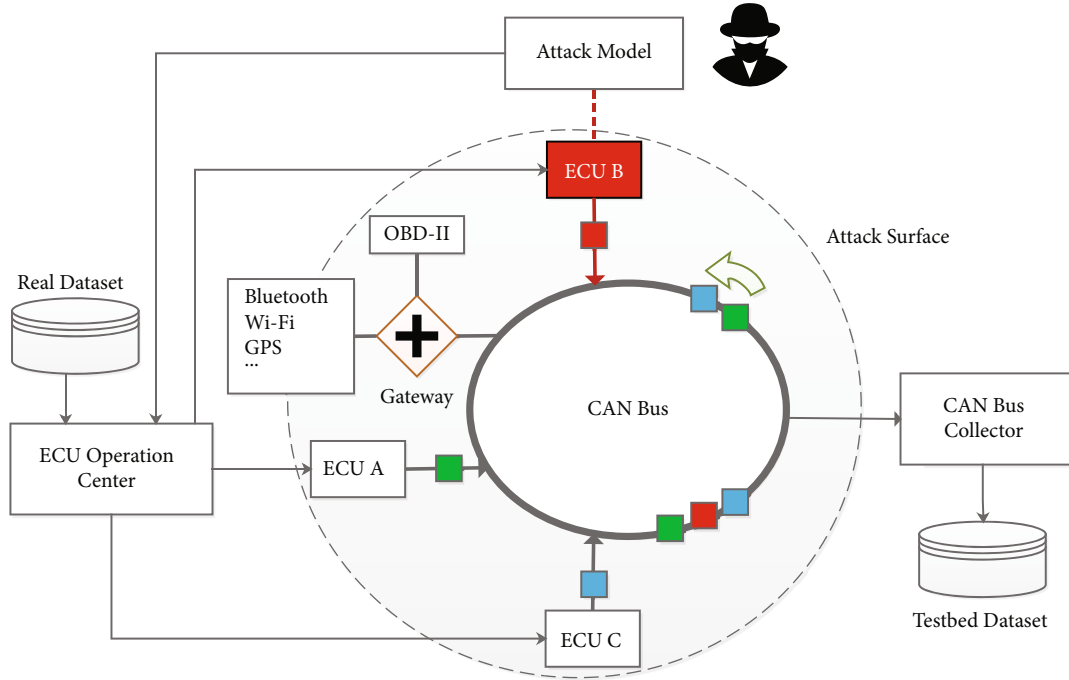


FIGURE 1: The framework of the CAN bus security testbed based on real vehicle datasets in the automotive CPS.

into the CAN bus and understands the CAN bus structure and ECU behavior patterns by observing the changes of the vehicle and ECU. According to the different injection content, fuzzy attack is categorized into fuzzy CAN ID attack and fuzzy payload attack. The fuzzy CAN ID attack is to observe the changes of the vehicle ECU by injecting an unknown CAN ID (generally below  $0 \times 700$ ) externally when the correspondence between the CAN ID and the ECU is uncertain. However, fuzzy payload attack means that the attacker has determined the corresponding relationship between CAN ID and ECU, but does not know the specific field meaning of the payload. Therefore, he or she will familiarize with the function of an ECU by injecting the content of the modified payload of the ECU. In this testbed framework, the hacker simulates a fuzzy attack on the CAN bus by sending control commands and specified data to the ECU Operation Center.

**3.2.2. Relay Attack.** If the attacker is not sure about the function of the CAN ID and the semantics of the payload, they will listen to the CAN bus, capture the data fragment, and then directly inject the CAN data frame into the current time point. In this scenario, both the real ECU and the ECU impersonated by the attacker are sending data frames, while the attacker does not know the internal semantics of the specific data frame. Therefore, the flooding method is often adopted in order to force the CAN bus between the injected data frame and the original data frame. Furthermore, there are ECU data frames, while the former is selected for transmission. For example, the injected data frames are sent to the CAN bus 10-100 times faster than the original CAN data frames [13]. The hacker simulates a replay attack on the CAN bus by sending control commands and specified data to the ECU Operation Center of the testbed.

**3.2.3. Spoof Attack.** If the attacker determines the semantics of the payload, the modified data frame will be injected into the current CAN bus for attacking, which is called fabrication [19]. The attack will show that there will be two ECUs with the same CAN ID on the bus sending normal data frames. For example, the attacker injects 100 km/h vehicle speed data, causing the vehicle speed on the dashboard to change from 40 km/h to 100 km/h. As attackers often align the injection speed with the fake data frame, it is challenging for security detection to identify which is a normal data frame. In this scenario, hackers implement spoofing attacks by sending control commands and specified data to the Operation Center to simulate a spoofed ECU.

**3.2.4. Suspension Attack.** The attacker intrudes into the ECU in some way and suspends its work for a period of time, such as causing the steering wheel ECU to interrupt for 5 seconds to send data frames through malicious code injection. As ECU pauses cause changes in bus traffic and the interruption of such messages, this kind of attack is often easier to implement in intrusion detection, whether by analyzing the time interval of CAN messages [20], information entropy, machine learning [21], or other methods [22]. In this scenario, the attacker temporarily suspends the target ECU by sending control commands and specified data to the Operation Center, thus completing the ECU suspending attack.

**3.2.5. DoS Attack.** The attacker causes the CAN bus crash by flooding the bus with a large number of random data frames in a short amount of time [7]. Since the CAN bus determines the sending of frames through the CAN ID priority arbitration mechanism, the attacker can choose to send the CAN ID with high priority for attacking, aiming to initially check the attack effect. Sometimes, the highest priority data frame



$0 \times 000$  is used to interfere with the sending and receiving of all ECUs on the CAN bus, thereby threatening the driving safety of vehicles. As DoS attacks easily lead to changes in traffic and CAN ID sequences, this kind of attack is often easier to implement in intrusion detection. In this framework, the attacker sends a large number of messages to the CAN bus by sending control commands and specified data in order to achieve the effect of DoS attack.

**3.2.6. Masquerade Attack.** The attacker is already familiar with the function of the CAN ID and the semantics of CAN payload, and he/she can perform various operations (such as suspend) on the relevant ECU in some way [18]. Subsequently, the attacker uses the ECU to send disguised CAN data. For example, the characteristics of the frame (periodic and aperiodic) include sending a well-designed attack data frame at an interval of 500 ms and simultaneously suspending the normal ECU. In addition, masqueraded ECU could also inject data frames at an interval of 500 ms. This kind of precision attack through semantics and camouflage is relatively difficult to detect. It is still very difficult to implement such an attack in a real vehicle. However, in this testbed framework, an attacker can simultaneously suspend a specified ECU and easily start a new ECU by sending a control command to perform a camouflage attack.

In addition to the above attacks, based on the openness and adaptability of the platform, researchers can add more attack scenarios as needed.

### 3.3. The Time Series Data Generation Method

**3.3.1. Delay Analysis.** The time series dataset generated by the security testbed in non-attack and attack environments needs to have low latency. Otherwise, it cannot better simulate the real vehicle environment. As a result, the delay of generating time series data by the security testbed is theoretically explored.

Supposing that there are two ECUs on the CAN bus, represented by  $A$  and  $R$ , respectively, and the ECU Operation Center is denoted by  $E$ ,  $M_i$  as the  $i$ -th CAN message sent by  $A$ , and  $s_i$  as the real timestamp of the message. Then,  $E$  sends the message to  $A$  according to the timestamp  $s_i$ , and  $A$  sends the message to the CAN bus. In this way, the sending scenario of real vehicle CAN messages is simulated.

To calculate the timestamp when  $R$  receives the message sent by  $A$ ,  $u_i$  denotes the delay of  $A$  receiving and processing the message sent from  $E$ , and  $c_i$  represents the delay of  $A$  sending due to bus arbitration failure, while  $d_i$  refers to the network delay of CAN bus transmission messages. Then, the timestamp when  $R$  receives the message is  $T_{rx,i}$ , as expressed below:

$$T_{rx,i} = s_i + u_i + c_i + d_i + n_i, \quad (1)$$

where  $n_i$  is the noise quantized by timestamp  $R$  [18], which is different from the ECU clock used in literature [10]. Since  $E$  distributes data through the  $A$  unified network clock, the clock shift of  $A$  is 0.

As  $\Delta T_{rx,i}$  denotes the timestamp interval between the  $i-1$ -th and  $i$ -th messages received by  $R$ , it is expressed as follows:

$$\Delta T_{rx,i} = \Delta s_i + \Delta u_i + \Delta c_i + \Delta d_i + \Delta n_i, \quad (2)$$

where  $\Delta X$  represents the interval of  $X$  between the  $i-1$ -th and  $i$  messages. Since the data length of each type of CAN messages of  $A$  is fixed,  $E[\Delta d_i] = 0$  [10]. However, when the zero-mean Gaussian noise distribution is satisfied,  $E[\Delta n_i] = 0$  [18]. In addition, as the lengths of each ECU and the ECU Operation Center are basically the same, as well as the control commands and programs programmed inside, the timestamp interval can be ignored when they receive the Operation Center  $E$ , whereas there may exist differences between ECUs due to different characteristics. For different processing times,  $E[\Delta u_i] \neq 0$ . At the same time,  $c_i$  judges the time conflict on the CAN bus based on the timestamp  $s_i$  and the delay  $u_i$ . Although  $E[\Delta u_i] \neq 0$ ,  $\Delta u_i$  is almost negligible compared with  $\Delta s_i$ , indicating that the probability of arbitration collision is very small on the platform. Therefore,  $E[\Delta c_i] = 0$ . Based on the above analysis, the expected value  $\delta_{rx}$  of the timestamp interval between the two messages before and after  $R$  reception can be expressed as follows:

$$\begin{aligned} \delta_{rx} &= E[\Delta T_{rx,i}] = E[\Delta s_i + \Delta u_i + \Delta c_i + \Delta d_i + \Delta n_i] \\ &= E[\Delta s_i] + E[\Delta u_i + \Delta c_i + \Delta d_i + \Delta n_i] \approx E[\Delta s_i] + E[\Delta u_i]. \end{aligned} \quad (3)$$

When  $A$  sends a message with a periodicity of  $T$ ,  $E[\Delta s_i] = T$ . Thus,  $\delta_{rx}$  can be denoted as follows:

$$\delta_{rx} = E[\Delta T_{rx,i}] \approx T + E[\Delta u_i]. \quad (4)$$

Since the delay  $u_i$  of the ECU processing the message itself is very small, basically within 1 ms,  $E[\Delta u_i]$  will become smaller, and the minimum periodic message interval observed from the public dataset is not less than 10 ms. Hence,  $E[\Delta u_i]$  is considered negligible compared to the periodic time  $T$ .

When  $A$  sends an aperiodic message,  $\delta_{rx}$  is equal to Expression (4). Although  $E[\Delta s_i]$  is not a constant, from the public dataset, the interval of aperiodic messages is much larger than 10 ms. Thus, compared with  $E[\Delta s_i]$ ,  $E[\Delta u_i]$  can be neglected.

Therefore, whether the time series data generated by the security testbed proposed in this paper is a periodic or aperiodic message, it is mainly associated with the timestamp  $s_i$  sent by the real dataset. When  $i$  is equal to 1, the timestamp of the first CAN message received by  $R$  is  $s_1 + u_1 + c_1 + d_1 + n_1$ , and the real dataset received is  $s_1$ . This is the difference between the two, that is, the entire time series dataset of the platform is offset backward from the real time series dataset. In fact, the value is so small that it can be almost negligible.

**3.3.2. Time Series Data Generation Algorithm.** In order to effectively lower the delay of CAN messages, the ECU Operation Center employs the multithread parallel transmission

mode to ensure that each ECU CAN sends data to the CAN bus according to the timestamp of the dataset. Algorithm 1 presents the method that the ECU Operation Center sends time series data to the ECU.

When there is only public real data in the dataset, Algorithm 1 is employed to verify the effectiveness of the security testbed and real vehicle experiments in order to ensure that the delay rate and packet loss rate can be kept low. When there is attack data, or when the bus is attacked, the dataset generated by the platform can be applied as an important basis for investigating the security of the CAN bus, especially the changing state of the CAN ID sequence on the bus at the moment of the attack.

In this study, Algorithm 2 is adopted to obtain the time series data on the CAN bus. The ECU can perform processes according to the instructions and data of the Operation Center. When there is an attack instruction, it sends the corresponding attack data. Since arbitration conflict may occur when sending a CAN message, the delay of arbitration failure should be considered in the actual sending time. Furthermore, Algorithm 2 provides the process of the security testbed, finally generating the time series dataset.

## 4. Experiments

In this section, firstly, the experiment setup and the component of the CAN bus security testbed in the automotive CPS are presented. Then, three evaluation criteria of delay, packet loss rate, and similarity are introduced. Finally, the experimental results are explored in detail.

*4.1. Experiment Setup.* Up to now, it is known that there are few public CAN bus attack datasets and most of the public data concerning security research, including normal behaviors and common attack behaviors come from the literature [6, 7]. The real dataset currently studied comes from the literature [6], and it will be used as the basic data support for the security testbed.

The security testbed of this paper is composed of the ECU Operation Center, the CAN bus, the CAN node, the collector, and various connecting lines. Figure 2 shows the prototype of the platform, while Table 1 describes its main components and corresponding specifications. The CAN bus is simulated by a breadboard, and the CAN node consists of an Arduino UNO board and a SeeedStudio CAN Shield to simulate the sending and receiving of CAN messages. The ECU Operation Center is implemented by a computer program, which transmits data and control commands to the CAN node in line with the CAN ID classification and timestamp, while the collector collects the messages on the CAN bus in real time through the Arduino program.

*4.2. Evaluation Metrics.* The performance of the security testbed is evaluated from two dimensions of stability and effectiveness. For the stability of the security testbed, two commonly used network performance evaluation indicators, namely, delay and packet loss rate, are adopted. For the effectiveness of the time series data generated by the plat-

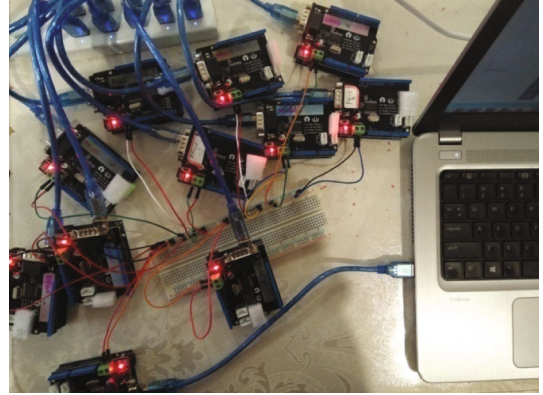


FIGURE 2: CAN bus security testbed prototype.

TABLE 1: Components of the CAN bus security testbed in the automotive CPS.

Component	Specification
ECU operation center	Windows Server 2012 R2
CAN node (ECU)	Arduino Uno (ATmega328)
CAN bus	CAN-BUS shield V1.2
Collector	500Kbps
Actual vehicle	Arduino Uno (ATmega328)
	CAN-BUS shield V1.2
	Toyota Camry 2010

form, the time series similarity is adopted for performing comparative analysis. The specific description is as follows.

*4.2.1. Delay.* Delay is an important indicator that must be considered in the study of platform performance [23]. It refers to the security testbed causing the message to be delayed in time during the process of CAN message transmission, due to various reasons containing sending processing, network transmission, bus arbitration, ECU processing, and receiving processing. Since the real dataset only has the time to receive the CAN message on the bus, and does not know the sending time of the message, this study takes the time interval from the first message to the last message received by the collector within a certain period of time as the delay. In the current work, the delay is divided into the overall delay and the delay of a single type of CAN message.

The overall delay is the time interval between the collector receiving the first message and the last message within a certain period of time. The specific definition is as follows.

*Definition 1.* During time  $T$ , if the timestamp of the collector receiving the first message is  $T_s$  and that of receiving the last message is  $T_e$ , the overall delay  $\Delta T$  can be defined as

$$\Delta T = T_e - T_s, \quad (5)$$

The delay of a single type of CAN message is the time interval from the first to the last message of the type received by the collector within a certain period of time. The specific definition is as follows.

*Definition 2.* During time  $T$ , if the collector receives the first message with message ID  $i$ , the timestamp is  $T_s^i$ , and the timestamp of the last message received is  $T_e^i$ . Next, the delay of ID message  $\Delta T^i$  is defined as

$$\Delta T^i = T_e^i - T_s^i. \quad (6)$$

To better evaluate the performance of the platform, [24] is taken as the reference for using the relative delay difference index. The relative delay difference in this paper refers to the distance between the message delay generated by the platform and the real vehicle message delay in a unit time interval relative to the message delay of the real vehicle. The specific definition is depicted below.

*Definition 3.* If the delay of real vehicle messages is recorded as  $\Delta T_1$  and that of platform messages is considered  $\Delta T_2$ , the relative delay difference  $\Delta \tilde{T}$  is defined as

$$\Delta \tilde{T} = \frac{\Delta T_2 - \Delta T_1}{\Delta T_1}. \quad (7)$$

**4.2.2. Packet Loss Rate.** During the process of sending CAN messages from the ECU Operation Center to the collector obtaining messages from the CAN bus, the message loss is called packet loss [25, 26] due to various reasons including transmission processing, network transmission delay, bus arbitration waiting, ECU processing, and receiving processing. The end-to-end packet loss rate refers to the percentage of the total number of messages lost in the process of transmitting messages from the sender to the receiver within the specified time interval to the total number of sent messages. In this paper, it is called the packet loss rate, and the metric is defined as follows.

*Definition 4.* For the sending and receiving ends of CAN messages, within the time  $T$ , if the total number of messages sent by the sender is recorded as  $N_s$ , and that of messages not received by the receiver is denoted as  $N_f$ , then the packet loss rate  $R$  is defined as

$$PLR = \frac{N_f}{N_s}. \quad (8)$$

When the platform simulates a non-attack scenario,  $T$  in this paper represents the running time of a real dataset, and  $N_s$  denotes the total number of messages in the real dataset. When the platform simulates an attack scenario,  $T$  and  $N_s$  need to be added to the attack time and the number of attack messages, respectively.

**4.2.3. Time Series Similarity.** To evaluate the effectiveness of the time series data generated by the platform, the similarity between the time series data generated by the platform and the real data should be compared. Since packet loss is inevitable in a strong real-time environment, the time series lengths of the two are often different. Therefore, the similarity cannot be well reflected by calculating the Euclidean dis-

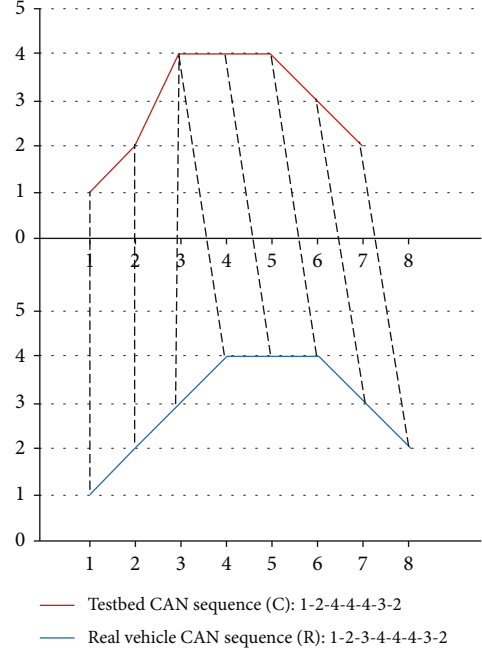


FIGURE 3: DTW similarity calculation example of two time series.

tance. Moreover, the current commonly used method aims to use dynamic time wrapping (DWT) [27, 28] to solve the existing problem. Apart from that, in this paper, the algorithm in [28] is adopted as the time series similarity calculation standard.

Supposing that there are two sequences  $R$  and  $C$ , the former is a certain type of CAN sequence in the real dataset, and the latter is the same type of CAN sequence generated by the platform, represented by  $R_m = \{r_1, r_2, \dots, r_{m-1}, r_m\}$  and  $C_n = \{c_1, c_2, \dots, c_{n-1}, c_n\}$ , respectively, where  $m$  is the sequence length of  $R$  and  $n$  refers to the sequence length of  $C$ . In terms of the steps of the DWT algorithm, it is first to calculate the distance matrix of each element of the two sequences  $R$  and  $C$  and then to find a path with the shortest distance sum from the upper left corner to the lower right corner of the matrix. In addition, it indicates that the shorter the path is, the higher the similarity of the two sequences is while the lower the similarity is.

Through time warping, the point at a certain time of sequence  $C$  corresponds to the point at multiple continuous moments of sequence  $R$  in order to achieve the goal of the minimum distance sum. Figure 3 displays an example of calculating the similarity of two time series  $R$  and  $C$ . By adopting the DTW method, the sum of the shortest distance of the two series can be obtained as 1.

### 4.3. Result Analysis

**4.3.1. Delay Analysis.** In this paper, the security testbed is tested 50 times, and the results are averaged. Since there are many CAN messages on the CAN bus, Table 2 lists the delay of common CAN messages in the platform and real datasets.

As shown in the table, the total relative delay difference of the platform is approximately 0.8%, suggesting that all



TABLE 2: Delay results in the testbed and the real vehicle.

CAN ID	Description	Rate	Real vehicle delay (ms)	Testbed delay (ms)	Relative delay difference (%)
All	All CAN messages	—	74352	74970	0.80
0B0	Speed of wheels 1 and 2	0.01	74352	74509	0.21
0B2	Speed of wheels 3 and 4	0.01	74353	74943	0.80
610	Vehicle speed	0.5	73481	73555	0.10
025	Steering angle	0.01	74352	74929	0.78
224	Brake pedal	0.32	74322	74440	0.18
2C1	Throttle	0.99	74324	74970	0.87
2C4	Engine speed	0.02	74333	74935	0.81
398	Fuel	Aperiodic	73081	73108	0.04
3B4	PRND	Aperiodic	72921	72990	0.09

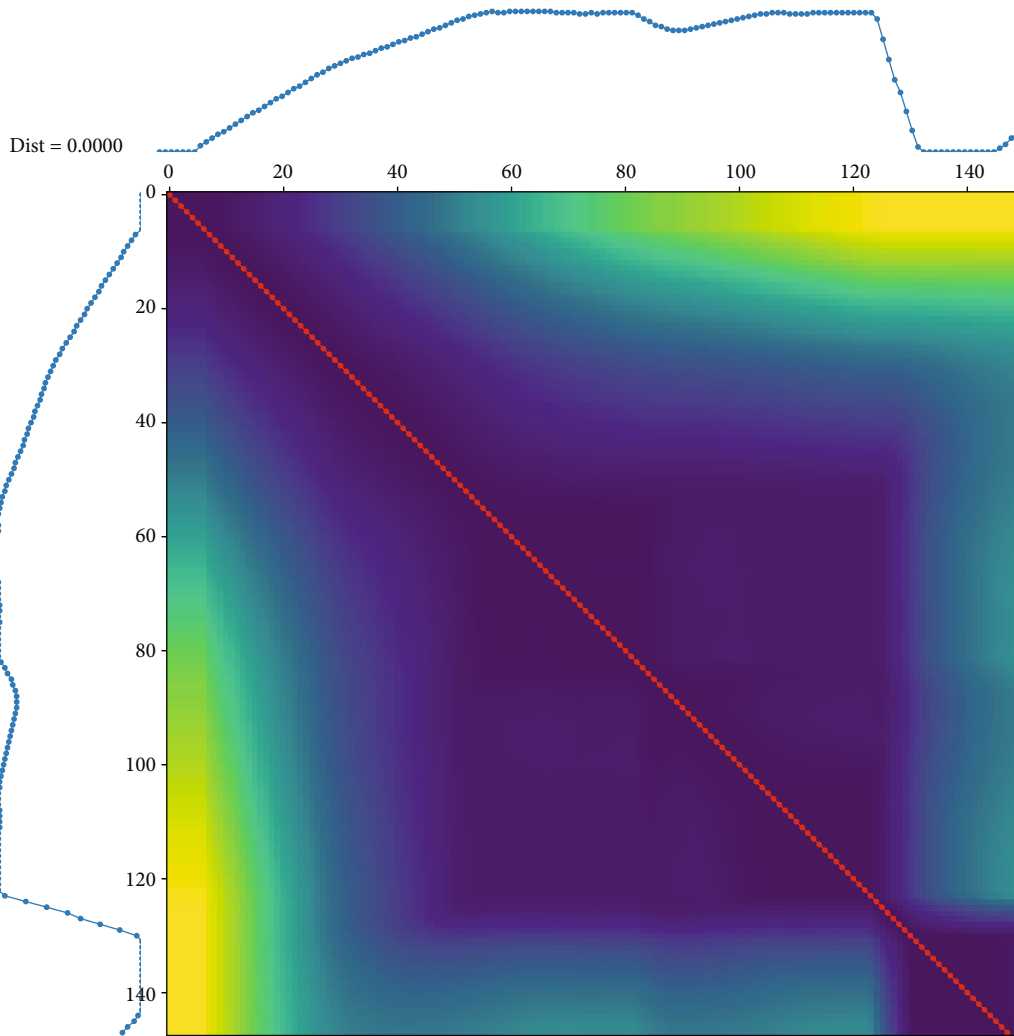


FIGURE 4: DTW similarity distance of CAN ID 610 between the real vehicle and the testbed.

messages are sent 0.8% later than the real vehicle. In addition, it can also be observed from a single type of CAN message that the relative delay difference of periodic and high-frequency messages is generally larger. For example, messages with CAN ID 0B2, 2C4, and 025 have a relative delay difference of periodic and low-frequency messages. In general, for smaller messages including the message with CAN ID 3B4, the relative delay difference of aperiodic messages will be smaller, such as the message with CAN ID 398. How-

ever, it is found that some messages are different, such as CAN ID 0B0 and 610, whose sending frequency is high, and relative delay difference is low. From the experimental process, it can be known that the situation is also related to the message type. For example, 0B2 and 0B0 belong to the wheels of the vehicle and are sent at the same frequency and almost simultaneously. Therefore, the 0B0 message with higher priority may block the sending of the 0B2 message, causing more long delay.

**4.3.2. Similarity Analysis.** In this paper, the similarity of the two time series between the real vehicle and the testbed is calculated according to the above DTW algorithm. In the following, the messages with lower frequency (CAN ID 610) and higher messages (CAN ID 0B0) as representatives are investigated.

Figure 4 shows all regular paths and the shortest paths with CAN ID 610. Obviously, the shortest distance of the two time series is 0, and the path becomes a diagonal line, indicating that the two are exactly the same. From the experimental results, it is also demonstrated that the two sequences are basically the same and the similarity is almost the same. Therefore, the packet loss rate is extremely low.

## 5. Conclusion

With the intelligent, networked, and electronic modern automobiles, the environment of the automotive CPS has become more complex, and the in-vehicle network, especially the CAN, has been threatened. Numerous scholars investigate the security issues of in-vehicle networks through software simulation and real vehicle experiments. In this paper, a CAN bus security testbed based on real vehicle data is proposed in order to help researchers build an open, adaptable, and low-risk infrastructure. To confirm the performance of the security testbed, firstly, the delay of CAN message sending and receiving is theoretically explored, demonstrating that the delay is mainly associated with the timestamp of real vehicle message sending from two aspects of periodicity and aperiodicity. Secondly, Algorithm 1 and Algorithm 2 are designed to complete the sending and receiving of platform messages and realize the simulation of six common attack behaviors. Finally, the security testbed is discussed in detail through relative delay difference, packet loss rate, and similarity. The experimental results demonstrate that the platform is featured by high stability and simulation.

In the next step, we plan to further study the latency of platform messaging, especially to improve the similarity between frequently sent messages and real vehicle time series data. Meanwhile, the attack model will be further enhanced, and the effectiveness of the attack model in this security testbed will be further confirmed.

## Data Availability

The data used to support the findings of the study are available at the Colorado State University (<https://www.engr.colostate.edu/~jdaily/tucrrc/ToyotaCAN.html>).

## Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

## Acknowledgments

This work was supported by the Key Research and Development Program of Zhejiang Province (No.2022C01125) and

grants from “the Fundamental Research Funds for the Provincial Universities”, Zhejiang Institute of Economics and Trade (No. 19SBYB06).

## References

- [1] W. Wu, R. Li, G. Xie et al., “A survey of intrusion detection for in-vehicle networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 919–933, 2020.
- [2] D. Shi, T. Wu, M. Xu, and L. Kou, “Intrusion detecting system based on temporal convolutional network for in-vehicle CAN networks,” *Mobile Information Systems*, vol. 2021, 13 pages, 2021.
- [3] G. Dupont, J. D. Hartog, S. Etalle, and A. Lekidis, “Evaluation framework for network intrusion detection systems for in-vehicle CAN,” in *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVEx)*, Graz, Austria, 2019.
- [4] H. Qin, M. Yan, and H. Ji, “Application of controller area network (CAN) bus anomaly detection based on time series prediction,” *Vehicular Communications*, vol. 27, article 100291, 2021.
- [5] L. B. Othmane, H. Weffers, M. M. Mohamad, and M. Wolf, “A survey of security and privacy in connected vehicles,” in *Wireless Sensor and Mobile Ad-Hoc Networks*, pp. 217–247, Springer, 2015.
- [6] R. Ruth, W. Bartlett, and Y. J. Dail, “Accuracy of event data in the 2010 and 2011 Toyota camry during steady state and braking conditions,” *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, vol. 5, no. 1, pp. 358–372, 2012.
- [7] H. Lee, S. H. Jeong, and H. K. Kim, “OTIDS: a novel intrusion detection system for in-vehicle network by using remote frame,” in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pp. 57–5709, Calgary, AB, Canada, 2017.
- [8] Z. A. Biron, S. Dey, and P. Pisu, “Real-time detection and estimation of denial of service attack in connected vehicle systems,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3893–3902, 2018.
- [9] T. Toyama, T. Yoshida, H. Oguma, and T. Matsumoto, *PASTA: portable automotive security testbed with adaptability*, Black Hat Europe, 2018.
- [10] M. Markovitz and A. Wool, “Field classification, modeling and anomaly detection in unknown CAN bus networks,” *Vehicular Communications*, vol. 9, pp. 43–52, 2017.
- [11] S. Nie, L. Liu, and Y. Du, *Free-fall: hacking tesla from wireless to can bus*, Black Hat USA, 2017.
- [12] S. Tuohy, M. Glavin, E. Jones, C. Hughes, and L. Kilmartin, “Hybrid testbed for simulating in-vehicle automotive networks,” *Simulation Modelling Practice and Theory*, vol. 66, pp. 193–211, 2016.
- [13] K. Fischer, “HACMS,” *ACM SIGAda Ada Letters*, vol. 32, no. 3, pp. 51–52, 2012.
- [14] C. Miller and C. Valasek, *Car hacking: for poories*, SyScan, 2014.
- [15] J. Ning, J. Wang, J. Liu, and N. Karto, “Attacker identification and intrusion detection for in-vehicle networks,” *IEEE Communications Letters*, vol. 23, no. 11, pp. 1927–1930, 2019.
- [16] Q. Luo and J. Liu, “Wireless telematics systems in emerging intelligent and connected vehicles: threats and solutions,” *IEEE Wireless Communications*, vol. 25, no. 6, pp. 113–119, 2018.

- [17] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 5–9, Baden-Baden, Germany, 2011.
- [18] K. T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *Proc. of the 25th USENIX Security Symposium*, pp. 911–927, Austin, TX, 2016.
- [19] K. Iehira, H. Inoue, and K. Ishida, "Spoofing attack using bus-off attacks against a specific ECU of the CAN bus," in *15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pp. 1–4, Las Vegas, NV, USA, 2018.
- [20] 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 *2016 International Conference on Information Networking (ICOIN)*, pp. 63–68, Kota Kinabalu, Malaysia, 2016.
- [21] H. M. Song, J. Y. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, article 100198, 2020.
- [22] S. Zander and S. J. Murdoch, "An improved clock-skew measurement technique for revealing hidden services," in *Proceedings of the 17th conference on Security symposium*, pp. 211–225, Berkeley, CA, USA, 2008.
- [23] N. Wang, *Research on Stability Analysis and Control Method of Networked Control System with Time-Delay*, Shandong University of Technology, Jinan, China, 2021.
- [24] K. Wu, *Research on Fast Algorithm and Link Delay Difference in Radiation Two-Step Method*, Hunan University, Changsha, China, 2020.
- [25] T. Mi, *The Research on Packet Loss Rate Based on the Measured Data*, Southeast University, Nanjing, 2015.
- [26] H. Lan, *Research on Key Technology for Packet Loss Estimation Based on Passive Measurement*, Southeast University, Nanjing, China, 2020.
- [27] J. Serra and J. L. Arcos, "An empirical evaluation of similarity measures for time series classification," *Knowledge-Based Systems*, vol. 67, pp. 305–314, 2014.
- [28] D. F. Silva and G. E. Batista, "Speeding up all-pairwise dynamic time warping matrix calculation," in *Proceedings of the 2016 SIAM International Conference on Data Mining*, pp. 837–845, Miami, Florida, USA, 2016.