IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 25, 2018, accepted September 8, 2018, date of publication September 17, 2018, date of current version October 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2870695

IDH-CAN: A Hardware-Based ID Hopping CAN Mechanism With Enhanced Security for Automotive Real-Time Applications

WUFEI WU[1], (Student Member, IEEE), RYO KURACHI[2], (Member, IEEE), GANG ZENG[2,3], (Member, IEEE), YUTAKA MATSUBARA[2], HIROAKI TAKADA[2], (Member, IEEE), RENFA LI[1], (Senior Member, IEEE), AND KEQIN LI[1,4], (Fellow, IEEE)

[1]College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
[2]Graduate School of Informatics, Nagoya University, Nagoya 464-8601, Japan
[3]Graduate School of Engineering, Nagoya University, Nagoya 464-8601, Japan
[4]Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

Corresponding author: Renfa Li (lirenfa@hnu.edu.cn)

This work was supported in part by the China Scholarship Council under Grant 201606130063, in part by the National Key Research and Development Plan of China under Grant 2016YFB0200405, and in part by the National Natural Science Foundation of China under Grant 61672217, Grant 61702172, and Grant 61502162.

ABSTRACT Cybersecurity is increasingly important for the safety and reliability of autonomous vehicles. The controller area network (CAN) is the most widely used in-vehicle network for automotive safety-critical applications. Enhancing the cybersecurity ability of CAN while considering the real-time, schedulability, and cost constraints becomes an urgent issue. To address this problem, a real-time, and schedulability analysis-guaranteed security mechanism [identification hopping CAN (IDH-CAN)] is proposed in this paper, which aims to improve the security performance of CAN under the constraints of automotive real-time applications. In order to support the operation of the IDH-CAN mechanism, an IDH-CAN controller is also designed and implemented on a field-programmable gate array, which can work as a hardware firewall in the data link layer of CAN to isolate cyberattacks from the physical layer. Meanwhile, to maximize the information entropy of the CAN message ID on the physical layer, the ID hopping table generation and optimization algorithms for IDH-CAN are also proposed. Then, information security evaluation experiments based on information entropy comparison are deployed. The simulation and practical evaluations demonstrate the effectiveness of the proposed mechanism in defending reverse engineering, targeted denial of service, and replay attacks without violating real-time and schedulability constraints.

INDEX TERMS Controller area network (CAN), cybersecurity, ID hopping, information entropy, in-vehicle network, real-time, schedulability analysis.

I. INTRODUCTION

Currently, a paradigm in the automotive industry has shifted from high-performance vehicles to comfortable, intelligent, safe and secure vehicles. However, cybersecurity is one of the most critical issues in autonomous driving technology [1], [2]. For the realization of autonomous driving, an increasing number of automotive application components are employed. Such a complex system is increasingly dependent on internal and external networks. State-of-the-art, on-board architectures of high-end automobiles can include more than 100 Electronic control units (ECUs) that are interconnected via heterogeneous communication networks [3], such as Controller Area Network (CAN), LIN, FlexRay, and automotive Ethernet. As in-vehicle networks are no longer isolated, autonomous vehicles face new vulnerabilities and attacks with the expansion of the external communication interface of automobiles. In the era of the upcoming autonomous driving vehicle, new security issues will surely arise. These potential attacks are discussed in [4]–[7].

The security attacks against CAN have attracted increasing research attention [4], [5], [8]. Unfortunately, the original CAN protocol does not consider network security issues, so it cannot provide security mechanisms such as message

authentication or data encryption [8]. Meanwhile, ECUs connected via the CAN bus are usually safety-critical functional systems (i.e., anti-lock braking system and electronic stability program), which need to guarantee strict end-to-end latencies, and also need to comply with the corresponding functional safety standards, such as functional safety-related Road Vehicles-Functional Safety standard ISO 26262 [9]. According to the ISO 26262 specification, CAN clusters in vehicles have high automotive safety integrity level (ASIL) requirements, which means that attacks on the CAN bus will not only lead to the disclosure of privacy, but also result in threats to the life and property of users and surrounding people in extreme circumstances.

In this study, a novel ID hopping CAN (IDH-CAN) mechanism is proposed to address the requirements of security and safety. Meanwhile, a hardware-based ID hopping CAN controller (IDHCC) is proposed, and several experiments are carried out to test the security performance of the proposed IDH-CAN. The main contributions of this study are as follows.

1) We propose a CAN security enhancement mechanism based on ID hopping (IDH-CAN) in this study. To the best of our knowledge, this study is the first work to consider the schedulability analysis of safety-critical systems when designing cybersecurity enhancements for CAN.
2) We propose an intellectual property (IP) core of the hardware-based IDHCC for IDH-CAN, which implements on a field programmable gate array (FPGA; Altera EP4CE22F17C6). Therefore, the security enhancement describes in this study is designed from the architectural perspective for in-vehicle network.
3) We propose two algorithms (generation and optimization) for the ID hopping table of IDH-CAN mechanism. The two algorithms aim to maximize the entropy of the arbitration IDs, which will be transmitted at the physical layer of IDH-CAN and will be used in the design deployment phase of the IDH-CAN mechanism.
4) The security performance of the proposed mechanism is analyzed from the perspective of information entropy. Experiments are conducted on a real board to verify the correctness of the security-enhanced CAN controller.

The rest of the paper is organized as follows. Section II introduces the background of this study. Section III describes the attack model of our method and the constraints of the security enhancement design for the CAN. Section IV describes our solution, which includes the design of the IDH-CAN mechanism, the startup and reply mechanism, and user guides, etc. Section V describes the ID hopping table generation and optimization algorithms for IDH-CAN. Section VI presents the hardware implementation of the IDH-CAN controller. Section VII evaluates the IDH-CAN in terms of resource consumption and security performance. Section VIII introduces the related works. Section IX concludes the paper.

II. BACKGROUND

A. AUTONOMOUS VEHICLE ENVIRONMENT

In the near future, autonomous vehicles will play numerous important roles in building new relationships among people, vehicles, and our city environment. According to the SAE International's J3016 standard [10], unmanned driving can be appropriately ranked in six levels (i.e., $0, 1, \ldots, 5$). An autonomous vehicle is always connected to an external network during operation or while parked [11]. Based on [2] and [11], Figure I depicts an overview of an autonomous vehicle environment from the perspective of networks.

- An array of sensors, such as LIDAR, RADAR, cameras, and GPS, are equipped on vehicle to collect information. These sensors provide autonomous vehicles with the ability to perceive the environment and make driving decisions without human interventions [2].
- A series of heterogeneous in-vehicle networks (e.g., CAN, Flex-Ray, LIN, and Ethernet) connect ECUs with distributed computing resources (CPU, GPU, and FPGA).
- A communication link connects the vehicle, other vehicles, base station, user's intelligent terminal, and so on. There are many ways to connect to external networks (e.g., 4G / 5G, Bluetooth, and Wi-Fi).
- A smart server center can provide various services for vehicles, such as the autonomous cruise control system. Server Center relies on modern ICT technologies such as machine learning, big data and cloud computing.

Current automotive electronic components are supplied by different manufacturers in the supply chain, which poses a challenge to system security deployment. With this in mind, we design IDHCC as an IP core for semiconductor fabrication chains, which works on the data link of the in-vehicle network structure, as indicated by the red circle in Figure I.

B. CAN OVERVIEW

The CAN protocol was invented by Robert Bosch GmbH in 1986 to meet the specific requirements of a automotive real-time applications, such as real-time message transmission, reliable operation in electromagnetic compatibility (EMC)/electromagnetic interference (EMI) environment, and cost effectiveness.

The CAN data field can transmit up to 8 bytes of data, and the ID field is 11 bits (29 bits in extended mode). At the data link layer, the CAN protocol uses broadcast communication to transmit messages, thereby all nodes on the bus can receive the messages simultaneously. The arbitration mechanism of CAN is that the message with high priority can continue to transmit data without affecting the bus collision decision time, while the low priority message to wait for the next idle state. The basic data frame structure of CAN 2.0 and a three-node arbitration instance are depicted in Figure 2, where the blue line represents the waveform transmitted on the physics bus. As described in [12], in the case of denial-of-service

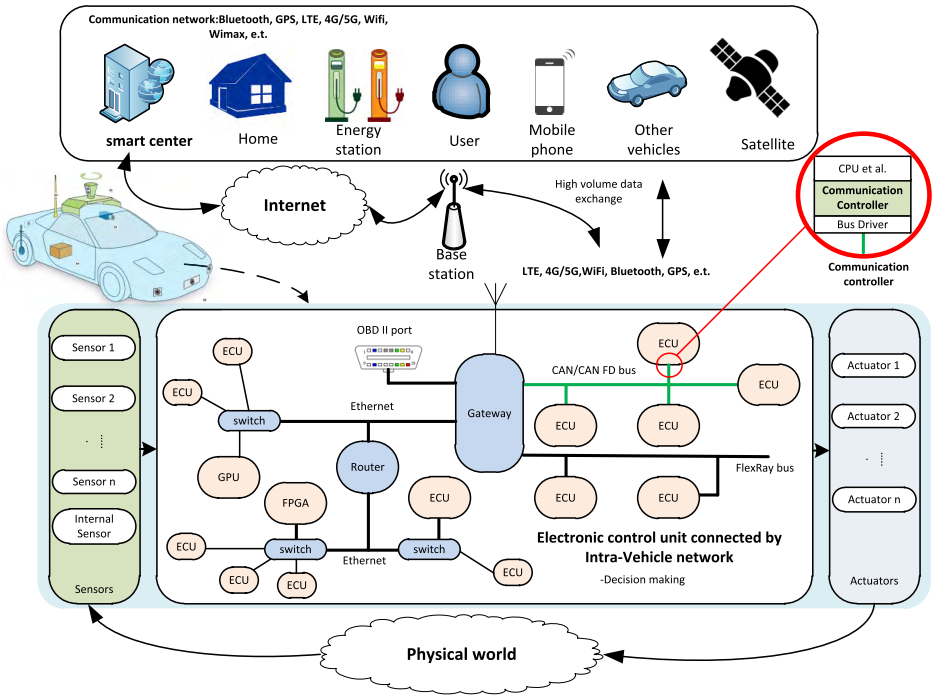


FIGURE 1. An overview of autonomous vehicles environment from the perspective of network.

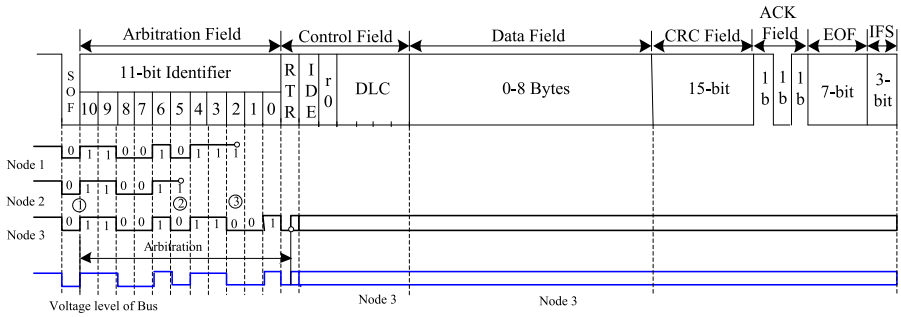


FIGURE 2. Standard CAN message frame and example of CAN protocol arbitration handling.

(DoS) attacks, this priority-based arbitration mechanism may cause the CAN network to be bus off.

III. ATTACK MODEL AND CONSTRAINTS

A. ATTACK MODEL

In our attack model, we assume that the attacker can access the CAN bus. The available access methods include, but are not limited to, Bluetooth, OBD_II, Wi-Fi, physical access, and USB ports. Since there is no message authentication and encryption in the CAN protocol, an attacker can easily reverse the CAN message. Once an attacker gains access to the CAN, the attacker can perform a series of attack actions such as sniffing, spoofing, replay and DoS attacks. To better describe our attack model, we build our attack model as shown in Figure 3. In our attack model, once a CAN network is illegally accessed by an attacker, the attacker can perform two types of attacks. The first type is that the attacker reads network logs and messages and then rewrites the ECU firmware to perform specific vehicle control operations. The second type is that attacker sends illegal messages directly over the network to perform attacks such as DoS and replay attacks. It should be noted that high-priority DoS attacks against CAN bus are the types of attacks we cannot currently solve in this study.

The state-of-the-art security enhancement researches of CAN focus on protecting message payloads, but our threat model focuses more on improving the security of ID fields. A CAN security enhancement mechanism based on ID hopping (IDH-CAN) is proposed in this study. The motivation of this study is to provide the countermeasures for sniffing and reverse engineering of CAN messages by integrating the ID masquerade techniques, which means that our method not only increases the difficulty of reverse engineering, but also

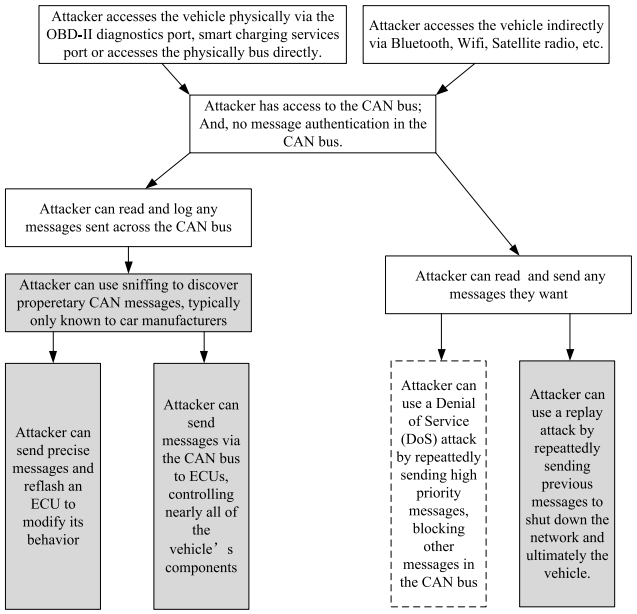


FIGURE 3. The attack model targeted by the our proposed method [13].

can resist most attacks against CAN, except for high-priority DoS attacks, as shown in Figure 3.

B. CONSTRAINTS OF CAN SECURITY ENHANCEMENT DESIGN FOR AUTOMOTIVE REAL-TIME APPLICATIONS

The original CAN was designed for isolated in-vehicle network, without taking into account cybersecurity issues. During the cybersecurity enhancement design for automotive real-time applications, various resource metrics need to be taken into account. Such as computation, storage, and communication bandwidth, etc. Furthermore, other important metrics such as reliability and schedulability also need to be considered during system design. Researchers have done a lot of research work to separating the attack source from the network layer to ensure autonomous vehicle secure. Considering the sate-of-the-art studies do not consider the schedulability of safety-critical systems. We identify the requirements to provide a secure CAN for vehicles as follows.

1) DATA FIELD

The traditional security encryption methods such as Hash Message Authentication Code (HMAC) consumes the limited data field of CAN. Considering the fact that a CAN frame can only can carry up to 8 bytes of data, the security mechanism design cannot ignore the size constraint of data field. Moreover, minimizing load data can help reduce transmission time and increase bandwidth utilization.

2) ID FIELD

In the CAN bus, each message is assigned a unique arbitration ID. The ID field in the basic CAN frame consists of 11 bits (29 bits for expanded frames). Moreover, in the automotive system, most of the IDs have already been used in the existing design. Thus, achieving message authentication and encryption by the modifications of ID field is considerably difficult.

3) REAL-TIME

ECUs connected via CAN are usually safety-critical components. According to the time requirement of functional safety, the communication between them needs to guarantee strict end-to-end latencies. However, most of the existing security enhancement methods will consume additional time (due to computation and communication overhead), which reduce the real-time performance of CAN messages [14]–[17].

4) SCHEDULABILITY ANALYSIS

System is schedulable means that all worst-case response time (WCRT) of messages can meet their deadlines, which is a critical safety issue. Automobile electronic systems are usually safety-critical system. Therefore, schedulability analysis is a critical step for designing the automotive real-time applications [18]. Nevertheless, uncertain delay will lead to a high difficultly in the scheduling of the system, especially in the autonomous vehicle environment, where in-vehicle networks share a large amount of safety-critical data between sensors and actuators.

IV. PROPOSED MECHANISM

In this section, we present a real-time guaranteed security mechanism for automotive real-time applications. The mechanism is based on anonymous ID technology to enable security enhancements in the in-vehicle network environment with low computational and communication overhead. To simplify the complexity of the problem, we make the following assumptions which are reasonable in practical applications [2], [19], [20]. 1) Each CAN message sender and receiver task has been assigned to a specific ECU in the early stages of the functional design of the automotive electronics system, and 2) the allocation of the IDs for the CAN messages have been completed from the application layer.

A. ID HOPPING CAN

The diagram of the proposed ID hopping CAN mechanism is depicted in Figure 4. An ECU in the IDH-CAN scenario contains two kinds of available IDs, where *App_IDs* represents the ID of CAN message in application software, and *Phy_IDs* represents the physical layer ID, which can be obtained by looking up the *ID_hopping_table*. And *ID_hopping_table* (with maximum 16 pages *Phy_IDs*) is generated in the design phase according to *App_IDs*, which will be written to the IDH-CAN controller (IDHCC) during the system deployment phase. The IDH-CAN implements constant switching and mapping between the physical layer transport ID and the application layer ID according to the message counter and *ID_hopping_table*. Through such isolation, the CAN network can be effectively improved against attacks from the physical layer, such as reverse engineering and replay attacks.

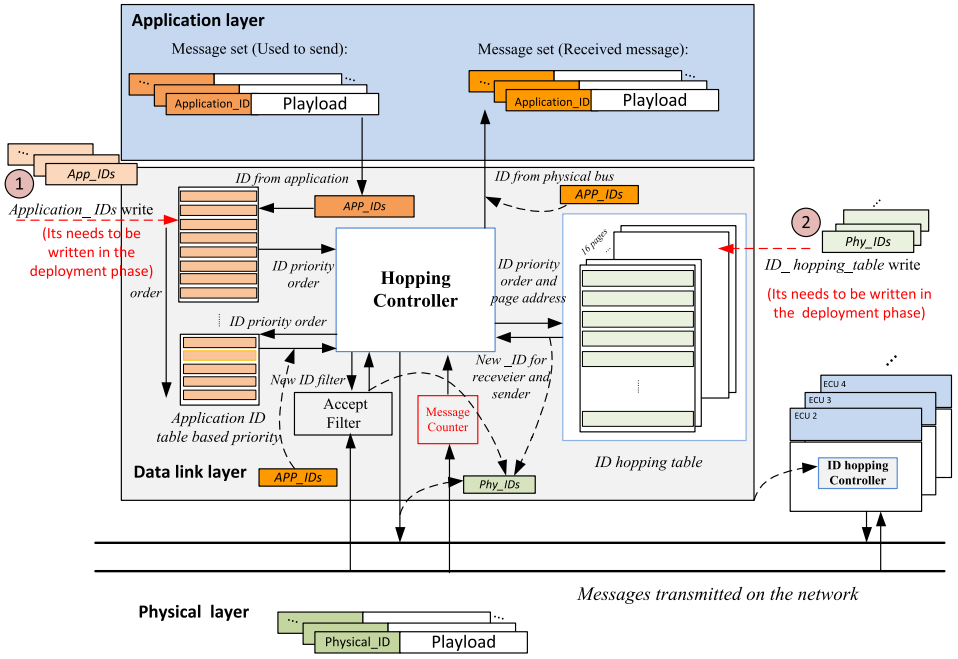


FIGURE 4. Diagram of the proposed ID hopping CAN mechanism.

1) FOR SINGLE CAN CLUSTER

Due to CAN's broadcast nature, the value of message counter is shared by all trusted nodes in single CAN cluster. Hopping action of IDH-CAN is triggered by message counter (for example, when the message counter is full, then a hopping occurs). Simultaneously, the relative priority order of messages can be determined. Based on the system's schedulability analysis model, the system can perform schedulability analysis. The ID hopping table and filter parameters of each node can be generated in the global ID range ($0x000 - 0x7FF$). For upgrading existing or designing new automotive application design, once relative priority order of the messages *App_IDs* is available for designer, then, the actually transmitted ID of message on CAN *Phy_IDs* will be more diversity under IDH-CAN mechanism.

As shown in Figure 4, once ID hopping occurs, physical message obtains another *Phy_IDs* by looking up the *ID_hopping_table*, but the priority order of the physical message and *Phy_IDs* is guaranteed be the same with that *App_IDs* (application message ID). Meanwhile, receiving ECUs use the new ID allocation and new filter registers to filter incoming data frames before verifying data integrity, which can reduce the time consumption due to deployment of security mechanisms.

2) FOR GATEWAY ENVIRONMENT

The message counters in different CAN clusters are usually different during the running of the automotive applications. However, the interconnection of multiple CAN clusters through the gateway is a very common phenomenon in the automotive electronics environment. For messages that need to be transmitted through the gateway, the available range of IDs and hopping mechanism in different CAN clusters are also different.

We use an example to illustrate how this differentiated hopping mechanism implements the transmission of messages across gateways in different CAN clusters. For example, if the message located in cluster 1 needs to be transmitted across the gateway to cluster 2. First of all, the *Phy_IDs*_1 located in cluster 1 maps to its *App_IDs*_1 in gateway node, and then *App_IDs*_1 maps to *App_IDs*_2. Then, according to the target cluster 2's hopping mechanism and ID hopping tables, the *App_IDs*_2 changed to *Phy_IDs*_2. Then, the message is transmitted. Namely, the IDs of cross-gateway messages need to be mapped twice: cluster 1 (*Phy_IDs*_1)→ gateway(*App_IDs*_1 → *App_IDs*_2) → cluster 2 (*Phy_IDs*_2), which means that message counters for ID hopping do not need to be synchronized between different CAN clusters.

A schematic of transmission conversion process of message IDs under ID hopping mechanism is depicted in Figure 5. In the ID hopping mechanism scenario, the ID hopping tables are stored in IDH-CAN controllers of transmitter node and receiver node respectively, the size of the table equal to *Phy_IDs* × *pages_number* (e.g., 4 × 11 in this example). *ID_hopping_table* is designed and generated during the design phase of the system. During the operation of the system, one page of *ID_hopping_table* is chosen as *Phy_IDs* in IDH-CAN controllers according to message counter, the priority order of *Phy_IDs* is guaranteed to fixed according to the order of *App_IDs*. For the sending nodes,

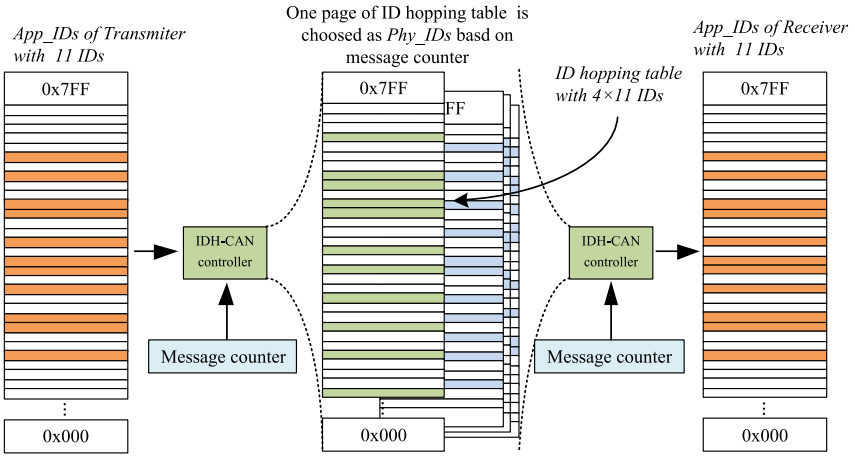


FIGURE 5. Schematic of transmission conversion process of message IDs under ID hopping mechanism.

IDHCC maps the *App_IDs* to the *Phy_IDs* according to the priority order of the message. Therefore, the actually transmitted message IDs on the physical layer will become more diverse than ordinary CAN messages without ID hopping. For receiving nodes, IDHCC maps the *Phy_IDs* to the *App_IDs* by looking up the *ID_hopping_table* based on the priority order of the message.

Meanwhile, the new filter registers of the receiving ECUs can be generated according to the current page of *ID_hopping_table*, which can be used to filter the incoming CAN messages before verifying data integrity. Invalid CAN messages are filtered without requiring any additional computation overhead, because the proposed ID hopping techniques are implemented in CAN controller by hardware.

According to the broadcast characteristics of the CAN, we can know that the messages transmitted on the CAN are shared for all nodes on the bus. Therefore, in this study, the arbitration ID hopping action for all ECUs in the system is synchronized through the message counter. For example, when the message counter in CAN controller detects that 8 messages are transmitted on the network, all nodes on the CAN bus will have an ID hopping action. In each *ID_hopping_table* page, the relative priority of all system IDs is constant. Therefore, once all messages are prioritized for the authentication node, the messages sent by the non-authenticated node will be rejected.

B. SCHEDULABILITY ANALYSIS

In automotive real-time applications, CAN messages usually have hard deadline constraints, which can be presented by D_m. Tasks on the receiving node may have multiple time requirements for the messages, but in a hard real-time environment such as an automotive real-time application, D_m is the strictest time constraint. Therefore, D_m must be satisfied in the security enhancement solution. A message is said to be schedulable if and only if its worst-case response time is shorter than or equal to its deadline ($R_m \leq D_m$). If a system is schedulable, it means that all messages in the system are schedulable.

Davis *et al.* [18] introduced that the message response time R_m is composed of three terms, namely the release jitter J_m, the queue delay $\mathcal{W}_m$, and the maximum transmission time C_m:

$$R_m = J_m + \mathcal{W}_m + C_m. \tag{1}$$

C_m represents the maximum transmission time, which is determined by the message payload s_m (1 to 8 bytes), the ID format (11 or 29 bit), and one bit transmission time. $\mathcal{W}_m$ represents the queue delay, which is determined by two factors, namely the blocking factor B_m due to non-preemptive message transmission and the interference due to higher priority messages (denoted by the set $hp(m)$), $\mathcal{W}_m$ can be obtained by:

$$\mathcal{W}_m^{n+1} = \max(B_m, C_m) + \sum_{\forall k \in hp(m)} \left\lceil \frac{\mathcal{W}_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k. \tag{2}$$

B_m represents the simple upper bound on the blocking factor, which is given by the transmission time of the longest message on the network:

$$B_m = \max_{\forall k \in lp(m)} \{C_k\}. \tag{3}$$

According to the Equations (2) and (3), the time available for ID hopping in a CAN bus system can be calculated as:

$$C_m = \left(g + 8S_m + 13 + \left\lfloor \frac{g + 8S_m - 1}{4} \right\rfloor\right) \tau_{bit}, \tag{4}$$

where S_m represents the data field of CAN that is assumed to be 8 bytes. g (i.e., the worst-case bit-stuffing length) is equal to 34 for standard frame format or 54 for extended format (29-bit ID field), and τ_{bit} is the bus bit rate. For 11-bit identifiers at 1 Mbps bit rate, C_m is equal to 0.136ms. Given that the time available for ID hopping is limited, we choose the hardware implementation for the ID hopping mechanism.

The schedulability analysis described in [18] has been commonly adopted by researchers and transferred to

industry in the form of commercial CAN shedulability analysis tools. These tools have been used in the in-vehicle network development phase of a large number of automobile manufacturers. However, to the best of our knowledge, the sate-of-the-art in-vehicle network cybersecurity enhancement studies do not consider the schedulability analysis of safety-critical systems. For example, given the $\mathcal{W}_m$ in CAN+ [17] and the fact that Identify-Anonymous CAN (IA CAN) [21] has been changed, their schedulability analysis model need to be changed inevitably. Therefore, one of our motivations is to design a cybersecurity enhanced CAN mechanism (IDH-CAN) with guaranteed schedulability of safety-critical system.

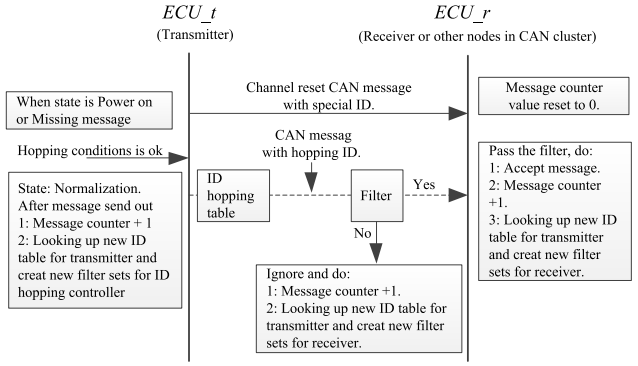


FIGURE 6. Process diagram of IDH-CAN.

C. OPERATION OF IDH-CAN

This section describes how to deploy the IDH-CAN to the automotive electronic system environment during the development phase of an automotive real-time application. For ease of understanding, we present a process diagram of IDH-CAN in Figure 6. The deployment of IDH-CAN can be divided into the following five steps:

- *Step 1:* Determine the number of messages N and their priority order $P\{p_1, p_2, \ldots, p_n\}$ according to the functional requirements of the system. The number of messages used in single CAN cluster is not large and is usually within 256 or fewer [22].
- *Step 2:* Assign an ID to each message-frame as a priority parameter. A total of 2048 possible message IDs can be placed in one CAN cluster. According to C_{2048}^{N}, the available ID combinations for *Phy_IDs* are very large.
- *Step 3:* Determine the acceptable message set named *RX_set* for each ECU at each ID page moment according to the relationship between ECUs and the messages obtained from Step 2. The filter registers of the ECU at each ID page moment can be obtained on the basis of *RX_set*. The new ID hopping page for the next ID page moment can be obtained by adding the filter registers to the ID hopping pages.
- *Step 4:* Write the *ID_hopping_table* and the *App_IDs* to the ID hopping controller in the design or upgrade phase of the automotive electronic system.
- *Step 5:* Startup or restart the system.

D. ID HOPPING SYNCHRONIZATION

ID hopping synchronization is especially important for IDH-CAN mechanism design. In this study, the message counter is used as ID hopping synchronization parameter (select one page from the *ID_hopping_table* as a new one), where the *ACK* signal in the data link layer is used to count the message counter. The ID hopping is performed on a cycle of 8. It is means that after 8 messages have transmitted over the CAN bus, each IDHCC can get this shared parameter (message counter), and ID hopping occurs on each node on the bus synchronously at this moment. Due to the relative priority order of the message is constant in each *ID_hopping_table* page, the priority order of the messages in the CAN cluster will be constant in each moment, and can be used as the parameter of the message mapping. Therefore, the schedulable analysis of the automotive real-time application system does not need to be redesigned, which is very important for the automotive real-time applications.

As depicted in Figure 7, from the perspective of the message, the ID of the message at each moment will correspond to a new ID based on the *ID_hopping_table* and priority order (according to the lookup *App_IDs* results) of the message. However, from the viewpoint of the ECUs, each combination of IDs can be seen as one ID page in one moment. The *ID_hopping_table*, as a 3D array, can be written to IDHCC and stored in secure memory space. Taking into account the limited register resources and cost factors, the table size in this study is divided to 4, 8 and 16 pages with 256 IDs per page.

E. STARTUP AND RECOVERY MECHANISM

Taking into account the requirements of robustness of automotive electronic systems, some special situations need to be considered in the IDH-CAN design. The ID hopping synchronization described in this study is based on the message counter of IDH-CAN controllers. If the message count on the node has a missing message, it will cause the node's ID hopping synchronization with the entire system to fail (i.e., missing messages). However, due to the good robustness and error recovery mechanism of CAN, the above-mentioned message loss situation is rarely seen in the CAN environment. Nevertheless, a robust recovery mechanism in automotive real-time applications for IDH-CAN is necessary, especially for safety-critical systems in autonomous vehicle.

1) STARTUP

In order to start the system, a startup message with special ID (such as $0x01$) is used in IDH-CAN. Once a network idle is detected by one node, and the message counter becomes equal to the threshold of hopping (e.g., 8 in this study), the node sends a startup message. If the other nodes receive the message, they will recover the ID hopping action, realize the reset, and start the system.

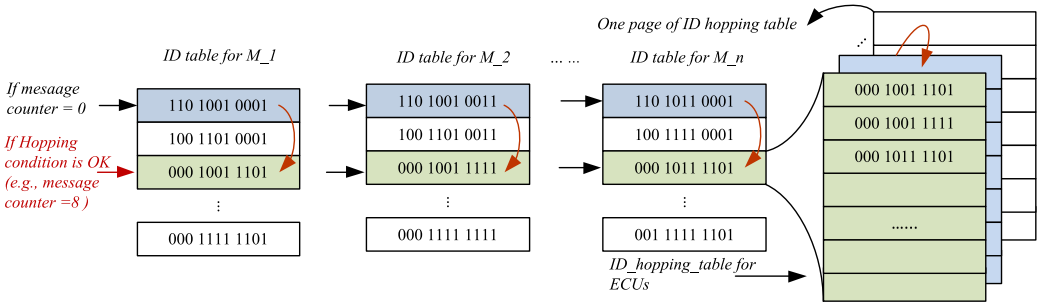


FIGURE 7. ID hopping synchronization can be observed from the perspective of messages and ECUs, respectively.

2) RECOVERY

Message counters between different ECUs may be offset (this condition causes synchronization failure), which will cause the ECUs to fail to send and receive messages. This type of error requires the reliable design of the upper application. To solve this problem, the IDH-CAN controller generates an interrupt to notify the upper software of synchronization errors. Once the application layer software receives the interrupt. Resetting the reset bit of the IDH-CAN controller will restart the ID hopping action. More details can be find in Section VI.

V. ID HOPPING TABLE DESIGN AND OPTIMIZATION ALGORITHMS

The *ID_hopping_table* is key part of the IDH-CAN mechanism. To obtain numerous non-formatting and diversified IDs combinations quickly, we design a heuristic algorithm to generate the *ID_hopping_table*. We also design an optimization algorithm based on simulated annealing to maximize the entropy of the ID transmitted in the physical layer of the CAN bus.

A. GENERATION OF ID HOPPING TABLE

Although 2048 possible message IDs actually exist in the CAN protocol, CAN implementations generally have message IDs far fewer than 256 in a CAN cluster. An enormous amount of combinations can be used for *ID_hopping_table*, and the *App_IDs* table is just one of them. Therefore, an efficient generation algorithm to quickly obtain these combinations is necessary.

Picking out k elements from n elements can construct one page of the *ID_hopping_table*. In the case of a guaranteed and fixed message priority order, the number of combinable ID assignments is related to message size of system. For instance, if there are 200 messages in the sub-network, the global allocation combination in sub-network is C_{2048}^{200}. Numerous possibilities exit in making *Phy_IDs* with several non-formatting and diversity combinations for the physical layer. Figure 7 depicts that message transmissions on the CAN bus have different *Phy_IDs* at different moments. Once the hopping occurs, ECU can acquire new *Phy_IDs* by looking up the *ID_hopping_table* according to the *App_IDs* of the message and the current value of the message counter.

Number of combinations can be defined as:

$$C_k^n = \binom{k}{n} = \frac{n!}{k!(n-k)!}. \tag{5}$$

Algorithm 1 depicts the complete design flow of the proposed *ID_hopping_table* generation algorithm. Briefly, the *ID_hopping_table* generation algorithm can be explained as how to generate the combinations of IDs with a fixed relative priority order.

Algorithm 1 ID-Hopping Table Generation ()

Input: *Available_IDs*, *N_pages*, k: Number of CAN message set

Output: The combinations of *Phy_IDs* maximum entropy results

1. $count \leftarrow 0$
2. **while** $count < N\_pages$ **do**
3. **for** $i = 0$ to N_pages **do**
4. **for** $j = 0$ to k **do**
5. Randomly select a number ID from *Available_IDs*
6. $Available\_IDs \leftarrow Available\_IDs - ID$
7. $Id\_table(i)(j) \leftarrow ID$
8. **end for**
9. Sort $Id\_table(i)$ in increasing order of ID priority
10. **for** $m = 0$ to $i - 1$ **do**
11. **if** $id\_table(i) == id\_table(m)$ **then**
12. $i \leftarrow i - 1$
13. **end if**
14. **end for**
15. **end for**
16. $count \leftarrow count + 1$
17. **end while**

In the Algorithm 1, the number of combinations that is equal to the pages of *ID_hopping_table* can be defined as *N_page*. Selection should be carried out among all available combinations. From the perspective of ECU, *ID_hopping_table* can combined into a 3D array, in this study, we set it to *IDs* [*N_page*][k] [11]. The time-consumed step of the algorithm is the sorting operation in Lines 9. The time complexity of traversing $|N|$ tasks is $O(\log |N|)$.

Algorithm 2 Optimization *ID_hopping_table* by Using Simulated Annealing ()

Input: ID_set, k_{max}
Output: ID_set_{best}

1. $ID\_set \leftarrow ID\_set_0$
2. $e \leftarrow H(ID\_set)$
3. $ID\_set_{best} \leftarrow ID\_set_0$; $e_{best} \leftarrow e$
4. $k \leftarrow 0$
5. **while** $k < k_{max}$ and $e < e_{max}$ **do**
6. $\quad ID\_set_{next} \leftarrow neighbour(ID\_set)$
7. $\quad e_{next} \leftarrow H(ID\_set_{next})$
8. $\quad$ **if** $random() < P(e, e_{next}, temp(k/k_{max}))$ **then**
9. $\quad\quad ID\_set \leftarrow ID\_set_{next}$; $e \leftarrow e_{next}$
10. $\quad\quad$ **if** $e > e_{best}$ **then**
11. $\quad\quad\quad ID\_set_{best} \leftarrow ID\_set$; $e_{best} \leftarrow e$
12. $\quad\quad$ **end if**
13. $\quad\quad k \leftarrow k + 1$
14. $\quad$ **end if**
15. $\quad$ **return** ID_set_{best}
16. **end while**

B. OPTIMIZATION OF ID HOPPING TABLE

In automotive electronic systems, IDs in CAN bus are generally divided according to different functions and domains (e.g., in-vehicle ECUs use IDs ranging from $0x000$ to $0x5FF$ and the automotive diagnostic tool uses an ID ranging from $0x700$ to $0x7FF$ [16]), meaning that cracking the content of IDs is easy. Wang *et al.* [22] collected 6.673 million CAN message from various vehicles and conduct entropy and the pattern analyses of the messages. The CAN messages were found to have low entropy with an average of 11.436 bits. In this study, we calculate the message ID's entropy using Shannon entropy definition. Assuming system X, its limited set of possible states is $x_1, x_2, \ldots, x_M$, then the information entropy of system X is:

$$H(X) = -\sum_{i=0}^{N} p(x_i) \log p(x_i), \tag{6}$$

where $p(x_i)$ is the probability of system X in state x_i.

ENTROPY OF PHYSICAL IDs

For the evaluation of the information entropy of CAN IDs, a CAN system model can be represented by $\Phi = (I, C, T)$, where $I = \{\iota_1, \iota_2, \iota_3, \cdots, \iota_n\}$ is a set of different IDs appearing within time T, and $C = \{c_1, c_2, c_3, \cdots, c_n\}$ is the set of periods or the minimum intervals of n different IDs that appear within time T. Subsequently, the entropy function of CAN IDs in period T can be expressed as:

$$H(I) = -\sum_{\iota \in I} p_\iota \log p_\iota \tag{7}$$

Assuming that the system is schedulable, it means that all CAN messages can meet their deadlines [18], and the total number of messages N_{total} in time T can be obtained by the period of messages in T and length of T:

$$N_{total} = \left(\frac{T}{c_1} + \frac{T}{c_2} + \cdots + \frac{T}{c_n}\right) = T \sum_{i \leftarrow 1}^{n} \frac{1}{c_i} \tag{8}$$

The number of ι_i appears in T is $n_i = T/c_i$, then the probability of $\iota_i$appears in T can be presented as $P(\iota_i)$:

$$P(\iota_i) = \frac{n_i}{N_{total}} = \frac{T}{c_i} \times \frac{1}{T \sum_{i=1}^{n} \frac{1}{c_i}} = \frac{1}{c_i \sum_{i=1}^{n} \frac{1}{c_i}} \tag{9}$$

Obviously $\sum_{i=1}^{n} P(\iota_i) = 1$, $P(\iota_i) > 0$, where $(i = 1, 2, \ldots, n)$. The self-information of ι_i is:

$$U_{\iota_i} = \log \frac{1}{P(\iota_i)} = \log c_i \sum_{i=1}^{n} \frac{1}{c_i} \tag{10}$$

Then, in the sampling period T, the entropy of ι_i in CAN bus is:

$$H_{\iota_i} = P(\iota_i) U(\iota_i) = \frac{\log c_i \sum_{i=1}^{n} \frac{1}{c_i}}{c_i \sum_{i=1}^{n} \frac{1}{c_i}} \tag{11}$$

The average entropy of IDs in sampling period T is:

$$H(I) = E[U(\iota_i)] = \sum_{i=1}^{n} H_{\iota_i} \tag{12}$$

Theorem 1: A high number of I leads to high information entropy and strong uncertainty.

Proof: Let x be $c_i \sum_{i=1}^{n} 1/c_i$, then acquire the function $f(x) = (\log x)/x$. In the case where x is greater than 0, the function is monotonically increasing, as I increases, H_i also increases and $H(I)$ increases. The theorem is proven. □

Therefore, we propose an *ID_hopping_table* optimization algorithm to get diverse ID combinations to form a *ID_hopping_table* with the maximized entropy in Algorithm 2. Algorithm 2 depicts the complete design flow of the proposed *ID_hopping_table* optimization algorithm. The simulated annealing (SA) has the advantage of relatively easy implementation and the ability to provide a reasonably good solution for many combinational optimization problems. Therefore, we chose SA to optimize the *ID_hopping_table* generated by Algorithm 1.

In Algorithm 2, *neighbour*()is a neighbour function that generates the candidates of ID combinations based on Algorithm 1. $H()$ is an energy function that calculates $H(ID\_set)$. And, ID_set_0 is an initial solution that is randomly generated. The main time complexity of Algorithm 2 is presented in Lines 5-9. Because the time complexity of the Algorithm 1 is $O(|N|^2)$. The time complexity of Algorithm 2 is $O(|N|^2 \times \log|N| \times |W|)$.

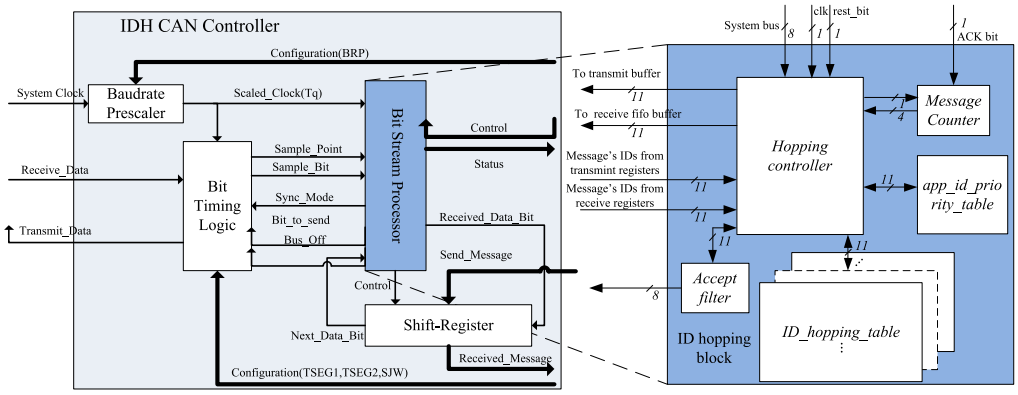


FIGURE 8. Structure diagram of the ID hopping CAN controller.

VI. HARDWARE-BASED IMPLEMENTATION

A. HARDWARE-BASED IMPLEMENTATION

In order to implement the IDH-CAN mechanism, an ID hopping CAN controller design named IDHCC is proposed in this study, which was published in [23] as our early work. And this paper will introduce its design implementation details. The IDHCC and is written by open source IP core from OpenCores community, which was designed by Igor [24]. The IP core is compatible with the SAJ1000 controller is based on the Verilog hardware description language, which supports the CAN 2.0A/B protocol, including CAN standard frames (11-bit IDs) and extended frames (29-bit IDs). CAN 2.0A/B has 64-byte receive FIFO, and the transmission rate can reach up to 1 Mbit/s.

As shown in Figure 8, four modules are presented in the top-level module of IDHCC. They are *ID_hopping_module*, *can_registers*, *can_btl*, and *can_bsp*.

The *ID_hopping_module* has two single modules, namely *app_id_priority_table* and *ID_hopping_table*. The *app_id_priority_table* gets from the priority order-sorted in *App_IDs*. In the ID hopping mechanism scenario, with the message counter synchronization on the bus, the ID hopping controller in the ECU controls the ID hopping process. The information security of CAN message transmission is enhanced under the condition that the relative priority of messages is kept unchanged.

1) HOPPING CONTROLLER

The main function of the hopping controller is to convert *App_IDs* to *Phy_IDs* when transferring message on the physical layer. Similarly, mapping the message ID of the physical layer to the application layer (convert *Phy_IDs* to *App_IDs*) is the reverse of this process. This transition trigger condition comes from the message counter of the IDHCC. Under the action of the hopping controller, new receive filter registers for receiver ECUs can be obtained from the new page of *ID_hopping_table*. The hopping controller finds the message ID of the corresponding physical layer by looking up *ID_hopping_table* according to the message counter value and the relative priority of the messages. The hopping controller module is connected to the *can_bsp* module via a 32-bit bus in the IDHCC.

2) APPLICATION ID TABLE

An application ID table is represented as *app_id_priority_table*, which contains the message IDs from the application layer software that need to be sent or received by ECU. The IDs in the *app_id_priority_table* are sorted by their priority. Therefore, when the application layer message ID (*App_IDs*) needs to be mapped to the physical layer message ID (*Phy_IDs*), or the physical layer message ID (*Phy_IDs*) needs to be restored to the application layer message ID ((*App_IDs*)), the *app_id_priority_table* can be used to find the priority of the message. The maximum size of the application ID table in this study is 256.

3) ID HOPPING TABLE

When designing the ID hopping table, we ensure that the system message ID combination (one page of *ID_hopping_table*) has a relative priority at each moment. This relative order of priority ensures that the relative priorities of the system messages remain unchanged. In this study, *ID_field_length* $\times$ *ID_depth*(maximum 256), and the data in each page of the table are sorted by the message's priority. The *ID_hopping_table* contains the message ID for the physical layer message to be transmitted on the CAN bus. In the hardware implementation process, we implement it through a 3-dimensional array based on Verilog on FPGA.

4) ACCEPT FILTER

In the early design stage of automotive real-time applications, the message sets that are received or sent by the nodes can be obtained through the system design and planning. Meanwhile, we can obtain the ECU node acceptance code and acceptance mask register according to the messages that the nodes need to receive. There are 4 acceptance code and 4 acceptance mask registers

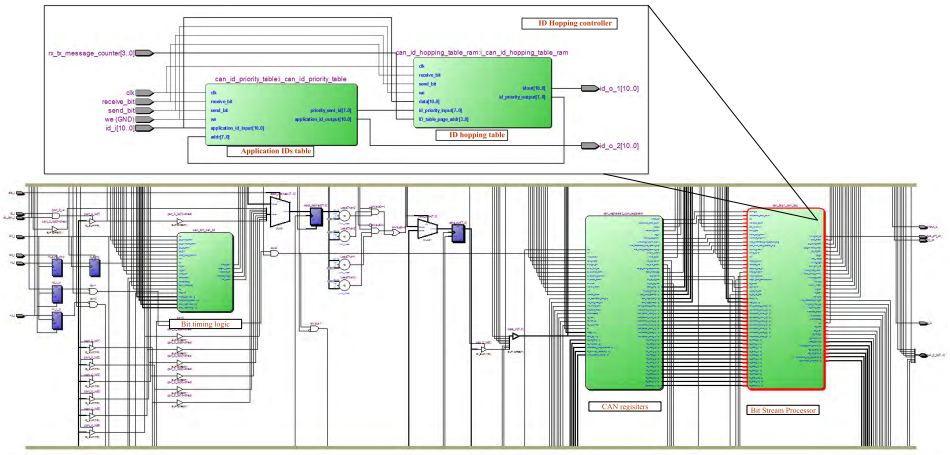


FIGURE 9. RTL view of the ID Hopping CAN controller.

TABLE 1. Comparison of resource consumption.

	CAN controller	ID hopping CAN controller	Compare
Device	EP4CE22F17C6	EP4CE22F17C6	Same
Total logic elements	2,313 / 22,320 (10 %)	2,362 / 22,320 (11 %)	+0.2%
Total combinational functions	1,927 / 22,320 (9 %)	1,996 / 22,320 (9 %)	+0.3%
Dedicated logic registers	1,175 / 22,320 (5 %)	1,265 / 22,320 (5 %)	+0.4%
Total pins	19 / 154 (12 %)	30 / 154 (19 %)	+7%

in the proposed IDHCC, where registers will be written to the *can_registers* module at the deployment phase and work in the *CAN_acf* module at the operation phase. The receiving nodes includes in the *CAN_acf* module, which determines whether to accept or discard a received data frame.

After completing the IDHCC design compilation in Quartus Prime, the RTL view of the ID Hopping CAN controller is shown in Figure 9. From the Figure 9 we can see that the ID hopping function is mainly done in the Bit Stream Processor (BSP) module of the CAN controller.

B. RESOURCE CONSUMPTION

Given that the automotive electronics systems are resource-constrained and cost-sensitive. The goal of the IDHCC hardware implementation is to minimize resource consumption. As described in Table 1, compared with ordinary CAN controllers, the consumption of total logic elements has increased by 0.2%, the consumption of total combinational functions has increased by 0.2%. The comparison of the proposed IDHCC with a normal CAN controller [24] shows that IDH-CAN consumes slightly increased hardware resources.

A security life cycle for cyber-physical automotive systems under the standard SAE J3061 [25] should be referenced to ensure security. In this way, the *ID_hopping_table* can be stored in a tamper-proof storage (e.g., a SHE or HSM secure memory) and can only be used by a legitimate ECU.

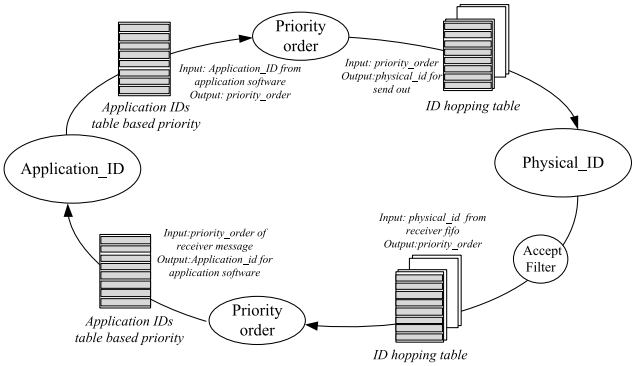


FIGURE 10. Data flow in IDH-CAN.

For a balanced consideration of cost and safety requirements, *N_page* is divided into three levels according to the ASIL. The ASIL with different *N_page* configurations and memory consumption is depicted in Table 2.

C. DATA FLOW IN IDH-CAN

The difference between IDH-CAN and the common CAN protocol is that at the data link layer of the CAN, a message ID conversion and restoration process are added. In order to more clearly show the IDH-CAN mechanism, we use the Figure 10 to express the data flow in IDH-CAN, which mainly includes the following two parts: transmitter and receiver link.

TABLE 2. ASIL and *N_pages* configuration.

ASIL	*N_page*	EEPROM
A	4	10Kbits
B	8	20Kbits
C	16	40Kbits

1) FOR TRANSMITTER LINK

After the ECU sends the message to the CAN controller by reading the register, IDHCC obtains the message priority by looking at the priority of *app_id_priority_table*, the *app_id_priority_table* will be written to IDHCC in design phase too. The *App_IDs* can then be converted to *Phy_IDs* by looking up the *ID_hopping_table* for sending (synchronized by the message counter). Therefore, the actual transmitted message ID on the physical layer of the CAN network is different.

2) FOR RECEIVER LINK

When a new message from a legal ECU is transmitted on the CAN bus, if the message ID passes the accepting filter of the IDHCC, then the message will be processed by the ECU. If this message comes from an illegitimate ECU (replay or insert attack from an attacker), this message will be discarded, because it cannot pass the synchronization of the ID transition and cannot pass through the ECU's acceptance filters. Legal messages will be received and processed by the ECU in two steps. First, the relative priority of the message can be obtained by looking up the *ID_hopping_table* based on the *Phy_IDs* and message counter of the ECU, and the *App_IDs* can be obtained by looking up the application IDs table according to the relative priority. Second, the ID for the application software layer is obtained.

D. WAVEFORM SIMULATION

Before deploying the proposed IDHCC to the FPGA platform, we design a test bench to verify the correctness of our IDHCC design. Our experimental environment is ModelSim ALTERA STARTER EDITION 10.1d, and the simulation experiments mainly include CAN message sending and receiving tests in IDHCC environment. Similar to the stage of automotive real-time application deployment, the first step in our test bench is to write *app_id_priority_table* and *ID_hopping_table* in IDHCC. The simulation test can be divided into the following aspects.

1) FOR TRANSMITTER

When the IDH-CAN controller is in the transmission state, as shown in Figure 10, the message ID that needs to be sent is 00000000000. However, the transmitted ID waveform on the physical bus is converted to 10101010100 as the waveform shows in Figure 11. If we keep sending the same message ID, the waveform of ID field is change according to the message counter. Hopping synchronization occurs based on *ACK* signal. The simulation demonstrates that the sending function of IDH-CAN controller is correct.

```
task send_frame_basic;    // CAN IP core sends frames
  begin
    //write register for message transmitter send out
    write_register(8'd10, 8'h00); // Writing ID[10:3] = 0x00000000
    write_register(8'd11, 8'h08); // Writing ID[2:0] = 0x001, rtr = 0, length = 8
    write_register(8'd12, 8'h56); // data byte 1
    write_register(8'd13, 8'h78); // data byte 2
    write_register(8'd14, 8'h9a); // data byte 3
    write_register(8'd15, 8'hbc); // data byte 4
    write_register(8'd16, 8'hde); // data byte 5
    write_register(8'd17, 8'hf0); // data byte 6
    write_register(8'd18, 8'h0f); // data byte 7
    write_register(8'd19, 8'hed); // data byte 8
```

The application layer's ID is written to the register

FIGURE 11. The message ID is written to the register.

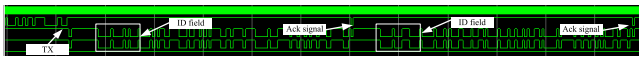


FIGURE 12. The waveform on the physical CAN bus.

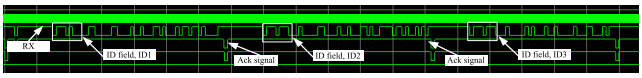


FIGURE 13. The waveform on the physical CAN bus.

```
# (490884400)
# (490924900) Reading register [20] = 0x0
# (490962400) Reading register [21] = 0x61
# (490999900) Reading register [22] = 0x12
# (491037400) Reading register [23] = 0x0
# (491074900) Reading register [24] = 0x62
# (491112400) Reading register [25] = 0x12
# (491149900) Reading register [26] = 0x34
# (491187400) Reading register [27] = 0x1
# (491224900) Reading register [28] = 0xc2
# (491262400) Reading register [29] = 0x12
# (491262500)
# (491299900)
# (491299900) Writing register [1] with 0x4
# (491337400) Rx buffer released.
```

ID1
ID2
ID3
The IDs read from the FIFO buffer

FIGURE 14. The message ID is written to the register.

2) FOR RECEIVER

The IDH-CAN controller will convert *Phy_IDs* to *App_IDs*. In the receiving state, after message passes through the CRC, and accept filter, the priority value is obtained by looking up the *ID_hopping_table*, and the ID correspond to the application layer is obtained. And, the *App_IDs* is stored in the 64-byte of the FIFO buffers for application software. As the waveform shown in Figure 13, the actual received ID waveform from the CAN bus is 11101000001 and 11101110001, but the ID written to the receiver FIFO register is converted to 00000000011 and 00000000101, as depicted in Figure 14.

The simulation experiments show that the IDH-CAN controller can work correctly and that the IDs transmitted on the physical layer change according to the message counter (triggered via the *ACK* signal).

VII. EVALUATION

The simulation results in Section VI-D demonstrated that IDHCC can implement the ID hopping mechanism correctly. To validate the effectiveness of IDH-CAN, we evaluate IDH-CAN from four aspects, namely resource consumption, performance analysis, entropy analysis, and security

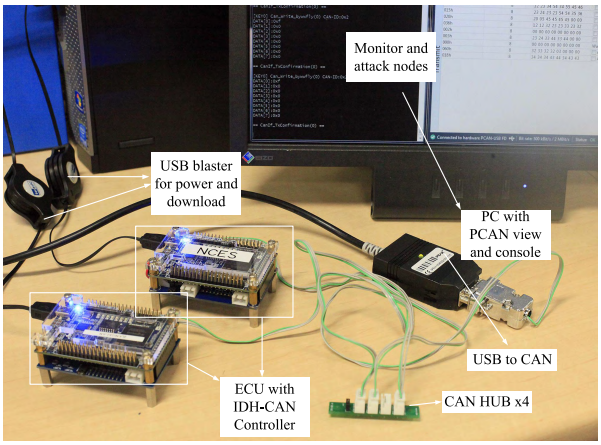


FIGURE 15. Evaluation platform.

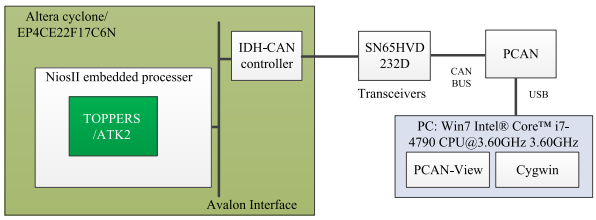


FIGURE 16. Block diagram of evaluation platform.

effectiveness. Several numerical examples are presented to confirm the proposed mechanism.

A. PERFORMANCE ANALYSIS

To evaluate the performance of the IDH-CAN controller, we carry out a practical deployment experiment, as shown in Figure 15. The block diagram of evaluation platform is presented in Figure 16. The evaluation platform used in this study is a DE0-Nano with Altera Cyclone/EP4CE22F17C6. In our experiments, the IDH-CAN controller is connected to the Nois II embedded processer by Avalon Interface. Our test application code runs on Toppers/ATK2 (http://www.toppers.jp). The experimental results on the FPGA platform show that the designed controller can reach 25MHz.

1) COMMUNICATION RESPONSE TIME

Given that synchronization is achieved using the message counter, the start message is fixed to the highest priority and fixed message counter cycles. The simulation experiments in Section VI-D demonstrated that IDHCC can work at 1 Mbps. Through theoretical analysis, the number of instructions in one cycle required for IDHCC to complete the hopping operation is 3. Theoretically, the time required for IDHCC to complete the hopping process is 240 μs, which is shorter than C_m 0.136 ms as described in Section IV-B.

2) SCHEDULABILITY ANALYSIS

Because the IDH-CAN needs not additional messages to achieve ID hopping synchronization, and the schedulability of CAN messages is guaranteed by the fixed priority order of the system messages, therefore, we can conclude that the proposed IDH-CAN can be applied to existing automotive real-time applications.

B. ENTROPY ANALYSIS

The greater information entropy means the diversity and uncertainty of the network ID, and it means that the network is more resistant to attacks. For the comparison of the entropy value of CAN IDs in different methods, we generate an SAE benchmark-based message set (the SAE benchmark message set originally included 53 signals [26]).

1) ENTROPY COMPARISON ANALYSIS

To compare the diversity of physical layer IDs between the normal CAN, the method mentioned in [27], and our algorithm, we design an experiment in which we observe the changes of information entropy in different sampling times. In this experiment, we consider the system with 200 messages in a single CAN cluster and the probability of messages is obtained on the basis of Equation (9). The average entropy of the IDs in a CAN cluster in the sampling period T is obtained with Equation (12). Figure 17(a) depicts the results of such comparison. For a normal CAN, the IDs of messages are fixed. Thus, the information entropy of CAN does not change, although the sampling period changes from 450ms to 2400ms. However, in IDH-CAN and the method proposed in [27], the type of ID increases with the increase in sampling time. Therefore, we observe that our solutions mostly have a larger entropy values than the normal CAN and method in [27].

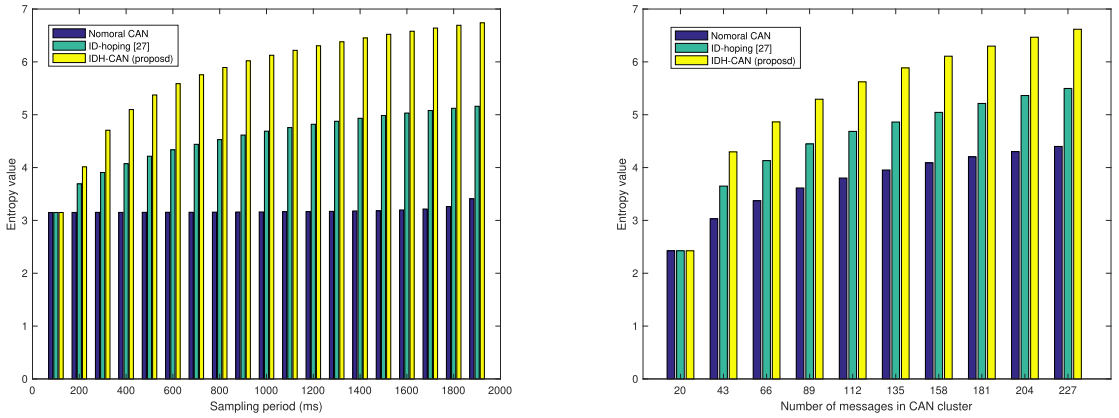
In practical automotive real-time applications, different CAN cluster usually has different message set sizes. Therefore, we design a comparison experiment of different methods under different message sets. In this experiment, we consider that the sampling period time is fixed at 2000ms, and the number of message sets ranges from 30 to 250. Figure 17 (b) illustrates that the physical IDs obtained through our method have larger entropy values in each size of message sets than other methods.

2) IMPACT OF ID HOPPING PAGE NUMBER

In this study, we balance cost and security requirements by designing IHD-CAN with different N_pages to correspond to different ASILs (as shown in Table 2). To compare the different entropy results of IDH-CAN with different N_pages, two similar experiments previously mentioned are conducted. Figure 18(a) and Figure 18(b) show that a larger number of N_pages equates to larger entropy value obtained from the IDH-CAN mechanism. It also indicates higher security performance and greater memory consumption.

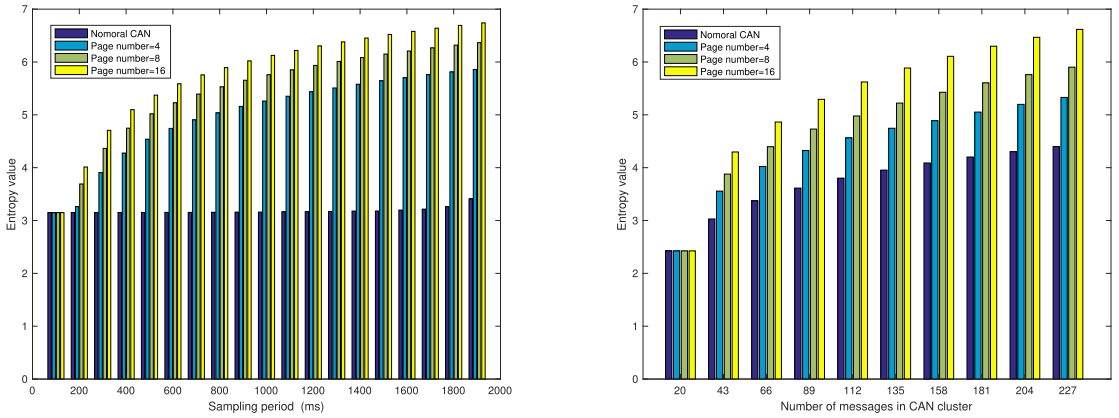
C. SECURITY AND EFFICIENCY COMPARISON

The security performance of IDH-CAN is analyzed and evaluated from the following aspects: targeted DoS attacks, replay attacks, and reverse engineering. Figure 15 shows the evalua-



(a) (b)

FIGURE 17. Comparative analysis of information entropy value between different CAN mechanisms. (a) Entropy value compare with different Sampling period. (b) Entropy value compare with different message number.



(a) (b)

FIGURE 18. Comparison and analysis of entropy values under different IDH-CAN parameter. (a) Entropy value compare with different ID hopping pages. (b) Entropy value compare with different message number.

tion platform, in which two nodes deploy IDHCC to transmit and receive CAN messages, and the PC, as the listener and attack node monitors and sends attacks.

1) REPLAY ATTACKS

In the IDH-CAN scenario that all ECUs in automotive electronics systems equipped with IDHCC, where N_pages represents the pages of the $ID\_hopping\_table$, $N_{num}\_App\_IDs$ represents the number of IDs in the application layer. The success rate of the replay attacks against the IDH-CAN system can be computed as

$$\frac{1}{N_{num}\_Phy\_IDs}, \tag{13}$$

where $N_{num}\_Phy\_IDs$ is the total number of IDs in the physical CAN bus with diversity of Phy_IDs. For normal CAN, $N_{num}\_Phy\_IDs$ is equal to $N_{num}\_App\_IDs$, but in IDH-CAN, $N_{num}\_Phy\_IDs$ is equal to

$$N_{num}\_App\_IDs \times N\_pages. \tag{14}$$

Therefore, the theoretical success rate of replay attacks against IDH-CAN is computed as

$$\frac{1}{N_{num}\_App\_IDs \times N\_pages}. \tag{15}$$

Obviously, this probability is fewer than that recorded in normal CAN (which is $\frac{1}{N_{num}\_App\_IDs}$). In the IDH-CAN scenario, because attackers cannot obtain ID hopping synchronization and cannot determine the priority order of the system messages, the diversity and unformatted of message IDs on the physical layer of the CAN network can increase the difficulty of reverse engineering and replay attacks.

2) TARGETED DoS ATTACKS

Targeted DoS attack which uses special messages for targeted ECU. For a physical attack under the proposed IDH-CAN mechanism, when ID hopping occurs, the ID in the previous page of the $ID\_hopping\_table$ becomes illegal in the next page of the $ID\_hopping\_table$. Therefore, IDH-CAN

TABLE 3. Performance comparison of CAN security enhancement methods.

	MAC[14], [15]	CAN+ [17], [28], [16]	IA-CAN[21]	ID hopping[27]	Proposed
Computational complexity	High	Middle	Low	Low	Low
Time delay	Middle	High	No	Low	No
Data-field consumption	High	No need	Middle	No need	No need
Additional messages	No need	Need	No need	Need	One
Schedulability analysis	Easy	Complex	Complex	Complex	Easy

mechanism can protect against targeted DoS attacks. The probability of a successful DoS attack for an unsecured CAN is 100%. However, with the proposed IDH-CAN, DoS attacks against individual nodes fail.

3) REVERSE ENGINEERING

Data collection analysis is usually the first step in the in-vehicle cyber attack process [16]. Wang et al. [22] found that CAN messages have low entropy with an average of 11.436 bits. Moreover, in-vehicle ECUs and the automotive diagnosis tool usually use fixed ID field [16]. Hence, in-vehicle networks are easy to be reverse engineered by attackers. In this study, the results of the entropy analysis in Section VII-B show that IDH-CAN has a larger entropy value in each message set. Therefore, IDH-CAN can improve the prevention of reverse engineering.

As presented in Table 3, IDH-CAN has the following advantages. First, compared with the MAC method, IDH-CAN does not need to consume data fields, and additional messages, simultaneously requires less computation time overhead. Second, IDH-CAN consumes less communication messages than CAN+, so it has higher bandwidth utilization. Third, compared with IA-CAN, our proposed method easily implements a schedulability analysis of CAN messages and is applicable to all types of CAN messages. Fourth, compared with the ID hopping method proposed in [27], the IDH-CAN makes the physical layer ID increasingly diversified and consumes a short processing time.

VIII. RELATED WORK

Previous researches have designed different security mechanisms to protect CAN from attacks and meet its security needs [14]–[17], which can be divided into three categories. The methods in the first category are authenticated by digital signature and MAC [14], [15], which will consume limited payload of CAN message frame and computing resources of node. Due to the resource constraints (calculation resources and bandwidth resources) and cost-sensitive features of automotive real-time applications, ID hopping is more suitable for security enhancement for CAN than message encryption and authentication. The second category is CAN+ [16], [17], a CAN-specific authentication model that requires additional messages. The third category is anonymous IDs. Han and Shin [21] introduced a new solution named Identity-Anonymous CAN (IA CAN), which enables CAN messages to be anonymized with message IDs at the physical layer.

The major disadvantage of the IA-CAN is that it cannot guarantee the worst-case response time (WCRT) of CAN messages. Therefore, the IA-CAN is not suitable for automotive real-time applications. Moreover, it only applies to periodic messages, which means that this method is not very practical for an event triggered CAN network.

The most relevant research to our work is the ID hopping mechanism proposed by Humayed and Luo [27]. The hopping synchronization in [27] is achieved through a special ID from the gateway, and the ID hopping is implemented based on software. Therefore, the conversion time of ID hopping is unavoidable. Our method is different from the method mentioned in [27] and mainly includes the following three aspects.

- First, method in [27] is software-based implementation, and ours is hardware-based. The advantage of the hardware implementation is that it does not require the ECU to consume additional computational overhead.
- Second, the hopping synchronization mechanism and parameter adopted between the two method is different, special ID is used in [27] to achieve synchronization, which means that the additional message is unavoidable. But the message counter is used in this study. Due to the CAN protocol's broadcast nature, the message counter is a shared parameter for all ECUs on the CAN bus. Therefore, our method requires fewer additional messages and computation overhead.
- Third, the method of *ID_hopping_table* generation is different. In [27], the next hopping IDs are equal to the previous IDs add offset after hopping in the runtime. However, in this study, *ID_hopping_table* is generated in the phase of system design, and stored in a tamper-proof storage (e.g., in a SHE or HSM secure memory) to ensure that only authenticated ECUs can access the table. The advantage of this method is that the IDs transmitted on the physical layer are more diverse.

IX. CONCLUSION

In this study, we proposed a real-time guaranteed security-enhanced IDH-CAN mechanism for automotive real-time applications. To implement this mechanism, a non-ISO IDH-CAN controller IP core is designed and implemented. To maximize the entropy of physical IDs used in IDH-CAN, we also designed the *ID_hopping_table* generation and

optimization algorithms for IDH-CAN. The advantage of this mechanism is that if the ECU is equipped with an IDH-CAN controller, the existing schedulability analysis model for CAN messages is applicable, which is especially important for safety critical systems in automotive electronic environments. Waveform simulation experiments verified the correctness of the IDH-CAN controller. The results of the entropy analysis experiments showed that the IDs transmitted over CAN bus have greater diversity, which can effectively improve the security performance of CAN at a low performance overhead. Finally, we compared and analyzed the effectiveness of IDH-CAN in enhancing cybersecurity. This research contributes to the literature by guaranteeing that CAN bus can defend systems from anti-engineering, sniffing, targeted DoS attacks, and replay attacks without requiring much computing and communications resources. The future research direction is to further improve the IDH-CAN design, such as further optimizing hardware resource consumption and improving system robustness.

ACKNOWLEDGMENT

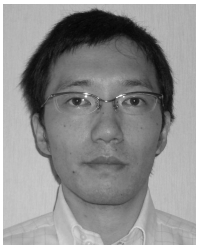
The authors would like to express their gratitude to the associate editor and anonymous reviewers for their constructive comments, which have helped to improve the quality of the paper. This paper was presented at the Proceedings of the Workshop on Security and Dependability of Critical Embedded Real-Time Systems (CERTS 2017) in conjunction with the IEEE Real-Time System Symposium (RTSS 2017), Paris, France, December 5, 2017.

This paper is an extended and revised version of the paper "IDHCC: A Security-Enhanced ID Hopping CAN Controller Design to Guarantee Real-Time" presented at the 2nd Workshop on Security and Dependability of Critical Embedded Real-Time Systems (CERTS2017), in conjunction with IEEE Real-Time System Symposium (RTSS 2017), Paris, France, December 5, 2017. In this journal version, we propose a new and optimized ID-CAN mechanism. Meanwhile, we further optimized the design of IDHCC. To obtain numerous non-formatting and diversified IDs combinations, we also develop an SA solution to optimize the proposed ID hopping table generation algorithm. We provide information security evaluation based on information entropy comparison experiments to show that our method provide improved results over existing work.

REFERENCES

[1] G. De La Torre *et al.*, "Driverless vehicle security: Challenges and future research opportunities," *Future Gener. Comput. Syst.*, to be published, doi: 10.1016/j.future.2017.12.041.
[2] B. Zheng, H. Liang, Q. Zhu, H. Yu, and C.-W. Lin, "Next generation automotive architecture modeling and exploration for autonomous driving," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2016, pp. 53–58.
[3] R. Kurachi, Y. Chen, H. Takada, and G. Zeng, "An integrated framework for topology design of can networks under real-time constraints," in *Proc. Veh. Technol. Conf.*, Sep. 2015, pp. 1–5.
[4] K. Koscher *et al.*, "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2010, pp. 447–462.
[5] J. Petit and S. E. Shladover, "Potential cyberattacks on automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 546–556, Apr. 2015.
[6] K.-K. R. Choo, R. Lu, L. Chen, and X. Yi, "A foggy research future: Advances and future opportunities in fog computing research," *Future Gener. Comput. Syst.*, vol. 78, pp. 677–679, Jan. 2018.
[7] S. Li, T. Tryfonas, and H. Li, "The Internet of Things: A security point of view," *Internet Res.*, vol. 26, no. 2, pp. 337–359, 2016.
[8] S. Checkoway *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. USENIX Conf. Secur.*, 2012, p. 6.
[9] *Road Vehicles—Functional Safety—Part 1: Vocabulary*, document ISO 26262-1, International Organization for Standardization, 2011.
[10] *Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*, SAE Standard J3016, SAE International, 2014.
[11] A. M. Wyglinski, X. Huang, T. Padir, L. Lai, T. R. Eisenbarth, and K. Venkatasubramanian, "Security of autonomous systems employing embedded computing and sensors," *IEEE Micro*, vol. 33, no. 1, pp. 80–86, Jan. 2013.
[12] K.-T. Cho and K. G. Shin, "Error handling of in-vehicle networks makes them vulnerable," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1044–1055.
[13] U. Hiroshi, K. Ryo, T. Hiroaki, M. Tomohiro, I. Masayuki, and H. Satoshi, "Security authentication system for in-vehicle network," *SEI Tech. Rev.*, vol. 81, pp. 44–55, Oct. 2015.
[14] C.-W. Lin and A. Sangiovanni-Vincentelli, "Cyber-security for the controller area network (CAN) communication protocol," in *Proc. Int. Conf. Cyber Secur.*, Dec. 2012, pp. 1–7.
[15] K. K. Dong, "A practical and lightweight source authentication protocol using one-way hash chain in can," M.S. thesis, Daegu Gyeongbuk Inst. Sci. Technol., Daegu, South Korea, 2017.
[16] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle CAN," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 993–1006, Apr. 2015.
[17] P. Mundhenk, S. Steinhorst, M. Lukasiewycz, S. A. Fahmy, and S. Chakraborty, "Lightweight authentication for secure automotive networks," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, 2015, pp. 285–288.
[18] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Syst.*, vol. 35, no. 3, pp. 239–272, 2007.
[19] G. Xie, G. Zeng, Y. Liu, J. Zhou, R. Li, and K. Li, "Fast functional safety verification for distributed automotive applications during early design phase," *IEEE Trans. Ind. Electron.*, vol. 65, no. 5, pp. 4378–4391, May 2018.
[20] G. Xie *et al.*, "Reliability enhancement towards functional safety goal assurance in energy-aware automotive cyber-physical systems," *IEEE Trans. Ind. Informat.*, to be published. [Online]. Available: https://ieeexplore.ieee.org/document/8409311/
[21] K. Han, A. Weimerskirch, and K. Shin, "Apractical solution to achieve real-time performance in the automotive network by randomizing frame identifier." in *Proc. Eur. Embedded Secur. Cars (ESCAR)*, 2015, pp. 13–29.
[22] E. Wang, W. Xu, S. Sastry, S. Liu, and K. Zeng, "Hardware module-based message authentication in intra-vehicle networks," in *Proc. 8th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, 2017, pp. 207–216.
[23] W. Wu, R. Kurachi, G. Zeng, Y. Matsubara, H. Takada, and R. Li, "IDHCC: A security-enhanced id hopping can controller design to guarantee real-time," in *Proc. 2nd Workshop Secur. Dependability Crit. Embedded Real-Time Syst. (CERTS)*, 2017, pp. 14–21. [Online]. Available: http://certs2017.uni.lu
[24] M. Igor. (2009). *Can Protocol Controller*. [Online]. Available: https://opencores.org/acc,view,igorm
[25] C. Schmittner, Z. Ma, C. Reyes, O. Dillinger, and P. Puschner, "Using SAE J3061 for automotive security requirement engineering," in *Proc. Int. Conf. Comput. Saf., Rel., Secur.*, 2016, pp. 157–170.
[26] T. Kindell, "Guaranteeing message latencies on control area network (CAN)," in *Proc. Int. CAN Conf.*, 1994, pp. 1–8.
[27] A. Humayed and B. Luo, "Using ID-hopping to defend against targeted dos on can," in *Proc. 1st Int. Workshop Safe Control Connected Auton. Vehicles*, 2017, pp. 19–26.
[28] A. I. Radu and F. D. Garcia, "Leia: A lightweight authentication protocol for CAN," in *Computer Security–ESORICS*. Berlin, Germany: Springer, 2016, pp. 283–300.

WUFEI WU (S'17) is currently pursuing the Ph.D. degree with the College of Computer Science and Electronic Engineering, Hunan University, China. His research interests include distributed systems, embedded computing systems, and cyber-physical systems. He is a Student Member of ACM and CCF.



RYO KURACHI (M'18) graduated in applied electronics from the Tokyo University of Science. He received the master's degree in management of technology from the Tokyo University of Science in 2007 and the Ph.D. degree in information science from Nagoya University in 2012. He is a Designated Associate Professor of the Center for Embedded Computing Systems, Nagoya University. He was with AISIN AW CO., LTD., as a Software Engineer before receiving his master's degree. His research interests include embedded systems and real-time systems. Within that domain, he has investigated topics such as in-vehicle networks and real-time scheduling theory and embedded system security.

GANG ZENG (M'03) received the Ph.D. degree in information science from Chiba University in 2006. From 2006 to 2010, he was a Researcher and then an Assistant Professor at the Center for Embedded Computing Systems, Graduate School of Information Science, Nagoya University. He is currently an Associate Professor at the Graduate School of Engineering, Nagoya University. His research interests mainly include power-aware computing, real-time embedded system design. He is a member of IPSJ.

YUTAKA MATSUBARA received the Ph.D. degree in information science from Nagoya University in 2011. He was a Researcher from 2009 to 2011 and a designated Assistant Professor in 2012 at Nagoya University, where he is currently an Assistant Professor at the Center for Embedded Computing Systems, Graduate School of Informatics. His research interests include real-time operating systems, real-time scheduling theory, and system safety and security for embedded systems. He is a member of USENIX and IPSJ.

HIROAKI TAKADA (M'92) received the Ph.D. degree in information science from the University of Tokyo in 1996. He was a Research Associate at the University of Tokyo from 1989 to 1997. He was a Lecturer and then an Associate Professor at the Toyohashi University of Technology from 1997 to 2003. He is currently a Professor at the Institute of Innovation for Future Society, Nagoya University. He is also a Professor and the Executive Director of the Center for Embedded Computing Systems, Graduate School of Informatics, Nagoya University. His research interests include real-time operating systems, real-time scheduling theory, and embedded system design. He is a member of ACM, IPSJ, IEICE, JSSST, and JSAE.

RENFA LI (M'05–SM'10) is currently a Professor of computer science and electronic engineering and the Dean of the College of Computer Science and Electronic Engineering, Hunan University, China. He is the Director of the Key Laboratory for Embedded and Network Computing of Hunan Province, China. He is also an Expert Committee Member of the National Supercomputing Center, Changsha, China. His major interests include computer architectures, embedded computing systems, cyber-physical systems, and Internet of Things. He is a member of the council of CCF and a senior member of ACM.

KEQIN LI (M'90–SM'96–F'15) is currently a Distinguished Professor of computer science with the State University of New York. He has published over 600 journal articles, book chapters, and refereed conference papers. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU–GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of Things, and cyber-physical systems. He has received several best paper awards. He is currently or has served on the editorial boards of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON SERVICES COMPUTING, the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.